Factoid API

This document does a quick summary of the API for Factoids and the Factoid wallet. At this point in time the wallet is a commandline driven program, intended to demonstrate the API more than to be a viable commerical wallet solution.

The first step is to install the factom client and factom wallet helpers. See the How To guides for setting up in your environment. You will need to run factomd and fctwallet. Note that from time to time over the next few months you will need to update factomd to continue to communicate with the network. Watch our technical blogfor notifications and updates.

The two APIs that are of interest are implemented by factomd and fctwallet. The first, **factomd**, is the client program that actually participates in the Factom Network. The second, **fctwallet**, provides common wallet functions, and it maintains the address book where private keys are kept. A third program, **walletapp**, provides for cold wallets, the generation of offline transactions, and the submission of offline transactions to the factom network.

The factom client **factomd** provides a RESTful interface found at http://localhost:8088 by default. None of the calls to factomd present any security issues, so factomd does not have to be colocated with programs creating and submitting transactions.

The factom wallet **fctwallet** provides a RESTful interface found at http://localhost:8089 by default. Calls to fctwallet allow for the creation of transactions against factoid addresses held in its wallet. Access to the API then must be kept secure.

The factom wallet **walletapp** is an alternative that only supports factoid functions. It communicates only with factomd. The *walletapp* supports cold storage, construction of offline transactions, and the submission of offline transaction to the factom network. We will soon be releasing a GUI that will run on top of the *walletapp* that will provide support for general factoid users.

The factom commandline wallet **factom-cli** is a wallet that supports factoid transactions, as well as general access to the Factom protocol. *factom-cli* uses the interfaces to *factomd* and *fctwallet* to implement its functionality. The main purpose of this program is to demonstrate the use of the factom APIs. It can also be used to script transaction processes against Factom.

Note: Examples of the API calls provided below can be executed in command line directly if curl is installed. The API calls have been prefixed with curl -X POST or curl -X GET depending on method.

factomd

This is a summary of the factomd API as pertains to trading Factoids. We will add detail on other calls as we go forward.

- Post **http://localhost:8088/v1/commit-chain/?**

  Commits a chain. The first step towards creating a new chain.

- Post **http://localhost:8088/v1/reveal-chain/?**

  Reveal the first entry in a chain. Required to complete the construction of a new chain.

- Post **http://localhost:8088/v1/commit-entry/?**

  Commits an entry. The first step in writing an entry to a chain.

- Post **http://localhost:8088/v1/reveal-entry/?**

  Reveal a new entry. Required to complete the writing of an entry into a chain.

- Post **http://localhost:8088/v1/factoid-submit/?**

  Submit transaction. Requires the encoded transaction as part of the call. For example, creating a transaction that sends 10 factoids from xxx to yyy might be encoded as:

  ```
  curl -X GET http://localhost:8088/v1/factoid-submit/httpp02015023e2886901010083ddb4b3006302ac3d
  a1a1e5eac31af88cdbb886f34470cc0415d1968d8637814cfac482f283dceb940025edb8b25808b6e6d
  48ad5ba67d0843eaf962c40f63c9b4df91b8fe7364ae872014b776d236585f2ed658ec9d24a4a65e08e
  f6074573f570b8b25a9d424b1d955d2caaa4d2cfe30eb8217844f8b28b8a47ce6dc3e5eecd03f30954c
  a3f0b64a63e0687f667bc3300bb33a0638953d442db2cd6fb4d27045318ec09463542c66305
  ```

  That seems like a pretty complex construction of data. Most users will use fctwallet to construct this call.

