Interacting with a Blockchain Service

Each Stratumn Service has its own subdomain with routes available to interact with it from the outside world. The routes are accessed using the **HTTP API**.

You can see the available routes by running:

```
$ stratumn-routes quickstart

> curl -X GET https://quickstart.stratumn.rocks/
>     Shows application information.
>
> curl -X POST https://quickstart.stratumn.rocks/maps -d '["title"]'
>     Executes function Agent#init(title) to create a new chain.
>
> curl -X GET https://quickstart.stratumn.rocks/links/:hash
>     Shows the link and evidence for the given hash.
>
> curl -X POST https://quickstart.stratumn.rocks/links/:hash/addMessage -d '["conte
nt","author"]'
>     Executes function Agent#addMessage(content, author) on the link with given li
nkhash.
```

HTTP API

The HTTP API allows interacting between the Client and the Service. It lets you:

- Create a new Chain Map or Map with only the root link in it
- Append a new Link to an existing Link in a Map. To do this: you call an agent method with the LinkHash of the existing Link you want the new Link to be appended to
- Fetch a Link in a Map
- Fetch the list of Maps under a Blockchain App
- Fetch the list of Links in a Map where only the meta being returned for each Link

For the first two, we would call the Agent Methods while for the rest, just a simple HTTP GET call to the Map would give us the results.

Using the HTTP API, when you call an Agent method to create a new Link, you would have to mention which Link in the Chain Map you would like to append this new Link to. A Link is created if and if only there has been a change in State from that of the previous Link which you wanted to append as there can never be a situation where consecutive Links have the same state.

Every time a Link is created, its cryptographic hash is timestamped on one or multiple blockchains, such as the Bitcoin Blockchain. Once created, a Link is immutable.

Create a new Chain Map

Let's execute the route to create a new chain map:

```
$ stratumn-routes quickstart -xp create-map "A conversation timestamped on the bloc
kchain"
```

> ☑ As you may have guessed, by calling the route, we passed the title to `init()` which created the first link of a Chain Map.
>
> The link is currently awaiting to be timestamped on the blockchain, as reflected by the value `.meta.evidence.state` .

It should respond with JSON content representing the Chainscript Segment that this Link belongs to. It will be similar to this:

JSON ()

```
{
"link": {
  "state": {
    "fileToNotarize": "test.PDF"
  },
  "meta": {
    "title" : "File Info for Notarizing",
    "tags" : ["notarize", "filename", "more_list", "of", "tags"],
    "priority": "0"
    "updatedAt": 1455532426303,
    "mapId": "56c11d0ea55773bc6e681fde",
    "agentHash": "f559625364c117e9bf07f9aa536cf72b1f1468e483b16f0b45e10e13faf8c8d9",
    "stateHash": "06faad7bf3f793c2ea5d89b0fbe906603f4ee15684e24350ff88a8d9ef7761d5",
    "prevLinkHash": "9b03d05c07fb44fe5aa472115cd918e886b71e0dbc6fe06ef976566c98272ee9",
    "action": "addFileToNotarize",
    "args": ["test.PDF"],

  }
},
"meta": {
  "linkHash": "fa871e81c469fa947eacd40f89dc5627a0cb3a96551a651c034787c752d4448e",
  "evidence": {
    "state": "COMPLETE",
    "merkleRoot": "855e019f67bc5ba09b7efb3a9e169bbe14e6abafa5b02d686607e25107db1b4c",
    merklePath: [
      {
        left: "fa871e81c469fa947eacd40f89dc5627a0cb3a96551a651c034787c752d4448e",
        right: "aa1331b6f1155fff5865116e5adcb6e4cdf0435fc2bd68053a4d24795f47dad3",
        parent: "39b43545134a903dd16be3d0b5391474ef82241acd33e8496b103ee8d378efea"
      },
      {
        left: "39b43545134a903dd16be3d0b5391474ef82241acd33e8496b103ee8d378efea",
        right: "9fb3164966231fca20c75a562d0b4c1e7adc68738ac6c179dd1ecf921845dbfd",
        parent: "fedacd347e84d39ab3122f78d762dbbc79646704deb90fbb118b140c85f4df96"
      },
      {
        left: "d7fa0ed22e4b58eb227a826a1d2f0ef564f5052172bd36fe612a2fa214d452c8",
        right: "fedacd347e84d39ab3122f78d762dbbc79646704deb90fbb118b140c85f4df96",
        parent: "7e7dec576382e948f4980e4295f6ef9b89748e4586723a7b9481ee20cfb14db5"
      },
      {
        left: "2c0ada2bdea45977acd0a4d56fda6f970f34df32a1e507e99cd7284c0038b22a",
        right: "7e7dec576382e948f4980e4295f6ef9b89748e4586723a7b9481ee20cfb14db5",
        parent: "fe7214757ed9d701a2ed112d01c1b17a734094444ee067dd2ee4c14907419a24"
      },
      {
        left: "dae80a5aa2b397285ce6ca079c78dad737099d293a8783d4b8680efff57e921e",
        right: "fe7214757ed9d701a2ed112d01c1b17a734094444ee067dd2ee4c14907419a24",
        parent: "5f9343488c91f22a0cfea5c9797e244c1a6d20872048e157e515a1172678ebcf"
      }
    ],
```

```
      "transactions": {
        "bitcoin:testnet": "15359dc0ef4c430d6219d07e0dd7be96442af20ffd6f79727559028a0651847
4"
      }
    },
    xStratumn: {
      totalTimeMs: 16.941432,
      loadApplicationCache: "miss",
      loadApplicationTimeMs: 5.387555,
      loadInputLinkCache: "miss",
      loadInputLinkTimeMs: 10.388065
    }
  }
}
```


