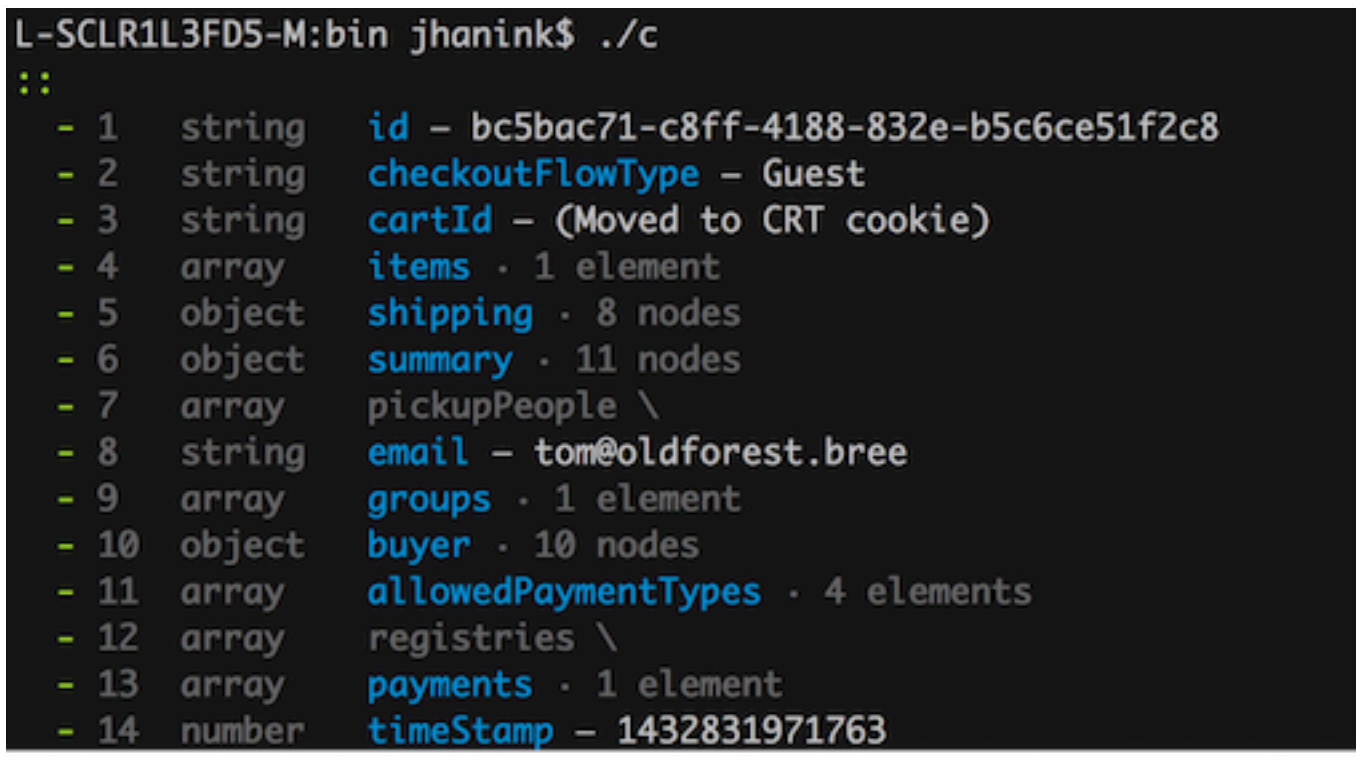
What is this?

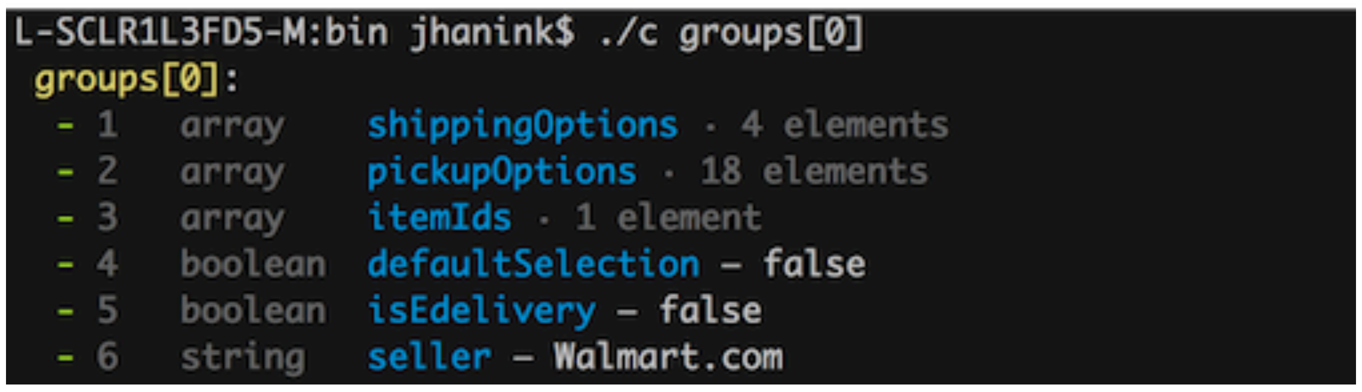
CLAPI - Command Line API processor (mainly for cart and checkout)

```
git clone https://gecgithub01.walmart.com/jhanink/clapi.git
cd clapi
npm install -g nodemon
npm install
cd bin
```