- Get **http://localhost:8088/v1/directory-block-head/?**

  Returns the hash of the directory block head. No parameters are needed. Returns a JSON string of the form:

  ```
  {"KeyMR":"f7eb0456b30b1a4b50867a5307532e92ddee7279ffc955ce1284cd142f94d642"}
  ```

- Get **http://localhost:8088/v1/directory-block-height/?**

  Returns the current directory block height.

  ```
  curl -X GET http://localhost:8088/v1/directory-block-height/
  ```

  Returned at the time of writing:

  ```
  {"Height":4585}
  ```

- Get **http://localhost:8088/v1/get-raw-data/([^/]+)**

  Returns the block assoicated with the given hash.

  ```
  curl -X GET http://localhost:8088/v1/get-raw-data/f7eb0456b30b1a4b50867a5307532e92ddee7279ffc955ce1284cd142f94d642
  ```

  returns:

  ```
  {"Data":"00fa92e5a291592f5f78c547560edceb8bc5ef142f20e9689fcd587557a2f3d18406d6e5ece9eacaa1c31d1371af60d6a9d5ea65654d1ff5698f7fb181d0ae4bc8582c093186dd2a14e83bbf53bb7cab230b1d0e2ce
  ```

  This data can be unmarshalled into the directory block struct used by Factom.

- Get **http://localhost:8088/v1/directory-block-by-keymr/([^/]+)**

  Returns the directory block assoicated with the given hash.

  ```
  curl -X GET http://localhost:8088/v1/directory-block-by-keymr/f7eb0456b30b1a4b50867a5307532e92ddee7279ffc955ce1284cd142f94d642
  ```

  returns:

  ```
  {"Header":{
     "PrevBlockKeyMR":"e9eacaa1c31d1371af60d6a9d5ea65654d1ff5698f7fb181d0ae4bc8582c0931",
     "SequenceNumber":4289,
     "Timestamp":1443711000},
     "EntryBlockList":[
        {"ChainID":"000000000000000000000000000000000000000000000000000000000000000a",
         "KeyMR":"98f7817976ed8ff9aa306834d98c145d7c0334d7057f89dd2f035df1b37946ae"},
        {"ChainID":"000000000000000000000000000000000000000000000000000000000000000c",
         "KeyMR":"9432448e6c7f56450804b42ed9c1653182efb6f48a5d8da2c22d1789e7dbff44"},
        {"ChainID":"000000000000000000000000000000000000000000000000000000000000000f",
         "KeyMR":"b642daa292af42dda109bc87cddd31647da6fef9f3f25129c3740ef4d72761a0"},
        {"ChainID":"df3ade9eec4b08d5379cc64270c30ea7315d8a8a1a69efe2b98a60ecdd69e604",
         "KeyMR":"789b0103e5f8358d7f8402264837986a2b29ac59be8a796dbbe75eecf6a853d9"}
    ]
  }
  ```

  This call returns the data held in a Directory Block digested into a JSON structure.

- Get **http://localhost:8088/v1/entry-block-by-keymr/([^/]+)**

  Returns an Entry Block structure. The call:

  ```
  curl -X GET http://localhost:8088/v1/entry-block-by-keymr/789b0103e5f8358d7f8402264837986a2b29ac59be8a796dbbe75eecf6a853d9
  ```

  Returns

```
{
  "Header":{
    "BlockSequenceNumber":2479,
    "ChainID":"df3ade9eec4b08d5379cc64270c30ea7315d8a8a1a69efe2b98a60ecdd69e604",
    "PrevKeyMR":"63833701a61a846ebe8d38a1c6ede6bf5d5516990c34372c7f7936812ec09bde",
    "Timestamp":1443711000
  },
  "EntryList":[
      { "EntryHash":"c8f4936962836cda0d8bf712653d97f8d8b5cbe675e495b6dfab6b2395c8b80a",
        "Timestamp":1443711360
      }
  ]
}
```

This is the structure of an Entry block, broken out into JSON.

- Get **http://localhost:8088/v1/entry-by-hash/([^/]+)", handleEntry)**

  Returns an Entry broken out into JSON. The following call:

  ```
  curl -X GET http://localhost:8088/v1/entry-by-hash/c8f4936962836cda0d8bf712653d97f8d8b5cbe675e495b6dfab6b2395c8b80a
  ```

  Returns:

  ```
  {"ChainID":"df3ade9eec4b08d5379cc64270c30ea7315d8a8a1a69efe2b98a60ecdd69e604",
  "Content":"7b22416e63686f725265636f7264566572223a312c22444248656967687422 3a343238382c224b65794d52223a226539656163616131633331643133373161663630643661396435656136353635346431666635
  ```

  Returns a particiular Entry's construction broken out into JSON.

