Sawtooth Bond 0.1 Application Specification

Overview

The Bond application provides a bond trading platform for Organizations to issue, buy, and sell bonds. The application also offers functionality to redeem coupon payments at regular intervals as well as bond redemption at the bond's maturity.

State

All Bond objects are serialized using Protocol Buffers in order to be stored inside of state. All of these objects are stored in separate, object-specific namespaces under the Bond namespace. In the case of hash collisions, any colliding objects are stored within lists in protobuf "Container" objects. All bond objects have an accompanying "Container" message beside the individual object message.

Bond

Bond objects contain all pertinent attributes for the functionality of this application as well as displaying detailed information on the bond itself. Organizations have holdings which represent stores of bonds, and most other transactions reference the issuing, buying, and selling of bonds.

```
message Bond {
    enum CouponType {
        FIXED = 0;
        FLOATING = 1;
    }

    enum CouponFrequency {
       QUARTERLY = 0;
       MONTHLY = 1;
       DAILY = 2;
    }

    message CorporateDebtRating {
       string agency = 1;
       string rating = 2;
    }

    // Identifier of the bond (ISIN).
    string bond_id = 1;

    // Ticker symbol of the organization which issued the bond.
    string issuing_organization_id = 2;

    // List of bond ratings by agency.
```

```
    repeated CorporateDebtRating corporate_debt_ratings = 3;

    // The yearly rate of interest paid in coupons.
    // Format: Thousandths of percents
    uint64 coupon_rate = 4;

    // If floating, the interest rate is calculated by adding coupon-rate
    // to the rate looked up using the benchmark index (Libor).
    CouponType coupon_type = 5;

    // How often the coupons are paid out.
    CouponFrequency coupon_frequency = 6;

    // Date the first coupon can be paid for the bond.
    // Format: UTC Timestamp
    uint64 first_coupon_date = 7;

    // Date on which the bond becomes worth it's face value and can be redeemed.
    // Format: UTC Timestamp
    uint64 maturity_date = 8;

    // Quantity of bonds.
    uint64 amount_outstanding = 9;

    // Par-value of the bond, to be paid upon maturity.
    // Format: Millionths of dollars
    uint64 face_value = 10;

    // Date on which the bond was first settled.
    // Format: UTC Timestamp
    uint64 first_settlement_date = 11;
}

message BondContainer {
    repeated Bond entries = 1;
}
```

Order

Orders display a potential action by an organization, created by participants on its behalf, that wishes to buy or sell a specific quantity of bonds. These orders are then matched with quotes in order to be settled, and handled in further transactions.

```
message Order {
    enum Action {
        BUY = 0;
        SELL = 1;
```

```
}

enum OrderType {
    MARKET = 0;
    LIMIT = 1;
}

enum Status {
    OPEN = 0;
    MATCHED = 1;
    SETTLED = 2;
}

// UUID of the order.
string order_id = 1;

// Whether this is a BUY or SELL order.
Action action = 2;

// Whether this is a MARKET or LIMIT order.
OrderType order_type = 3;

// Ticker symbol or pricing source of the
// organization the order is on behalf of.
string ordering_organization_id = 4;

// ISIN of the bond the order is for.
string bond_id = 5;

// The maximum price the ordering firm is willing to pay for a bond or
// the minimum price they are willing to sell, depending on whether
// this is a buy or sell order. If order_type is MARKET, this field should
// not be set. If order_type is LIMIT, one of limit_price or limit_yield
// must be set.
// Format: Millionths of dollars
uint64 limit_price = 6;

// The minumum yield at which the ordering firm is willing to buy a bond a
// bond or the maximum yield at which they are willing to sell, depending
// on whether this is a buy or sell order. If order-type is MARKET, this
// field should not be set. If order-type is LIMIT, one of limit-price or
// limit-yield must be set.
// Format: Thousandths of percents
uint64 limit_yield = 7;

// Quantity the firm wishes to buy or sell in USD as a multiple of the
// bond's par value.
```

```
    // Format: Millionths of dollars
    uint64 quantity = 8;

    // Quote which is matched to this order.
    string quote_id = 9;

    // OPEN means that the order has been created in the system and is
    // waiting to be matched to a quote. MATCHED means that the order has been
    // matched up with a quote which represents the highest sell price or
    // lowest buy price which matches the requested quantity. SETTLED means
    // that the counterparties had sufficient assets in their holdings to
    // settle the trade and that asset transfer has occurred.
    Status status = 10;

    // Time that the order is created in relation to the current clock.
    // Format: UTC Timestamp
    uint64 timestamp = 11;
}

message OrderContainer {
    repeated Order entries = 1;
}
```

