Manifold Technology™

Manifold Liquidity Platform™ - Serverless

MLP-S Quick Start & REST API Documentation

V0.1.2 - 8/8/2018

Requirements:

AWS CLI installed on a machine with internet access.

Getting Started:

Launching MLP-S from the Amazon marketplace will create a single EC2 instance that is used to deploy the serverless components. Once MLP-S deployment has been initialized, monitor cloudformation in the AWS console to check the deployment status (https://console.aws.amazon.com/cloudformation/home). Watch for the mlp-serverless stack to report “CREATE_COMPLETE”.

Once the mlp-serverless stack has successfully deployed, remove the temporary deployment instance by deleting the cloudformation stack that was defined during the marketplace process

The mlp-init lambda function can then be called to create the mlp-s admin account (with corresponding admin password), set the name of the virtual currency, and set the total amount of currency that be created for use within the blockchain.

Note - Additional currency cannot be added into the system after blockchain initialization.

```
aws lambda invoke --function-name mlp-init --invocation-type RequestResponse --payload
'{ "admin": "password", "currency": "currency_name", "amount": number}' init.log
```

Check the output of the init.log file to verify that the init function ran successfully.
Init.log will also include the endpoint for specific blockchain (“baseUrl”)

Optionally, you may configure MLP to publish uberhashes to the Bitcoin network as part of a minimally sized transaction.

To do so add these arguments to the payload JSON:

- wallet1 - either a Bitcoin wallet in WIF format OR a BTC private key corresponding to an address with assigned value
- wallet2 - another Bitcoin wallet in WIF format OR a BTC private key corresponding to an address with assigned value

Both wallet1 and wallet2 need to have some Bitcoins in them.

Network either 'testnet' or 'livenet' - defaults to testnet so that real Bitcoins are not spent

```
aws lambda invoke --function-name mlp-init --invocation-type RequestResponse --payload
'{ "admin": "password", "currency": "name", "amount": number, "wallet1":
"~/wallet1.WIF", "wallet2": "~/wallet2.WIF", "network": "testnet"}' init.log
```

The uberhash publishing process will periodically transfer a minimal amount of satoshis back and forth between wallet1 and wallet2.

After this command is executed, init.log should contain output like:

```
{
	"messages": ["Using Bitcoin testnet network", "Bitcoin integration enabled"],
	"adminUserId": "58f1ff25-e758-4af1-a0fc-d50d2bb3eee7",
	"initEventId": "6a856f3e-bec7-41c6-a43e-e63b60307c73",
	"baseUrl": "https://20l1scai0j.execute-api.us-west-2.amazonaws.com/prod"
}
```

MLP-S Concepts:

Important

Users vs Accounts
Users own and control accounts. A user has one or more accounts associated with it.
Each account can be used to store blockchain currency.

There are three publicly accessible accessable endpoints:

```
/config
/register
/login
```

While MLP-S uses a cognito user pool for authentication, these endpoint have been provided to abstract the implementation details and provide a simple method for end users to interact with the system.

Hit the register endpoint to create a new user account in the system.

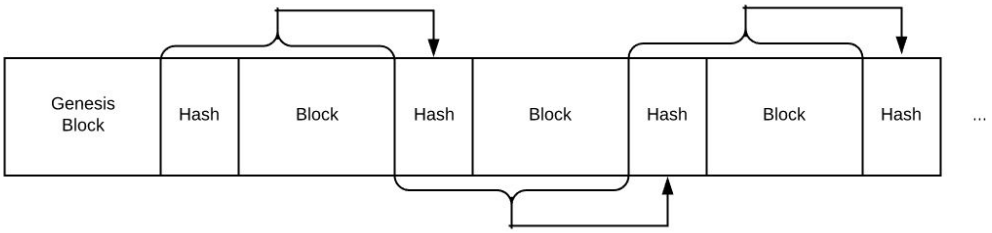
Then use the same username/password to authenticate with the login endpoint.

A successful login will include a response thats has an "idToken". The corresponding value of the “jwtToken” key is what is then used in the “Authentication” header in order to access the other endpoints which require cognito user pool auth.

Each user created in the system has an account, which is a where blockchain currency is stored. New users do receive any currency so the admin user must transfer some amount to that user’s account in order for them to have a balance. Additional accounts for a given user can be created using the “/account” endpoint.

