SECURE
DIGITAL MARKETS

Introduction

Welcome to the SDM API.

The websocket API provides the user with access to real-time market data subscription, orders execution and balance information. The REST API provides historical market data.

Documentation Version: 1.0.0

API Keys

If you do not have an API key and a signature assigned, you will need to obtain them first. You can do so by using the following method.

- Email SDM support staff at: tech@sdm.co.

You will need separate API Keys for connecting to the sandbox and production environments.

Websocket API

Websocket Address

Your websocket endpoint is trade-sandbox.sdm.co in sandbox, and trade.sdm.co in production.

Authentication

Websocket connections must be authenticated by signing the request and sending the signature in HTTP headers.

> *Connect using python 2 or 3*

```
# to install websocket lib:
#     $ pip install websocket-client
from websocket import create_connection
import datetime
import hmac
import hashlib
import base64

api_key = "<apikey>"
api_secret = "<apisecret>"
utc_now = datetime.datetime.utcnow()
utc_datetime = utc_now.strftime("%Y-%m-%dT%H:%M:%S.000000Z")

host = "<websocket host>" # trade-sandbox.sdm.co, for example
path = "/ws/v1"
params = "\n".join([
    "GET",
    utc_datetime,
    host,
    path,
])
hash = hmac.new(
    api_secret.encode('ascii'), params.encode('ascii'), hashlib.sha256)
hash.hexdigest()
signature = base64.urlsafe_b64encode(hash.digest()).decode()
header = {
    "ApiKey": api_key,
    "ApiSign": signature,
    "ApiTimestamp": utc_datetime,
}

ws = create_connection("wss://" + host + path, header=header)

while True:
    print(ws.recv())
```

Header	Description
`ApiKey`	Your SDM API key
`ApiTimestamp`	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`ApiSign`	base64-encoded signature

The signature is a base64 encoded sha256 HMAC using the API secret of the following:

`GET\n<ApiTimestamp>\n<host>\n<path>`

For example,

`GET\n2019-02-13T05:17:32.000000Z\ntrade-sandbox.sdm.co\n/ws/v1`

REST API

REST Endpoint

REST Endpoints are available for OHLCV data.

Authentication

REST requests must be authenticated by signing the request and sending the signature in HTTP headers.

> *Connect using python 2 or 3*

```
# to install requests lib:
#     $ pip install requests
import requests
import datetime
import hmac
import hashlib
import base64

api_key = "<apikey>"
api_secret = "<apisecret>"
utc_now = datetime.datetime.utcnow()
utc_datetime = utc_now.strftime("%Y-%m-%dT%H:%M:%S.000000Z")

host = "<host>" # trade-sandbox.sdm.co, for example
path = "/v1"
query = ""
body = ""

params = [
    "GET",
    utc_datetime,
    host,
    path,
]
endpoint = "https://" + host + path
if query:
    endpoint += "?" + query
    params.append(query)
if body:
    params.append(body)

payload = "\n".join(params)
hashvalue = hmac.new(
    api_secret.encode('ascii'), payload.encode('ascii'), hashlib.sha256)
hashvalue.hexdigest()
signature = base64.urlsafe_b64encode(hashvalue.digest()).decode()
headers = {
    "ApiKey": api_key,
    "ApiSign": signature,
    "ApiTimestamp": utc_datetime,
}

print(requests.get(url=endpoint, headers=headers))
```

Header	Description
`ApiKey`	Your API key

`ApiTimestamp`	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`ApiSign`	base64-encoded signature

The signature is a base64 encoded sha256 HMAC using the API secret of the following:

`GET\n<ApiTimestamp>\n<host>\n<path>[\nquery][\nbody]`

`query` and `body` should only be included if they are not empty.

For example,

`GET\n2019-02-13T05:17:32.000000Z\ntrade-sandbox.sdm.co\n/v1/symbols/BTC-USD/ohlcv/1D`

Rate Limits

There is no limit to how many concurrent websocket connections you're allowed, though this may change in the future. If you expect to have more than ~10 active connections, please contact tech@sdm.co.

Protocol

General notes

- Timestamps: In all API requests, the user has the option to provide a timestamp (ts) in an ISO-8601 UTC string format with microsecond resolution. While optional, we recommend taking advantage of this field, as it will both enable our system to monitor client-server latency and allow certain security features.
- All system responses are tied to a specific client request. The user should provide a unique reqid (Request ID) on each request, and the system will echo this reqid on the corresponding responses.

Hello

> *The server sends a hello message on a successful connection*

```
{
  "type": "hello",
  "ts": "2019-10-16T10:41:23.168807Z",
  "session_id": "1109RQ13KXR00"
}
```

After a client connects, the server will send an unsolicited hello message.

Parameter	Type	Required	Description
`type`	string	Y	Always "hello".
`ts`	number	Y	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`session_id`	string	Y	A session ID that uniquely identifies this websocket session.

Requests

> *An example request that subscribes to data*

```
{
  "reqid": 1,
  "type": "subscribe",
  "streams": [
    {
      "name": "MessageType",
      "Arg1": 1
    }
  ],
  "ts": "2019-02-13T05:17:32.000000Z"
}
```

Clients issue requests to subscribe to data and execute actions like placing orders. A request should be a JSON message with the following keys:

Parameter	Type	Required	Description
`reqid`	number	Y	A number that identifies this request and will be included on responses to this request.
`type`	string	Y	The type of message sent. This document describes all valid message types.
`ts`	number	N	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`

`reqid` cannot be equal to 0.

Responses

> *An example response with data*

```
{
  "reqid": 1,
  "seq": 1,
  "type": "MessageType",
  "action": "Update",
  "data": [
    {
      "Key": "Value"
    }
  ],
  "ts": "2019-02-13T05:17:32.000000Z"
}
```

The server responds to requests with one or more response messages that reference the request by echoing back the `reqid` field.

Key	Type	Required	Description
`reqid`	number	Y	A number that relates this response to a request.
`seq`	number	Y	The sequence number for this response per request. `seq` begins at 1 on the first message with this `reqid` and increments by one for each message sent on this `reqid`.
`type`	string	Y	The type of message sent. This document describes all valid message types.
`tag`	string	N	An optional tag from the stream request.
`ts`	number	Y	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`.
`initial`	bool	N	If this is initial data for a request, the `initial` flag will be set.
`action`	string	N	"Update" or "Remove"
`total_count`	number	N	The total number of records for this stream.

`seq` is for debug purposes only, the client is not required to do any sequencing.
If a request requires responses with different types, then multiple responses may have the `initial` flag set for different types.
`action` is either `Update` or `Remove` and tells the client if the given entity should be removed or added/updated. The key for a given entity depends on the message type.

Types

In all messages below the described types are encoded as below:

Type	Encoding
datetime	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
decimal	A string encoded decimal; eg `"4.327"` represents the decimal 4.327

Customer Websocket Subscriptions

Subscribe for Stream

Send a subscription request to begin a stream of data.

Request structure

Key	Type	Required	Description
`reqid`	int	Y	Request ID - will be echoed back in the response structure
`type`	string	Y	"subscribe" for initializing a subscription
`tag`	string	Y	Optional user generated tag to attach to stream data
`streams`	object[]	Y	Streams to subscribe to
→ `name`	string	Y	Subscription name
→ <`Subscription Parameters`>	---	----	Any required or optional parameters described for the subscription names below

Stream Update Structure

Key	Type	Required	Description
`reqid`	int	Y	Echoed back request ID
`type`	string	Y	Echoed back subscription name to define the type of the data
`seq`	int	Y	Incrementing sequence number of messages for request ID starting with 1
`initial`	boolean	Y	Flag to indicate if this message is the initial state of the stream
`ts`	datetime	Y	Timestamp of message publication
`data`	object[]	Y	List of response data structured per subscription type below
`next`	string	N	If paging, tag for next page of data

Amend an Existing Stream

Amends a subscription request by Request ID. Response structure remains the same, just the content of the `data` will change per the amendment.

Request structure

Key	Type	Required	Description
`reqid`	int	Y	Request ID - will be echoed back in the response structure
`type`	string	Y	"subscribe" for initializing a subscription
`tag`	string	Y	Optional user generated tag to attach to stream data
`streams`	object[]	Y	Streams to subscribe to
→ `name`	string	Y	Subscription name
→ <`Subscription Parameters`>	---	----	Any required or optional parameters described for the subscription names below

Continue a Paged Stream

If you request a stream that is paged, the data on the stream will be delivered with a `next` field populated. If given this field, request the next set of data with the below structure. Response continues as with the original stream.

