Datagen Hyperledger Block Process

Context and Problem Statement

The Duty Drawback application integrates with a hyperledger block chain provided by Dubai customs. As part of that integration, events must be processed in the order they are created. The Datagen system must consume and verify all events. If an event is of business value, the content should be further processed by the trans comm system.

This operation introduces the following challenges:

- For events to be received by datagen, subscriptions must be established requesting the correct starting point (block number)
- Block numbers are not limited to one transaction, multiple events can share the same block number.
- Datagen subscribes to multiple different events, each with their own block number.
- All relevant events must be consumed and communicated within the system.
- Many events maybe considered not relevant, but must reliably be determined as such.
    - This verification can be accomplished by known data points, which in some corner cases can be subject to race conditions, for example. the speed in which data is stored vs the time to receive an event.

Datagen Subscriptions

When establishing a subscription between Datagen and Hyperledger, a request is made from Datagen to Hyperledger Gateway, requesting a subscription for a given contract or block event. Within Transcomm we subscribe to the following events:

- Contract - CLAIM_STATUS_CHANGE
- Contract - DECLARATION_STATUS_CHANGE
- Contract - ChainCode

an example of such a request is as follows:

```
var data = qs.stringify({
  'channelName': 'dhlecomchannel',
  'chaincodeName': 'businesscontract',
  'eventCategory': 'Contract',
  'eventName': 'CLAIM_STATUS_CHANGE',
  'eventType': 'full',
  'startblock': '0',
  'callbackURL': 'http://20.107.203.150/datagen/hl-events/contract'
});
```

One key component is the property, `startblock`. The `startblock` instructs the subscription where to begin. In the example, `0` is the starting block, which would ensure our subscription will consume all events from the start indefinitely. If we provide a `endblock` the subscription would only last until that block is

reached. A risk with a request with no `endblock` is we simply do not know how many events we will receive once the subscription is established, this can be mitigated via the use of the `endblock` but as previously stated, if we leverage this approach our subscription has a fixed time to live. Therefore introducing the need to monitor and re-subscribe continuously.

Hyperledger Blocks

When events are received from the hyperledger, they contain a block number. Events can share block numbers so long as they were committed to the blockchain within the same time frame. The impact of this is, if we consume an event with the block number `15` we cannot reliably determine we've consumed all events for block `15`. If an issue were to occur (such as the datagen crashes) we *should* not start consuming from our highest known block number, but rather our second highest known.

Events relevancy

Events coming into the datagen should only be processed if they contain a known Transaction Id (TxId) or known hyperledger key. If considered irrelevant, the event can be skipped.

Currently, if an event is deemed relevant, the event is recorded and stored. This record servers an additional purpose, the block number for that record is used as input for the subscriptions at creation. This is particularly important in moments where the server is re-started. When the server starts its highest known block is based on these records which is the input for the subscriptions. A side effect of this however is, if all events consumed are not relevant, they will be processed multiple times up until a relevant process is received. This is because irrelevant events and their blocks are not currently being stored.

Decision Drivers

- Prevention of possible data loss.
- Low impedance on performance.
- high resiliency.

Decision Outcome

- All events that are processed are recorded with their given contract event.
- Subscription block numbers are stored and determined independently (no longer shared block number).
- If an event fails to process, the service is interrupted and other events are not processed until the problem is resolved.
- Subscriptions all start from the second highest known block in an attempt to mitigate missing any events.

Positive Consequences

- Subscriptions have more control over which block is the starting point.
- Low risk of events being skipped due to a failure in processing.
- No added complexity.

Negative Consequences

- Processing times can be slow*
- If a logjam occurs, a resolution is required else there is a deadlock.

Process times can be slow

As mentioned in the context, there exists a condition where our system cannot store the txId of a given transaction quicker than the event for that transaction is received. To mitigate this, there is a retry/backoff period to verify has txId been created (configurable). By default this is three attempts one second apart. However, if the system consumes multiple irrelevant events, the system is forced to verify in that period if the transaction may become relevant.

Other mitigate options considered:

- Elevate a timestamp for the event. Sadly, the events from hyperledger do not appear to contain a timestamp making this option not possible.
- Configure backoff and retry to be lower - this does introduce risk of skipping when shouldn't occur.
- If known prior that many blocks aren't relevant, preconfigure the blocks to skip.