- Get **http://localhost:8088/v1/chain-head/([^/]+)**

  Returns the KeyMR of the first Entry in an Entry Chain. The call:

  ```
  curl -X GET http://localhost:8088/v1/chain-head/df3ade9eec4b08d5379cc64270c30ea7315d8a8a1a69efe2b98a60ecdd69e604
  ```

  Returns

  ```
  {"ChainHead":"bfd814a3b9a4356e04c816fe4ce1a53198953ab321912d60dacba766950e5591"}
  ```

- Get **http://localhost:8088/v1/entry-credit-balance/([^/]+)**

  Returns the balance at the given Entry Credit address. For example, the call:

  ```
  curl -X GET http://localhost:8088/v1/entry-credit-balance/748be8327d20fee4365e6b5a3dca7df1e59da47e9ebd99129ba84d58d4d0726b
  ```

  Might return (depending on the balance at that address at the time):

  ```
  {"Response":"4000","Success":true}
  ```

  This would indicate that the decoded Entry Credit address (EC2eUoDPupuQXm5gxs1sCBCv3bbZBCYFDTjaFQ6iRaAKfyXNqjEJ) decodes to the hex: 748be8327d20fee4365e6b5a3dca7df1e59da47e9ebd99129ba84d58d4d0726b and has a balance of 4000 entry credits.

- Get **http://localhost:8088/v1/factoid-balance/([^/]+)**

  Returns the Factoid balance at the given address. For example, the call:

```
curl -X GET http://localhost:8088/v1/factoid-balance/f6e117ea838cb652e9cfc3b29552d5887800a7ba614df0bd8c13e171eddc5897
```

Returns:

```
{"Response":"1210268000","Success":true}
```

Note that, like Bitcoin, Factoids use fixed point to indicate parts of a coin. so 12.10268000 represents 12.10268 factoids.

- Get **http://localhost:8088/v1/factoid-get-fee/**

  Returns the current exchange rate for Entry Credits. So the call:

  ```
  curl -X GET http://localhost:8088/v1/factoid-get-fee/
  ```

  might return

  ```
  {"Fee":100000}
  ```

  indicating that .001 Factoids will purchase 1 Entry Credit.

- Get **http://localhost:8088/v1/properties/",handleProperties)**

  Returns the version numbers of various components of Factom. For example at the time of writing, the call:

  ```
  curl -X GET http://localhost:8088/v1/properties/
  ```

  Returns:

  ```
  {
    "Protocol_Version":1005,
    "Factomd_Version":3002,
    "Fctwallet_Version":0
  }
  ```

fctwallet

- Get **http://localhost:8089/v1/factoid-balance/([^/]+)**

  Return the factoid balance at the given Factoid address. The call can take an address name known by your wallet, a Factoid address, or a hex representation of the address (less base 58 and checksums).

  For example, for a given wallet, the following calls:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-balance/FAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  curl -X GET "http://localhost:8089/v1/factoid-balance/9e72fa1dbdac30b557c857a1dcdca04b4ae748e52dc492e1f85f6af6f29f6534"
  curl -X GET "http://localhost:8089/v1/factoid-balance/FactomAddress01"
  ```

  Will return:

  ```
  {"Response":"1210680000","Success":true}
  ```

  Should all retrieve the same balance from the same address, assuming that your address book had an entry FactomAddress01 with the private key for FA3ArvkijVcgrFVj45PBgGBfWm1MWAEjV1SbVxSFiUNT6s9F7AQb.

- Get **http://localhost:8089/v1/entry-credit-balance/([^/]+)**

  Return the Entry Credit balance for the specified address. The call can take an address name known by your wallet, an Entry Credit address, or a hex representation of the address (less base 58 and checksum).

For example, for a given wallet and Entry Credit address, the calls:

```
curl -X GET "http://localhost:8089/v1/entry-credit-balance/ECxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
curl -X GET "http://localhost:8089/v1/entry-credit-balance/748be8327d20fee4365e6b5a3dca7df1e59da47e9ebd99129ba84d58d4d0726b"
curl -X GET "http://localhost:8090/EntryCreditAddress001"
```

Will Return