View and search a JSON data tree as simply as navigating a file-system



```
L-SCLR1L3FD5-M:bin jhanink$ ./c
::
  - 1   string   id - bc5bac71-c8ff-4188-832e-b5c6ce51f2c8
  - 2   string   checkoutFlowType - Guest
  - 3   string   cartId - (Moved to CRT cookie)
  - 4   array    items · 1 element
  - 5   object   shipping · 8 nodes
  - 6   object   summary · 11 nodes
  - 7   array    pickupPeople \
  - 8   string   email - tom@oldforest.bree
  - 9   array    groups · 1 element
  - 10  object   buyer · 10 nodes
  - 11  array    allowedPaymentTypes · 4 elements
  - 12  array    registries \
  - 13  array    payments · 1 element
  - 14  number   timeStamp - 1432831971763
```



```
L-SCLR1L3FD5-M:bin jhanink$ ./c groups[0]
 groups[0]:
  - 1   array    shippingOptions · 4 elements
  - 2   array    pickupOptions · 18 elements
  - 3   array    itemIds · 1 element
  - 4   boolean  defaultSelection - false
  - 5   boolean  isEdelivery - false
  - 6   string   seller - Walmart.com
```

Features

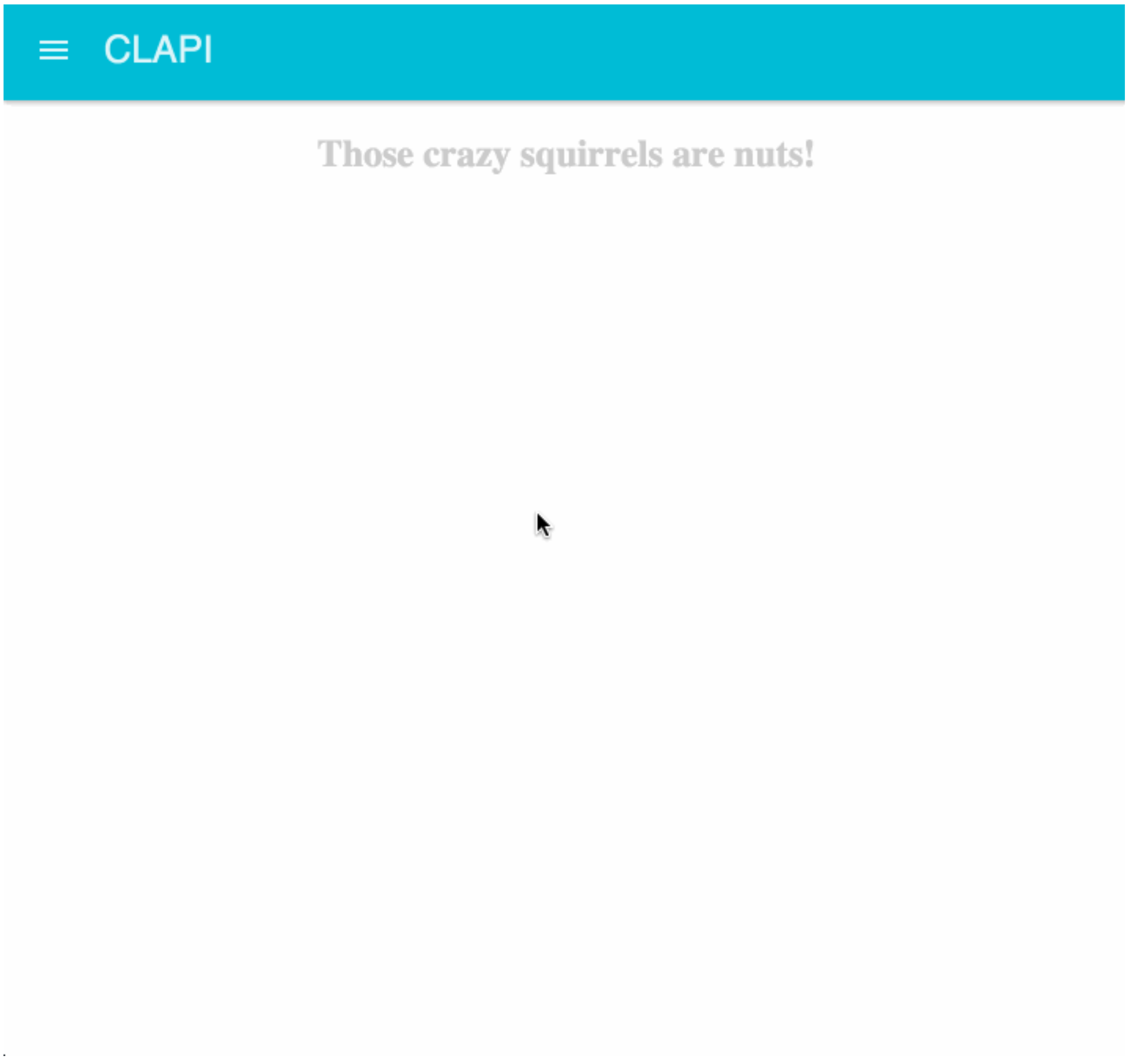
- Call an API and view in JSON, PRETTY, or Interactive mode
- Use the CLAPI result buffer to grep, pipe, or process with custom functions
- Prefer to use a GUI? see the next section, "Using the UI"