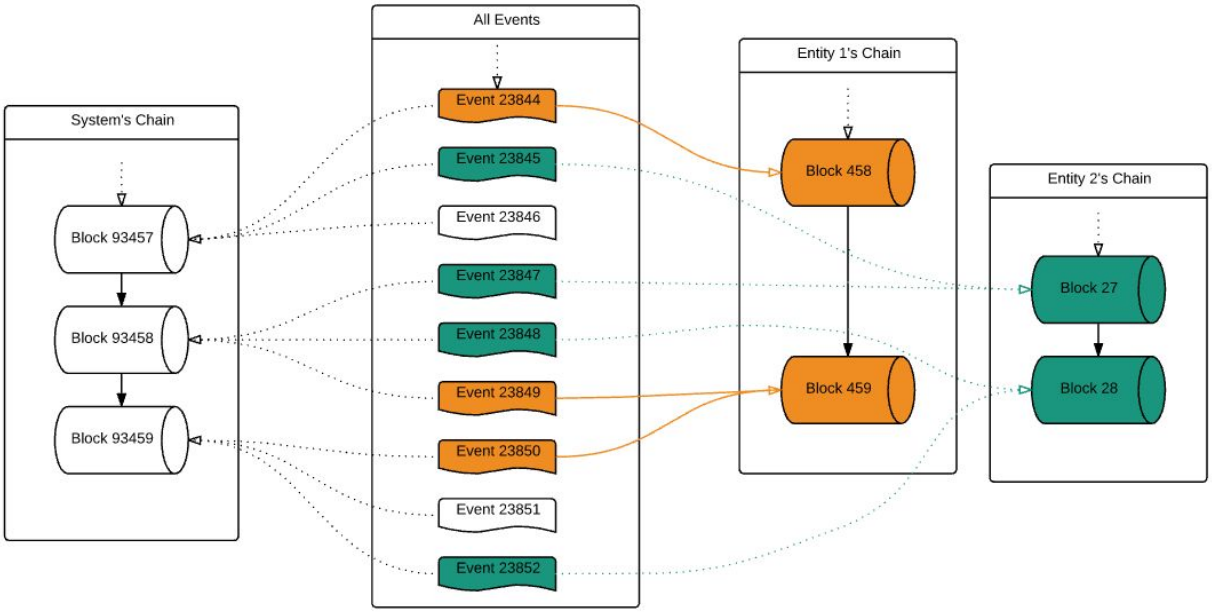
In addition to currency transfers, any arbitrary data can be stored in the blockchain using the “/event” endpoint. This endpoint requires a “chainId” in the URI, which can be retrieved using the “/chain” endpoint.

All transactions and events interacting with the MLP-S are stored in the blocks in the blockchain. Each block has a corresponding cryptographic hash that can be used to verify that no modification to that block has occurred after the fact.



The hash of each block is included in the hash of the subsequent block, linking the chain together.

Each user in the system has their own private blockchain. Each of these blockchains compose a portion of the larger system chain:



In addition to individual blockchain block hashes, MLP-S includes the concept of metahashes and uberhashses.

Metahash, within MLP-S, is a single cryptographic hash of all events and transactions on the full system chain that have occured within the system over the previous hour.

Uberhash is a single cryptographic hash of all events and transactions on the full system chain that have occured within the system over the previous 24 hours. If bitcoin integration is enabled in the system, the uberhash is what is written as metadata to the bitcoin blockchain.

REST API

<table>
<tr><th colspan="2">REST Interface Properties</th></tr>
<tr><td>Authentication</td><td>Unless otherwise noted, all endpoints require the HTTP Authorization header to be populated with a Cognito user ID token (JWT Token).

Endpoints that require admin level privileges will result in a 403 error when invoked by a non-admin user.</td></tr>
<tr><td>System Users</td><td>admin user is created as part of the <code>mlp-init</code> function</td></tr>
<tr><td>Parameter Placement</td><td>
<table>
<tr><th>GET</th></tr>
<tr><td>Lookup of a specific, single resource, is done via path variables in the URI.
e.g. /account/{accountId}</td></tr>
<tr><td>Querying or filtering, which may return 0..n results in a collection, is done via query parameters.

e.g. /events/{chainId}?startTime=2018-04-01T12:30:00.000Z
&endTime=2018-04-10T12:30:00.000Z</td></tr>
<tr><th>POST</th></tr>
<tr><td>Insert for a new resource.
e.g. /account