Request structure

Key	Type	Required	Description
`reqid`	int	Y	The request ID to continue the page, must match the original request ID
`type`	string	Y	"page" for continuing the paged stream
`data`	object[]	Y	Stream request data to continue
→ `name`	string	Y	Subscription name to continue the page
→ `after`	string	Y	Delivered `next` value from the paged stream payload

Cancel an Existing Stream

Cancels an existing stream by Request ID. No response, unless there is an error. Stream will just stop.

Request structure

Key	Type	Required	Description
`reqid`	int	Y	Request ID to cancel
`type`	string	Y	"cancel" for cancelling the subscription

Websocket Commands

Commands can be submitted via the websocket API.

Send a Command

This is the general structure for sending a command to the websocket API. The `type` field is the command name, specified as one of the options listed below. The `data` required for each command is described below. There could be an error response to the command, otherwise any published response data should be consumed via the associated subscription above.

Submission Structure

Key	Type	Required	Description
`type`	string	Y	The type of the command to send
`data`	object[]	Y	Any requisite data that needs to be submitted along with the command

Security Master

Overview

SDM maintains a security master which contains up-to-date information about the securities supported over API that are available to trade. The user should use the below streams when subscribing to the Security Master data.

Security

Request for stream of available securities for trading. On initial request, a snapshot of all available securities is provided. All subsequent messages are deltas to the securities snapshot.

> *Request for Security Sub*

```
{
  "reqid": 2,
  "type": "subscribe",
  "streams": [
    {
      "name": "Security"
    }
  ]
}
```

> *Response stream is open*

```
{
  "reqid": 2,
  "type": "Security",
  "ts": "2021-09-14T22:11:47.512168Z",
  "initial": true,
  "seqNum": 1,
  "data": [
    {
      "Timestamp": "2021-09-14T22:11:47.512168Z",
      "UpdateAction": "Update",
      "SecurityID": 2,
      "Symbol": "ETH-USD",
      "MinPriceIncrement": "0.01",
      "MinSizeIncrement": "0.00000001",
      "MinAmtIncrement": "0.01",
      "MinimumSize": "0.00000001",
      "MaximumSize": "10000000",
      "QuoteCurrency": "USD",
      "BaseCurrency": "ETH",
      "DefaultPriceIncrement": "0.01",
      "DefaultSizeIncrement": "0.0001",
      "PriceDisplaySpec": "M.m",
      "SizeDisplaySpec": "M.m",
      "NormalSize": "2",
      "ProductType": "Spot",
      "PositionCurrency": "ETH",
      "SettlementCurrency": "USD",
      "NotionalMultiplier": "1",
      "SizeBuckets": [
        {
          "Size": "0"
        }
      ],
      "DisplaySymbol": "ETH-USD",
```

```
      "Description": "Ethereum/U.S. Dollar"
    }
  ]
}
```

Subscription Parameters

Key	Type	Required	Description
`Symbols`	[]string	N	Optional list of symbols to filter the subscription on

Response

Key	Type	Required	Description
`Timestamp`	datetime	Y	Timestamp of message publication
`UpdateAction`	string	Y	`Update` or `Remove` where the latter indicates the Security is no longer available
`SecurityID`	string	Y	System ID for Security - for information only
`Symbol`	string	Y	`Symbol` for security
`MinPriceIncrement`	decimal	Y	Minimum Price Increment for Security
`MinSizeIncrement`	decimal	Y	Minimum Size Increment for Security. This increment is used to validate orders and RFQS expressed in base currency of the security.
`MinAmtIncrement`	decimal	N	Minimum Amount Increment for Security. This increment is used to validate orders and RFQS expressed in quote currency of the security.
`MinimumSize`	decimal	Y	Minimum Size for an Order on this Security
`MaximumSize`	decimal	N	Maximum Size for an Order on this Security
`QuoteCurrency`	string	Y	Currency in which the Security is quoted
`BaseCurrency`	string	Y	Base currency for the Security
`EndTime`	datetime	N	Date this security was Disabled
`DefaultPriceIncrement`	decimal	Y	Default price increment used in all Market Data and Orders
`DefaultSizeIncrement`	decimal	Y	Default size increment used in all Market Data and Orders
`PriceDisplaySpec`	string	Y	String specification describing the price display
`NormalSize`	decimal	N	Normal size of Security
`ProductType`	string	N	Product type of Security
`PositionCurrency`	string	Y	Currency in which any position on this Security is tracked
`SettlementCurrency`	string	N	Currency in which any settlement on thes Security is negotiated
`NotionalMultiplier`	decimal	N	Notional Multiplier of Security
`Expiration`	decimal	N	Expiration of Security
`SizeBuckets`	[]decimal	Y	Default array of Size Buckets for Market Data Snapshot Streams
`DisplaySymbol`	string	Y	Display symbol for Security
`Description`	string	Y	Description of Security

Currency

Request for stream of available currencies used within trading. On initial request, a snapshot of all available currencies is provided. All subsequent messages are deltas to the currency snapshot.

> *Request for Currency Sub*

```
{
  "reqid": 3,
  "type": "subscribe",
  "streams": [
    {
      "name": "Currency"
    }
  ]
}
```

> *Response stream is open*

```
{
  "reqid": 3,
  "type": "Currency",
  "ts": "2021-09-14T22:11:47.512168Z",
  "initial": true,
  "seqNum": 1,
  "data": [
    {
      "Timestamp": "2020-09-30T17:56:08.552227Z",
      "UpdateAction": "Update",
      "CurrencyID": 2,
      "Symbol": "USD",
      "MinIncrement": "0.01",
      "DefaultIncrement": "0.01",
      "Description": "U.S. Dollar"
    },
    {
      "Timestamp": "2020-05-13T13:24:58.983054Z",
      "UpdateAction": "Update",
      "CurrencyID": 17,
      "Symbol": "ETH",
      "MinIncrement": "0.00000001",
      "DefaultIncrement": "0.0001",
      "Description": "Ethereum"
    }
  ]
}
```

Subscription Parameters

Key	Type	Required	Description
`Symbols`	[]string	N	Optional list of currency symbols to filter the subscription on

Response

Key	Type	Required	Description
`Timestamp`	datetime	Y	Timestamp of message publication
`CurrencyID`	string	Y	System ID for Currency - for information only
`Symbol`	string	Y	`Symbol` for currency
`MinIncrement`	decimal	Y	Minimum increment for Currency
`DefaultIncrement`	decimal	Y	Default increment used in all Market Data and Orders
`Description`	string	Y	Description of Security

Market Data

Overview

This section outlines how to subscribe to market data via the SDM API.

MarketDataSnapshot

Request stream of MarketDataSnapshot messages. To stream security pricing.

> *Request for MarketDataSnapshot Sub*


```
{
  "reqid": 5,
  "type": "subscribe",
  "streams": [
    {
      "name": "MarketDataSnapshot",
      "Symbol": "BTC-USD"
    }
  ]
}
```


> *Response stream is open*


```
{
  "reqid": 5,
  "type": "MarketDataSnapshot",
  "ts": "2021-09-14T22:20:49.862930Z",
  "initial": true,
  "seqNum": 1,
  "data": [
    {
      "Timestamp": "2021-09-14T22:20:49.860957Z",
      "Symbol": "BTC-USD",
      "Status": "Online",
      "Bids": [
        {
          "Price": "46817.27965000",
          "Size": "1.00000000"
        },
        {
          "Price": "46816.08025000",
          "Size": "2.00000000"
        }
      ],
      "Offers": [
        {
          "Price": "46868.59755873",
          "Size": "1.00000000"
        },
        {
          "Price": "46870.33345500",
          "Size": "2.00000000"
        }
      ]
    }
  ]
}
```

Subscription Parameters

Key	Type	Required	Description
`Symbol`	string	Y	Security to request
`Throttle`	string	N	An optional throttle duration for updates, defaults to configured `MinMarketDataThrottle`. A throttle is specified by a time duration + units. For example: "300ms", "500us". Valid time units are "ns", "us", "ms", "s".
`PriceIncrement`	decimal	N	Price increment for levels to enable price bucketing
`Depth`	decimal	N	Maximum number of bid / offer levels to stream prices for
`SizeBuckets`	[]decimal	N	A list of sizes to return price levels for; defaults to configured Security SizeBuckets

Response

Key	Type	Required	Description
`Timestamp`	datetime	Y	Timestamp of update
`Symbol`	string	Y	Security for update
`Status`	string	Y	Status of update; `Online` or `Offline`
`Bids`	object[]	Y	Full list of Bids
→ `Price`	decimal	Y	Limit price of the level.
→ `Size`	decimal	Y	Size on this level.
`Offers`	object[]	Y	Full list of Offers
→ `Price`	decimal	Y	Limit price of the level.
→ `Size`	decimal	Y	Size on this level.