Organization

Organizations are entities that can issue, own, buy, and sell bonds, as well as pay out and redeem bond coupons. Organizations also contain several sub-messages:

- Holding sub-message objects, which may be bonds, differentiated by their unique IDs or currency amounts in USD.
- Authorization sub-messages list specific participants by public key and an associated role. A participant's associated role ultimately determines the validity of certain transactions signed by the individual participants. While participants are the entities which submit transactions, the majority of these transactions are done on behalf of their associated organization.
- Receipt sub-messages represent either the redemption of bond coupons at regular intervals between the bond's issue date and its maturity date, or redemption of the bond itself once the maturity date has been reached.

```
message Organization {
    enum OrganizationType {
        TRADING_FIRM = 0;
        PRICING_SOURCE = 1;
    }

    message Holding {
```

```
        enum AssetType {
            CURRENCY = 0;
            BOND = 1;
        }

        // Whether the asset is CURRENCY or a BOND
        AssetType asset_type = 1;

        // If asset_type is BOND, the asset_id will contain the bond_id. If
        // asset_type is CURRENCY, this will be the string 'USD'.
        string asset_id = 2;

        // The current balance of the holding.
        // Format: Millionths of dollars, or quantity of bonds
        uint64 amount = 3;
    }

    message Authorization {
        enum Role {
            MARKETMAKER = 0;
            TRADER = 1;
        }

        // Public key of a participant.
        string public_key = 1;

        // Whether the participant is a MARKETMAKER or TRADER.
        Role role = 2;
    }

    message Receipt {
        enum PaymentType {
            COUPON = 0;
            REDEMPTION = 1;
        }

        // ID of the bond for which this payment occurred.
        string bond_id = 1;

        // Whether it was a coupon or redemption payment.
        PaymentType payment_type = 2;

        // Date the coupon became due. Required if payment_type is COUPON.
        // Format: UTC Timestamp
        uint64 coupon_date = 3;

        // Amount received.
```

```
        // Format: Millionths of dollars
        uint64 amount = 4;

        // Time that the receipt is created in relation to the current clock.
        // Format: UTC Timestamp
        uint64 timestamp = 5;
    }

    // Ticker symbol for organizations with organization type 'TRADING FIRM'
    // uniquely identifying the organization;
    // Pricing source for organizations with organization type 'PRICING SOURCE'
    // is a four-letter code representing organizations responsible for
    // providing quotes to the market.
    // Ticker symbol or pricing source provided dependent on organization type
    string organization_id = 1;

    // Name of the organization.
    string name = 2;

    // Type of the organization.
    // Determines the format of organization_id
    OrganizationType organization_type = 3;

    // Industry of the organization.
    string industry = 4;

    // List of the assets that the organization owns.
    repeated Holding holdings = 5;

    // List of participant public keys and their roles within the organization.
    repeated Authorization authorizations = 6;

    // List of coupon and redemption receipts.
    repeated Receipt receipts = 7;
}

message OrganizationContainer {
    repeated Organization entries = 1;
}
```