```
{"Response":"4000","Success":true}
```

Assuming that your wallet had an entry EntryCreditAddress001 with the private key for the given public address.

- Get **http://localhost:8089/v1/factoid-generate-address/([^/]+)**

  Generate an address, and create an entry in your wallet to hold said address. Addresses are created from a deterministic hash, so if you back up your wallet, then your wallet can be restored even if some of the addresses were created after the backup.

  The call:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-generate-address/fctAddress0001"
  ```

  will create an address fctAddress0001, and assign it a new private key.

- Get **http://localhost:8089/v1/factoid-generate-ec-address/([^/]+)**

  Generate an Entry Credit address, and create an entry in your wallet to hold said address. Addresses are created from a deterministic hash, so if you back up your wallet, then your wallet can be restored even if some of the addresses were created after the backup.

  The call:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-generate-ec-address/ECAddress0001"
  ```

  will create an address ECAddress0001, and assign it a new private key.

- Get **http://localhost:8089/v1/factoid-generate-address-from-private-key/(.*)**

  This call is used to import a factoid private key in hex from another source. Provided a private key and a name. For example:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-generate-address-from-private-key/?name=addr01&privateKey=85d6755c286c6f139b1696ca74b0c14da473beadc37b2ec6273f2a92ce8d7c88"
  ```

  would import the given private key, and store it in the wallet under addr001 and return the public key. Note that importing private keys in this fashion requires a fresh backup of the wallet for safety.

- Get **http://localhost:8089/v1/factoid-generate-ec-address-from-private-key/(.*)**

  This call is used to import an entry credit private key in hex from another source. Provided a private key and a name. For example:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-generate-ec-address-from-private-key/?name=addr001&privateKey=3ffa892f2445286a06c0dc591d7fa557d16701e44ec1cbee2930f7d7dfb62d57"
  ```

  would import the given private key, and store it in the wallet under addr001 and return the public key. Note that importing private keys in this fashion requires a fresh backup of the wallet for safety.

- Get **http://localhost:8089/v1/factoid-generate-address-from-human-readable-private-key/(.*)**

  This call is used to import a factoid private key in human readable form from another source. Provided a private key and a name. For example:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-generate-address-from-human-readable-private-key/?name=addr001&privateKey=Fsxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  ```

  would import the given private key, and store it in the wallet under addr001 and return the public key. Note that importing private keys in this fashion requires a fresh backup of the wallet for safety.

- Get **http://localhost:8089/v1/factoid-generate-ec-address-from-human-readable-private-key/(.*)**

  This call is used to import an Entry Credit private key in human readable form from another source. Provided a private key and a name. For example:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-generate-address-from-human-readable-private-key/?name=addr001&privateKey=Esxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  ```

  would import the given private key, and store it in the wallet under addr001 and return the public key. Note that importing private keys in this fashion requires a fresh backup of the wallet for safety.

- Get **http://localhost:8089/v1/factoid-generate-address-from-token-sale/(.*)**

  Accepts the 12 words provided by Koinify during the crowd sale, and generates the corresponding entry in the wallet. For example:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-generate-address-from-token-sale/?name=koinifyAddr&mnemonic=<12 words separated by %20>"
  ```

  Returns the public key

- Post **http://localhost:8089/v1/factoid-new-transaction/([^/]+)**

  Creates a new transaction, and assoicates that transaction with a key. This key is used in other operations to add inputs, add outputs, add entry credit outputs, pay the fee, sign the transaction, and submit it. Example:

  ```
  curl -X POST "http://localhost:8089/v1/factoid-new-transaction/trans"
  ```

  Response

  ```
  {"Response":"Success building a transaction","Success":true}
  ```

  Which creates a transaction named 'trans'. We will use this transaction in the following commands.