HTTP entity-body contains the JSON representation of that account object.</td></tr>
<tr><th>DELETE</th></tr>
<tr><td>Delete a specific resource.
e.g. /account/<account UUID></td></tr>
</table>
</td></tr>
<tr><td>Time/Date Parameters</td><td>All time/date parameters are defined in ISO 8601 format unless otherwise noted. The time zone is UTC.

e.g. 2018-04-01T12:30:00.000Z</td></tr>
<tr><td>Error Handling</td><td>Any request that cannot be satisfied will receive as a response the appropriate HTTP Status Code (https://en.wikipedia.org/wiki/List_of_HTTP_status_codes) and an error message in the following format:</td></tr>
</table>

	{
	"Error Message": "The specified account 9e9b4bef-b998-44ff-8a30-e3ff4089e16a could not be found."
	}

Multiple Inputs	Where explicitly allowed, multiple values can be passed to a single query parameter.
JSON	All request payload and response body data types are JSON unless explicitly noted otherwise. See Examples table at the end of this file for examples of the JSON data structures used in the API.

GET /config

Public endpoint to return useful configuration information, specifically the Cognito user pool and client ids necessary to use the Cognito API, and the configured virtual currency for the MLP instance. Does not require User Token Authorization.

Response Body:

```
{
	"currency": "<currency>",
	"userPoolId": "<Cognito user pool ID>",
	"clientId": "<Cognito user pool client ID>"
}
```

POST /register

Public helper endpoint to assist in creation of new users without having to use the Cognito API. Does not require User Token Authorization.

JSON Payload Body:

```
{
	"username": "<username>",
	"password": "<password>"
}
```

Passwords must meet default Cognito password requirements of 8 characters including an uppercase and lowercase letter and a number.

Response Body:


```
{
    "user": {
        "username": "<username>",
        "pool": {
            "userPoolId": "us-east-2_IsTMR4oOd",
            "clientId": "6rrhjdnlhe20ifqq874qlajq2k",
            "client": {
                "endpoint": "https://cognito-idp.us-east-2.amazonaws.com/",
                "userAgent": "aws-amplify/0.1.x js"
            },
            "advancedSecurityDataCollectionFlag": true
        },
        "Session": null,
        "client": {
            "endpoint": "https://cognito-idp.us-east-2.amazonaws.com/",
            "userAgent": "aws-amplify/0.1.x js"
        },
        "signInUserSession": null,
        "authenticationFlowType": "USER_SRP_AUTH"
    },
    "userConfirmed": true,
    "userSub": "d57c5682-5317-47d0-bc32-0a910d535dd1"
}
```

POST /login

Public helper endpoint to allow logging in to MLP without having to use the Cognito API. Hit this endpoint after calling register. Does not require User Token Authorization.

Returns the result of calling AmazonCognitoIdentity.CognitoUser.authenticateUser

JSON Payload Body:

```
{
       "username": "<username>",
       "password": "<password>"
}
```

