BigchainDB Driver (Version 2) Documentation

11/14/16

TODO: Note that the new version of BigchainDB is still pending on adding threshold conditions and transactions belonging to multiple users for the Python Library. These are pending on integration into the new driver. Also, perhaps at a later iteration, it may be fruitful to add Hash-locked Conditions and Timeout Conditions. Finally, the driver takes in a flag (0 for bigchain keys and 1 for Eris keys). It remains to test the case when the flag is zero.

Note: One endpoint is no longer relevant with respect to the new model of BigchainDB. It is the default endpoint for adding data. This endpoint added data with the key of the federation node. Thus, there is only one endpoint for adding data. The ‘addData’ endpoint now requires the public key of the user, whereas before it was optional.

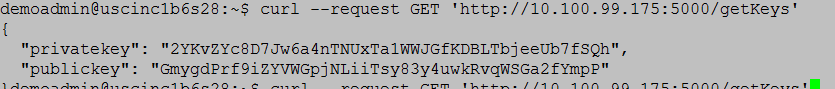
The following endpoints have been tested to work:

**ENDPOINT:** /getKeys

**DESCRIPTION:** This endpoint generates a key pair.

**TESTING REQUEST:** curl --request GET 'http://10.100.99.175:5000/getKeys'

**OUTPUT:**



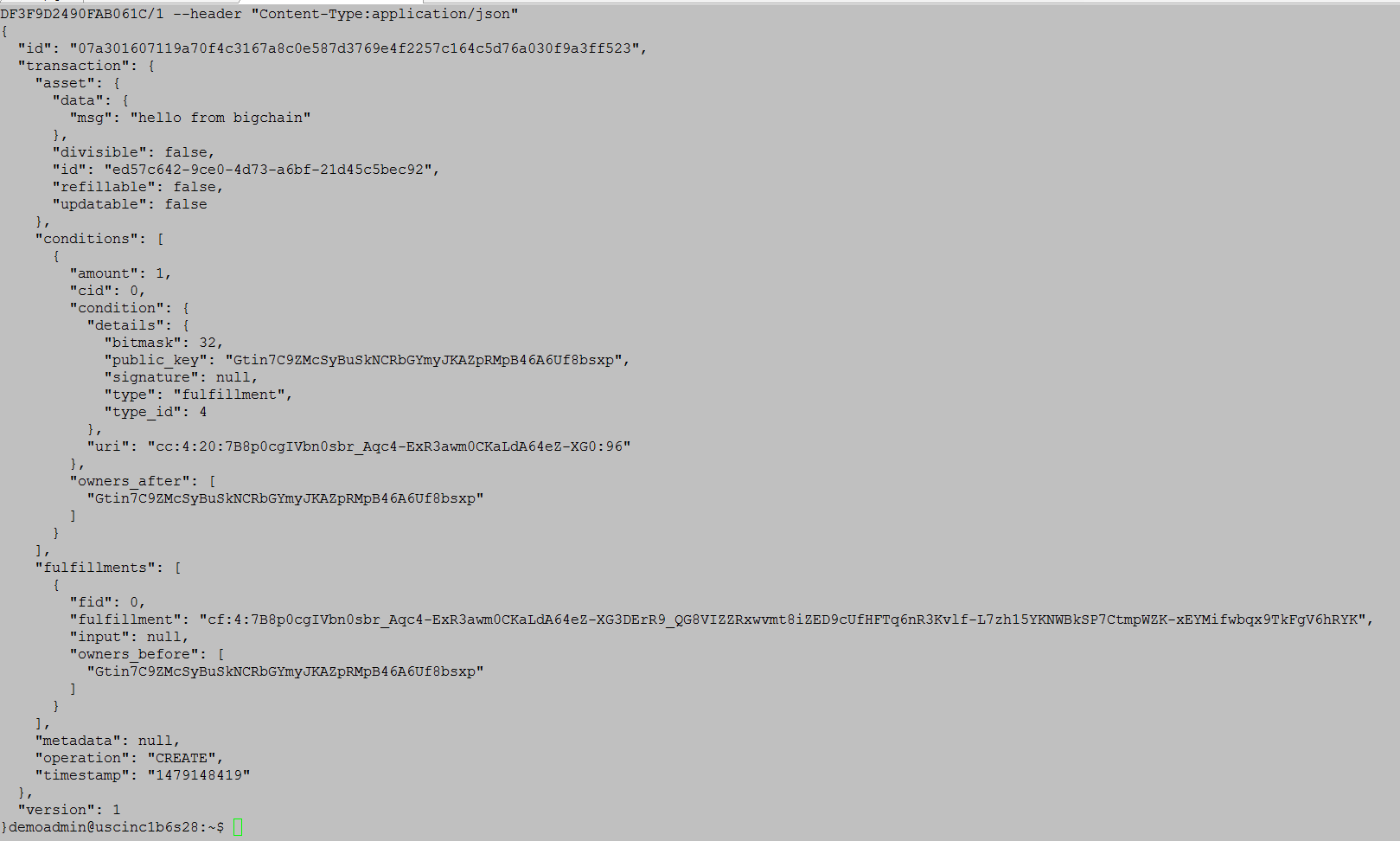
**ENDPOINT:** /addData/<pub1>/<flag>

**DESCRIPTION:** This endpoint adds a new digital\_asset for a user. <pub1> is the user’s public key. <flag> will be 0 if the public key is the user’s bigchain key; flag will be 1 if the public key is the user’s Eris key. Input is the JSON object (digital asset payload). Note that with the new version of bigchaindb, the input JSON object must contain the key “data”!

**TESTING REQUEST:** This was tested twice with the same public key. The reason is the first time, internally, the bigchaindb key pair is generated. The second time, it is retrieved. This was the request that was tested:

curl -X POST -d '{"data":{"msg":"hello from bigchain"}}' http://10.100.99.175:5000/addData/D43DFF3D0809BCBAA91AE759769D7B181C31652279B2E5DDF3F9D2490FAB061C/1 --header "Content-Type:application/json"

**OUTPUT:**



**ENDPOINT:** /getTransaction/<tx\_ID>

**DESCRIPTION:** This endpoint returns transaction data. Input is the transaction ID.

**TESTING REQUEST:** For testing, we use the transaction ID from the transaction above. The request is displayed below:

curl --request GET 'http://10.100.99.175:5000/getTransaction/07a301607119a70f4c3167a8c0e587d3769e4f2257c164c5d76a030f9a3ff523’

**OUTPUT:**



**ENDPOINT:** /transaction

**DESCRIPTION:** This endpoint implements a transaction of a digital asset. Input is in JSON format: “PubTo” (the public key for who will receive the asset), “PubFrom” (the private key of the sender of the asset), “txID” (the transaction ID) and “flag” (0 if dealing with Bigchain Keys; 1 if dealing with Eris Keys).

**TESTING REQUEST:**

(Part 1—Create an Asset): curl -X POST -d '{data":{"msg":"hello from bigchain"}}' http://10.100.99.175:5000/addData/a/1 --header "Content-Type:application/json"

(Part 2—Transfer the Asset): curl -X POST -d '{"PubTo": "b" ,"PubFrom":"a", "flag": 1, "txID":"4527cdcd41fa4730d6161bd097caee91de971bf137faee70946436021a41a675"}' http://10.100.99.175:5000/transaction --header "Content-Type:application/json"

**OUTPUT (From Part 2):**