CurrencyConversion

Request for stream of currency conversion rates.

> *Request for CurrencyConversion Sub*

```
{
  "reqid": 4,
  "type": "subscribe",
  "streams": [
    {
      "name": "CurrencyConversion",
      "EquivalentCurrency": "USD",
      "Currencies": ["BTC", "ETH", "BCH-USD"],
      "Throttle": "15s",
      "Tolerance": "0.0002"
    }
  ]
}
```

> *Response stream is open*

```
{
  "reqid": 4,
  "type": "CurrencyConversion",
  "ts": "2021-09-14T22:16:18.604996Z",
  "initial": true,
  "seqNum": 1,
  "data": [
    {
      "Timestamp": "2021-09-14T22:16:18.604674Z",
      "EquivalentCurrency": "USD",
      "Currency": "ETH",
      "Rate": "3368.16",
      "Status": "Online",
      "ConversionPath": "(ETH-USD)"
    },
    {
      "Timestamp": "2021-09-14T22:16:18.604674Z",
      "EquivalentCurrency": "USD",
      "Currency": "BTC",
      "Rate": "46841.35",
      "Status": "Online",
      "ConversionPath": "(BTC-USD)"
    },
    {
      "Timestamp": "2021-09-14T22:16:18.604674Z",
      "EquivalentCurrency": "USD",
      "Currency": "BCH-USD",
      "Rate": "639.530000000000",
      "Status": "Online",
      "ConversionPath": "(BCH-USD)"
    }
  ]
}
```

Subscription Parameters

Key	Type	Required	Description
`EquivalentCurrency`	string	Y	Quote currency to base the conversion rates on
`Currencies`	[]string	N	List of currencies and securities to generate conversion rates for; defaults to all available currencies
`Throttle`	string	N	Optional throttle interval for conversion rate updates. Minimal / default value: 10s
`Tolerance`	decimal	N	Optional param to indicate what conversation rate percent change should trigger sending out an update. Minimal / default value: 0.0001 which means 1 bps

Response

Key	Type	Required	Description
`Timestamp`	datetime	Y	Timestamp of message publication
`EquivalentCurrency`	string	Y	Currency symbol for equivalent currency
`Currency`	string	Y	Currency symbol for base currency on conversion
`Rate`	decimal	Y	Currency conversion rate

Orders

Overview

This section outlines how to send orders using the SDM API. To send orders, first subscribe to the ExecutionReport stream. All order responses and updates are sent as `ExecutionReport`'s'. Send requests with the NewOrderSingle, OrderCancelRequest, OrderCancelReplaceRequest, and OrderControlRequest messages.

A typical trading strategy would do the following:

1. Subscribe to `MarketDataSnapshot` for the relevant securities
2. Subscribe to `ExecutionReport` with `StartDate` of the `Timestamp` of the last `ExecutionReport` processed to recover order state.
3. Subscribe to `Trade` with `StartDate` of the `Timestamp` of the last `Trade` processed to recover any missed trades.
4. Maintain a map of (`OrderID` , last received `ExecutionReport`) for open orders.
5. Maintain a set of `ClOrdID` for each pending request.
6. To submit an order, first generate a new `ClOrdID` (see instructions below). Send a `NewOrderSingle` specifying the `ClOrdID` , `Side` , `Price` , `OrderQty` , `Strategy` , etc. for that request.
7. The response for the request will be received on the `ExecutionReport` stream with the `ClOrdID` being the `ClOrdID` specified on the `NewOrderSingle` . If the order is accepted (`ExecType=PendingNew`), then add this order to the set of open orders.
8. To cancel or modify the open order, generate a new `ClOrdID` and send an `OrderCancelRequest` or `OrderCancelReplaceRequest` specifying the the `ClOrdID` , `OrigClOrdID` from the previous successful request, and remaining parameters.
9. Fills will be received as `ExecutionReport` with `ExecType=Trade` .

To generate a `ClOrdID`, we recommend using a UUID4 or another globally unique identifier. `ClOrdID` must be unique daily and among an y open order and must be less than 36 characters.
The orders API roughly follows the same semantics as the FIX Protocol, though with some notable differences, see Orde r State Change Matrices for more details.

Supported Order Strategies

SteadyPace

A schedule-based algorithm that divides an order into suborders (clips) and submits them at equal, user defined intervals.

SteadyPace Parameters

Parameter	Type	Required	Description
`ClipSize`	Qty	Y	A quantity (subset of the total order size) to send on each execution of the schedule.
`ClipInterval`	string	Y	Time interval between individual clips specified as an interval string.
`StartTime`	string	N	Time at which this order will activate and begin sending orders. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`EndTime`	string	N	Optional expire time for the order. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Variance`	decimal	N	Optional degree of randomization for clip sizes.

An interval or duration string is a possibly signed sequence of decimal numbers, each with optional fraction and a unit suffix, such as `300ms` or `2h45m`. Valid time units are `ns`, `us` (or `µs`), `ms`, `s`, `m`, `h`.
A variance of `0.1` (implying a variance of `10%`) for a clip size of `1.0` would result in clips ranging between `0.9` and `1.1`.

StopLimit

This strategy will only place the order once the specified stop price has been met. For a Buy StopLimit order, the stop price must be *above* the current market price. For a Sell StopLimit order, the stop price must be *below* the current market price.

StopLimit Parameters

Parameter	Type	Required	Description
`TriggerPrice`	Price	Y	The price that must be met to trigger execution of the order with the configured limit price.
`EndTime`	string	N	Optional expire time for the order. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`

Order

Request stream of Order updates.

> *Request for Order Sub*


```
{
  "reqid": 7,
  "type": "subscribe",
  "streams": [
    {
      "name": "Order"
    }
  ]
}
```


> *Response stream is open*


```
{
  "reqid": 7,
  "type": "Order",
  "ts": "2021-09-14T22:26:44.518538Z",
  "initial": false,
  "seqNum": 3,
  "data": [
    {
      "Timestamp": "2021-09-14T22:26:44.505519Z",
      "Symbol": "BTC-USD",
      "OrderID": "b35b1c3b-a304-4224-919f-9db1319de188",
      "ClOrdID": "d7635e40-15aa-11ec-b0a2-2554a9e1e7a4",
      "SubmitTime": "2021-09-14T22:26:44.457050Z",
      "ExecID": "c73fcf77-aaa1-46e7-9260-f625d6416646",
      "Side": "Buy",
      "ExecType": "New",
      "OrdStatus": "New",
      "OrderQty": "0.10000000",
      "OrdType": "Market",
      "Currency": "BTC",
      "LeavesQty": "0.10000000",
      "CumQty": "0",
      "AvgPx": "0",
      "TimeInForce": "FillOrKill",
      "LastPx": "0",
      "LastQty": "0",
      "LastAmt": "0",
      "LastFee": "0",
      "CumAmt": "0",
      "DecisionStatus": "Active",
      "AmountCurrency": "USD",
      "CustomerUser": "tom@company.com",
      "Parameters": {
        "ErrorAction": "Pause"
      }
    }
  ]
}
```

Subscription Parameters

Key	Type	Required	Description
`StartDate`	string	N	If provided, the subscription will return orders that were submitted after this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`EndDate`	string	N	If provided, the subscription will return orders that were submitted before this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Symbol`	string	N	If provided, Symbol of the security to get the orders for
`Statuses`	string[]	N	If provided, comma-separated Statuses of orders to include e.g. New,Filled
`OrderID`	string	N	If provided, filter by `OrderID`
`RFQID`	string	N	If provided, filter by `RFQID`

Response

An `Order` update having a `data` field containing an array that has a similar structure to `ExecutionReport`

ExecutionReport

Request stream of Execution Reports.