Response Body:


```
{
    "idToken": {
```

```
        "jwtToken":
"eyJraWQiOiJ5V1JVb0FQaERXXC9HbWNHZW1RXC9ITHQrM2ZHQ1ZFZXpPZU5hQlV4VVUwd3c9IiwiYWxnIjoiUlMyNTYif
Q.eyJzdWIiOiJkNTdjNTY4Mi01MzE3LTQ3ZDAtYmMzMi0wYTkxMGQ1MzVkZDEiLCJhdWQiOiI2cnJoamRubGhlMjBpZnFx
ODc0cWxhanEyayIsImV2ZW50X2lkIjoiMmQ0YmUxNzUtNDgxOS0xMWU4LWI5NTktNTlkZmNkZjNhOWMxIiwidG9rZW5fdX
NlIjoiaWQiLCJhdXRoX3RpbWUiOjE1MjQ2MTMzNjAsImlzcyI6Imh0dHBzOlwvXC9jb2duaXRvLWlkcC51cy1lYXN0LTIu
YW1hem9uYXdzLmNvbVwvdXMtZWFzdC0yX0lzVE1SNG9PZCIsImNvZ25pdG86dXNlcm5hbWUiOiJUeWxlclAiLCJleHAiOj
E1MjQ2MTY5NjAsImlhdCI6MTUyNDYxMzM2MH0.M3OpaQPXFJO5c9HWl4dx0oPYFORJmpKFcs_iZLZoKcRWFRLZH6-jROy5
Aa0XqDgX1WxLA1vINDZJuJEC477XW270f6yTdt8d0mxliS0zwDhm5qBgX84_ZvkIlTqHYbLwf8AQ7oOI73tazOMuCRU9ds
Ajh7SyVkcDKbAWqTetOmPvrsFwPSb_mTuvKK_dzogj1VAnPQ-B9e6n8rDJSA-5DHpMlYXpFMRHVQDHKi-u40FT9syy1hmW
q8QclmTFcXQ2IZ_wZCh4y_3bYqj71o71qRwv-n1Ihd7TpQFdXIgbyC-q9BsL7Av9yAKDcDETs8oLsATcg6N5_Vr5fbwqlQ
bwLQ",
        "payload": {
            "sub": "d57c5682-5317-47d0-bc32-0a910d535dd1",
            "aud": "6rrhjdnlhe20ifqq874qlajq2k",
            "event_id": "2d4be175-4819-11e8-b959-59dfcdf3a9c1",
            "token_use": "id",
            "auth_time": 1524613360,
            "iss": "https://cognito-idp.us-east-2.amazonaws.com/us-east-2_IsTMR4oOd",
            "cognito:username": "<username>",
            "exp": 1524616960,
            "iat": 1524613360
        }
    },
    "refreshToken": {
        "token":
"eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAifQ.RXzRKNwKASkISQ5hABLI9K9GWyPxwT
ENPE29RM1ZbKBWZ5lmavQwvTkzneiI6ilfk-IBWgFoTXkW28YSMl8C3JYAnbOd1S5LdI9_fgAV9JhGXj1TnLwSBk6-HRxU
PcnhYpTzHrl9EePMjtRg5FxkP61i0AeXSbLprbJ4kPx_gSCDyB3PPWdhVJvW3MLTmjEnVrjGY6zVXuOajLOMONaNjt6f92
MrcKgXfd_ebIlOmeG026jzbqZ4pKXAX2rsw6SliakYomqTzuuoDDUwh4NHm7Cb1zomkXupiGs9SNVkMAYTJO3ZbDwbswaD
h8G3_MgJzVE-rK5Y4_ln8v_PcNW5rw.qhnn0n53wCdRvjRz.pyWGKibSN3KqHHpdWEciky7BEEWqY4Oh6HGOmA12YSVMfu
Y2E3yFve6UK1th7RVshuj4Ahv5-V9D0P8OItU0dkVuV5KgpiXto5kZjnzD2W9rgm3vaWSWunkt4gbPMNU5GeaCM1Q7FpSh
I5ez4nDPSBIrdEEuM6fIO8G0zwWQtSMFLJj_G1XZZLwIJndB7FaAXlSWtBUeo3eVS5s6nWWkunZvVb8MEQfdTCzU7Hq-Ld
UsNbI7sqwNJP1E-6Y12QFYCXbSg7Ygq-w6Qx71onVDkG9X2q_cRmqNBe-duR0OZFzDwjm5iKMd_GKn-0mEiJnrH62eoKh5
rmMOBrNWd6AFb4GDElc0rcBlzU7-xvjt1uMh5-_TSXKBCamRqDwSk_hnsXw0Fmii_AfIFDBp2Q51A4TBOwzoSRsBK-eADJ
W5A1_cBATLVGzeDmwylySnZKe7I8e6M3ELbsqTwLTxbi52yQ0kps0qmp6TOfteY6qNybInsIB5zI1t6m6sAyz802y2HYim
