Hazeian

Technical Implementation Paper

GitHub:
https://github.com/HazeianNetwork/

Walkthrough:
https://medium.com/@hazeian/hazeian-proof-of-concept-679e598b4a52

1. Developer Resources

Please note, this section is technical in nature. It takes a deep dive into the usage of the Hazeian Connect API by giving developers examples of how they can interact with the service.

Non-technical users will be able to administer every part of their marketplace and user account using the dedicated management site.

The API can be accessed through standard HTTPS calls - easily implementable on any language and device. Client libraries will be built in all major programming languages, handling authentication, logging and errors and allowing developers to easily integrate with the network.

1.1 Markets

The market endpoint will allow users to view, create and modify their marketplaces.

1.1.1 Get Markets

A marketplace owner can return their markets using a GET request. This request is authenticated and a user token must have market view permissions.

For example:

Request: GET /market/

Response:


```
{
    "id": "28148186-97f5-4471-877e-3fa898d3f6b1",
```

```
    "title": "Hazeian Exchange",
    "escrow_ids": ["358f42f4-5d5c-48ca-beae-5640aaa74ac2"],
...
```


1.1.2 Update Markets

A marketplace owner can update a market using a POST request. This request is authenticated and a user token must have market write permissions. When a marketplace is updated, Hazeian Connect will validate the data and call the Ethereum market contract to update details.

For example:

Request: POST /market/MARKET_ID/


```
{
    "title": "New Exchange Name"
}
```


1.2 Advertisements

The advertisement endpoint will allow users to view, retrieve and create advertisements on a marketplace, as well as see the overall history of their own advertisements such as price and unit changes. It will also allow marketplace owners to manage advertisements on their marketplace.

1.2.1 Get Advertisement

Advertisements can be retrieved using a HTTP GET request and the advertisement ID. Unauthenticated users will have access to basic advertisement fields.

For example:

Request: GET /advertisement/MARKET_ID/ADVERTISEMENT_ID

Response:


```
{
   "id": "28148186-97f5-4471-877e-3fa898d3f6b1",
   "title": "Hazeian Token",
   "units_available": 1000,
   "units_sold": 100,
   "seller_id": "bb7741ac-d45f-4fbc-ba96-9d032a59a382",
   "escrow_ids": ["358f42f4-5d5c-48ca-beae-5640aaa74ac2"],
   "price": 0.01,
   "currency": "ETH"
}
```


Calling the advertisement endpoint without an ID will return a list of the current live advertisements on a market.

1.2.2 Create Advertisement

To create an advertisement a user can call the endpoint with a POST request. This request is authenticated and a user token must have advertisement write permissions. Market owners can also set limits on advertisement creation and a user must not exceed these limits to create a new advertisement. When this endpoint is called, Hazeian Connect will validate a user can create the advertisement and then add it to the marketplace.

For example:

Request:

POST /advertisement/MARKET_ID/


```
{
   "title": "Hazeian Token",
   "units_available": 1000,
   "escrow_ids": ["358f42f4-5d5c-48ca-beae-5640aaa74ac2"],
   "price": 0.01,
   "currency": "ETH"
}
```


Response:


```
{"id": "f2e9e594-7cbb-4fca-a24c-db3ddaa4ab26"}
```


1.2.3 Update Advertisement

The advertisement can be updated at any time, for example if the seller wanted to change the price to account for currency fluxuations. To update an advert, the seller would perform a POST request to the specific advertisement ID. The marketplace owner can set limits on the frequency of price updates.

This request is authenticated and a user token must have advertisement write permissions for the advertisement in question.

For example:

Request: POST /advertisement/MARKET_ID/ADVERTISMENT_ID


```
{
   "price": 0.02
}
```

1.3 Trades

The trade endpoint allows users to view, create and control trades for a marketplace. All trade requests are authenticated and marketplace owners can set limits on trades for different groups of users. For example, new users maybe limited in the size of trades they can create. A user token will be verified before the request is processed.

1.3.1 Get Trade

A user can call the trade endpoint to get details of all their trades or a specific trade. Since active trades are stored in the Hazeian Core market contract, Hazeian Connect will retrieve this information along with supplementary information from the database when returning trade details..

Trades are stored in the blockchain using a unique hashing system. Hazeian Connect will calculate the trade hash on the user's behalf and pull the data from the market contract, allowing users to call trades using a simple trade identifier.

For example:

Request: GET /trade/MARKET_ID/TRADE_ID

Response:


```
{
   "id": "28148186-97f5-4471-877e-3fa898d3f6b1",
   "advertisement_id": "f2e9e594-7cbb-4fca-a24c-db3ddaa4ab26",
   "escrow_id": "c793c1ba-db9b-4499-b6bf-adc3172c435f",
   "units": 2
}
```

1.3.2 Create Trade

A buyer can use the trade endpoint to start a new trade with a seller. To accomplish this they send the advertisement ID, along with the number of units and escrow service selected (if any).