> *Request for Execution Report Sub*

```
{
  "reqid": 6,
  "type": "subscribe",
  "streams": [
    {
      "name": "ExecutionReport"
    }
  ]
}
```

> *Response stream is open*

```
{
  "reqid": 6,
  "type": "ExecutionReport",
  "ts": "2021-09-14T22:23:11.920903Z",
  "initial": false,
  "seqNum": 2,
  "data": [
    {
      "Timestamp": "2021-09-14T22:23:11.903074Z",
      "Symbol": "BTC-USD",
      "OrderID": "1b6b882b-e2fc-4774-ad2d-9db4df536f29",
      "ClOrdID": "58b9adb0-15aa-11ec-b0a2-2554a9e1e7a4",
      "SubmitTime": "2021-09-14T22:23:11.903074Z",
      "ExecID": "4b423f54-4a54-4679-9ee7-1f5f417ec5f5",
      "Side": "Buy",
      "TransactTime": "2021-09-14T22:23:11.883000Z",
      "ExecType": "PendingNew",
      "OrdStatus": "PendingNew",
      "OrderQty": "0.20000000",
      "OrdType": "Market",
      "Currency": "BTC",
      "LeavesQty": "0.20000000",
      "CumQty": "0",
```

```
      "AvgPx": "0",
      "TimeInForce": "FillOrKill",
      "LastPx": "0",
      "LastQty": "0",
      "LastAmt": "0",
      "LastFee": "0",
      "CumAmt": "0",
      "AmountCurrency": "USD",
      "CustomerUser": "tom@company.com",
      "Parameters": {
        "ErrorAction": "Pause"
      }
    }
  ]
}
```

Subscription Parameters

Key	Type	Required	Description
`StartDate`	string	N	If provided, the subscription will return execution reports that were published at or after this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`EndDate`	string	N	If provided, the subscription will return execution reports that were published before this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Symbol`	string	N	If provided, Symbol of the security to get the executions for
`Statuses`	string[]	N	If provided, comma-separated statuses of orders to include e.g. New,Filled
`OrderID`	string	N	If provided, filter by `OrderID`
`RFQID`	string	N	If provided, filter by `RFQID`

Response

Key	Type	Required	Description
`Timestamp`	datetime	Y	Timestamp of the message
`Symbol`	string	Y	Symbol of the order security
`OrderID`	string	Y	Server assigned Order ID, will be a UUID
`ClOrdID`	string	Y	Client assigned Order ID for the last request
`OrigClOrdID`	string	Y	Original client assigned
`SubmitTime`	datetime	Y	Time of original order submission
`ExecID`	string	Y	Server assigned ID of this update, will be a UUID
`Side`	string	Y	"Buy" or "Sell"
`TransactTime`	datetime	N	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`ExecType`	string	Y	Describes the specific execution (e.g. Pending Cancel) while OrdStatus will always identify the current order status (e.g. Partially Filled). Valid values: `New`, `Trade`, `Canceled`, `Replaced`, `PendingCancel`, `Stopped`, `Rejected`, `PendingNew`, `Restated`, `PendingReplace`, `CancelRejected`, `ReplaceRejected`, `PendingResume`, `Resumed`, `PendingPause`, `Paused`, `Triggered`, `Started`
`OrdStatus`	string	Y	Identifies current status of order. Valid values: `New`, `PartiallyFilled`, `Filled`, `Canceled`, `PendingCancel`, `Rejected`, `PendingNew`, `PendingReplace`.
`OrderQty`	decimal	Y	Order quantity
`OrdType`	string	Y	Order type. Valid values: `Market`, `Limit`, or `RFQ`
`Price`	decimal	N	Order limit price, required when `OrdType=Limit`
`Currency`	string	Y	Currency of `Quantity`
`LeavesQty`	decimal	Y	Quantity open for further execution. If the OrdStatus is Canceled or Rejected (in which case the order is no longer active) then LeavesQty could be 0, otherwise LeavesQty = OrderQty – CumQty
`CumQty`	decimal	Y	Total quantity filled. Always in Currency units (BTC of a BTC-USD order)
`AvgPx`	decimal	N	Average filled price of the order. Only valid if `CumQty > 0`. Does not include fees
`TimeInForce`	string	Y	Specifies how long the order remains in effect. Valid values: `GoodTillCancel`, `FillAndKill`, or

			`FillOrKill`
`LastPx`	decimal	N	Last price, specified when `ExecType=Trade`
`LastQty`	decimal	N	Last qty, specified when `ExecType=Trade`
`LastAmt`	decimal	N	Last amount, specified when `ExecType=Trade`
`LastFee`	decimal	N	Last fee, specified when `ExecType=Trade`
`LastFeeCurency`	string	N	Last fee currency, specified when `ExecType=Trade`. Note this may be different from the cumulative fee
`CumAmt`	decimal	N	Total amount filled. Always in Amount currency (USD of a BTC-USD order)
`FeeCurency`	string	N	Cumulative fee currency
`OrdRejReason`	string	N	Order reject reason, specified when ExecType=Rejected. Valid values: `UnknownSymbol`, `OrderExceedsLimit`, `TooLateToEnter`, `UnknownOrder`, `DuplicateOrder`, `StaleRequest`, `InternalError`, `BrokerOption`, `RateLimit`, `ForceCancel`, `ImmediateOrderDidNotCross`, `PostOnlyOrderWouldCross`, `InvalidStrategy`, `QuoteRequestTimeout`, `QuoteExpired`
`CxlRejReason`	string	N	Cancel reject reason, specified when `ExecType=CancelRejected` or `ExecType=ReplaceRejected`. Valid values: `UnknownOrder`, `Broker`, `OrderAlreadyInPendingCancelOrPendingReplaceStatus`, `DuplicateClOrdID`, `TooLateToCancel`, `StaleRequest`, `RateLimit`, `InvalidStrategy`, `Other`
`QuoteID`	string	N	Server assigned QuoteID for RFQ Orders
`AmountCurrency`	string	Y	Currency of `Amount`
`SessionID`	string	Y	SessionID that submitted this order
`CancelSessionID`	string	N	If specified, the order will be cancelled when this session is closed
`Strategy`	string	N	Name of the Order Strategy to use
`RFQID`	string	N	Server assigned RFQID for RFQ Orders
`AllowedSlippage`	decimal	N	AllowedSlippage in absolute value (ie. 0.001 = 10 bps) if specified on order
`Text`	string	N	Human readable error message
`CustomerUser`	string	N	The customer user associated with this execution report.

NewOrderSingle

Command used to submit a new order either against any provided liquidity or an open RFQ. Results of this command will be visible in the `Order`, `ExecutionReport` and `Trade` streams which will track the lifecycle of the order.

> *Submit NewOrderSingle*

```
{
  "reqid": 12,
  "type": "NewOrderSingle",
  "data": [
    {
      "Symbol": "BTC-USD",
      "ClOrdID": "4d489920-15da-11ec-b5e7-7f4881f01b7d",
      "Side": "Buy",
      "TransactTime": "2021-09-15T04:06:28.530000Z",
      "OrderQty": "0.5",
      "OrdType": "Limit",
      "Price": "43000",
      "Currency": "BTC",
      "TimeInForce": "GoodTillCancel",
      "Strategy": "Limit",
      "Parameters": {
        "ErrorAction": "Pause"
      }
    }
  ]
}
```

Command Data

Key	Type	Required	Description
`Timestamp`	datetime	N	Timestamp of the message
`Symbol`	string	Y	Symbol of the security to submit an order on
`ClOrdID`	string	Y	Unique Client Order ID for this request, usually a UUID
`Side`	string	Y	Side of the order, either `Buy` or `Sell`
`OrderQty`	decimal	Y	Order quantity in units of Currency
`OrdType`	string	Y	Order type. Valid values: `Market`, `Limit`, or `RFQ`
`Price`	decimal	N	Order limit price, required for `Limit` orders
`TransactTime`	datetime	N	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Currency`	string	N	The currency that the quantity is specified in. If not specified, defaults to the base currency for the symbol
`QuoteID`	string	N	Optional QuoteID to trade on. Must be specified when `OrdType=RFQ`
`RFQID`	string	N	Optional RFQID to trade on. Must be specified when `OrdType=RFQ`
`TimeInForce`	string	Y	Specifies how long the order remains in effect. Valid values: `GoodTillCancel`, `FillAndKill`, or `FillOrKill`
`Strategy`	string	N	Optional order strategy to use. For algorithmic orders `SteadyPace`, `StopLimit' are supported
`AllowedSlippage`	decimal	N	Optional allowed price slippage for Limit order placed against RFQ
`Parameters`	object	N	Optional order strategy parameters

OrderCancelRequest

Command used to cancel an open order. Results of this command will be visible in the `Order` and `ExecutionReport` streams.