VJOjC_WrQYgIsn_ojtzPOyQDWbX9J7JvKsVRPvKCsvD766jinO7ke9fJfoHZbeNXpMqXYCh1ftsiw3flrUud7Eo9NWx2WQ
5hizG2MSqL8wd7U27q81KJ-iLmU45YE-yeEIirbTmxF9qT9OeYY3wkW4Tkjt2FlGX5RXmOGuZIOIRNW7U5mdzHaS5JcX7q
v8qZD6JWLSXdhbBa531J3y_KBlPPr7LpCrYsxv3yJ_y75TBLBXkMb4CJj3gaIbPgz1Nat7P0LwB4I6TT6Rt0MRzBpsqdKW
8soqAQIXue06s0HraMA1oeICbg86Jjvzbd3unagMUZxNgfr6u6BtRXQnbgNcqoK8_NfnBi9un25KTMp-iA3OTb9V4eoa-N
GCaPXCz53zbSHyQx-DcE_yZbA_06-JtWxhTLSSZtKwyFM6AZpdLEMxSuE362sQ7-6w0WpFu33MAsiAPM1DOzp1h4l-iEWl
8OnE1lamYYEQWojFTMfohsj-LV7a0JqZsQgbP_h3FwUsRXCBaW90VIApvB03DvAeYIvCMWVQg0FDiRDhHnGrxuAA_9sarC
QTgmrvZPb1PYjz67yetZuqCyoysnN782OahjjSppyQ2CUY0zt64cKla9uoyvFguoXUP9RX5M3-PB_gjtY6XEu3OwMd9m5I
OrSTeV1Kb4-6Ql2fl0t1FZ68xb8OBUyWVZ1lylyO4-D42dAH2uHZaupHog7wmVLPL_MMDi9mHECgoWmgxwwCGMVdYjplXU
fHr3GCsHh_31gppgWR4PSw.oMmbeslTviNyAm7Wq6ZA6w"
    },
    "accessToken": {
        "jwtToken":
"eyJraWQiOiJcLytqSmlIQnMxS1pzdEJPNzExNXZiNWdMXC9qd1RvR0RrUEpsK2J6RFVDU0U9IiwiYWxnIjoiUlMyNTYif
Q.eyJzdWIiOiJkNTdjNTY4Mi01MzE3LTQ3ZDAtYmMzMi0wYTkxMGQ1MzVkZDEiLCJldmVudF9pZCI6IjJkNGJlMTc1LTQ4
MTktMTFlOC1iOTU5LTU5ZGZjZGYzYTljMSIsInRva2VuX3VzZSI6ImFjY2VzcyIsInNjb3BlIjoiYXdzLmNvZ25pdG8uc2
lnbmluLnVzZXIuYWRtaW4iLCJhdXRoX3RpbWUiOjE1MjQ2MTMzNjAsImlzcyI6Imh0dHBzOlwvXC9jb2duaXRvLWlkcC51
cy1lYXN0LTIuYW1hem9uYXdzLmNvbVwvdXMtZWFzdC0yX0lzVE1SNG9PZCIsImV4cCI6MTUyNDYxNjk2MCwiaWF0IjoxNT
I0NjEzMzYwLCJqdGkiOiJkMjU3YTYyMS1lZmIwLTRjYzUtOThhNy0xYzE3Njc1ZDNmODUiLCJjbGllbnRfaWQiOiI2cnJo
amRubGhlMjBpZnFxODc0cWxhanEyayIsInVzZXJuYW1lIjoiVHlsZXJQIn0.bT59swuAIXv6x4OWrNdkUVy_DZhyZ0g6Ad
Uzd0betJMfvk1FOtl6jmg5A3jXPyk0hv8sU76ysaz-qHm3_8dZElQN5r6V_FtF2H8OqXscsDAycQkxrls3fgQgULxyzqcu
kHS-zZ6cdZt13tfmcfeVkUtl4oRa22DW9LoViTciJq_Ws8ux8Jh5tdfyqAyES6JuBDTU1plRMnIIpb7s6elsGnOl7XFGhd
```

```
tIuhRK_PXc_CNj4SFAhcZ_Mur60a5-p7SjuhveqXBncy3X790gShFRp-Eq0XjGij1xoFUA1T6x86jZwtg66yR1oHZNEgv2
n-afBJQX_P69LUGLaZRUSG7xJA",
        "payload": {
            "sub": "d57c5682-5317-47d0-bc32-0a910d535dd1",
            "event_id": "2d4be175-4819-11e8-b959-59dfcdf3a9c1",
            "token_use": "access",
            "scope": "aws.cognito.signin.user.admin",
            "auth_time": 1524613360,
            "iss": "https://cognito-idp.us-east-2.amazonaws.com/us-east-2_IsTMR4oOd",
            "exp": 1524616960,
            "iat": 1524613360,
            "jti": "d257a621-efb0-4cc5-98a7-1c17675d3f85",
            "client_id": "6rrhjdnlhe20ifqq874qlajq2k",
            "username": "<username>"
        }
    },
    "clockDrift": 0
}
```


