GetSandbox Mocks

Giving internal control, around [possibly shaky] vendor API's

Lewis Cowles. March 2022

What

Network-level mock services, that act as drop-in replacements for vendor API's

Why

Overcome vendor limitations, and keep clean services that are unaware of being mocked.

Where

Mocks deploy like most other micro services

Running locally

Local set-up

1. From `Dockerfile`, find the docker image name. It will look similar to `346121089040.dkr.ecr.eu-west-2.amazonaws.com/mock-sandbox:latest`.

2. Pull the docker image (login to AWS via the command-line `awslogin`, if you haven't already).

```
docker pull 346121089040.dkr.ecr.eu-west-2.amazonaws.com/mock-sandbox:latest
```

3. Go to the `checkout-mock` app root directory - the same directory as the `Dockerfile`.

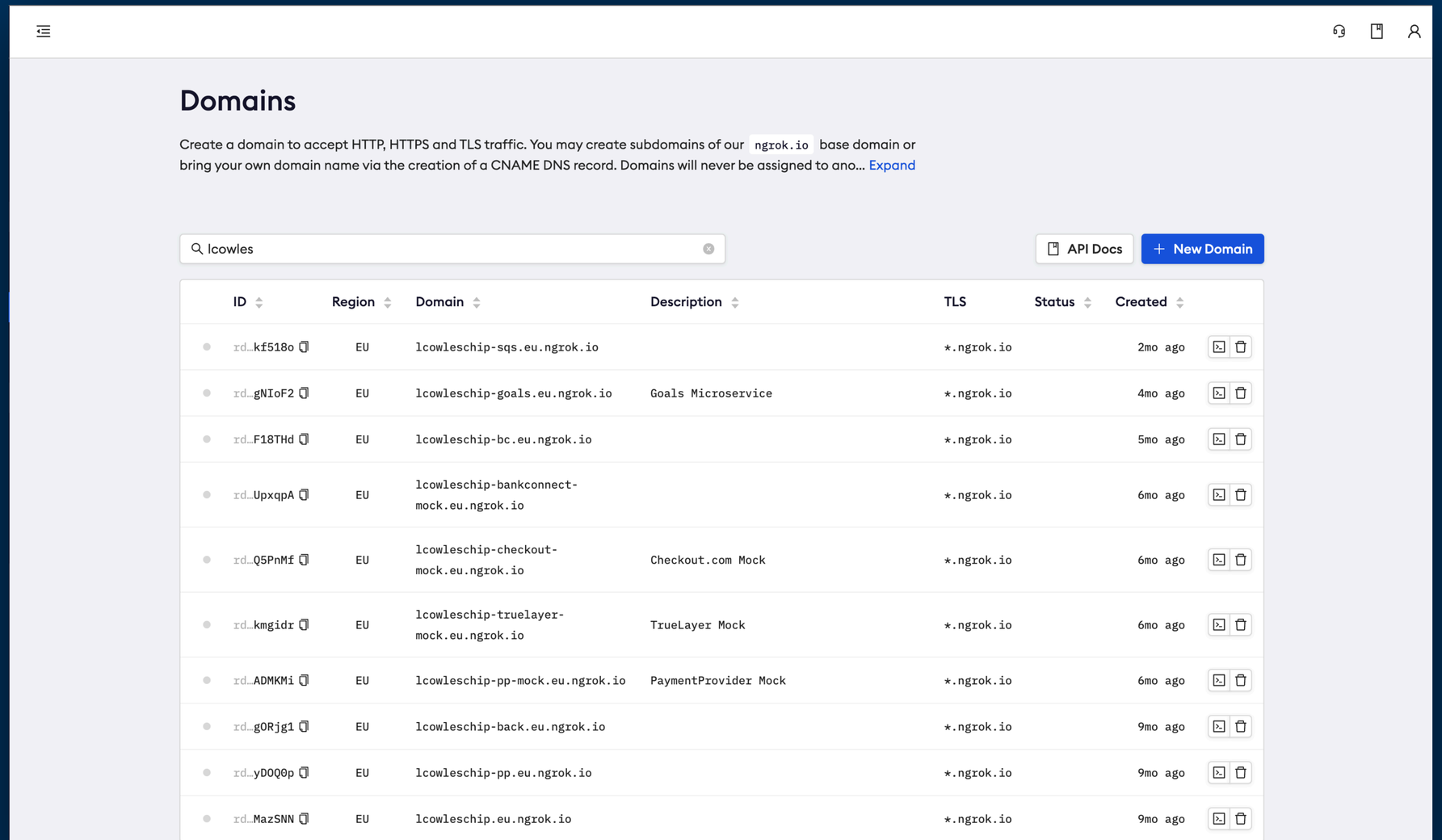
4. Run the docker container.

```
docker run -v $(pwd):/base -p 8081:80 -p 8082:90 -it  346121089040.dkr.ecr.eu-west-2.amazonaws.com/mock-sandbox:
```

Mock app is available at localhost:8081

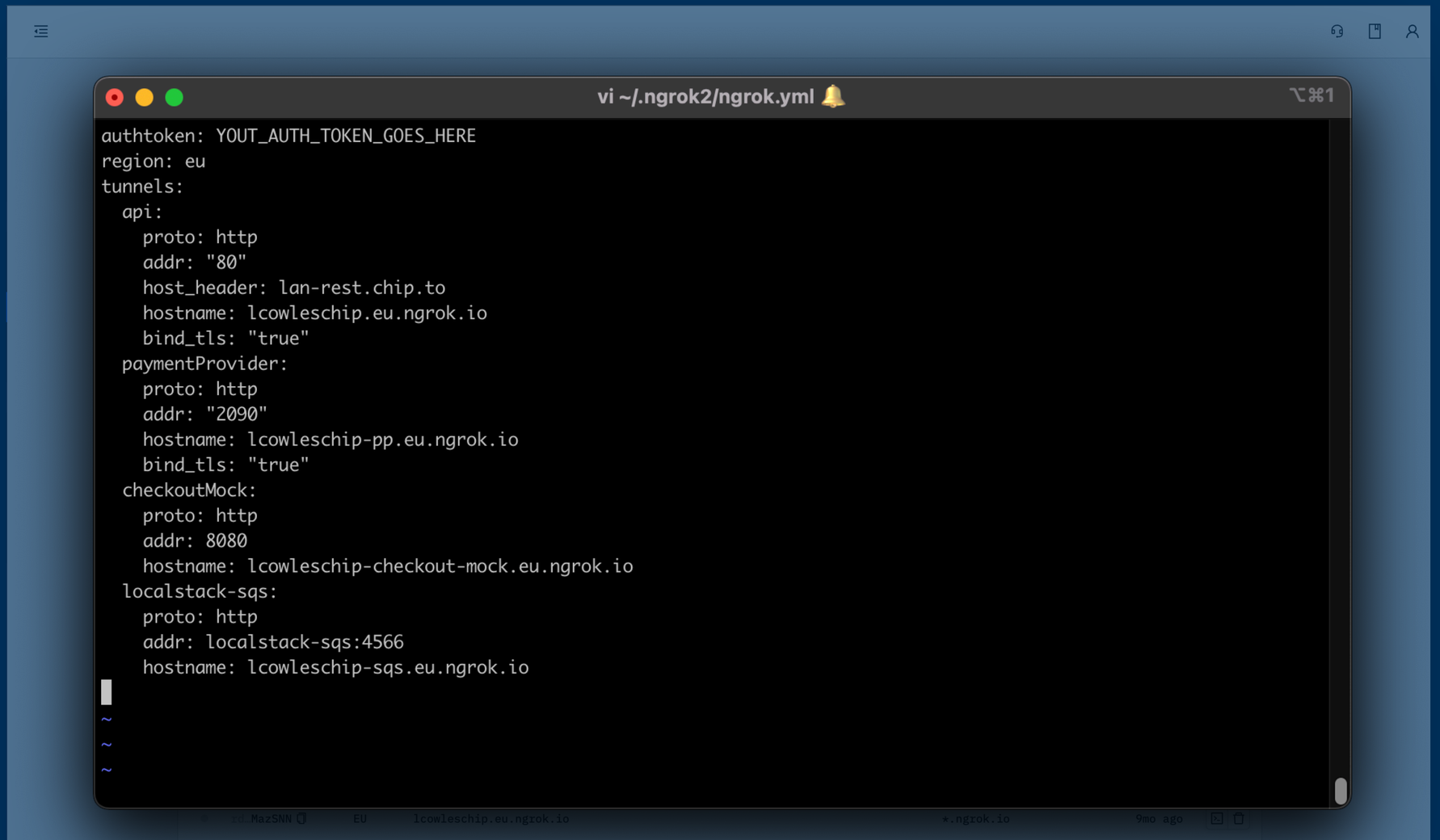
Inspecting web-traffic

[Documenting existing systems]



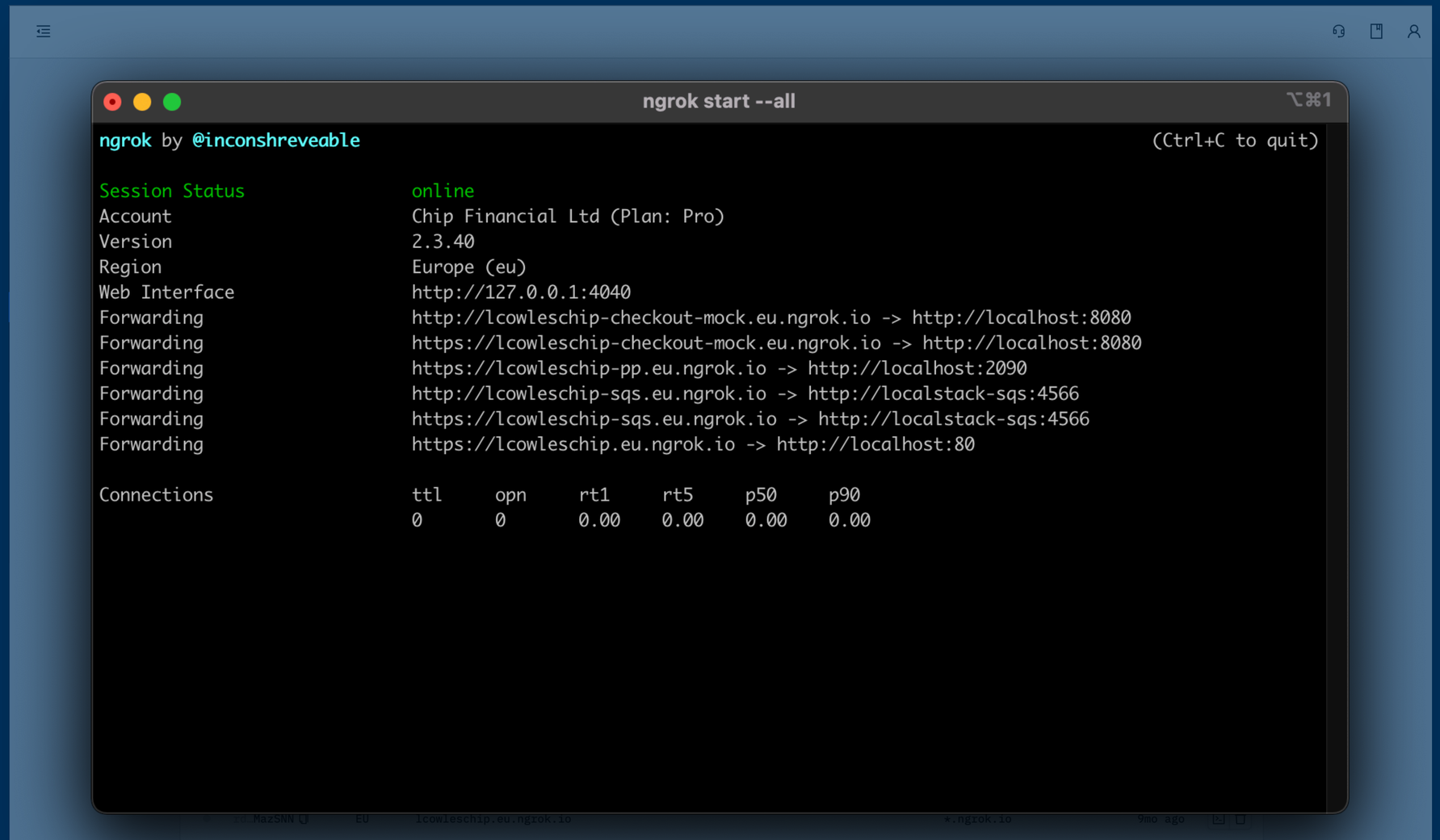
Inspecting web-traffic

[Documenting existing systems]