Once the seller has accepted the trade, the trade data will sent to Hazeian Core to be stored on the market contract. Because the creation of the trade costs a small amount of gas, waiting for confirmation prevents buyers from spamming sellers with fake orders.

Hazeian Connect will then store the trade details and update the market contract.

For example:

Request:

POST /trade/MARKET_ID/


```
{
   "units": 2,
   "advertisement_id_id": "f2e9e594-7cbb-4fca-a24c-db3ddaa4ab26",
   "escrow_id": "358f42f4-5d5c-48ca-beae-5640aaa74ac2",
   "price": 0.01
}
```


Response: {"id": "987baafe-2933-4b0a-92e0-da8e76cb8ad9"}

1.3.3 Accept Trade

Sellers can use the trade endpoint to return a list of their current or pending trades. Once a trade is accepted, Hazeian Connect will call the market contract to add the trade to the blockchain.

Request: POST /trade/MARKET_ID/TRADE_ID/

```
{
   "state": "accepted"
}
```

1.3.4 Withdraw Funds

After a trade is complete the buyer and seller are able to withdraw funds to their accounts. Hazeian Connect will automatically monitor the state of the trade and call the market contract to withdraw funds to the buyer and seller when the trade is complete, removing the need for users to manually check the status of their trades.

1.3.5 Cancel Trade

In some cases the buyer or seller may wish to cancel a trade. For obvious reasons this can only happen if a certain set of circumstances are met; such as when the buyer and seller both agree to cancel. When these circumstances are met the buyer or seller can send a DELETE request to the trade endpoint with the trade id. If the trade is allowed to be cancelled, Hazeian Connect will return a 204 (No Content) response, otherwise it will return a 403 (Access Denied). Hazeian Connect will make all the required checks during the request, allowing the user to safely make the call at any time.

For example:

Request: DELETE /trade/MARKET_ID/TRADE_ID

1.3.6 Dispute Trade

The dispute endpoint allows either the buyer or seller to dispute a trade if something goes wrong. Calling the dispute endpoint will give permission for the escrow service to decrypt all messages and resolve the trade. The dispute status will be returned when retrieving the trade details from the trade endpoint.

For example:

Request: POST /trade/MARKET_ID/TRADE_ID/

```
{
   "state": "dispute"
}
```

The escrow service will then be able to take control of the trade and either release or refund user funds.

1.4 Messaging

Once a trade has begun, Hazeian Connect will setup a secure, end-to-end encrypted messaging channel. To ensure messages cannot be read by Hazeian, they must first be encrypted before being sent to the endpoint - a process that is handles in the provided client libraries.

To send a secure message, a user would first encrypt it using the other users public key, before sending it within a POST request to the endpoint.

Request: POST /trade/MARKET_ID/TRADE_ID/message

```
{
   "encrypted_message": "MIIBIjANBgkqhkiG9w0BAQEFAAO...
}
```


1.5 Feedback

The feedback endpoint allows users to view and send feedback for other users, markets and escrow services. If a trade completes successfully, feedback is left automatically. However, escrow services and users can leave negative feedback in the case of a dispute. Users will be able to see accurate feedback scores for their own profiles, while other users will show aggregated scores and sellers will be able to restrict access to advertisements based on a users feedback score.

Feedback is returned as a field when getting user or market data.

Request: GET /trade/MARKET_ID/USER_ID/

Response:


```
{
   "feedback_score": "100",
   "feedback_count": "500",
}
```


1.6 Account Management

The API will allow users to manage all aspects of their user account or marketplaces. This includes changing details, generating or deleting keys and modifying group members and permissions. Access will also be given to account statistics, showing real-time and historical data for API usage, advertisements, tradesand users.

When accessing account management endpoints, Hazeian Connect will check the users authorization against their account permissions.

2. Client Libraries

Hazeian will provide client libraries for all major programming languages, starting with Python, Java, Swift, JavaScript, Ruby and Go. Further libraries will follow based on a community vote.

The client libraries will handle authentication, logging and error handling, allowing developers to access the core functionality of Hazeian Connect with a single line of code for each action.

For example, a developer can use the Python library to return trade details in three lines of code. The the background, Hazeian Connect will check the calling user has the required access, retrieve the trade details from the market contract, supplement these details with data from the datastore and return them to the user.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import hazeian

conn = hazeian.connect(API_KEY, API_SECRET, MARKET_ID)
conn.get_trade(TRADE_ID)
```