Use the UI

React UI components plus material-ui theme

```
# start an auto hot-reload server
npm start

# open an auto hot-reloading UI
open -a "Google Chrome" http://localhost:3000

# open an auto hot-reloading UI with update indicator
open -a "Google Chrome" http://localhost:8080/webpack-dev-server/build/bundle
```

Run the API commands from the GUI



Commands

Add new requests here

Available commands:

→ STAGE : Cart & Checkout

```
  ./get-customer 688ddfc5-181f-46b5-a0e7-8dc139146253       # customerId
  ./get-customer node-1@wm.com                              # email
```

```
  ./create-gift-card 100                                    # amount
```

```
./create-temp-card 688ddfc5-181f-46b5-a0e7-8dc139146253    # customerId
```

```
./create-cart 688ddfc5-181f-46b5-a0e7-8dc139146253         # customerId

./create-cart 688ddfc5-181f-46b5-a0e7-8dc139146253 \
    --EVAL cart.id | pbcopy                                # create cart, copy to clipboard

./get-cart 6a6f9ddb-8e95-4083-9efe-d1bbb544d03b            # cartId
./clear-cart 6a6f9ddb-8e95-4083-9efe-d1bbb544d03b          # cartId
```

```
./add-to-cart 989CF1FB215E4C579A273357D8DE5111             # offerId
./add-to-cart 9875792                                      # itemId
./update-cart-item  `./get-cart --EVAL=items[0].id` 5      # id, quantity   (not USItemId)
./delete-cart-item  `./get-cart --EVAL=items[0].id`        # id             (not USItemId)
```

→ STAGE : Other

```
./fetch-inventory-report --NEW                             # fetches the latest report
./fetch-inventory-report --EVAL result[0]                  # get first item from cached result
./fetch-inventory-report --FUNC printFetchedItems          # print condensed report
```

```
./get-iro-offers 989CF1FB215E4C579A273357D8DE5111          # get IRO offers by offerId
./get-iro-offers 17753319                                  # get IRO offers by USItemId
./get-iro-offers --MORE                                    # print cached result
./get-iro-offers --EVAL status                             # print IRO status (OK, PARTIAL..)

./get-iro-offers --upc 084522601117                        # get IRO offers by upc
./get-iro-offers --wupc 0084522601117                      # get IRO offers by wupc

./get-receipt 26686011496922631859                         # get receipt data by TC#
```

```
./create-purchase-contract                                 # create 1hg sample purchase contract

./get-purchase-contract \
   53f91076-0f31-4456-b214-74e3741b7d77                    # get purchase contract by id
```

→ STAGE : Set Configs

edit the file `~/clapi-config-override.json` or use the `config-set` commands.

```
./config-set-cid                                            # set cid for purchase-contract
```