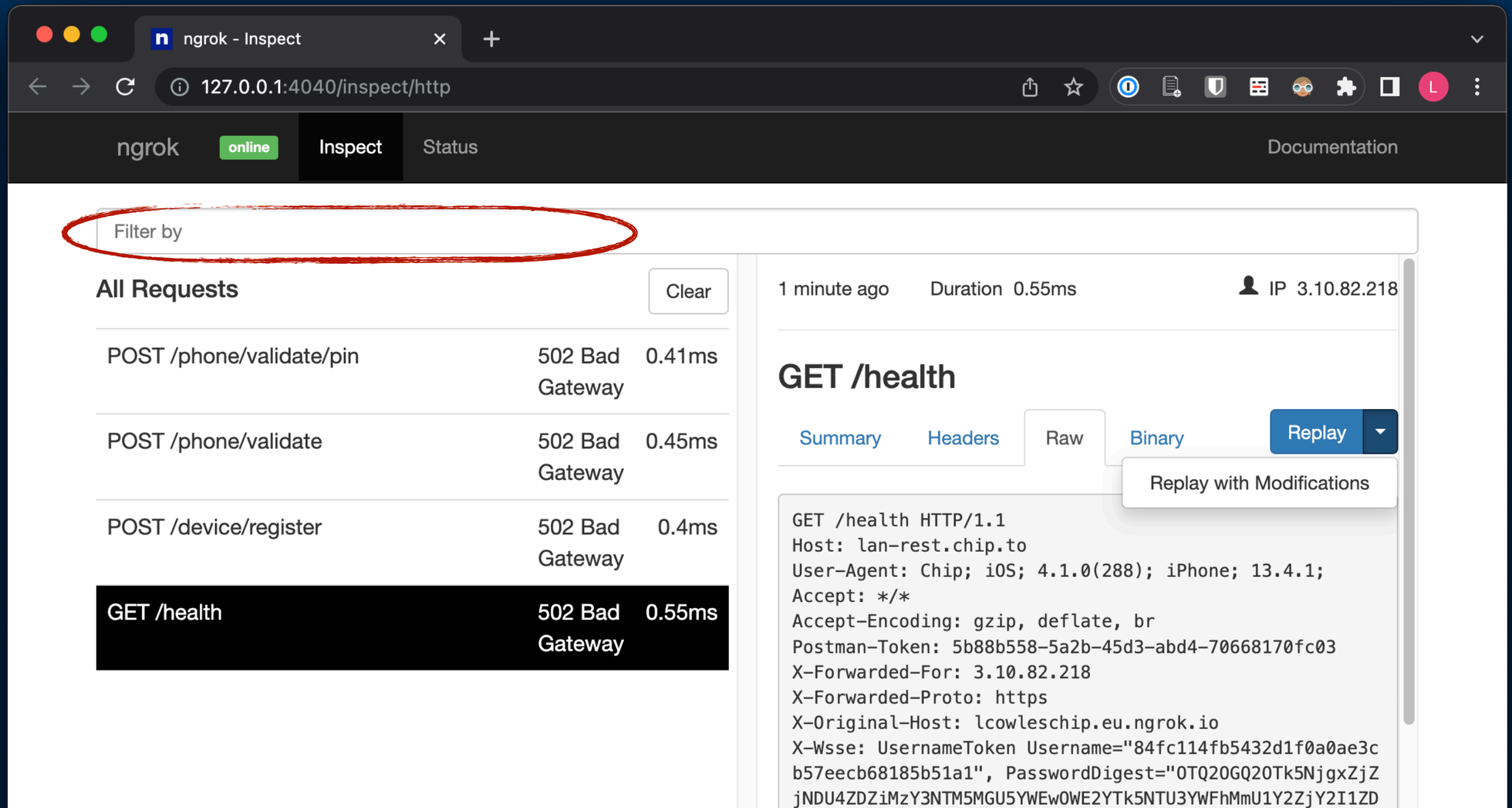
Inspecting web-traffic

[Documenting existing systems]



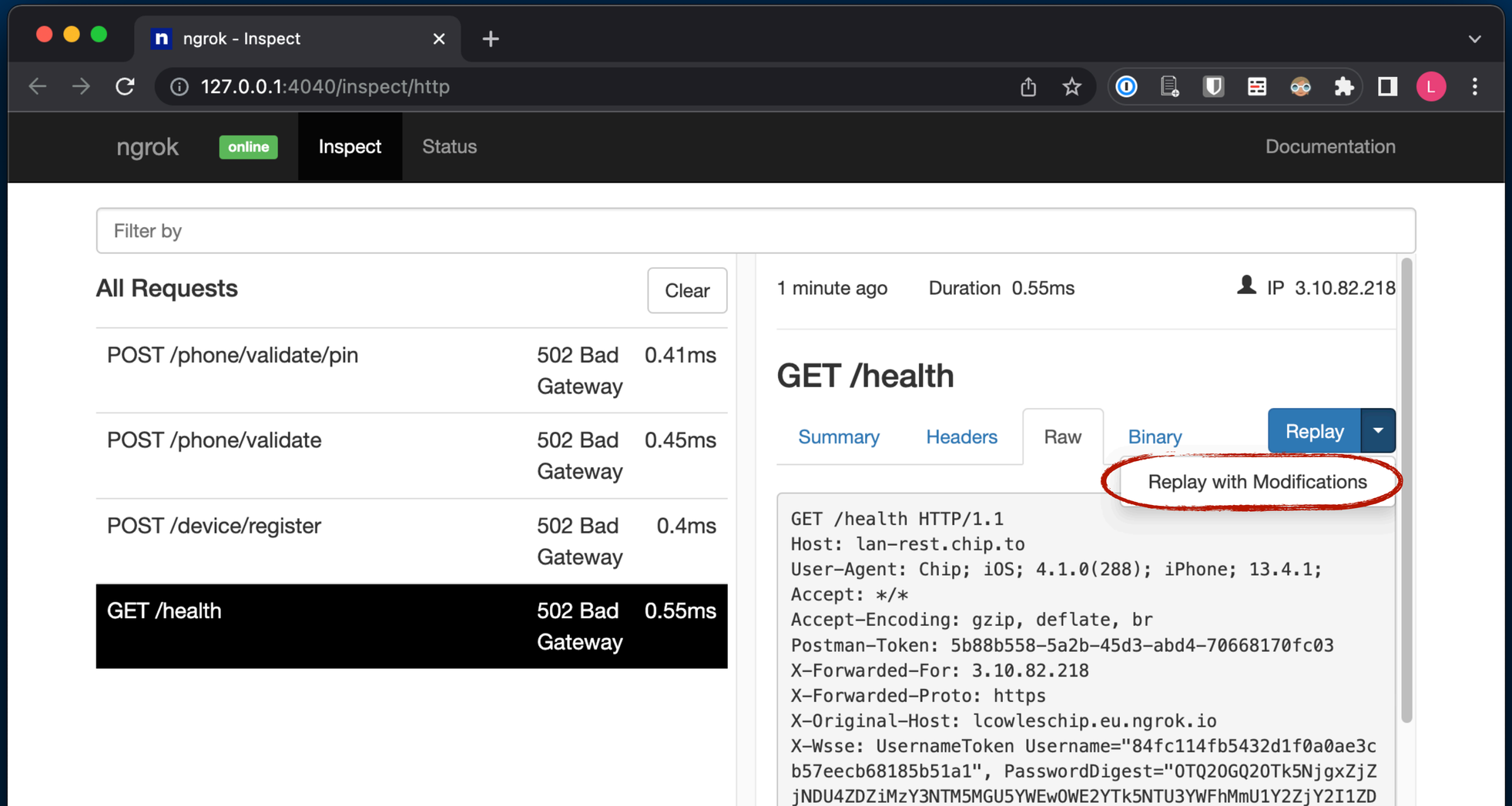
Inspecting web-traffic

[Documenting existing systems]



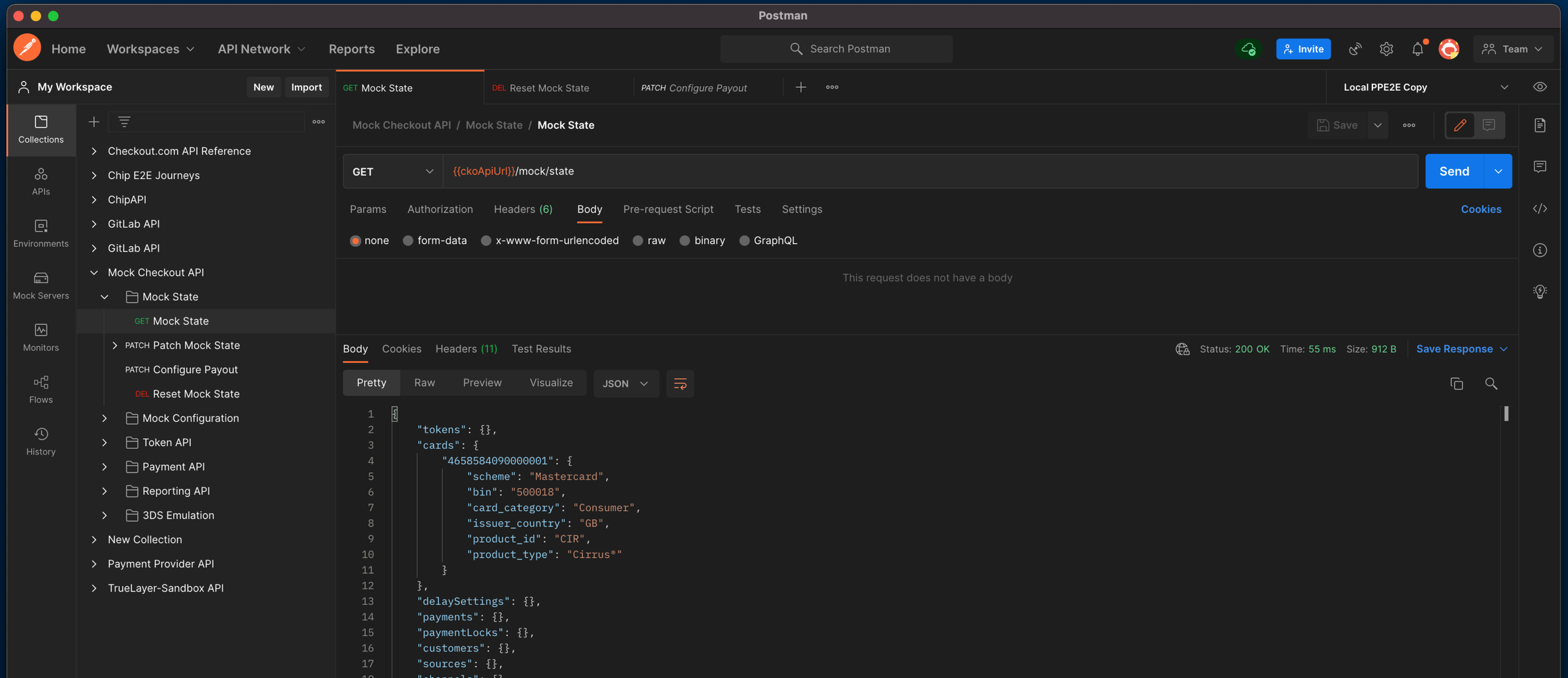
Inspecting web-traffic

[Documenting existing systems]