Note - the "idToken" - "jwtToken" value is what is passed in for the “Authorization” field in the header for all of the subsequent REST calls.

POST /account

Creates a new account for the current user.
All users get a default account at account creation, this endpoint creates an additional account that can be used to store the blockchain currency.

No POST Body required.

Returns the UUID eventId of the corresponding account creation event.
This is also the new account’s id.

GET /account/{accountId}

Returns the specified account. If this is invoked by someone other than the admin or the account owner, an error occurs. If the account does not exist, a 404 response will be returned.

Response Body:


```
{
       "id": "<unique account identifier>",
       "ownerId": "<id of the user that owns this account"
```

```
        "balances": {
                "<currency>": <amount>,
                ...
        },
        "metadata": {<optional JSON data>}
}
```

DELETE /account/{accountId}

Deletes the specified account.
Returns the eventId of the corresponding account deletion event.

If this is invoked by someone other than the admin or the account owner, an error occurs.
If the account does not exist, a 404 response will be returned.

GET /account/{accountId}/history

Returns any events that involved the specified account.
If this is invoked by someone other than the admin or the account owner, an error occurs.
If the account does not exist, a 404 response will be returned.

The following query parameters are accepted:

`startTime`	Return events codified after the specified date. Can be submitted as number of milliseconds since the epoch or as an ISO8601 formatted string.
`endTime`	Return events codified before the specified date. Can be submitted as number of milliseconds since the epoch or as an ISO8601 formatted string.

Response Body:

```
[
        {
                "chainId": "<unique blockchain identifier>",
                "hash": "<hash of this event>",
                "id": "<unique event identifier>",
                "type": "<event type>"
```

```
                "userId": "<id of the user that submitted this event>",
                "data": {<event object data>},
                "created": "<ISO8601 timestamp of when this event was created>",
                "objectId": "<the account id>",
                "inverseId": "<if this event triggers a corresponding event, the id of that
        event>"
        }
        …
]
```

GET /accounts

Returns the accounts owned by the current user.

Response Body:

```
[
        {
                "id": "<unique account identifier>",
                "ownerId": "<id of the user that owns this account"
                "balances": {
                        "<currency>": <amount>,
                        ...
                },
                "metadata": {<optional JSON data>}
        },
        ...
]
```

POST /transfer

Transfers a currency amount between two accounts.

During the initialization step, the administrator account specifies the amount of currency created. This initial amount needs to be distributed from the admin account (airdropped) to other user accounts in order to be accessed and usable by others.

If this is invoked by someone other than the admin or the from account owner, an error occurs.
If either of the accounts does not exist, a 404 response will be returned.
If the from account does not have enough of the currency to transfer, an error occurs.

Accepts the JSON body:

```
{
	“from”: “<source account id>”,
	“to”: “<destination account id>”,
	“currency”: “<currency>”,
	“amount”: <amount to transfer>
}
```

Returns the id of the transfer event.

POST /event/{chainId}

Utility endpoint to submit generic event data to the specified blockchain. If this is invoked by someone other than the admin or the blockchain owner, an error occurs. Returns the eventId of the new event.

Accepts the JSON body:

```
{
	"type": "<an event type string>",
	"data": {<any JSON data>}
}
```

GET /chain

Returns the blockchain for the current user.

Response Body:

```
{
	"id": "<unique blockchain identifier>",
	"lastHash": "<hash of the last block codified in this blockchain>",
	"ownerId": "<id of the user that owns this blockchain>"
	"blockCount": <number of blocks in the chain>,
	"eventCount": <number of events codified in the chain>,
}
```

GET /chain/{chainId}

Returns the specified blockchain. If this is invoked by someone other than the admin or the blockchain owner, an error occurs. If the blockchain does not exist, a 404 response will be returned.

Response Body:

```
{
	"id": "<unique blockchain identifier>",
	"lastHash": "<hash of the last block codified in this blockchain>",
	"ownerId": "<id of the user that owns this blockchain>"
	"blockCount": <number of blocks in the chain>,
	"eventCount": <number of events codified in the chain>,
}
```

GET /block/{chainId}

Returns the last block in the specified blockchain. If this is invoked by someone other than the admin or the blockchain owner, an error occurs. If the blockchain does not exist, a 404 response will be returned.

Response Body:

```
{
	"chainId": "<unique blockchain identifier>",
	"hash": "<block hash string>",
	"prevHash": "<hash of block codified before this one>",
	"ownerId": "<id of the user that owns the blockchain>",
	"created": "<ISO8601 timestamp of when this block was created>",
	"events": [<array of event objects that were codified in this block>]
}
```

GET /block/{chainId}/{blockHash}

Returns the specified block in the specified blockchain. If this is invoked by someone other than the admin or the blockchain owner, an error occurs. If the blockchain/block hash combination does not exist, a 404 response will be returned.