Participant

Participants are individual entities who may be associated with certain organizations and given certain authorizations in order to submit transactions on the organization's behalf. Participants are not able to hold bonds or currency.

```
message Participant {
```

```
    // Public key associated with the participant.
    string public_key = 1;

    // Username of the participant.
    string username = 2;

    // Ticker symbol of the participant's organization.
    string organization_id = 3;
}

message ParticipantContainer {
    repeated Participant entries = 1;
}
```

Quote

Quotes are statements sent out by organizations which state either their willingness to buy or sell a certain bond, which they may have in their holdings or wish to acquire. Quotes can remain open until the organization is unable to complete a round lot order, the sell or buy quantity is less than 100,000 bonds or until it is closed by its owning organization.

```
message Quote {
    enum Status {
        OPEN = 0;
        CLOSED = 1;
    }

    // UUID of the quote.
    string quote_id = 1;

    // Ticker symbol or pricing source of the
    // organization the quote is on behalf of.
    string organization_id = 2;

    // ID of the bond the quote is for.
    string bond_id = 3;

    // Minimum price for which the firm is willing to sell the bond.
    // Format: Millionths of dollars
    uint64 ask_price = 4;

    // Maximum quantity the firm is willing to sell.
    uint64 ask_qty = 5;

    // Maximum price for which the firm is willing to buy the bond.
    // Format: Millionths of dollars
    uint64 bid_price = 6;
```

```
    // Maximum quantity the firm is willing to buy.
    uint64 bid_qty = 7;

    // Whether the quote is OPEN or CLOSED. Quotes are OPEN if they have
    // sufficient remaining buy and sell quantities and have not been
    // explicitly cancelled. Quotes move to CLOSED status if they can no
    // longer fulfill a round-lot order (buy or sell quantity < 100000) or if
    // they are explicitly cancelled by an authorized member of the firm
    // associated with the quote.
    Status status = 8;

    // Time that the quote is created in relation to the current clock.
    // Format: UTC Timestamp
    uint64 timestamp = 9;
}

message QuoteContainer {
    repeated Quote entries = 1;
}
```

Settlement

Settlements are created when an order and quote are matched and the holdings of each organization are able to be traded. Once orders have a status of “matched,” a participant associated with the ordering organization may choose to settle that matched order and quote. Assuming each organization has the sufficient holdings to complete the transaction, the order will change to have a status of “closed” and the holdings of each organization will have been traded. Thus, signalling a successful settlement.

```
message Settlement {
    enum Action {
        BUY = 0;
        SELL = 1;
    }

    // ID of the order which is being settled.
    string order_id = 1;

    // Ticker symbol or pricing source of the organization on
    // whose behalf the order was entered.
    string ordering_organization_id = 2;

    // Ticker symbol or pricing source of the organization that
    // supplied the quote matched with the order.
    string quoting_organization_id = 3;
```

```
    // Buy or sell settlement from the perspective of the ordering organization.
    Action action = 4;

    // Quantity of bonds that are transferred.
    uint64 bond_quantity = 5;

    // Amount of cash that is transferred.
    // Format: Millionths of dollars
    uint64 currency_amount = 6;
}

message SettlementContainer {
    repeated Settlement entries = 1;
}
```

Transactions

Bond Payload

All transactions are carried in a payload object to allow for the transaction to be executed by its specific transaction handler.

```
message BondPayload{
    enum Action {
        CREATE_ORGANIZATION = 0;
        UPDATE_ORGANIZATION = 1;
        UPDATE_ORGANIZATION_AUTHORIZATION = 2;

        CREATE_PARTICIPANT = 3;
        UPDATE_PARTICIPANT = 4;

        CREATE_ORDER = 5;

        CREATE_QUOTE = 6;

        CREATE_SETTLEMENT = 7;

        CREATE_RECEIPT = 8;

        CREATE_HOLDING = 9;
        DELETE_HOLDING = 10;

        CREATE_BOND = 11;
    }

    // Type of transaction data the payload contains
    Action action = 1;

    // Payload data
    bytes content = 2;
}
```