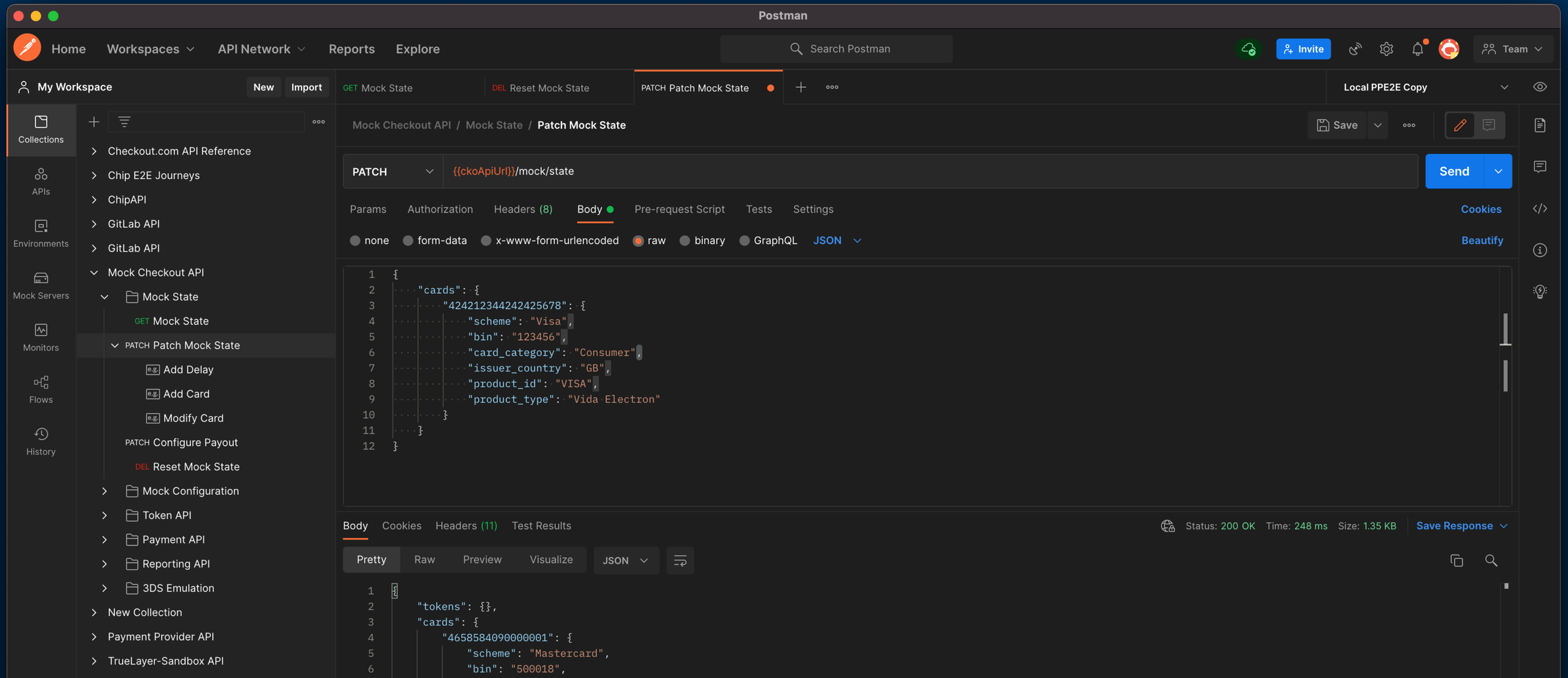
State

[How to persist data in a mock?]



State

[How to persist data in a mock?]



State

[How to persist data in a mock?]

```
exports.resetState = function() {
    state.tokens = {};
    state.cards = {
        "4658584090000001": defaultCardDetails
    };
    state.delaySettings = {};
    state.payments = {};
    state.paymentLocks = {};
    state.customers = {};
    state.sources = {};
    state.payout = {
        statement_id: '190110B107654',
        id: 'CKO_MOCK_PAYOUT',
        status: 'Remitted',
        currency_payout: 'GBP',
        carried_forward_amount: 0,
        current_period_amount: 0,
        net_amount: 0,
        payout_fee: '-5',
        channel: 'save'
    };
    state.channels = {};
};
exports.getChannel = function(req) {
    return state.channels[req.headers.Authorization] ?? DEFAULT_CHANNEL;
}
```

State

[How to persist data in a mock?]

```
exports.resetState = function() {
    state.tokens = {};
    state.cards = {
        "4658584090000001": defaultCardDetails
    };
    state.delaySettings = {};
    state.payments = {};
    state.paymentLocks = {};
    state.customers = {};
    state.sources = {};
    state.payout = {
        statement_id: '190110B107654',
        id: 'CKO_MOCK_PAYOUT',
        status: 'Remitted',
        currency_payout: 'GBP',
        carried_forward_amount: 0,
        current_period_amount: 0,
        net_amount: 0,
        payout_fee: '-5',
        channel: 'save'
    };
    state.channels = {};
};
exports.getChannel = function(req) {
    return state.channels[req.headers.Authorization] ?? DEFAULT_CHANNEL;
}
```

Timeouts

[Testing third-party without stubbing your code]

Common Code

```
const getDelaySettings = function() {
    return state.delaySettings || {};
}
exports.getDelaySettings = getDelaySettings;
exports.delayForRequest = function(req, res) {
    const reqSanitized = `${req.method} ${req.path}`;
    const delayTime = getDelaySettings()[reqSanitized] ?? 0;
    console.log(`REQUEST_DELAY ${reqSanitized}: ${delayTime}`, '');

    if (delayTime > 0) {
        res.setResponseDelay(delayTime);
    }
}
```

Timeouts

[Testing third-party without stubbing your code]

Common Code

```
const getDelaySettings = function() {
    return state.delaySettings || {};
}
exports.getDelaySettings = getDelaySettings;
exports.delayForRequest = function(req, res) {
    const reqSanitized = `${req.method} ${req.path}`;
    const delayTime = getDelaySettings()[reqSanitized] ?? 0;
    console.log(`REQUEST_DELAY ${reqSanitized}: ${delayTime}`, '');

    if (delayTime > 0) {
        res.setResponseDelay(delayTime);
    }
}
```

Timeouts

[Testing third-party without stubbing your code]

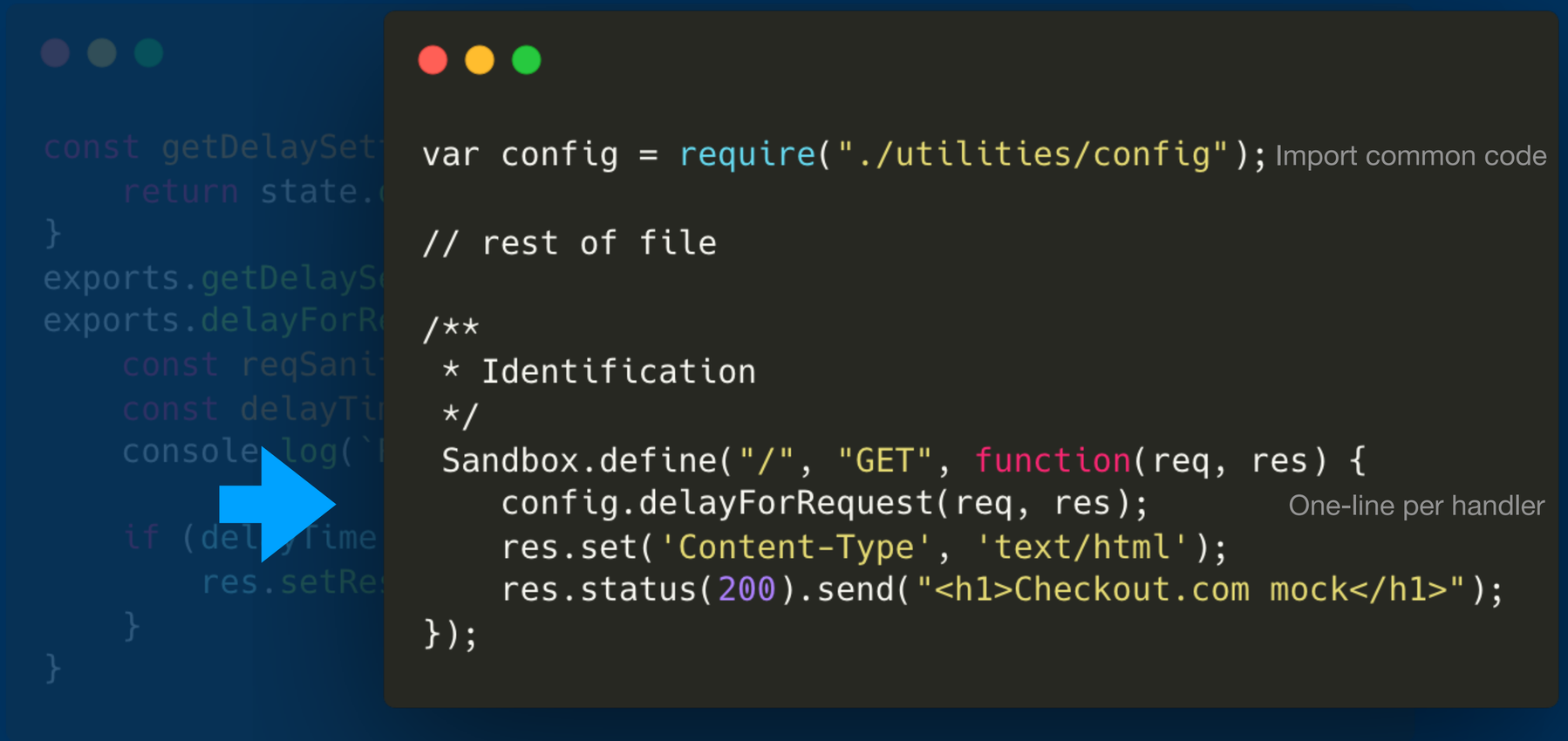
```
var config = require("./utilities/config");

// rest of file

/**
 * Identification
 */
 Sandbox.define("/", "GET", function(req, res) {
    config.delayForRequest(req, res);
    res.set('Content-Type', 'text/html');
    res.status(200).send("<h1>Checkout.com mock</h1>");
});
```