> *Submit OrderCancelRequest*

```
{
  "reqid": 13,
  "type": "OrderCancelRequest",
  "data": [
    {
      "ClOrdID": "c0be17e0-15da-11ec-b5e7-7f4881f01b7d",
      "OrderID": "16a55364-4248-4575-aa0b-dddd3f694ca3",
      "TransactTime": "2021-09-15T04:09:42.238000Z"
    }
  ]
}
```

Command Data

Key	Type	Required	Description
`Timestamp`	datetime	N	Timestamp of the message
`Symbol`	string	Y	Symbol of the security to submit an order on
`ClOrdID`	string	Y	Unique Client Order ID for this request, usually a UUID
`OrigClOrdID`	string	N	Client Order ID of the order to cancel, one of `OrderID`, `OrigClOrdID` is required
`OrderID`	string	N	Order ID of the order to cancel, one of `OrderID`, `OrigClOrdID` is required
`TransactTime`	datetime	Y	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`

OrderCancelReplaceRequest

Command used to modify an open order. Results of this command will be visible in the `Order` and `ExecutionReport` streams.

> *Submit OrderCancelReplaceRequest*

```
{
  "reqid": 14,
  "type": "OrderCancelReplaceRequest",
  "data": [
    {
      "Price": "44000",
      "Symbol": "BTC-USD",
      "OrderID": "16a55364-4248-4575-aa0b-dddd3f694ca3",
      "ClOrdID": "b5e52570-15da-11ec-b5e7-7f4881f01b7d",
      "TransactTime": "2021-09-15T04:09:24.039000Z",
      "OrderQty": "0.5",
      "Parameters": {
        "ErrorAction": "Pause"
      }
    }
  ]
}
```

Command Data

Key	Type	Required	Description
`Timestamp`	datetime	N	Timestamp of the message
`Symbol`	string	Y	Symbol of the security to submit an order on
`ClOrdID`	string	Y	Unique Client Order ID for this request, usually a UUID
`OrigClOrdID`	string	N	Client Order ID of the order to cancel, one of `OrderID`, `OrigClOrdID` is required
`OrderID`	string	N	Order ID of the order to cancel, one of `OrderID`, `OrigClOrdID` is required
`OrderQty`	decimal	Y	Order quantity in units of Currency
`Price`	decimal	N	Order limit price, required for `Limit` orders
`TransactTime`	datetime	Y	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Strategy`	string	N	Optional order strategy to use. For algorithmic orders `Pegged`, `TWAP`, `FullAmount`, `TimeSliced`, and `SteadyPace` are supported
`EndTime`	datetime	N	Optional expire time for the order. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Parameters`	object	N	Optional order strategy parameters

OrderControlRequest

Command used to `Pause` or `Resume` an open order. Results of this command will be visible in the `Order` and `ExecutionReport` streams.

> *Submit OrderControlRequest*

```
{
  "reqid": 17,
  "type": "OrderControlRequest",
  "data": [
    {
      "ClOrdID": "9eb07120-15da-11ec-b5e7-7f4881f01b7d",
      "OrderID": "16a55364-4248-4575-aa0b-dddd3f694ca3",
      "Action": "Pause"
    }
  ]
}
```

Command Data

Key	Type	Required	Description
`Timestamp`	datetime	N	Timestamp of the message
`ClOrdID`	string	Y	Unique Client Order ID for this request, usually a UUID
`OrigClOrdID`	string	N	Client Order ID of the order to submit the control request for, one of `OrderID`, `OrigClOrdID` is required
`OrderID`	string	N	Order ID of the order to submit the control request for, one of `OrderID`, `OrigClOrdID` is required
`TransactTime`	datetime	Y	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Action`	string	Y	Action to apply on the order, either `Pause` or `Resume`

Order State Change Matrices

Order State Change Matrices

The orders API follows FIX 4.4 semantics closely. The main difference is that instead of sending separate OrderCancelReject messages, two extra ExecType are supported: CancelRejected and ReplaceRejected. The order state change matrices below are heavily inspired by the FIX documentation.

Filled order

Time	Request (ClOrdID,OrigClOrdID)	Response (ClOrdID,OrigClOrdID)	ExecType	OrdStatus	OrderQty	CumQty	LeavesQty	LastQty	Comment
1	NewOrderSingle(A)				10				
2		ExecutionReport(A)	PendingNew	PendingNew	10	0	10		
3		*ExecutionReport(A)*	*Rejected*	*Rejected*	10	0	10		*If order is rejected*
3		ExecutionReport(A)	New	New	10	0	10		
4		ExecutionReport(A)	Trade	PartiallyFilled	10	2	8	2	
5		ExecutionReport(A)	Trade	PartiallyFilled	10	3	7	1	
6		ExecutionReport(A)	Trade	Filled	10	10	0	7	

Orders with CumQty > 0 will receive an Execution Report that will have `CumQty` for the filled size of the order. Orders that are partially filled and canceled will receive an ExecutionReport with a terminal state (Canceled) and a CumQty > 0

Canceled order

Time	Request (ClOrdID,OrigClOrdID)	Response (ClOrdID,OrigClOrdID)	ExecType	OrdStatus	OrderQty	CumQty	LeavesQty	LastQty	Comment
1	NewOrderSingle(A)				10				
2		ExecutionReport(A)	PendingNew	PendingNew	10	0	10		
3		*ExecutionReport(A)*	*Rejected*	*Rejected*	10	0	10		*If order is rejected*
3		ExecutionReport(A)	New	New	10	0	10		
4	OrderCancelRequest(B,A)								
5		ExecutionReport(B,A)	*CancelRejected*	*New*	10	0	10		*If cancel is rejected*
5		ExecutionReport(B,A)	PendingCancel	PendingCancel	10	0	10		
6		ExecutionReport(B,A)	*CancelRejected*	*New*	10	0	10		*If cancel is rejected*
7		ExecutionReport(B,A)	Canceled	Canceled	10	0	0		

Replace to increase quantity

Time	Request (ClOrdID,OrigClOrdID)	Response (ClOrdID,OrigClOrdID)	ExecType	OrdStatus	OrderQty	CumQty	LeavesQty	LastQty	Comment
1	NewOrderSingle(A)				10				
2		ExecutionReport(A)	PendingNew	PendingNew	10	0	10		
3		ExecutionReport(A)	New	New	10	0	10		
4	OrderCancelReplaceRequest(B,A)				11				
5		ExecutionReport(B,A)	*ReplaceRejected*	*New*	10	0	10		*If replace is rejected*
5		ExecutionReport(B,A)	PendingReplace	PendingReplace	10	0	10		
6		ExecutionReport(B,A)	*CancelRejected*	*New*	10	0	10		*If replace is rejected*
7		ExecutionReport(B,A)	Replaced	New	11	0	0		
8		ExecutionReport(B)	Trade	PartiallyFilled	11	1	10	1	

Replace during fill

Time	Request (ClOrdID,OrigClOrdID)	Response (ClOrdID,OrigClOrdID)	ExecType	OrdStatus	OrderQty	CumQty	LeavesQty	LastQty	Comment
1	NewOrderSingle(A)				10				
2		ExecutionReport(A)	PendingNew	PendingNew	10	0	10		
3		ExecutionReport(A)	New	New	10	0	10		
4	OrderCancelReplaceRequest(B,A)				8				
5		ExecutionReport(A)	Trade	PartiallyFilled	10	1	9	1	Fill before replace is received
6		ExecutionReport(B,A)	PendingReplace	PendingReplace	10	1	9		
7		ExecutionReport(A)	Trade	PendingReplace	10	3	7	2	Fill before replace is processed
8		ExecutionReport(B,A)	Replaced	New	8	3	5		
9		ExecutionReport(B)	Trade	Filled	8	8	0	5	

Post Trade

Trade

Upon Subscription, the server will return all Trade for this client as long as the optional constraints are satisfied. Server will also send a Trade update when the order receives more fills or new fills arrive on a new order that came in as long as the optional constraints are satisfied.