Create Organization

The CreateOrganization transaction allows a participant to create an organization object, which enables participants to complete further transactions on behalf of the organization.

```
message CreateOrganization {

    // The name of the organization
```

```
    string name = 1;

    // Type of the organization.
    Organization.OrganizationType organization_type = 2;

    // The industry of the organization.
    string industry = 3;

    // Ticker symbol is for organizations with organization type 'TRADING FIRM'
    // uniquely identifying the organization.
    // Pricing source is for organizations with organization type 'PRICING SOURCE'
    // is a four-letter code representing organizations responsible for
    // providing quotes to the market.
    // Ticker symbol or pricing source provided dependent on organization type.
    string organization_id = 4;

    // List of participant public keys and their roles
    // within the organization.
    repeated Organization.Authorization authorizations = 5;
}
```

A CreateOrganization transaction is invalid if the one of the following occurs:

- The provided name or ID already exist
- If the organization type is PRICING SOURCE, if the ID is not a four-character string
- The name, ID, or organization type are not provided
- Invalid formatting of certain attributes, including authorizations or organization type

Update Organization

The UpdateOrganization transaction allows participants to update an organization's name or industry. The organization the participant is associated with is updated in this transaction.

```
message UpdateOrganization {

    // The name of the organization.
    string name = 1;

    // The industry of the organization.
    string industry = 4;
}
```

An UpdateOrganization transaction is invalid if the provided name are not unique or if both attributes are not provided. Additionally, the participant must have an association with an organization.

Update Organization Authorization

The UpdateOrganizationAuthorization transaction allows participants to alter the authorization list the organization holds in order to authenticate certain transactions. The organization the participant is associated with is updated in this transaction.

```
message UpdateOrganizationAuthorization {

        enum AuthorizationAction {
                ADD = 0;
                REMOVE = 1;
        }

        // Either 'add' or 'remove'
        // If set to 'add', then this is a request to add an entry to the list
        // If set to 'remove', this is a request to remove an entry
        AuthorizationAction authorization_action = 1;

        // Public key of the participant whose authorization is being altered
        string public_key = 2;

        // Role to update the specified participant entry to
        // Roles grant permissions for a participant to act on behalf of the
        // Organization in some capacity
        Organization.Authorization.Role role = 3;
}
```

An UpdateOrganizationAuthorization transaction is invalid if one of the following occurs:

- If the public key specified is not already within the organization's authorizations
- If the action is ADD, the participant is already in the organization's authorization list
- If the action is REMOVE, the participant is not in the organization's authorization list
- The role is invalid

Create Participant

The CreateParticipant transaction allows for the creation of a participant object, which is a user who is then able to create organizations and submit specific transactions on behalf of organizations.

```
message CreateParticipant {

    // The public key associated with the participant used to sign
    // transactions.
    string public_key = 1;

    // Username of the participant.
```

```
    string username = 2;

    // The ticker symbol or pricing source of the participant's
    // organization.
    string organization_id = 3;
}
```

This transaction will be invalid if if there is already a participant with the specified public key. This transaction is also invalid if the public key is not provided.

Update Participant

The UpdateParticipant transaction allows for attributes of a specified participant to be altered. The participant associated with the public key in the transaction header is updated.

```
message UpdateParticipant {

    // Username of the participant.
    string username = 1;

    // The ticker symbol or pricing source of the participant's
    // organization.
    string organization_id = 2;
}
```

An UpdateParticipant transaction is invalid if the username and organization ID are empty strings.