Import common code

One-line per handler

Timeouts

[Testing third-party without stubbing your code]

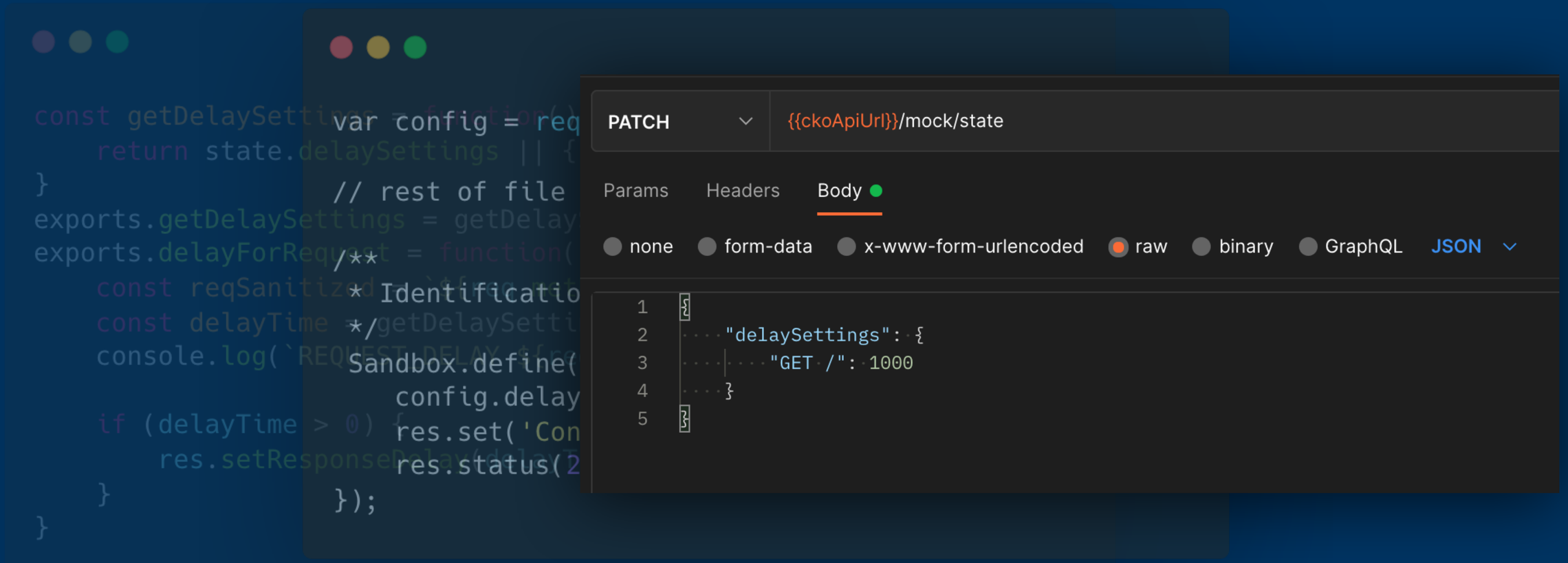


```
var config = require("./utilities/config");
```
Import common code

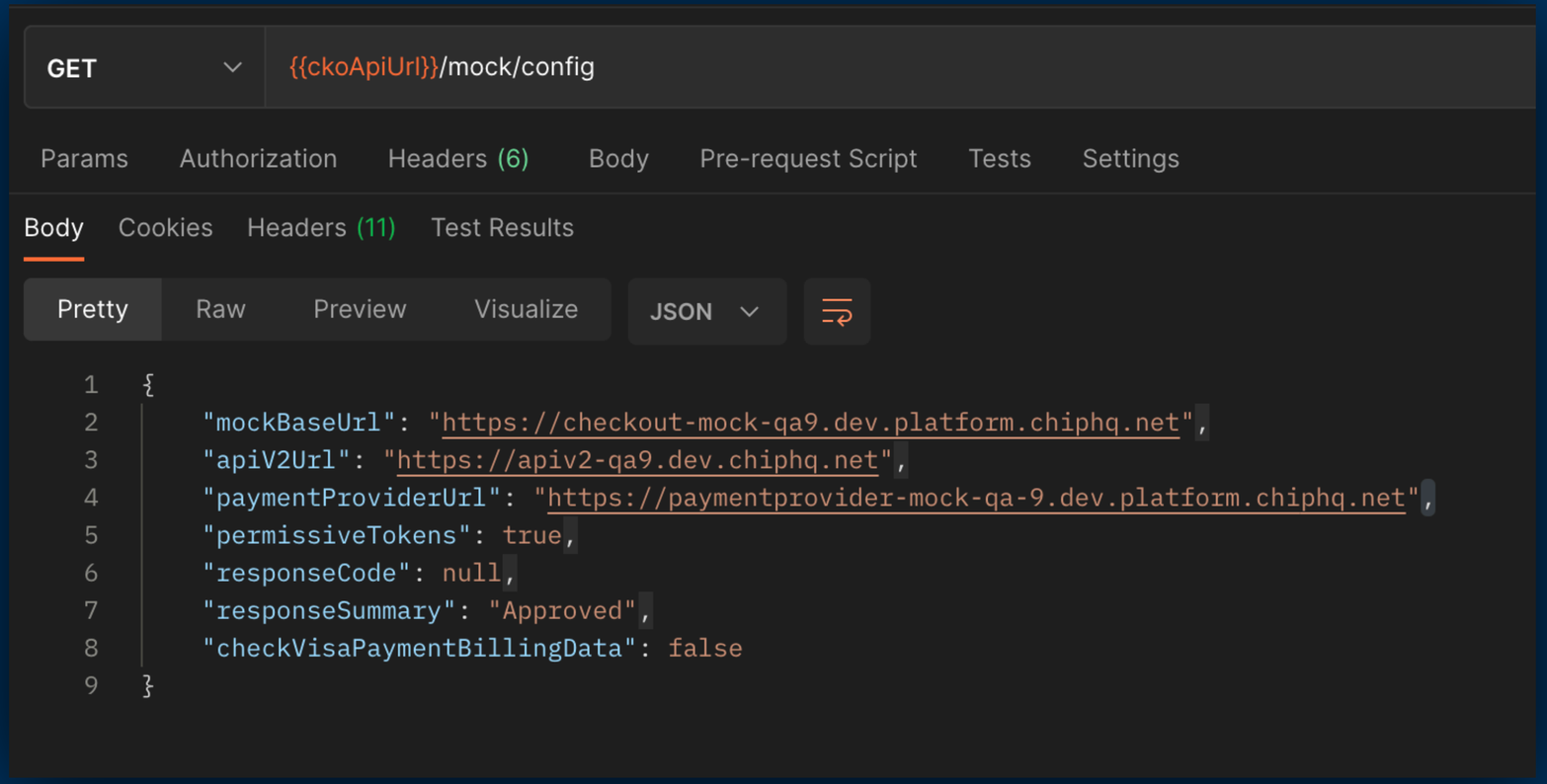
```
// rest of file

/**
 * Identification
 */
 Sandbox.define("/", "GET", function(req, res) {
    config.delayForRequest(req, res);
    res.set('Content-Type', 'text/html');
    res.status(200).send("<h1>Checkout.com mock</h1>");
});
```
One-line per handler

Timeouts

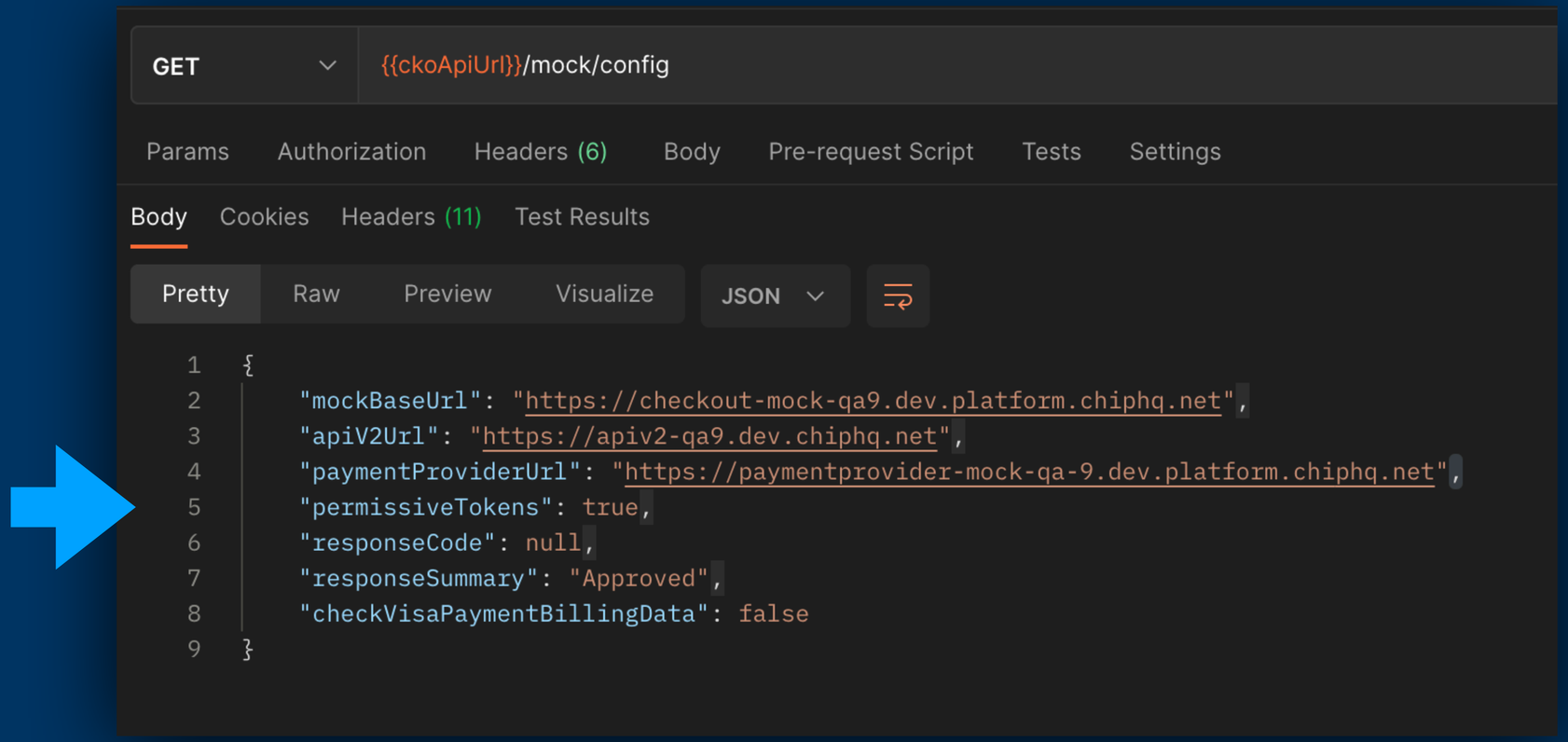
[Testing third-party without stubbing your code]



