REST APIs

Introduction

- Sawtooth provides a REST API for clients to interact with a validator using common HTTP/JSON standards
- It is RESTful API, which is an application program interface that uses HTTP requests to work on data
- REST : Representational State Transfer Technology
- In General API(Application Program Interface), is a code that allows 2 software programs to communicate with each other

Introduction

- Four basic commands of REST API are GET,PUT,POST & DELETE
- GET : Retrieve a resource
- PUT : Change the state of a resource
- POST : To create that resource
- DELETE : To remove that resource
- The resource can be object,file or even a block

Basic Properties

- Sawtooth provides a REST API for interacting with the validator
- Intended as a simple interface for client use
- REST API’s are stateless in nature

  (Retains nothing during executions)
- Requests are generally done on HTTP format and responses are done in JSON format

JSON

- JavaScript Object Notation
- It is a data interchange format
- Merits : Easy for humans to read and write

  Easy for machines to pause & generate
- It is language independent but use conventions of c,c++,java etc…

  (Ideal for data interchange languages)
- Outputs are obtained as key value pairs(set of linked items)

Properties(Continues)

- Executions by a REST API is an entirely different process
- It is not a part of the validator and hence, which once running allows transactions to be submitted and blocks to be read, with a common language neutral interface
- As validator is redesigned and improved, The REST API will grow with it, providing a consistent interface that meets the need of application developers in the future.

Properties(Continues)

- The validator acts like a black box
- The REST API just submits requests & fetch responses
- It is not used for validator communications.(It is not used by TP to communicate with a validator or by one validator to talk to other)
- We have efficient mechanisms for this type of communications
- ZMQ/PROTOBUF interface : ZeroMQ is used for Decentralised messaging & Communications and Protobuf for serialization

Properties(Continues)

- It offers no inbuilt authorization mechanism

  (Can be provided using third parties like servers,proxy servers etc..)
- REST API is comprehensively documented using the OPEN API specification, formatted as a YAML file
- YAML : YAML Ain't Markup Language

  (Human readable data serialization language and generally uses for configuration files)

Query Parameters

- Many routes support query parameters to help specify how a request to the validator should be formed. Not every endpoint supports every query, and some endpoints have their own parameters specific to just to them.

Head	Index / ID of head block
Count	no of resources to fetch
Start	Start paging
Limit	No of items to return
Reverse	If list should be reversed
Wait	Time to wait till commit
Min , Max , Sort	----------------

ENDPOINTS

- The endpoints include RESTful references to resources stored in the Sawtooth ledger that clients might be interested in, like blocks and transactions, as well as RESTish metadata, like batch status.

- A GET request fetches one or many resources, depending on whether or not a particular resource identifier was specified

- It specifies the location in which resource to be fetched

ENDPOINTS

- GET/batches : The batches stored on the blockchain, referenced by id

  Query Parameters : head,start,limit,reverse

  Status Codes : 200,400,500,503

- GET/batches/{batch_id} : Fetches a particular transaction

  Query Parameters : batchid{String}=Batchid

  Status Codes : 200,400,500,503,404

- GET/batch-statuses : Fetches the committed statuses for a set of batches

  (Committed, Invalid, Pending, Unknown)

  Query Parameters : id{string}, wait

  Status Codes : 200,400,500,503,404

ENDPOINTS

- GET/state : Fetches data for the current state

  Query Parameters : head,start,limit,reverse

  Status Codes : 200,400,500,503

- GET/state/{address} : Fetches a particular leaf from a current state

  Query Parameters : head

  Status Codes : 200,400,500,503,404

- GET/blocks : Fetches a list of blocks from the validator

  Query Parameters : Head,start,limit,reverse

  Status Codes : 200,400,500,503

ENDPOINTS

- GET/blocks/{block-id} : Fetches a particular transaction

  Query Parameters : Block id

  Status Codes : 200,400,500,503,404
- GET/transactions : Fetches a paginated list of transactions from the validator

  Query Parameters : head,start,limit,reverse

  Status Codes : 200,400,500,503
- GET/transactions/{transaction_id} : Fetches a particular transaction

  Query Parameters : transaction_id

  Status Codes : 200,400,4034,500,503

ENDPOINTS

- GET/receipts : Fetches the receipts for a set of transactions.

  Query Parameters : id

  Status Codes : 200,400,500,503

- GET/peers : Fetches the endpoint of the authorised peers of the validator

  Status Codes : 200,400,500,503

- GET/status : Fetches information pertaining to the status of the validator

  Status Codes : 200,400,500,503

ENDPOINTS

- POST/batches : It accepts a protobuf formatted batchlist & submits to validator

  Query Parameters :

  Status Codes : 202,400,429,500,503

- POST/batch_statuses : Fetches the committed status for a set of batches

  Query Parameters : wait

  Status Codes : 200,400,500,503,

- POST/receipts : Fetches the receipts for a set of transaction

  Query Parameters : wait

  Status Codes : 200,400,500,503

HTTP Status Codes

In order to improve clarity and ease parsing, the REST API supports a limited number of common HTTP status code

STATUS CODE	TITLE
202	Accepted
400	Bad Request
429	Too Many Request
500	Internal Server error
503	API unable to reach validator
200	OK
404	Not found

Response

- Results will be sent in a JSON envelope with at least four properties
- { Data, Head, Link, Paging)
  - Data – the requested resource
  - Head – the id of the head block of the chain
  - Link – a link to the resource fetched
  - Paging – information about how further pages can be fetched

Response – Errors

- If something goes wrong while processing a request, the REST API will send back a response envelope with only one property: "error". That error will contain three values which explain the problem that occurred:

  code - machine-parsable code specific to this particular error

  title - short human-readable headline for the error

  message - longer more detailed explanation of what went wrong

- While the title or message of an error may change or be reworded over time, **the code is fixed**, and will always refer to the same error.

Error Response Example

```
{
  "error": {
    "code": 30,
    "title": "Submitted Batches Invalid",
    "message": "The submitted Batch List is invalid. It was poorly formed, or has an invalid signature."
  }
}
```

EXAMPLE CODE

```
http = require('http')


//GET data
function getData(){
    http.get("http://localhost:8008/state", function(response){
        console.log(response)
    })
}
```

EXAMPLE CODE

```
//POST data
function postData(data){
    var post_options = {
                                method: 'POST',
                                headers:  {'Content-Type': 'application/json'},
                               }
    var post_req = http.request("http://localhost:8008/state",options ,function(response){
         console.log(response)
    })
    post_req.write(data)
    post_req.end()
}
```

K B A ™

THANK YOU