Create Order

The CreateOrder transaction allows for further trading actions. The organization the participant is associated with is the ordering organization. The order is given the default status of OPEN upon creation.

```
message CreateOrder {

    // Whether this is a buy / sell order
    Order.Action action = 1;

    // Whether this is a market / limit order
    Order.OrderType order_type = 2;

    // ID of the bond the order is for
    string bond_id = 3;

    // The maximum price the ordering firm is willing to pay for a bond or
```

```
    // the minimum price they are willing to sell, depending on order_type
    // If order_type is "MARKET", this field should not be set
    // If order_type is "LIMIT", limit-price or limit-yield must be set
    // Format: Millionths of dollars
    uint64 limit_price = 4;

    // The minimum yield at which the ordering firm is willing to buy a
    // bond or the maximum yield at which they are willing to sell
    // If order-type is "MARKET", this field should not be set
    // if order-type is "LIMIT", limit-price or limit-yield must be set
    // Format: Thousandths of percents
    uint64 limit_yield = 5;

    // The quantity the firm wishes to buy or sell in USD as a multiple of
    // the bond's par value
    // Format: Millionths of dollars
    uint64 quantity = 6;
}
```

This transaction is invalid if one of the following occurs:

- The name, order type, bond ID, and quantity are not provided
    - If the order type is LIMIT, the limit price or limit yield must also be provided
    - If the order type is MARKET, the limit price and limit yield must not be provided
- An invalid action is provided
- Invalid formatting of attributes

Create Quote

The CreateQuote transaction allows a participant to create quotes, which are then available to be matched with any open orders. The organization the participant is associated with is the organization associated with the quote. The order is given the default status of OPEN upon creation.

```
message CreateQuote {

    // ID of the bond the quote is for
    string bond_id = 1;

    // The minimum price for which the firm is willing to sell the bond
    uint64 ask_price = 2;

    // The maximum quantity the firm is willing to sell
    uint64 ask_qty = 3;

    // The maximum price for which the firm is willing to sell the bond
    uint64 bid_price = 4;
```

```
    // The minimum quantity the firm is willing to sell
    uint64 bid_qty = 5;
}
```

A CreateQuote transaction is invalid if one of the following occurs:

- All of the attributes are not provided
- The signing participant is not an authorized MARKETMAKER for its associated organization
- Invalid formatting of attributes

Create Holding

The CreateHolding transaction allows a participant to create either a bond or currency holding for their associated organization. An organization may have a single holding with the currency asset type and multiple holdings of unique bonds.

```
message CreateHolding {

    // Whether the asset is currency or a bond
    Organization.Holding.AssetType asset_type = 1;

    // If asset_type is BOND, the asset-id will contain the bond_id
    // For asset_type of CURRENCY, this will be the string 'USD'
    string asset_id = 2;

    // The current balance
    uint64 amount = 3;
}
```

A CreateHolding transaction is invalid if the organization already has a holding with that asset ID.

Delete Holding

The DeleteHolding transaction allows for organizations to remove existing holdings. The organization associated with the participant who submits the transaction must have a holding with the asset ID in order for the .

```
message DeleteHolding {

    // If asset_type is BOND, the asset-id will contain the bond_id
    // For asset_type of CURRENCY, this will be the string 'USD'
    string asset_id = 1;
}
```

A DeleteHolding transaction is invalid if the signing participant is not associated with an organization or the organization does not have a holding with the specified asset ID.

Additionally, the transaction is invalid if the asset ID is not provided.

Create Settlement

The CreateSettlement transaction allows for the trading of holdings between organizations. A quote and order are able to be settled once the order has been matched to a quote. The creation of a settlement object is instigated by a participant authorized as a TRADER of an associated organization.

```
message CreateSettlement {

    // The ID of the order which is being settled
    string order_id = 1;
}
```

A CreateSettlement transaction is invalid if the order ID is not provided. Additionally, the order specified must have a status of MATCHED and the organizations associated with quote and order must have the sufficient holdings to complete the transaction.