> *Request for Trade Sub*


```
{
  "reqid": 8,
  "type": "subscribe",
  "streams": [
    {
      "name": "Trade",
      "StartDate": "2021-09-14T00:00:00.000000Z"
    }
  ]
}
```


> *Response stream is open*


```
{
  "reqid": 8,
  "type": "Trade",
  "ts": "2021-09-14T22:29:32.255949Z",
  "initial": true,
  "seqNum": 1,
  "data": [
    {
      "Timestamp": "2021-09-14T22:23:11.989845Z",
      "Symbol": "BTC-USD",
      "OrderID": "1b6b882b-e2fc-4774-ad2d-9db4df536f29",
      "TradeID": "a0ec46f2-6d0e-451a-a4f1-2a0be7d32d26",
      "Side": "Buy",
      "TransactTime": "2021-09-14T22:23:11.964531Z",
      "ExecType": "Trade",
      "Currency": "BTC",
      "Price": "47085.090668824440",
      "Quantity": "0.20000000",
      "Amount": "9417.02",
      "Fee": "0",
      "CustomerUser": "tom@company.com",
      "TradeStatus": "Confirmed",
      "AggressorSide": "Buy",
      "AmountCurrency": "USD",
      "DealtCurrency": "BTC"
    },
    {
      "Timestamp": "2021-09-14T22:26:44.515692Z",
      "Symbol": "BTC-USD",
      "OrderID": "b35b1c3b-a304-4224-919f-9db1319de188",
      "TradeID": "17f9da19-d3bd-4ba2-b29a-1f31516458f2",
      "Side": "Buy",
      "TransactTime": "2021-09-14T22:26:44.485836Z",
      "ExecType": "Trade",
      "Currency": "BTC",
      "Price": "46945.3659525",
      "Quantity": "0.10000000",
      "Amount": "4694.54",
      "Fee": "0",
```

```
      "CustomerUser": "tom@company.com",
      "TradeStatus": "Confirmed",
      "AggressorSide": "Buy",
      "AmountCurrency": "USD",
      "DealtCurrency": "BTC"
    }
  ]
}
```

Request stream of Trade messages.

Subscription Parameters

Key	Type	Required	Description
`StartDate`	string	N	If provided, the subscription will return trades that were executed after this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`EndDate`	string	N	If provided, the subscription will return trades that were executed before this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Symbol`	string	N	If provided, Symbol of the security to get the orders for
`OrderID`	string	N	If provided, filter by `OrderID`
`RFQID`	string	N	If provided, filter by `RFQID`

Response

Key	Type	Required	Description
`Timestamp`	datetime	Y	Timestamp of the message
`OrderID`	string	Y	Server assigned Order ID, will be a UUID
`TradeID`	string	Y	Server assigned Trade ID, will be a UUID
`Side`	string	Y	"Buy" or "Sell"
`TransactTime`	datetime	Y	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`ExecType`	string	Y	Describes the specific execution (e.g. Pending Cancel) while OrdStatus will always identify the current order status (e.g. Partially Filled). Valid values: `New`, `Trade`, `Canceled`, `Replaced`, `PendingCancel`, `Stopped`, `Rejected`, `PendingNew`, `Restated`, `PendingReplace`, `CancelRejected`, `ReplaceRejected`, `PendingResume`, `Resumed`, `PendingPause`, `Paused`, `Triggered`, `Started`
`Currency`	string	Y	Currency of `Quantity`
`Price`	decimal	N	Order limit price, required when `OrdType=Limit`
`Quantity`	decimal	Y	Trade quantity
`Amount`	decimal	Y	Trade amount
`Fee`	decimal	Y	Trade fee
`FeeCurency`	string	N	Cumulative fee currency
`TradeStatus`	string	Y	Status of the trade. Valid values: `Pending`, `Confirmed`, `Canceled`
`QuoteID`	string	N	Server assigned QuoteID for RFQ Orders
`AmountCurrency`	string	Y	Currency of `Amount`
`RFQID`	string	N	Server assigned RFQID for RFQ Orders

`CustomerUser`	string	N	The customer user associated with this trade.

Response

Key	Type	Required	Description
`CustomerUserID`	string	Y	Server assigned ID of this Customer User
`Email`	string	Y	Email of Customer User
`DisplayName`	string	Y	Friendly name for customer user
`Config`	object[]	Y	List of user config settings
→ `Timestamp`	datetime	Y	Timestamp of config setting update
→ `CustomerUserID`	string	Y	Customer User for this config setting
→ `Mode`	string	Y	Setting whether config is enabled, values `Enabled` or `Disabled`
→ `Key`	string	Y	User config key
→ `Value`	string	Y	User config value

RFQ

Overview

This section outlines how to request quotes from multiple dealers simultaneously and accept them using the SDM API. To send a quote request, first subscribe to the Quote stream, and the ExecutionReport stream as with orders. All quote responses are sent as Quote updates, and all order responses and updates are sent as `ExecutionReport`.

To request a quote, send a QuoteRequest message. The server responds with one or many Quote messages as the streaming prices update. An in-progress Quote can be cancelled by sending a QuoteCancelRequest message. Quotes time out automatically after 3 minutes by default.

To accept a quote, send a NewOrderSingle (RFQ) message.

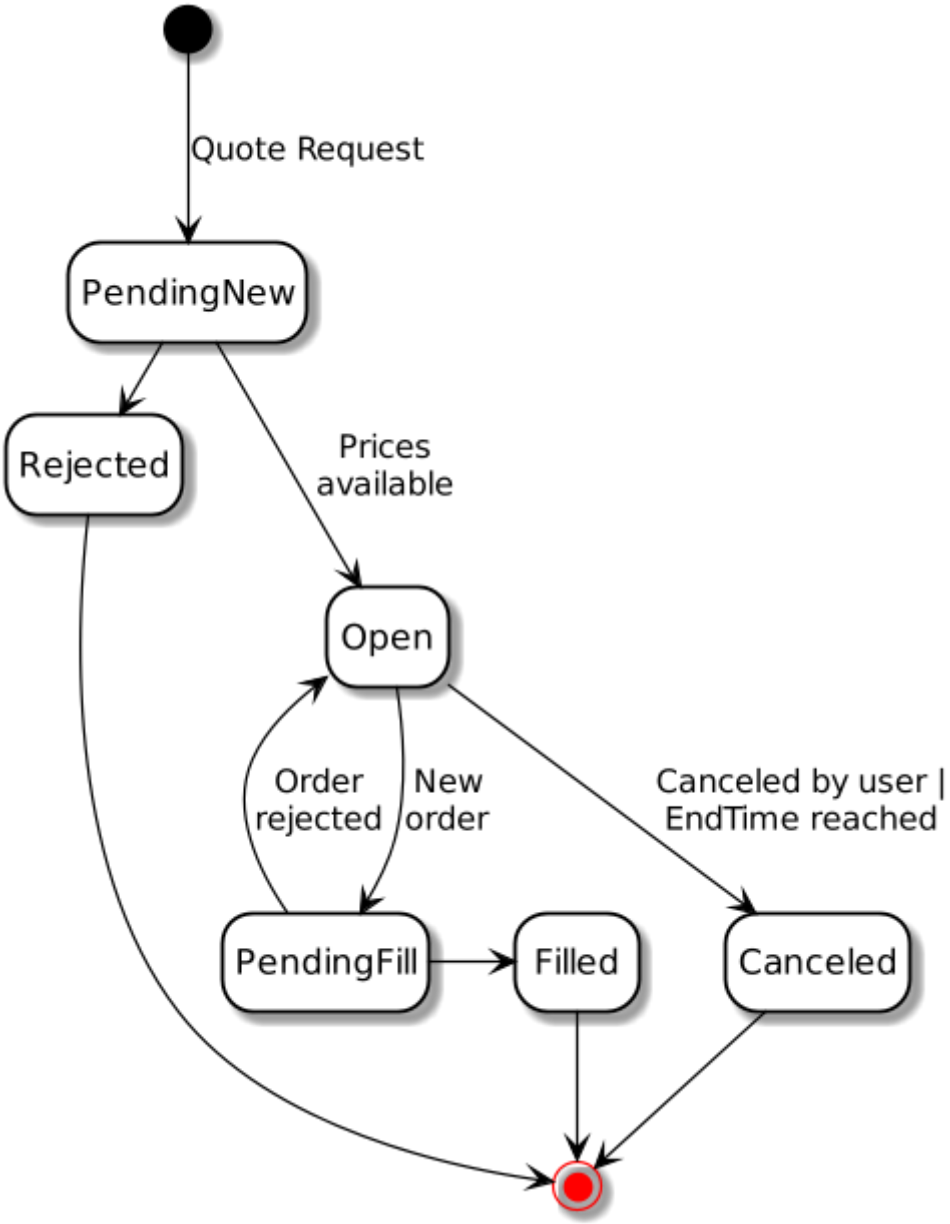
A typical workflow to request and accept a quote would be:

1. Subscribe to `Quote`.
2. Subscribe to `ExecutionReport`.
3. Subscribe to `Trade` with `StartDate` of the `Timestamp` of the last `Trade` processed to recover any missed trades.
4. **To request a quote, first generate a new `QuoteReqID` (see instructions below). Send a `QuoteRequest` specifying the `QuoteReqID`, `OrderQty`, `Currency` etc. for that request.**
5. `Quote` updates will be received that are able to be priced.
6. To accept a quote that is received, send a `NewOrderSingle` referencing the `RFQID` and parameters depending on `OrdType`
7. Fills will be received as `ExecutionReport` with `ExecType=Trade`.
8. `QuoteStatus` will change to `Filled` when the `Quote` is successfully accepted.

To generate a `QuoteReqID`, we recommend using a UUID4 or another globally unique identifier. `QuoteReqID` must be unique and with length less than or equal to 36 characters.

State Machine

See below for a state diagram for `QuoteStatus`:


Quote Request
PendingNew
Rejected
Prices
available
Open
Order
rejected
New
order
Canceled by user |
EndTime reached
PendingFill
Filled
Canceled

Quote

Request stream of Quotes.

> *Request for Quote Sub*

```
{
  "reqid": 9,
  "type": "subscribe",
  "streams": [
    {
      "name": "Quote"
    }
  ]
}
```

> *Response stream is open*

```
{
  "reqid": 9,
  "type": "Quote",
  "ts": "2021-09-14T22:31:27.214025Z",
  "initial": false,
  "seqNum": 3,
  "data": [
    {
      "Timestamp": "2021-09-14T22:31:27.204209Z",
      "Symbol": "BTC-USD",
      "Currency": "BTC",
      "RFQID": "8688ceab-ea67-416c-8ce1-9b63d6c0fe4e",
      "QuoteReqID": "7feb22f0-15ab-11ec-b0a2-2554a9e1e7a4",
      "QuoteStatus": "Open",
      "QuoteID": "20e240c5-9a82-4731-809e-91b3a2bf2f49",
      "SubmitTime": "2021-09-14T22:31:27.183751Z",
      "OrderQty": "0.30000000",
      "AmountCurrency": "USD",
      "EndTime": "2021-09-14T22:31:42.183751Z",
      "BidPx": "46836.27",
      "BidAmt": "14050.88",
      "OfferPx": "46879.47",
      "OfferAmt": "14063.85",
      "ValidUntilTime": "2021-09-14T22:31:28.204209Z",
      "CustomerUser": "tom@company.com"
    }
  ]
}
```

Subscription Parameters

Key	Type	Required	Description
`StartDate`	string	N	If provided, the subscription will return quotes for rfqs that were submitted after this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`EndDate`	string	N	If provided, the subscription will return quotes for rfqs that were submitted before this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Symbol`	string	N	If provided, Symbol of the security to get the orders for
`RFQID`	string	N	If provided, filter by `RFQID`

Response

Key	Type	Required	Description
`Timestamp`	datetime	Y	Timestamp of the message
`Symbol`	string	Y	Symbol of the order security
`Currency`	string	Y	Currency of `Quantity`
`RFQID`	string	N	Server assigned RFQID for RFQ Orders
`QuoteReqID`	string	Y	Client assigned Quote Req ID for the quote request
`QuoteStatus`	string	Y	Status of the quote, one of `PendingNew`, `Open`, `PendingCancel`, `Canceled`, `PendingFill`, `Filled`, `Rejected`
`QuoteID`	string	Y	Server assigned ID for this quote update
`QuoteRequestRejectReason`	string	Y	Quote request reject reason, specified when `ExecType=Rejected`. Valid values: `UnknownSymbol`, `OrderExceedsLimit`, `TooLateToEnter`, `UnknownOrder`, `DuplicateOrder`, `StaleRequest`, `InternalError`, `BrokerOption`, `RateLimit`, `ForceCancel`, `ImmediateOrderDidNotCross`, `PostOnlyOrderWouldCross`, `InvalidStrategy`, `QuoteRequestTimeout`, `QuoteExpired`
`SubmitTime`	datetime	Y	Time of original quote request submission
`OrderQty`	decimal	Y	Order quantity
`AmountCurrency`	string	Y	Currency of `Amount`
`EndTime`	datetime	Y	End time of RFQ; an ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Side`	string	Y	"Buy" or "Sell"
`TradedPx`	decimal	N	The price of the trade if the quote is filled. This will be an all-in price that includes fees
`TradedQty`	decimal	N	The quantity of the trade if the quote is filled in units of `Currency`
`TradedAmt`	decimal	N	The amount of the trade if the quote is filled in units of `AmountCurrency`
`TradedSide`	string	N	The side of the trade if the quote is filled
`BidPx`	decimal	N	Bid quote price
`BidAmt`	decimal	N	Bid quote amount in units of `AmountCurrency`
`OfferPx`	decimal	N	Offer quote price
`OfferAmt`	decimal	N	Offer quote amount in units of `AmountCurrency`

`ValidUntilTime`	datetime	Y	The expire time of the quote. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`Text`	string	N	Human readable error message
`CustomerUser`	string	N	The customer user associated with this quote.

QuoteRequest

Command used to request a quote thereby opening a new RFQ. Results of this command will be visible in the `Quote` stream.

> *Submit QuoteRequest*

```
{
  "reqid": 15,
  "type": "QuoteRequest",
  "data": [
    {
      "QuoteReqID": "8e9540d0-15db-11ec-b5e7-7f4881f01b7d",
      "OrderQty": "0.75",
      "Currency": "BTC",
      "Symbol": "BTC-USD",
      "TransactTime": "2021-09-15T04:15:27.581000Z",
      "Parameters": {}
    }
  ]
}
```

Command Data

Key	Type	Required	Description
`Timestamp`	datetime	N	Timestamp of the message
`Symbol`	string	Y	Symbol of the security to submit an order on
`Currency`	string	Y	The currency that the quantity is specified in.
`QuoteReqID`	string	Y	Unique ID for this request, usually a UUID
`Side`	string	N	Optional side for a one sided request, either "Buy" or "Sell". Leave blank for a two way quote
`OrderQty`	decimal	Y	Requested quantity in units of Currency
`TransactTime`	datetime	N	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`

QuoteCancelRequest

Command used to cancel an open RFQ. Results of this command will be visible in the `Quote` stream.