→ OPTIONS : DISPLAY FORMAT, EXPRESSION EVALUATION, CUSTOM FUNCTIONS

```
# output modes and options
./get-customer                                             # default output
./get-custmer --JSON                                       # JSON output
./get-customer --JSON | more                               # JSON piped to more
./get-customer --LESS                                      # default piped to less
./get-customer | grep accountType                          # default, grep for accountType

# interactive object navigation mode
./get-cart -i                                              # list obj props under root node
./get-cart -i cart                                         # list obj props under named node
./get-cart -i cart --NOCOLOR | pbcopy                      # remove color codes before copy
```

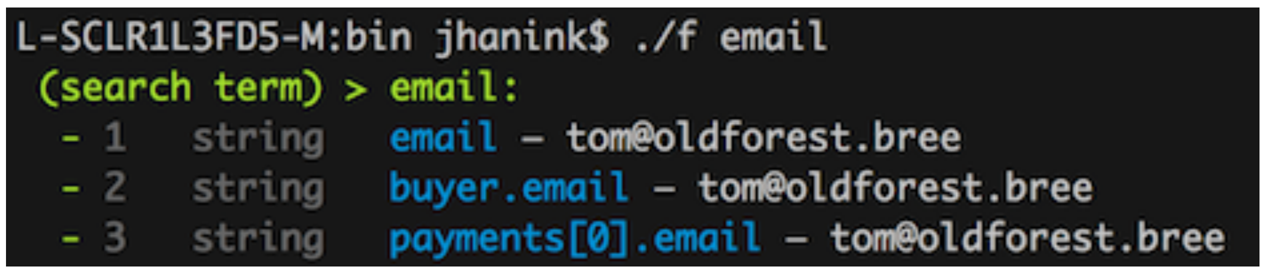
```
./get-iro-offers -i \
     payload[0].product.productAttributes["has-mercury"]   # use dashes instead of spaces in keys

# evaluate fixed nodes
./get-customer --EVAL payload.person.customerAccountId     # eval object for a json property
./get-cart --EVAL cart.id                                  # eval object for a json property
./get-cart --EXPR obj.cart.id                              # eval from root object reference

# call custom functions
./get-cart --FUNC printCartItems                           # run a custom function on result
```

All the above commands save to the CLAPI buffer and can be immediately driven by ./clapi (below)

→ CLAPI : INTERACTIVE MODE

```
# implicitly save a result to the clapi buffer
./get-cart

# explicitly save a json file to the clapi buffer
./clapi -f=../samples/SAMPLE.json

# interactive mode against the existing clapi buffer
./clapi
./clapi summary

# use the ./clapi alias ./c
./c summary.shippingCosts
./c summary.shippingCosts[0]

# turn datatype display on
export CLAPI_SET_DATATYPE=ON
./c

# turn datatype display off
export CLAPI_SET_DATATYPE=OFF
./c
```

```
# find all matches by property name (deep search)
# using starts-with + case-insensitive matching
./cf isAssociate
./cf isassoc

# clapi-find aliases: ./cf, ./find, ./f
./cf email
./find email
./f email
```



```
L-SCLR1L3FD5-M:bin jhanink$ ./f email
 (search term) > email:
  - 1   string   email - tom@oldforest.bree
  - 2   string   buyer.email - tom@oldforest.bree
  - 3   string   payments[0].email - tom@oldforest.bree
```

```
# functions
./c -f=../samples/missingFulfillmentPrices.json
./c --FUNC=missingFulfillmentPrices
```

→ CLAPI : MOCKS MODE

- The CLI and UI return mock data when MOCKS mode is ON.
- The CLI will display a visual cue `========== clapi mocks mode ON`
- The Clapi UI response data will contain an extra property `MOCK-DATA:true`

```
# turn mocks mode on and run any command
export CLAPI_SET_MOCKS=ON
./get-customer
./c

# turn mocks mode off
export CLAPI_SET_MOCKS=OFF
```

→ ETC

```
# get torbit headers
./curl-torbit http://www-e6.walmart.com -i
```

→ TIPS & TRICKS

```
# find which nodes have values
./c -f=../samples/missingFulfillmentPrices.json && \
for var in $(seq 0 20); \
do \
  echo "--- $var" && \
  ./c test[$var].storefrontPricing.currentPrice.currentValue; \
done;

# find selected purchase contract shipping options
./c -f=../samples/purchaseContract.json;
for i in {0..3}; do ./c groups[0].shippingOptions[$i].selected; done;
```

Contribute

- Open a github issue to request a new feature

Related Documentation

- https://confluence.walmart.com/display/PGSCARTXO/Cart-Service-App+API
- https://confluence.walmart.com/display/PGSCARTXO/New+Checkout+Service+API
- http://mobile-qa-ci.homeoffice.wal-mart.com:8080/view/Item%20check/job/Inventory_check/
- See additional project goals