Create Receipt

The CreateReceipt transaction allows an organization to be able to either receive a bond coupon payment or bond pay out. Receipts are created either once a coupon benchmark has been reached which is set at regular intervals or once the bond's maturity date has been reached and the organization can receive a pay out.

```
message CreateReceipt {

    // Whether it was a coupon or redemption payment
    Organization.Receipt.PaymentType payment_type = 1;

    // The ID of the bond for which this payment occurred
    string bond_id = 2;

    // The date that the coupon became due
    // Required if payment-type is COUPON
    // Format: UTC Timestamp
    uint64 coupon_date = 3;

    // Amount received
    // Format: Millionths of dollars
    uint64 amount = 4;
}
```

The CreateReceipt transaction is considered invalid if one of the following occurs:

- Invalid payment type is provided
- Each associated organization has insufficient holdings to complete the transaction

- If the payment type is REDEMPTION, the maturity date of the bond specified must have been reached
- If the payment type is COUPON, there must be sufficient time between coupon payments and receipt objects
- Invalid formatting of attributes

Create Bond

The CreateBond transaction allows for the creation of bond objects, which serve as the object referenced in almost every transaction. Bonds may only be created by participants, already associated with an organization, and authorized as a MARKETMAKER.

```
message CreateBond {

        // Identifier of the bond (ISIN)
        string bond_id = 1;

        // List of bond ratings by agency
        repeated Bond.CorporateDebtRating corporate_debt_ratings = 2;

        // The yearly rate of interest paid in coupons
        // Format: Thousandths of percents
        uint64 coupon_rate = 3;

        // If floating, the interest rate is calculated by adding coupon-rate
        // to the rate looked up using the benchmark index (Libor)
        Bond.CouponType coupon_type = 4;

        // How often the coupons are paid out
        Bond.CouponFrequency coupon_frequency = 5;

        // The date the first coupon can be paid for the bond
        // Format: UTC Timestamp
        uint64 first_coupon_date = 6;

        // The date on which the bond becomes worth it's face value
        //  and can be redeemed
        // Format: UTC Timestamp
        uint64 maturity_date = 7;

        // Quantity of bonds
        uint64 amount_outstanding = 8;

        // The par-value of the bond, to be paid upon maturity
        // Format: Millionths of dollars
```

```
        uint64 face_value = 9;

        // Date on which the bond was first settled
        // Format: UTC Timestamp
        uint64 first_settlement_date = 10;

}
```

The CreateBond transaction is considered invalid if one of the following occurs:

- Bond ID, amount outstanding, maturity date, coupon type, coupon rate, coupon frequency, first coupon date, face value are not provided
- The provided bond ID is not unique
- The signing participant is not authorized as a MARKETMAKER for the associated organization
- Invalid formatting of attributes

Addressing

Namespace Prefix

Bond objects are stored under the namespace by taking the first six characters of the SHA-256 hash of the string “bond namespace”: d16f4b

Address Construction

After its namespace prefix, the next six characters are a string based on the object’s type, determined using the first six characters of the SHA-256 hash of the object type string (i.e., “organization”, “participant”, etc.) :

- **Organization:** af3a1b
- **Participant:** c0cde0
- **Holding:** e77b14
- **Settlement:** a75c43
- **Receipt:** 6f3286
- **Quote:** 632724
- **Order:** 3eeb7e
- **Bond:** f21dea

The remaining 58 characters are determined by its type (All hashes are determined using SHA-256):

Organization:

- First 58 characters of the hash of the organization’s unique ID

Participant:

- First 58 characters of the hash of its public key

Holding:

- First 22 characters of the hash of the owning organization’s unique ID
- First 36 characters of the hash of the asset ID
    - If currency, ‘USD’, otherwise the bond ID

Settlement:

- First 22 characters of the hash of the ordering organization’s unique ID
- First 36 characters of the hash of the order ID

Receipt:

- First 22 characters of the hash of the payee’s ID (Owner of the bond)
- First 36 characters of the hash of the bond ID

Quote:

- First 22 characters of the hash of the organization's ID
- First 36 characters of the hash of the bond ID

Order:

- First 22 characters of the hash of the organization's ID
- First 36 characters of the hash of the bond ID

Bond:

- First 58 characters of the hash of the bond's ID