Sawtooth Simple Supply Application Specification

Overview

The Sawtooth Simple Supply application is a simplified version of Sawtooth Supply Chain. It allows users to track the provenance and location of goods as they move through a supply chain.

State

All Simple Supply objects are serialized using Protocol Buffers before being stored in state. These objects are Agents and Records. As described in the Addressing section below, these objects are stored in separate sub-namespaces under the Simple Supply namespace. To handle hash collisions, all objects are stored in lists within protobuf “Container” objects.

Records

Records represent the goods being tracked by Simple Supply. Almost every transaction references some Record. A Record contains a unique identifier and lists containing the history of its owners and locations.

```
message Record {
    message Owner {
```

```
        // Public key of the agent who owns the record
        string agent_id = 1;

        // Approximately when the owner was updated, as a Unix UTC timestamp
        uint64 timestamp = 2;
    }

    message Location {
        // Coordinates are expected to be in millionths of a degree
        sint64 latitude = 1;
        sint64 longitude = 2;

        // Approximately when the location was updated, as a Unix UTC timestamp
        uint64 timestamp = 3;
    }

    // The user-defined natural key which identifies the object in the
    // real world (for example a serial number)
    string record_id = 1;

    // Ordered oldest to newest by timestamp
    repeated Owner owners = 2;
    repeated Location locations = 3;
}
```

Records whose addresses collide are stored in a list alphabetically by identifier.

```
message RecordContainer {
    repeated Record entries = 1;
}
```

Agents

Agents are entities that can own, transfer, and update Records.

```
message Agent {
    // The agent's unique public key
    string public_key = 1;

    // A human-readable name identifying the agent
    string name = 2;

    // Approximately when the Agent was registered, as a Unix UTC timestamp
```

```
    uint64 timestamp = 3;
}
```

Agents whose keys have the same hash are stored in a list alphabetized by public key.

```
message AgentContainer {
    repeated Agent entries = 1;
}
```

Addressing

Simple Supply objects are stored under the namespace obtained by taking the first six characters of the SHA-512 hash of the string `simple_supply`:

```
>>> hashlib.sha512('simple_supply'.encode('utf-8')).hexdigest()[:6]
'212bd8'
```

After its namespace prefix, the next two characters of a Simple Supply object's address are a string based on the object's type:

- Agent: `00`
- Record: `01`

The remaining 62 characters are determined by its type:

- Agent: The first 62 characters of the hash of its public key
- Record: the first 62 characters of the hash of its identifier

Transactions

Transaction Payload

All Simple Supply transactions are wrapped in a tagged payload object to allow for the transaction to be dispatched to the appropriate handling logic.

```
message SimpleSupplyPayload{
    enum Action {
```

```
        CREATE_AGENT = 0;
        CREATE_RECORD = 1;
        UPDATE_RECORD = 2;
        TRANSFER_RECORD = 3;
    }

    // Whether the payload contains a create agent, create record,
    // update record, or transfer record action
    Action action = 1;

    // The transaction handler will read from just one of these fields
    // according to the action
    CreateAgentAction create_agent = 2;
    CreateRecordAction create_record = 3;
    UpdateRecordAction update_record = 4;
    TransferRecordAction transfer_record = 5;

    // Approximately when transaction was submitted, as a Unix UTC timestamp
    uint64 timestamp = 6;
}
```

Any transaction is invalid if its timestamp is greater than the validator's system time.

Create Agent

Create an Agent that can interact with Records. The `signer_pubkey` in the transaction header is used as the Agent's public key.

```
message CreateAgentAction {
    // A human-readable name identifying the new agent
    string name = 1;
}
```

A CreateAgentAction transaction is invalid if there is already an agent with the signer's public key or if the name is the empty string.

Create Record

When an agent creates a record, the record is initialized with that agent as the owner.

```
message CreateRecordAction {
    // The user-defined natural key which identifies the object in the
    // real world (for example a serial number)
```

```
    string record_id = 1;

    // Coordinates are expected to be in millionths of a degree
    sint64 latitude = 2;
    sint64 longitude = 3;
}
```

A CreateRecordAction transaction is invalid if one of the following conditions occurs:

- The signer is not registered as an agent
- The identifier is the empty string
- The identifier belongs to an existing record
- The latitude and longitude are not provided and valid

Update Record

An UpdateRecordAction transaction contains a `record_id`, a `latitude`, and a `longitude`. It can only be (validly) sent by an agent who is the owner of the record.

```
message UpdateRecordAction {
    // The id of the record being updated
    string record_id = 1;

    // Coordinates are expected to be in millionths of a degree
    sint64 latitude = 2;
    sint64 longitude = 3;
}
```

An UpdateRecord transaction is invalid if one of the following conditions occurs:

- The record does not exist
- The signer is not the owner of the record
- The latitude and longitude are not provided and valid

Transfer Record

A TransferRecordAction transaction transfers the ownership of a record from one agent to another. It can only be (validly) sent by an agent who is the owner of the record.

```
message TransferRecordAction {
    // The id of the record for the ownership transfer
```

```
    string record_id = 1;

    // The public key of the agent to which the record will be transferred
    string receiving_agent = 2;
}
```

A TransferRecordAction transaction is invalid if one of the following conditions occurs:

- The signer is not the owner of the record
- The record does not exist
- The receiving agent does not exist