README

Data Format

The training datasets contains 4 files, each with 1M rows using the following format (separated by the tab character):

Timestamp Node ID Ref-ID User Activity Resource

There are 4 types of activities (may be different in the test data):

REQ_RESOURCE, VIEW_RESOURCE, FILE_ACCESS, RISK_ASSESSMENT

Also, there are 11 types of resources (may be different in the test data):

GTEx, TOPMed,
MOD_FlyBase, MOD_GMOD, MOD_Gene_Ontology_Consortium,
MOD_MGI, MOD_Reactome, MOD_SGD,
MOD_UCSC_Genome_Bioinformatics, MOD_UniProt_KnowledgeBase, MOD_WormBase

An example use case for the log is as follows. In a node N at time T1, a user U first requests (REQ_RESOURCE) the resource MOD_MGI, and the ID of the record is X. Also, the Ref-ID for this record is X. Then at T2, this user U later views (VIEW_RESOURCE) the same resource, under the same request. This view record will have a different ID (e.g., Y), and the Ref-ID would still be X (because the view activity still refers to the original request). Finally at time T3, the user U accessed the file (FILE_ACCESS) of the same resource for the same request. Again the access record will have a new ID (e.g., Z), with the same Ref-ID (i.e., X). The log would look like:

```
...
T1   N   X   X   U   REQ_RESOURCE    MOD_MGI
…
T2   N   Y   X   U   VIEW_RESOURCE   MOD_MGI
…
T3   N   Z   X   U   FILE_ACCESS      MOD_MGI
…
```

The testing data will be in the same format, and may contain a different number of records (but in a similar scale) or different values for the activities and resources.

Additional Rules of Solution

The logging solution should be designed in such a way as to be optimized for reading and storing live changes to a log system, instead of compressing and handling a million entries at once. Therefore, the solution may be tested by sending batches of lines (e.g. 1000 lines, or

even 1 line) at a time. To simulate a real world logging solution, any sort of buffering method (which may lead to a loss of data) is not allowed.

Also, the solution should be able to store the data on one node of the blockchain network and retrieve on any other node, without any other method of communication between machines. The solution should be symmetric (i.e., identical software for each node), and no other computing/storage resources will be available, only the virtual machines provided for testing. Furthermore, the solution should be standalone, such that upon installation it should be able to read and write to an existing chain without any external setup or configuration, aside from telling it what chain to use.