- Post **http://localhost:8089/v1/factoid-delete-transaction/([^/]+)**

  Delete the specified transaction under construction by name.

  ```
  curl -X POST "http://localhost:8089/v1/factoid-delete-transaction/trans"
  ```

  Response

  ```
  {"Response":"Success deleting transaction","Success":true}
  ```

  Removes the transaction 'trans'. To continue to build a transaction named trans, you would need to recreate 'trans'.

- Post **http://localhost:8089/v1/factoid-add-fee/(.*)**

  Add the needed fee to the given transaction. This call calculates the needed fee, and adds it to the specified input. The inputs and outputs must be exactly balanced, because this call isn't going to mess with unbalanced transactions as how to balance can be tricky.

  ```
  curl -X POST http://localhost:8089/v1/factoid-add-fee/ -d "key=trans&name=FAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  ```

  Response

  ```
  {"Response":"Added 0.153318 to FAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx","Success":true}
  ```

  Assuming the given Factoid address is an input to trans, this adds the fee to that address.

- Post **http://localhost:8089/v1/factoid-add-input/(.*)**

  Add the given input to the transaction specified.

```
curl -X POST http://localhost:8089/v1/factoid-add-input/ -d "key=trans&name=FAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&amount=10000000"
```

Response

```
{"Response":"Success adding Input","Success":true}
```

Adds an input from the given address to the transaction trans. The number of factoids (12) will be presented in fixpoint notation, i.e. (1200000000)

- Post **http://localhost:8089/v1/factoid-add-output/(.*)**

  Add the given output to the transaction specified.

  ```
  curl -X POST http://localhost:8089/v1/factoid-add-output/ -d "key=trans&name=FAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&amount=10000000"
  ```

  Response

  ```
  {"Response":"Success adding output","Success":true}
  ```

  Adds an output to the given address to the transaction trans. The number of factoids (13) will be presented in fixpoint notation, i.e. (1300000000)

- Post **http://localhost:8089/v1/factoid-add-ecoutput/(.*)**

  Add the given Entry Credit Output to the transaction specified. Note that Entry Credit Outputs are denominated in Factoids. How many Entry Credits are alloted depends upon the exchage rate of factoids to entry credits in place at the time of the transaction. For example:

  ```
  curl -X POST http://localhost:8089/v1/factoid-add-ecoutput/ -d "key=trans&name=ECxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&amount=10000000"
  ```

  Response

  ```
  {"Response":"Success adding Entry Credit Output","Success":true}
  ```

  Adds an ecoutput to the given entry credit address to the transaction trans. Assume a factoid to Entry Credit exchange rate of .001. Then the number of Entry Credits (1000) will be determined by the factoids in the output (1) divided by the factoid to entry credit rate (.001). The factoids converted to entry credits will be presented in fixpoint notation, i.e. (100000000 == 1 factoid)

- Post **http://localhost:8089/v1/factoid-sign-transaction/(.*)**

  Sign the given transaction.

  ```
  curl -X POST "http://localhost:8089/v1/factoid-sign-transaction/trans"
  ```

  Response