Response Body:

```
{
	"chainId": "<unique blockchain identifier>",
	"hash": "<block hash string>",
	"prevHash": "<hash of block codified before this one>",
	"ownerId": "<id of the user that owns the blockchain>",
	"created": "<ISO8601 timestamp of when this block was created>",
	"events": [<array of event objects that were codified in this block>]
}
```

GET /event/{chainId}/{eventId}

Returns a specific event in a blockchain. If this is invoked by someone other than the admin or the blockchain owner, an error occurs. If the event does not exist, a 404 response will be returned.

Response Body:

```
{
        "chainId": "<unique blockchain identifier>",
        "hash": "<hash of this event>",
        "id": "<unique event identifier>",
        "type": "<event type>"
        "userId": "<id of the user that submitted this event>",
        "data": {<event object data>},
        "created": "<ISO8601 timestamp of when this event was created>",
        "objectId": "<optional id of an MLP object that this event references>",
        "inverseId": "<if this event triggers a corresponding event, the id of that event>"
}
```

GET /events/{chainId}

Returns events in the specified blockchain that matches optional filter parameters. If this is invoked by someone other than the admin or the blockchain owner, an error occurs. The following query parameters are accepted:

`type`	Return events with the specified event type
`objectId`	Return events with the specified objectId
`startTime`	Return events codified after the specified date. Can be submitted as number of milliseconds since the epoch or as an ISO8601 formatted string.
`endTime`	Return events codified before the specified date. Can be submitted as number of milliseconds since the epoch or as an ISO8601 formatted string.

If no filter parameters are submitted, this will return all events in the specified blockchain. This case may lead to a very expensive operation.

Response Body:

```
[
        {
                "chainId": "<unique blockchain identifier>",
                 "hash": "<hash of this event>",
                "id": "<unique event identifier>",
                "type": "<event type>"
                 "userId": "<id of the user that submitted this event>",
                "data": {<event object data>},
                 "created": "<ISO8601 timestamp of when this event was created>",
                "objectId": "<optional id of an MLP object that this event references>",
                 "inverseId": "<if this event triggers a corresponding event, the id of that
        event>"
        }
        …
]
```

GET /uberhash

Admin only endpoint to return the current uberhash. If an uberhash has not been generated this will return a 404 error.

Uberhashes are generated daily.

Response Body:

```
{
        "hash": "<current uberhash string>",
        "prevHash": "<previous uberhash string or null>",
        "txHash": "<if Bitcoin integration enabled, hash of transaction that contains this
uberhash>",
        "created": "<ISO8601 timestamp of when this uberhash was created>",
        "hashCount": <number of metahashes used to calculate this uberhash>,
        "metaHashes": [array of all the metahashes used to calculate this uberhash]
}
```

GET /uberhash/{hash}

Admin only endpoint to return the specified uberhash. If the uberhash does not exist this will return a 404 error.

Response Body:

```
{
        "hash: "<current uberhash string>",
        "prevHash: "<previous uberhash string or null>",
```

```
	"txHash": "<if Bitcoin integration enabled, hash of transaction that contains this
uberhash>",
	"created": "<ISO8601 timestamp of when uberhash was created>",
	"hashCount": <number of metahashes used to calculate this uberhash>,
	"metaHashes": [array of all the metahashes used to calculate this uberhash]
}
```

GET /metahashes

Admin only endpoint to return all metahashes generated by MLP.

Metahashes are generated hourly.

Response Body:

```
[
	{
		"hash": "<metahash string>",
		"blockCount": <number of blocks in this metahash>,
		"created": "<ISO8601 timestamp of when this metahash was created>"
	},
	...
]
```

GET /metahash/{hash}

Admin only endpoint to return the specified metahash. If the metahash does not exist this will return a 404 error.

Response Body:

```
{
	"hash": "<metahash string>",
	"blockCount": <number of blocks in this metahash>,
	"created": "<ISO8601 timestamp of when this metahash was created>",
	"blocks": [array of all the block hashes user to calculate this metahash]
}
```

DELETE /user/{userId}

Admin only endpoint to delete users. If the user does not exist, a 404 response will be returned. Returns the eventId of the corresponding user deletion event.