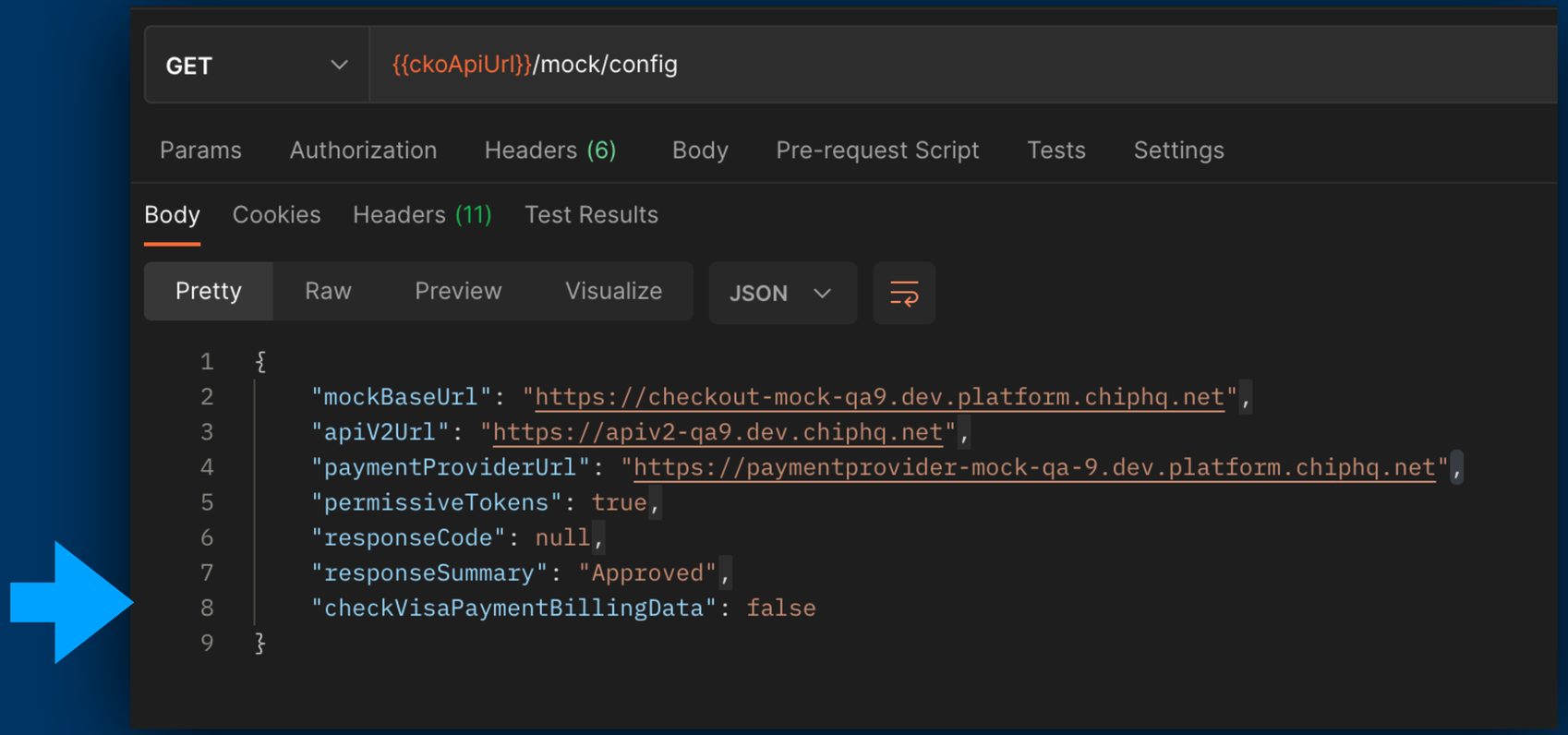
Adding controls



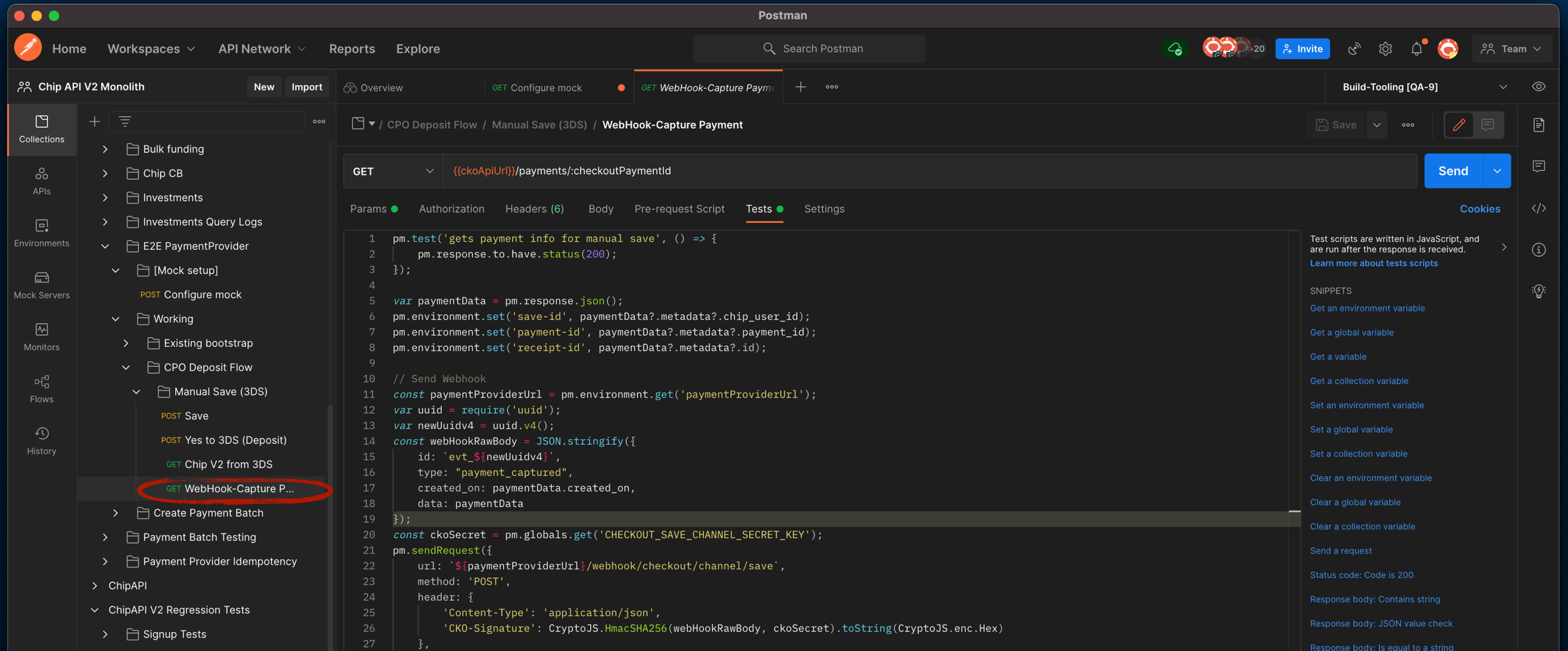
Adding controls



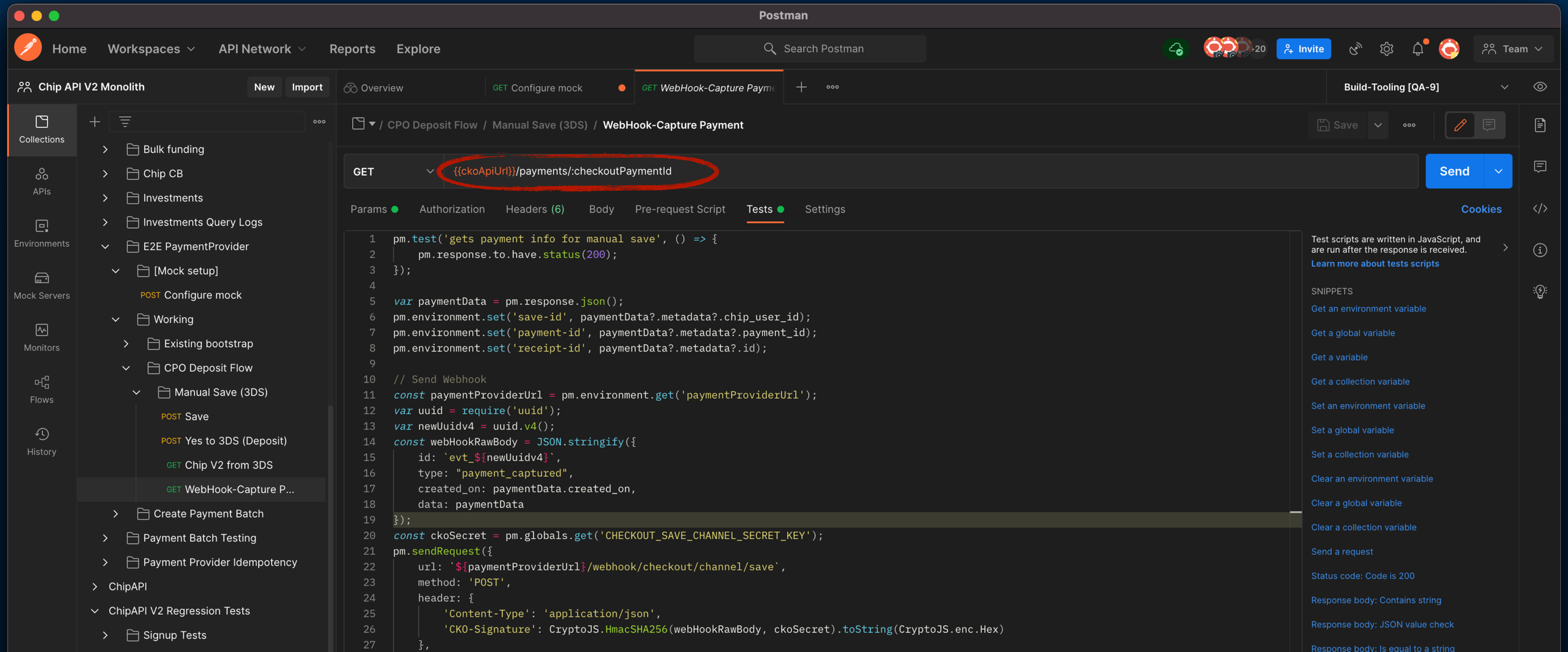
Adding controls



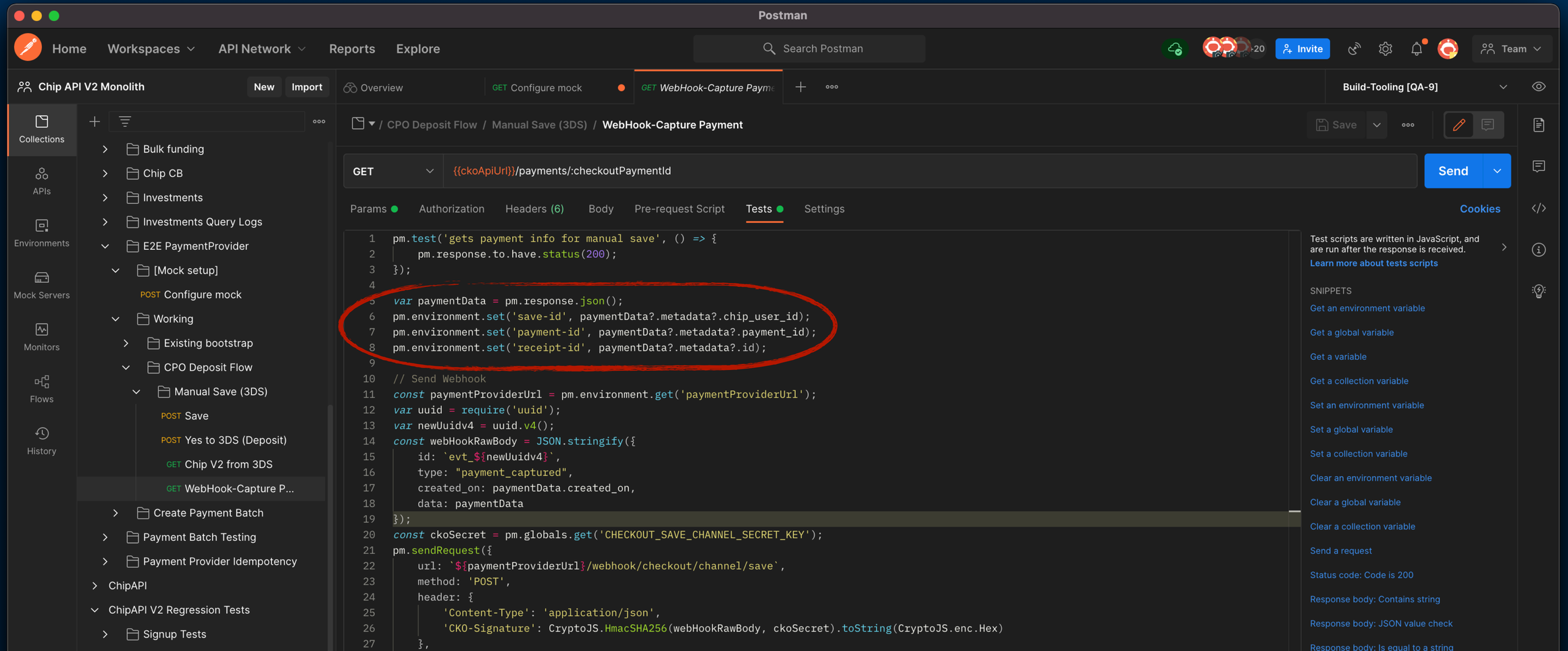
Coordinating web hooks [E2E,SSI]