  ```
  {"Response":"Success signing transaction","Success":true}
  ```

  Signs the transaction 'trans'.

- Post **http://localhost:8089/v1/compose-submit-chain/([^/]+)**

Create a JSON object that may be used in the factomd calls to commit-chain and reveal-chain

$ curl -X POST -H 'Content-Type: application/json' -d '{"ExtIDs":["foo", "bar"], "Content":"Hello Factom!"}' localhost:8089/v1/compose-chain-submit/app

Returns

{"ChainID":"92475004e70f41b94750f4a77bf7b430551113b25d3d57169eadca5692bb043d","ChainCommit":
{"CommitChainMsg":"0001521deb5c7891ac03adffe815c64088dc98ef281de1891c0f99a63c55369c1727dc73580cbcc309ee55fa780ce406722b7a074138c994c859e2eda619bbad59b41775b51176464cb77fc08b6ef6767dcc315b4729a871071053cfe4af5a6

{"Entry":"0092475004e70f41b94750f4a77bf7b430551113b25d3d57169eadca5692bb043d000a0003666f6f000362617248656c6c6f20466163746f6d21"}}

- Post **http://localhost:8089/v1/compose-submit-entry/([^/]+)**

Create a JSON object that may be used in the factomd calls to commit-entry and reveal-entry

$ curl -i -X POST -H 'Content-Type: application/json' -d '{"ChainID":"5c337e9010600c415d2cd259ed0bf904e35666483277664d869a98189b35ca81", "ExtIDs":["foo", "bar"], "Content":"Hello Factom!"}' localhost:8089/v1/compose-entry-submit/app

Returns

{"EntryCommit":
{"CommitEntryMsg":"0001521dc2d47d32cbdd3fc21889e22cc408ae0b0c120662c0873331cc5ce8ebdc1b6722968ce20179a1ad273d890287e5d4f16d2669c06c523b9e48673de1bfde3ea2fda309ac92f4f4b4d52cc6b228b9b621b1b1969ab46bfa4f80379e14d
{"Entry":"005c337e9010600c415d2cd259ed0bf904e35666483277664d869a98189b35ca81000a0003666f6f000362617248656c6c6f20466163746f6d21"}}

- Post **http://localhost:8089/v1/commit-chain/([^/]+)**

  Sign a binary Chain Commit with the specified entry credit key and submit it to the factomd server

- Post **http://localhost:8089/v1/commit-entry/([^/]+)**

  Commit an entry to an Entry Chain

- Post **http://localhost:8089/v1/factoid-submit/(.*)**

  Submit a transaction to Factom. This call takes a named JSON parameter. For example, to submit a transaction named trans, you need the following call:

  ```
  curl -X POST http://localhost:8089/v1/factoid-submit/\\{\"Transaction\":\"trans\"\\}
  ```

  Response

  ```
  {"Response":"Success Submitting transaction","Success":true}
  ```

- Get **http://localhost:8089/v1/factoid-validate/(.*)**

  Not currently implemented.

- Get **http://localhost:8089"/v1/factoid-get-fee/(.*)**

  Get the current exchange rate in number of Factoids per Entry Credit For example:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-get-fee/"
  ```

  Response

  ```
  {"Response":"0.006666","Success":true}
  ```

- Get **http://localhost:8089"/v1/properties/**

  Get the version numbers of all the components of the Factom client, fctwallet, factomd, and the protocol For example:

  ```
  curl -X GET "http://localhost:8089/v1/properties/"
  ```

  Response

  ```
  {"Response":"Protocol Version: 0.1.5\nfactomd Version: 0.3.4\nfctwallet Version: 0.1.4\n","Success":true}
  ```

- Get **http://localhost:8089/v1/factoid-get-addresses/**

Get the address list held in the wallet For example:

```
curl -X GET "http://localhost:8089/v1/factoid-get-addresses/"
```

- Get **http://localhost:8089/v1/factoid-get-transactions/**

  Get all the transactions currently under construction, along with the key used to reference them. For example:

  ```
  curl -X GET "http://localhost:8089/v1/factoid-get-transactions/"
  ```

- Post **http://localhost:8089/v1/factoid-get-processed-transactions/(.*)**

  If pass in 'all' then all transactions are returned. If an address, then all the transactions that use the address as an input, output, or entry credit output will be returned. The transactions are returned as text. For example:

  ```
  curl -X POST http://localhost:8089/v1/factoid-get-processed-transactions/ -d "address=FAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  curl -X POST http://localhost:8089/v1/factoid-get-processed-transactions/ -d "address=<addrname>"
  curl -X POST http://localhost:8089/v1/factoid-get-processed-transactions/ -d "cmd=all"
  ```

- Post **http://localhost:8089/v1/factoid-get-processed-transactionsj/(.*)**

  If pass in 'all' then all transactions are returned. If an address, then all the transactions that use the address as an input, output, or entry credit output will be returned. The transactions are returned as an array of JSON objects. For example:

  ```
  curl -X POST http://localhost:8089/v1/factoid-get-processed-transactionsj/ -d "address=FAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
   curl -X POST http://localhost:8089/v1/factoid-get-processed-transactionsj/ -d "address=<addrname>"
   curl -X POST http://localhost:8089/v1/factoid-get-processed-transactionsj/ -d "cmd=all"
  ```