BFTX JSON Validation

* It needs a JSON example to test BFTX JSON validator.

**Done, but it would be great a real example.**

**JSON** as input. The design is according to the sample XML. (Navia)

* It is necessary to modify “notify\_address” instead of “nofity\_address”.

**Done.**

* Should all number fields be floats or integers?
  + BOL\_NUM
  + REF\_NUM
  + vessel
  + port\_of\_loading
  + port\_of\_discharge
  + gross\_weight (This could be float)
  + freight\_payable\_amt
  + freight\_adv\_amt
  + date\_shipped[type]
  + Num\_bol
* That structure has some null fields, so If these fields are always null, do we need to leave them in the JSON structure? Yes, but it’s necessary to check what is the type of the value.
  + Consignee
  + sig for master\_info, agent\_for\_master and agent\_for\_owner
* It is necessary to research each field, create a specification, restrictions (required, optional, not null, conditional, minimum and maximum characters, etc) per field. Define what is valid data and describe what each field is for. It will Solve with the API documentation in HTML format.
* I had to change the name of field “issue\_details[properties]” with “issue\_details[issue\_details\_properties]” because there are repeated properties in other fields. [Check the way is coded, in nested strcture, ir shouldn’t be a problem]
* I had to change the name of field “master\_info[properties]” with “master\_info[master\_info\_properties]” because there are repeated properties in other fields. [Check the way is coded, in nested strcture, ir shouldn’t be a problem]
* I had to change the name of field “agent\_for\_master[properties]” with “agent\_for\_master[agent\_for\_master\_properties]” because there are repeated properties in other fields. [Check the way is coded, in nested strcture, ir shouldn’t be a problem]
* I had to change the name of field “agent\_for\_owner[properties]” with “agent\_for\_owner[agent\_for\_owner\_properties]” because there are repeated properties in other fields. [Check the way is coded, in nested strcture, ir shouldn’t be a problem]
* Which object is gonna be received in the first element on BFTX JSON? ("type": "object") And also in issue\_details, master\_info, agent\_for\_master and agent\_for\_owner fields. It’s part of general JSON structure.
* BFTX JSON Structure has BOL\_NUM and num\_BOL and both elements are numeric, are these different elements? We don’t know now.
* Do we have to validate lowercase, uppercase, etc?
  + Keys: It’s case sensitive.
  + Values: a-z A-Z both validate and case preserved. Case not validated.
* Some values may have spaces between words, will BFTX JSON structure allow spaces just for these values? Or for any part in the structure? Yes, but it’s necessary to check if we can receive more than one space.
* For all format fields after any date field, the structure has a string as value, what will be all set of posibilities for date-time format? Unix date formats or Japanese ISO YYYY-MM-DD T12:00:00+10GMT or Bitcoin Block Height + Hash Root Value
* Do we need to save the complete JSON in Database? O each value in one database field or column? Blockchain is like a log file : it keeps a ledger, a record and an archival quality data set. In this regards – it saves complete JSON converted with Google Protobuff down to a line level protocol – and requires SQL-less persistent storage – like a key-value pair. [some pegging of JSON origin structure being required for reverse generation (decompression up into emittabel JSON under search and transaction conditions) However as a protocol the storage mechanish in application will vary amount client and implemeantations. <specifically we provide for the lite client (Mobile) full nodes [Docker distributed developer choice of DB (Redis/LevelDB/MyNewMgic Best Keystore DB – Apache X project whatever] vs some SaaS platforms – even if opearrated by Blockfreight, Inc. sauch as [https://aws.amazon.com/nosql/**key**-**value**/](https://aws.amazon.com/nosql/key-value/) (AWS)
* Where will be the source of the BFTX JSON structure? In a file or Database.

Bugs

* Check the nth element when there are repeated elements. (format, first\_name, last\_name, sig, etc)