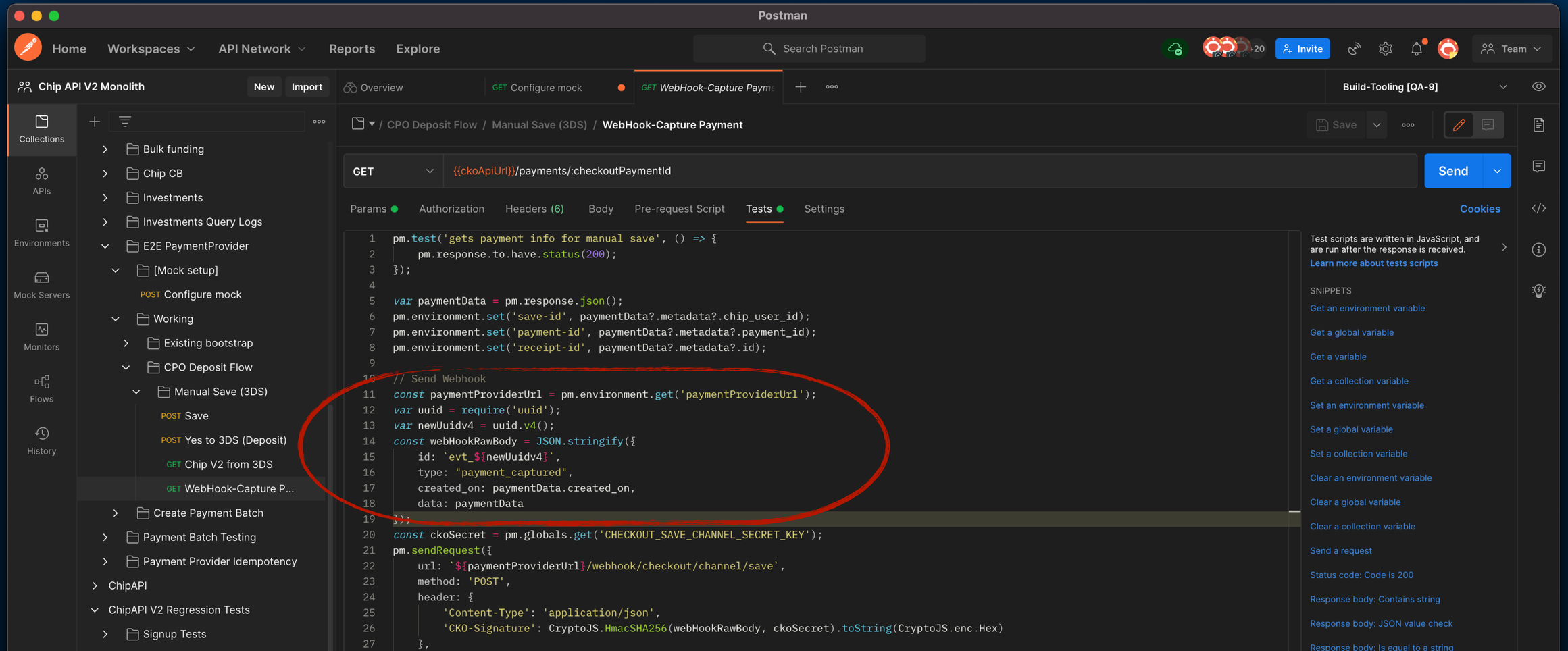
Coordinating web hooks [E2E,SSI]



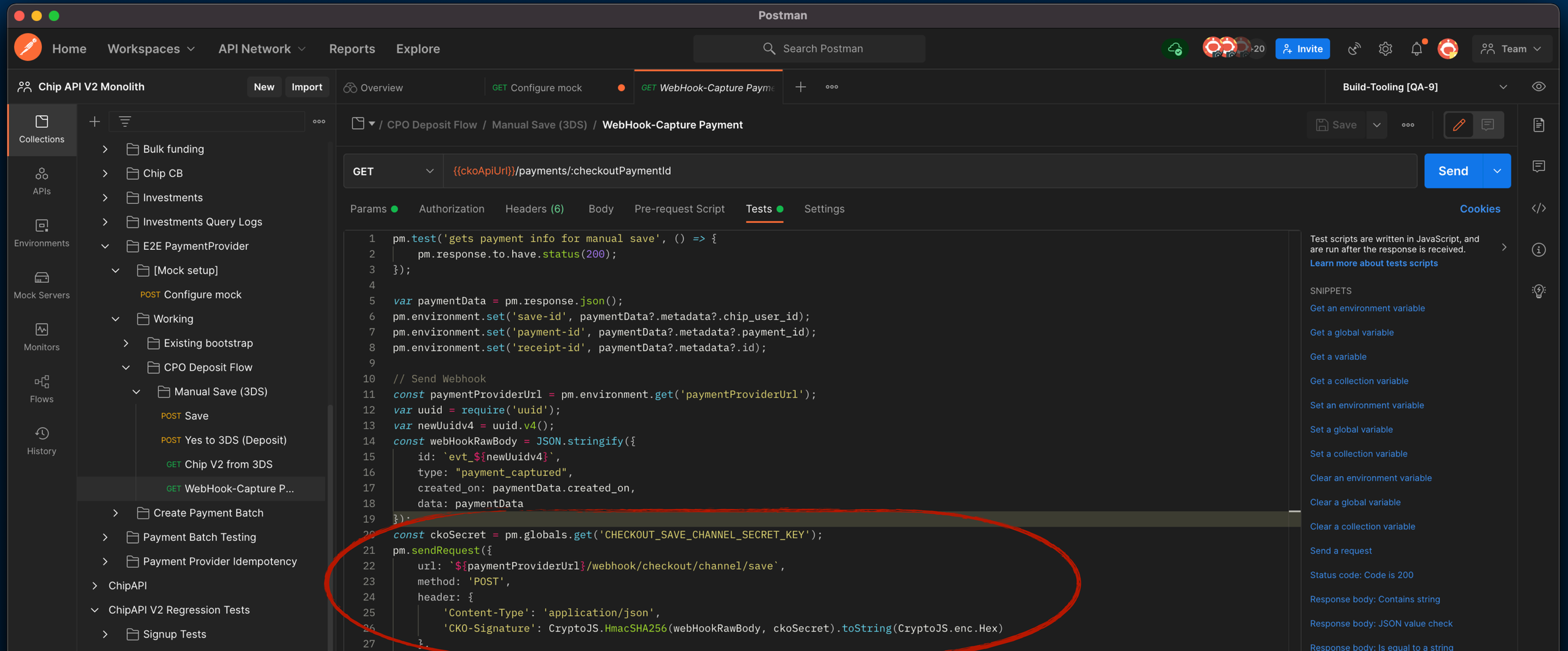
Coordinating web hooks [E2E,SSI]



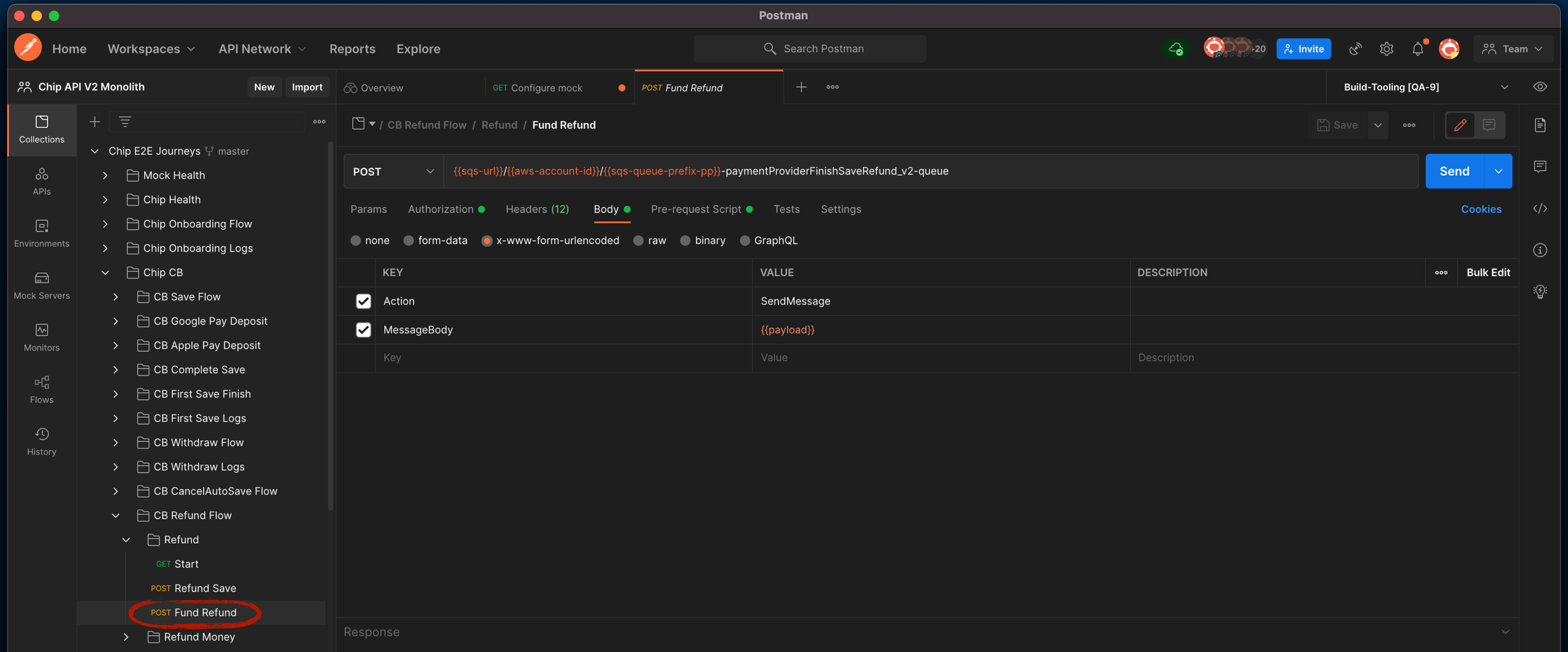
Coordinating web hooks [E2E,SSI]