> ❶ **Anatomy of the above Chainscript JSON**
>
> `link` is the actual link
>
> `link.state` is the user defined state which captures a specific step
>
> `link.meta` is the meta data of the link
>
> `link.meta.title` is a user defined title for the state
>
> `link.meta.tags` is used to "color" chains so that they can be indexed in a multidimensional graph
>
> `link.meta.priority` is used to "order" links in a chain so that they can be sorted in a multidimensional graph
>
> `link.meta.mapId` is the chain ID common to all the links in the chain
>
> `link.meta.agentHash` is the hash of the agent
>
> `link.meta.stateHash` is the hash of the state
>
> `link.meta.prevLinkHash` is the hash of the previous link, if any
>
> `link.meta.action` is the name of the method that was called to append this link
>
> `link.meta.args` is the arguments that were passed to the method, if any
>
> meta is the meta data of the Chainscript
>
> `meta.linkHash` is the hash of the link
>
> `meta.evidence` is information that shows the link hash was inserted in one or multiple blockchains
>
> `meta.evidence.state` is the state of the proof, either QUEUED or COMPLETE
>
> `meta.evidence.merkleRoot` is the root of the Merkle tree which is inserted in one or multiple blockchains
>
> `meta.evidence.merklePath` is the path to go from the link hash to the root in the Merkle tree
>
> `meta.evidence.merklePath[].left` is the left hash of the Merkle node

> `meta.evidence.merklePath[].right` is the right hash of the Merkle node
>
> `meta.evidence.merklePath[].parent` is the parent hash of the Merkle node
>
> `meta.evidence.transactions` is the transactions that were created in one or multiple blockchains
>
> `meta.evidence.transactions.[blockchain]:[network]` is a transaction ID or hash, in this example it is [bitcoin]:[testnet]

Fetch a Link

The link is currently awaiting to be timestamped on the blockchain, as reflected by the value `.meta.evidence.state` .

You can fetch a link at anytime by running:

```
$ stratumn-routes -xp quickstart show-link fa871e81c469fa947eacd40f89dc5627a0cb3a96
551a651c034787c752d4448e
```

> Replace `fa871e81c469fa947eacd40f89dc5627a0cb3a96551a651c034787c752d4448e` with the value you have in `link.meta.linkHash` .

Within ten minutes, the evidence will be inserted, and fetching the link will show the evidence.

Append a Link to the Chain Map

Let's add a message and append a link to the chain. Simply run:

```
$ stratumn-routes -xp quickstart create-link fa871e81c469fa947eacd40f89dc5627a0cb3a
96551a651c034787c752d4448e addMessage "Hello there" "Stephan"
```

You should get a response like this:

JSON ()

```
{
  "link": {
    "state": {
      "title": "A conversation timestamped on the blockchain",
      "createdAt": 1455906044106,
      "messages": [
        {
          "content": "Hello there",
          "author": "Stephan",
          "createdAt": 1455906146633
        }
      ]
    },
    "meta": {
      "mapId": "56c75cf7a10fe1627fd0fa13",
      "agentHash": "1bee388c6ca7ae8984cb78caeeb77d63fb4af46287c96433e5d6a2f17f567991",
      "stateHash": "eecacfc780bba8a7aa2f73d21ddd92cb12888d81b849657ee5528808738e1000",
      "prevLinkHash": "fa871e81c469fa947eacd40f89dc5627a0cb3a96551a651c034787c752d4448e"
    }
  },
  "meta": {
    "linkHash": "616f7dbf832d46aa68df034e6752c21ae218819335181d597e1dc03697d14585",
    "xStratumn": {
      "agent": {
        "name": "/silly_bardeen",
        "responseTimeMs": 19.492376999999998
      },
      "totalTimeMs": 27.051294
    },
    "evidence": {
      "state": "QUEUED"
    }
  }
}
```


We just added a message and created a new link by calling `addMessage`. As you can see it has the hash of the previous link, so you can trace the whole history of the chain map.