FALCONX

RFQ RFS API integration FAQs

Q: How can I obtain prices for a market?

There are multiple ways to obtain prices for any market. Please select the approach that best suits your use case.

1. RFQ - Request for price over a REST call for a specific quantity. https://app.falconx.io/docs/rfq-rfs#get-quote-new-v3-endpoint
2. RFS - Subscribe for a continuous price feed over a web-socket connection.https://app.falconx.io/docs/rfq-rfs#websocket-prices
3. RFS - Subscribe for a continuous price feed over a FIX connection. https://app.falconx.io/docs/rfq-rfs#fix-api-docs

Q: How are orders placed?

There are multiple channels available for placing different order types.

1. RFQ execution
    1. REST endpoints
2. Market order
    1. REST endpoints
    2. FIX
3. Limit FOK order
    1. REST Endpoint
    2. FIX
4. Limit GTC/GTX order
    1. Websocket connection
    2. FIX
5. Twap Order
    1. Websocket connection [TBD]
    2. FIX

Q: How is order status obtained?

A:

For all orders, an ACK/order fill status will be provided through the channel used to place the order. Further order status can be checked using this API.
https://app.falconx.io/docs/rfq-rfs#get-order-history

Q : I have trouble connecting/authenticating over the REST endpoint.
Ans: Troubleshooting Steps

1. Verify that you are connecting to the correct host.
2. Verify that you are using the correct path as specified in the API documentation.
3. Verify that you are using the correct method (GET/POST) as specified in the API documentation.
4. Verify that you have the correct headers set in your request.
    a. Expected common headers

```
'FX-ACCESS-SIGN': 'signature_b64',
'FX-ACCESS-TIMESTAMP': 'timestamp',
'FX-ACCESS-KEY': 'api_key',
'FX-ACCESS-PASSPHRASE': 'passphrase',
'Content-Type': 'application/json'
```

5. Verify that the API key being utilized has the appropriate permissions enabled.
6. Please re-examine the timestamp included in the headers above to confirm it is not outdated.
7. Ensure that your signature generation algorithm is accurate and adheres to the method outlined in the API documentation. Refer to **Annexure A** below for further verification.
8. Verify if the body/payload used for generating signature matches the actual contents of the messages

Q : I have trouble connecting/authenticating over Web-Socket.
Ans: Troubleshooting Steps

1. Verify that you are connecting to the correct host.
2. Verify that you are using the correct path as specified in the API documentation.
3. Verify that you have the correct auth request within 30 seconds of making the connection.
    a. Format of auth request over websocket

```
Unset
{
    "action": "auth",
    "api_key": "xxyyzz",
    "passphrase": "aabbcc",
    "signature": "xt64fsrt18_uid",
    "timestamp": 16524352778,
    "request_id": "my_sample_request"
}
```

4. Verify that the API key being utilized has the appropriate permissions enabled.
5. Please re-examine the timestamp included in the headers above to confirm it is not outdated.
6. Ensure that your signature generation algorithm is accurate and adheres to the method outlined in the API documentation. Refer to **Annexure A** below for further verification.
7. Verify if the body/payload used for generating signature matches the actual contents of the messages

Q: Why is my IP restricted from integration?

A: For REST and Websocket connections, you might receive the error code REQUEST_IP_RESTRICTED if your IP is not authorized to access the Falconx environment.

1. Please ensure that the correct IP/CIDR range is whitelisted by your account admin for the API key that you are using.
2. Ensure that your external-facing IP belongs to the list whitelisted in the above step.
    1. You can check your external IP by running this command from the machine you are trying to connect to Falconx.
    2. curl ifconfig.me

Q: Why does the Websocket client drop after a few seconds/minutes?

A: Websocket connections for Market data receive many price updates per second. It can

be up to 400 ticks per second for a given market level combination. So for multiple subscriptions over a connection, it can go up to 20,000 messages per second. This volume can be overwhelming for a system that can't process this many messages per second, and it may eventually drop the connection due to the TCP_BUFFER limit on the client's system.

1. To check if this is the issue, you can run this (or a similar) command on a Linux-based system to see the SEND and RCV Queues on your system and observe if they are increasing over the connection's duration.
2. Command - ss -nmtp

Another suggestion is to divide the subscriptions over multiple connections so that the load is distributed across multiple connections/systems.

Q: Why am I not receiving price updates over websocket?
A:
Please ensure that

1. You have an active web-socket connection.
2. You have successfully authenticated on that connection and have successful authentication acknowledgement.
3. You have sent a subscription for the market and have received successful acknowledgement for the same.

Annexure A.

Verify your signature algorithm. Put in inputs in this function and see if output from your implementation matches the output from this functions

```
import base64
import hmac
import hashlib

def generate_signature(secret_key, method, url, timestamp, body=None):
    """
    Generates a signature for authentication.
```

```
    :param secret_key: Your secret key (base64 encoded).
    :param method: HTTP method (e.g., 'GET', 'POST').
    :param url: The full URL path of the request.
    :param timestamp: The timestamp to be used in the signature.
    :param body: The request body as a string (default is None).
    :return: A base64-encoded signature.
    """
    request_body = body if body else ''
    message = timestamp + method + url + request_body
    hmac_key = base64.b64decode(secret_key)
    signature = hmac.new(hmac_key, message.encode(), hashlib.sha256)
    signature_b64 = base64.b64encode(signature.digest()).decode()

    return signature_b64

# Example usage
secret_key = 'your_secret_key_base64_encoded'
method = 'POST'
url = '/v1/orders'
timestamp = str(time.time())
body = '{"order": "details"}'

signature = generate_signature(secret_key, method, url, timestamp, body)
print("Generated Signature:", signature)
```