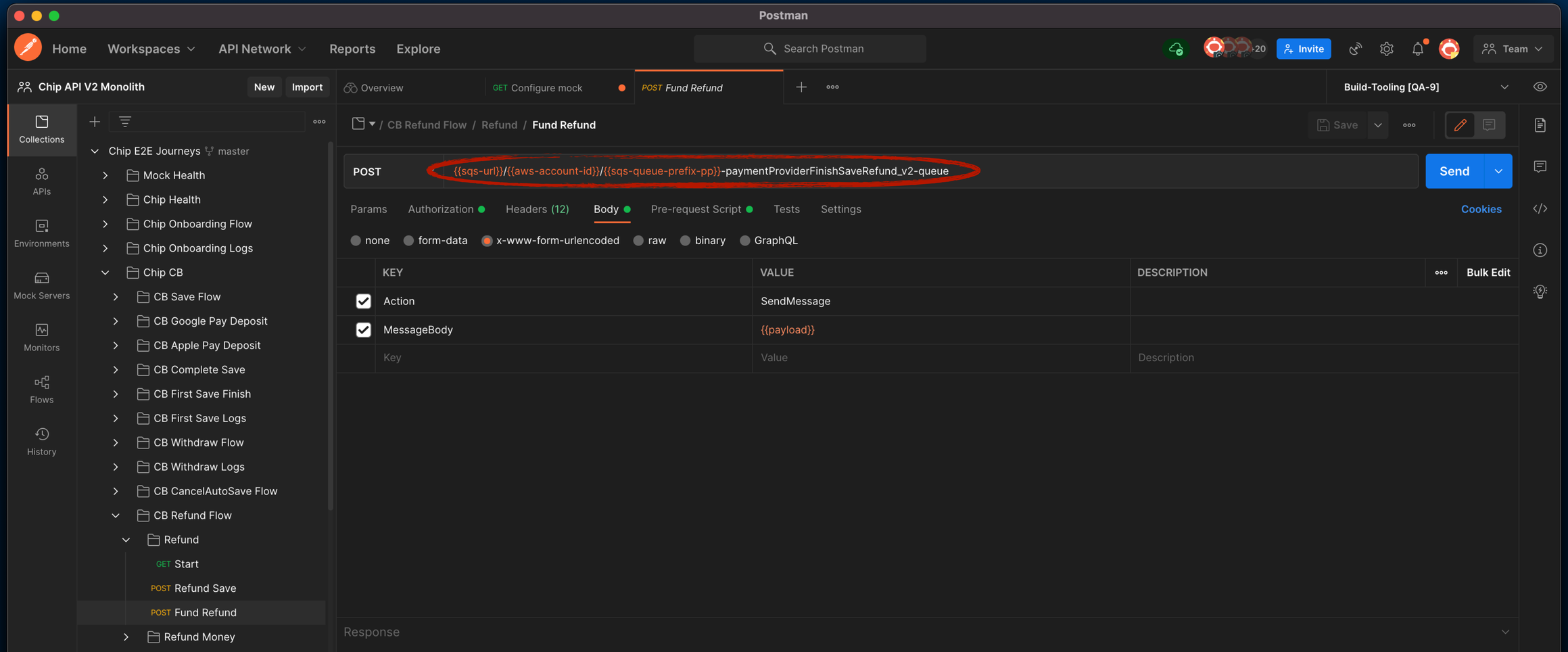
Coordinating web hooks [E2E,SSI]



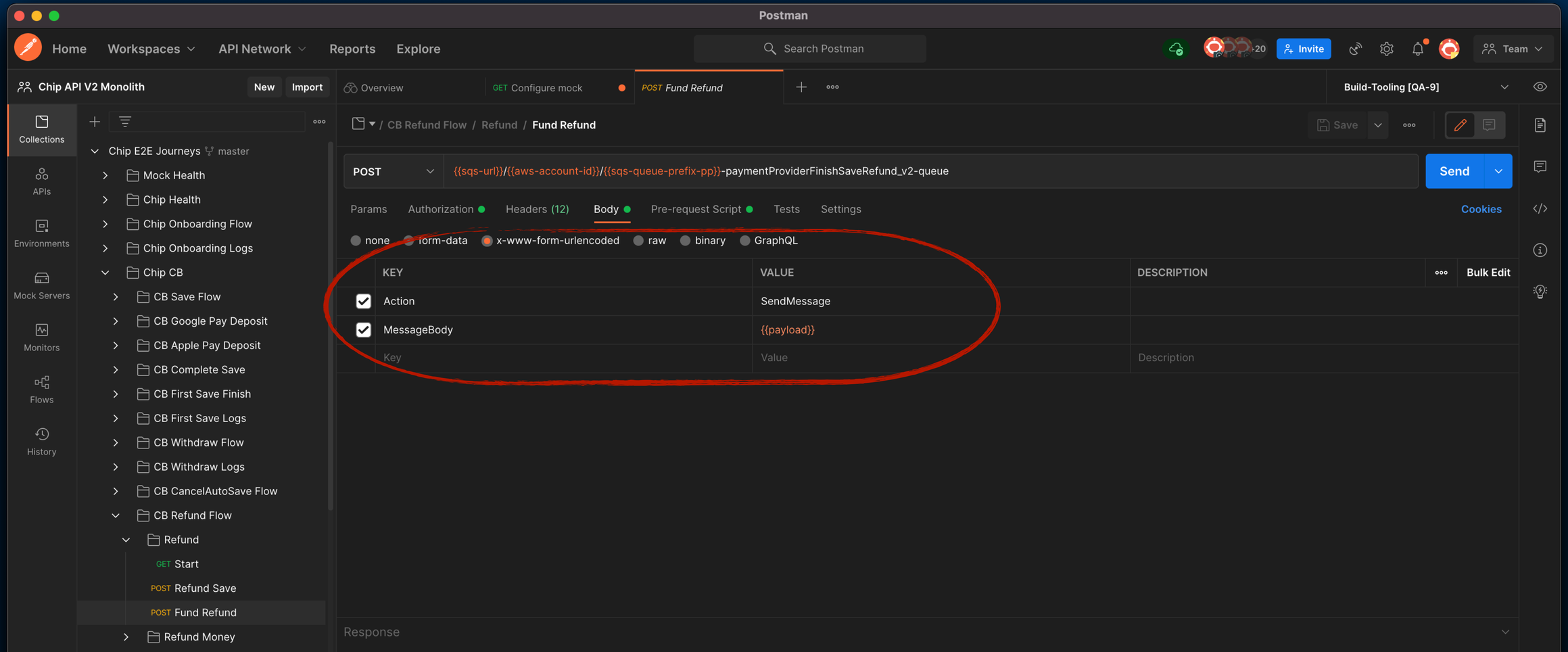
Managing SQS [E2E,SSI]



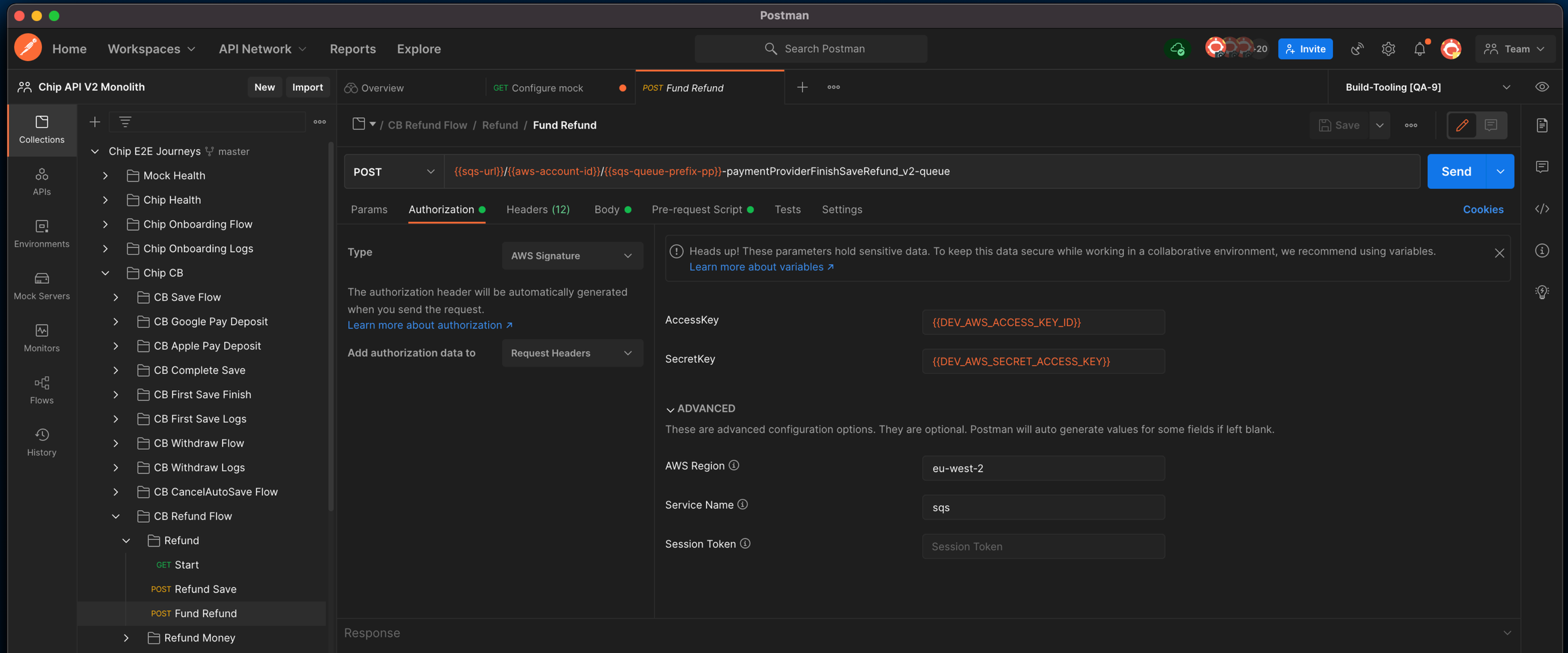
Managing SQS [E2E,SSI]



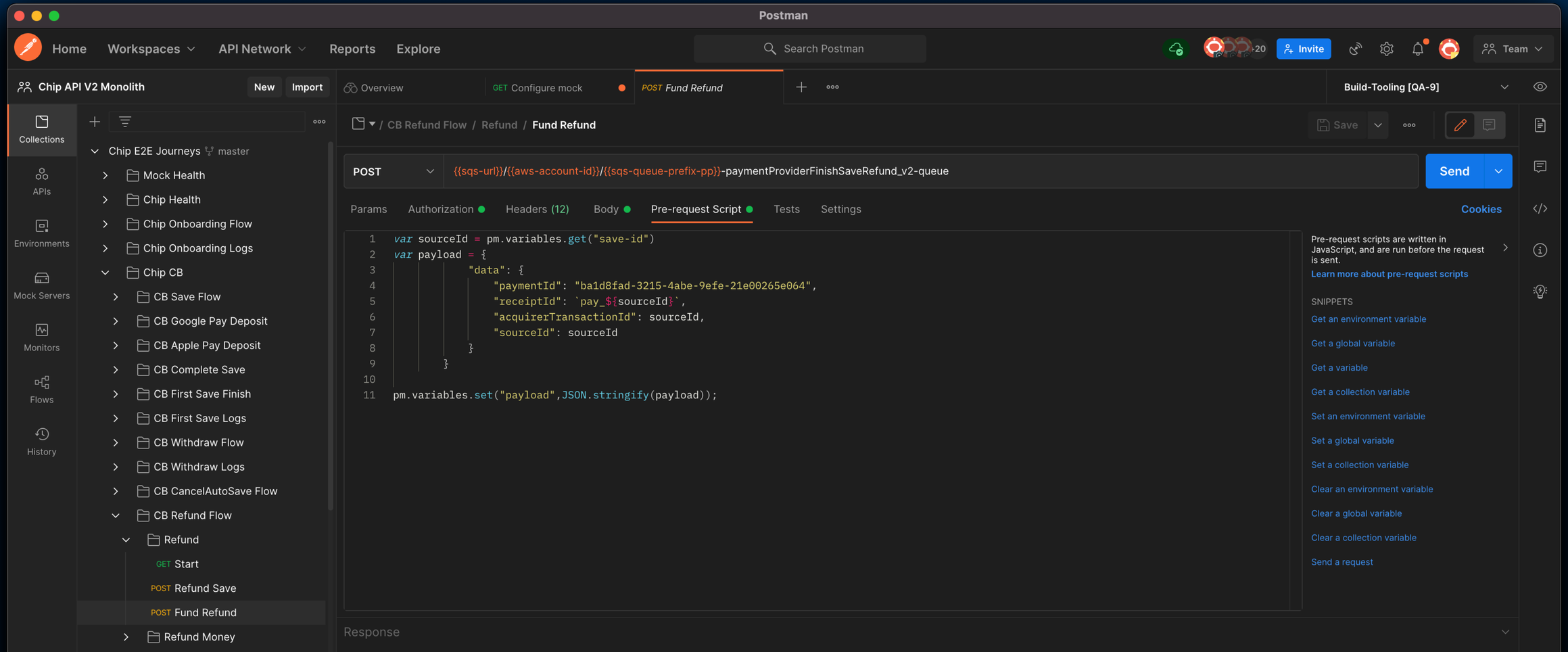
Managing SQS [E2E,SSI]



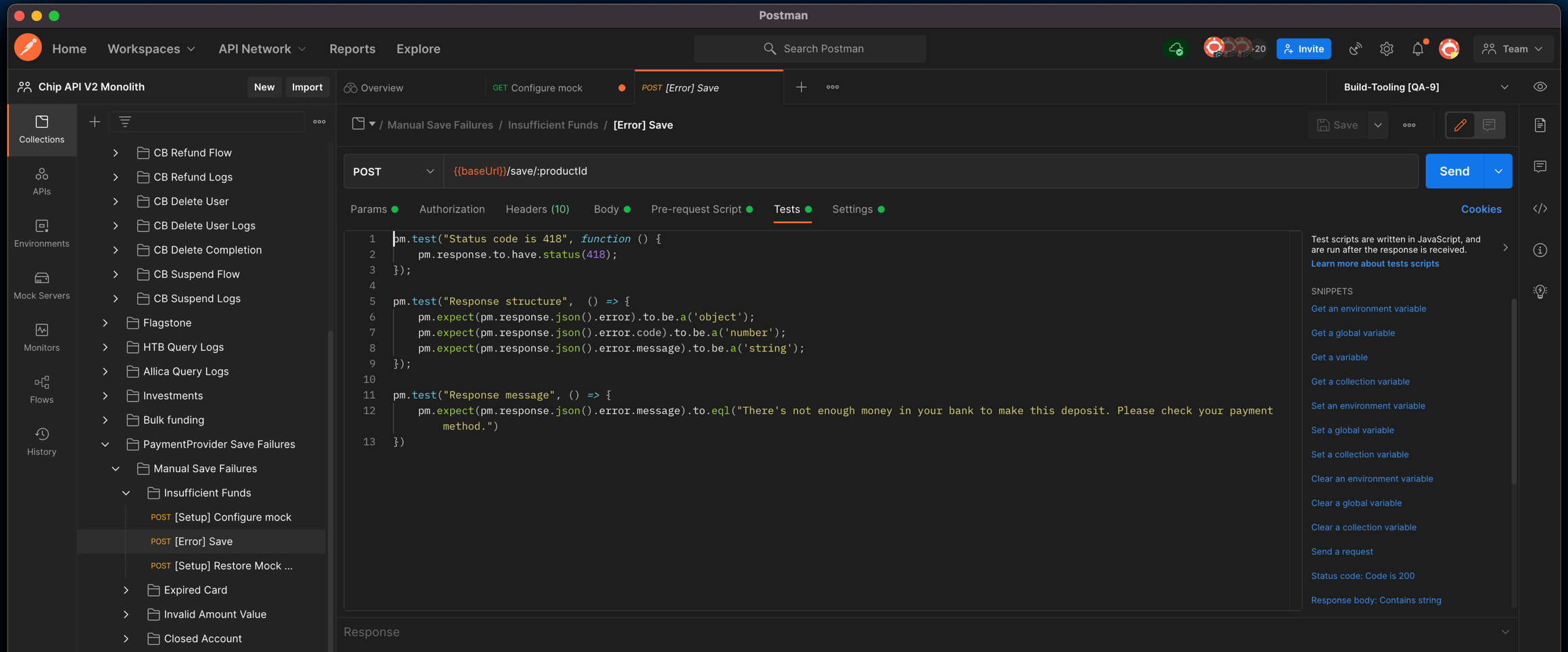
Managing SQS [E2E,SSI]



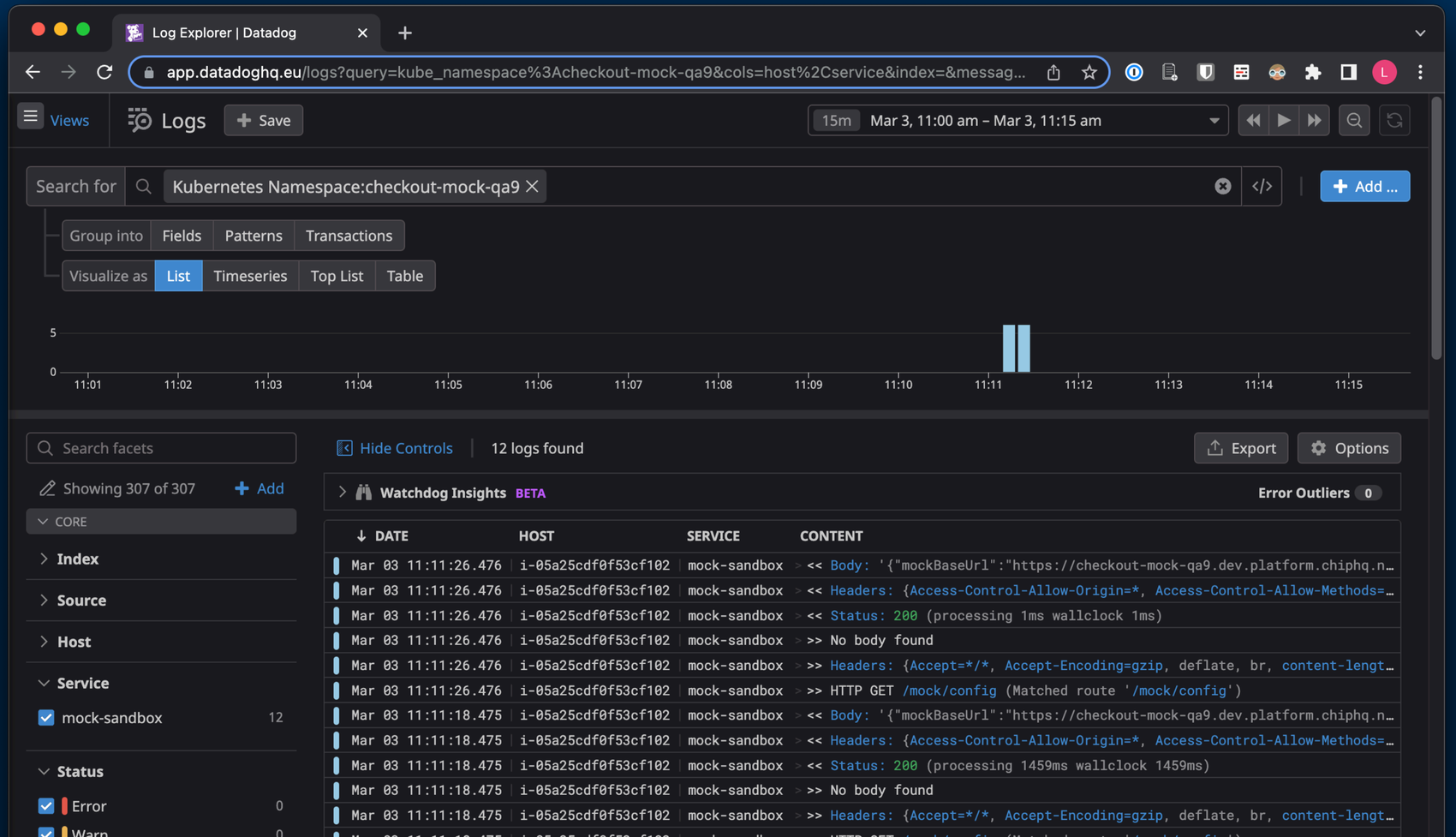
Managing SQS [E2E,SSI]



Coordinating sequences of interactions [E2E,SSI]



Monitoring / Logs



Future work

- Future without postman?
- Proxy in-front of mock for SQS / WebHooks?
- Templating
- Deployment
- Workshop on sandbox mocks?
- Ideas?

Questions?