> *Submit QuoteCancelRequest*

```
{
  "reqid": 16,
  "type": "QuoteCancelRequest",
  "data": [
    {
      "QuoteReqID": "8e9540d0-15db-11ec-b5e7-7f4881f01b7d",
      "RFQID": "f47bfb99-4d1e-4179-b62c-a96e83cc93ba",
      "TransactTime": "2021-09-15T04:15:24.581000Z"
    }
  ]
}
```

Command Data

Key	Type	Required	Description
`QuoteReqID`	string	Y	`QuoteReqID` from the request to cancel
`RFQID`	string	Y	`RFQID` to cancel, required and must align with `QuoteReqID`
`TransactTime`	datetime	Y	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`

Balances and Credit

Balance

Request for stream of Balance updates

> *Request for Balance Sub*

```
{
  "reqid": 11,
  "type": "subscribe",
  "streams": [
    {
      "name": "Balance",
      "EquivalentCurrency": "USD"
    }
  ]
}
```

> *Response stream is open*


```
{
  "reqid": 11,
  "type": "Balance",
  "ts": "2021-09-16T16:17:06.892914Z",
  "initial": true,
  "seqNum": 1,
  "data": [
    {
      "Currency": "USD",
      "Amount": "-1928479.77953598",
      "AvailableAmount": "-1928479.77953598",
      "Equivalent": {
        "Currency": "USD",
        "Amount": "-1928479.78",
        "AvailableAmount": "-1928479.78"
      }
    },
    {
      "Currency": "BTC",
      "Amount": "40.15480673",
      "AvailableAmount": "40.15480673",
      "Equivalent": {
        "Currency": "USD",
        "Amount": "1930241.56",
        "AvailableAmount": "1930241.56"
      }
    },
    {
      "Currency": "ETH",
      "Amount": "11.00000000",
      "AvailableAmount": "11.00000000",
      "Equivalent": {
        "Currency": "USD",
        "Amount": "39753.78",
        "AvailableAmount": "39753.78"
      }
    },
    {
      "Currency": "BCH",
      "Amount": "23.68787010",
```

```
      "AvailableAmount": "23.68787010",
      "Equivalent": {
        "Currency": "USD",
        "Amount": "15066.67",
        "AvailableAmount": "15066.67"
      }
    },
    {
      "Currency": "ADA",
      "Amount": "10330.26832917",
      "AvailableAmount": "10330.26832917",
      "Equivalent": {
        "Currency": "USD",
        "Amount": "25340.27",
        "AvailableAmount": "25340.27"
      }
    }
  ]
}
```


Subscription Parameters

Key	Type	Required	Description
`Currencies`	string[]	N	Optional list of currencies to filter these balance updates by, default is all available currencies
`EquivalentCurrency`	string	N	If provided, will provide converted equivalent amounts of each currency in the provided Equivalent Currency

Response

Key	Type	Required	Description
`Currency`	string	Y	Currency of this balance update
`Amount`	decimal	Y	Current balance amount in the specified currency
`AvailableAmount`	decimal	Y	Amount available for trading right now
`Equivalent`	object[]	N	If requested equivalent balance amount, will provide a Balance update in the specified equivalent currency
→ `Currency`	string	N	Requested Equivalent Currency
→ `Amount`	decimal	N	Amount of this balance in the equivalent currency
→ `AvailableAmount`	decimal	N	Equivalent amount of amount available for trading right now

Exposure

Request for stream of Exposure updates

> *Request for Exposure Sub*

```
{
  "reqid": 11,
  "type": "subscribe",
  "streams": [
    {
      "name": "Exposure"
    }
  ]
}
```

> *Response stream is open*

```
{
  "reqid": 11,
  "type": "Exposure",
  "ts": "2021-09-16T16:17:06.892914Z",
  "initial": true,
  "seqNum": 1,
  "data": [
    {
      "ExposureCurrency": "USD",
      "Exposure": "1000000",
      "ExposureLimit": "5000000",
      "Status": "Online",
      "Timestamp": "2021-09-16T16:17:06.892914Z"
    }
  ]
}
```

Subscription Parameters

None

Response

Key	Type	Required	Description
`ExposureCurrency`	string	Y	The currency all exposure values are expressed in
`Exposure`	decimal	Y	Current credit utilization expressed in the `ExposureCurrency`
`ExposureLimit`	decimal	Y	Credit limit expressed in the `ExposureCurrency`
`Status`	status	Y	Status of the exposure information. Possible values: `Online`, `Offline`
`Timestamp`	status	Y	The last time the credit was updated as an ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`.

BalanceTransaction

Request stream of Balance Transaction Updates

> *Request for BalanceTransaction Sub*

```
{
  "reqid": 9,
  "type": "subscribe",
  "streams": [
    {
      "name": "BalanceTransaction"
    }
  ]
}
```

> *Response stream is open*


```
{
  "reqid": 9,
  "type": "BalanceTransaction",
  "ts": "2023-02-10T16:49:18.300588Z",
  "initial": true,
  "seqNum": 1,
  "data": [
    {
      "Timestamp": "2023-02-10T16:50:12.562757Z",
      "ClReqID": "06b78113c1454174b9e403a1245d67ea",
      "TransactionID": "1D2FBR4J00R00",
      "Revision": 0,
      "TransactionType": "Withdrawal",
      "MarketAccount": "default",
      "Quantity": "1",
      "Currency": "ETH",
      "Status": "PendingApproval",
      "LastUpdateTime": "2023-02-10T16:50:12.556229Z"
    }
  ]
}
```

Subscription Parameters

None

Response

Key	Type	Required	Description
`Timestamp`	datetime	Y	Timestamp of the message
`Currency`	string	Y	Currency of `Quantity`
`ClReqID`	string	Y	Client assigned Request ID
`Status`	string	Y	Status of the quote, one of `PendingApproval`, `PendingTransfer`, `Rejected`, `Completed`
`TransactionID`	string	Y	Server assigned ID for this transaction
`LastUpdateTime`	datetime	Y	Time of most recent update to the Transaction
`Quantity`	decimal	Y	Trnasaction quantity
`MarketAccount`	string	Y	Account to apply this transaction to
`Revision`	string	Y	Revision of this transaction

NewWithdrawRequest

Command used to request a new withdrawal

> *Submit NewWithdrawRequest*

```
{
  "reqid": 34,
  "type": "NewWithdrawRequest",
  "data": [
    {
      "Quantity": "1",
      "Currency": "ETH",
      "MarketAccount": "default",
      "RoutingInfo": {
        "WalletAddress": "XXYYZZ",
        "Memo": "",
        "DestinationTag": ""
      },
      "ClReqID": "06b78113c1454174b9e403a1245d67ea"
    }
  ]
}
```

Command Data

Key	Type	Required	Description
`Currency`	string	Y	The currency that the quantity is specified in.
`ClReqID`	string	Y	Unique ID for this request, usually a UUID
`Quantity`	decimal	Y	Requested quantity in units of Currency
`TransactTime`	datetime	N	An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`
`RoutingInfo`	object	N	Provides routing info for the transaction such as WalletAddress

User Administration

User

Request for updates for this customer user. No subscription parameters.

> *Request for User Sub*


```
{
  "reqid": 10,
  "type": "subscribe",
  "streams": [
    {
      "name": "User"
    }
  ]
}
```


> *Response stream is open*


```
{
  "reqid": 10,
  "type": "User",
  "ts": "2021-09-15T03:33:07.545223Z",
  "initial": true,
  "seqNum": 1,
  "data": [
    {
      "CustomerUserID": "174K1Q9GC0C00",
      "Email": "tom@company.com",
      "DisplayName": "Tom",
      "Config": [
        {
          "Timestamp": 1628004805337309202,
          "CustomerUserID": "174K1Q9GC0C00",
          "Key": "recentSymbols",
          "Value": "{\"BTC-USD\":[1626111502357],\"ADA-USD\":[1626732116743],\"BCH-USD\":[1628004751665]}"
        },
        {
          "Timestamp": 1628004805377207692,
          "CustomerUserID": "174K1Q9GC0C00",
          "Key": "symbol",
          "Value": "\"BTC-USD\""
        }
      ]
    }
  ]
}
```

UserConfig

Command used to update user config on behalf of session user. Results of this command will be visible in the `User` stream.

> *Submit UserConfig*

```
{
  "reqid": 18,
  "type": "UserConfig",
  "data": [
    {
      "Key": "symbol",
      "Value": "\"ETH-BTC\""
    }
  ]
}
```

Command Data

Key	Type	Required	Description
`Timestamp`	datetime	N	Timestamp of the message
`Mode`	string	N	Optional mode on config, either `Enabled` or `Disabled`
`Key`	string	Y	Key of config
`Value`	string	N	Value of config

OHLCV

The OHLCV API provides data on the specified symbols.

GET /v1/symbols/<_symbol_>/ohlcv/<_resolution_>

> *Get the open, high, low, close, and volume parameters for a given symbol and resolution*

```
GET /v1/symbols/BTC-USD/ohlcv/1h
```

Query Parameters

Name	Type	Description
`startDate`	string	If provided, will return data starting at this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`.
`endDate`	string	If provided, will return data ending at this time. An ISO-8601 UTC string of the form `2019-02-13T05:17:32.000000Z`.

> *Example response*


```
{
  "type": "OHLCV",
  "ts": "2019-09-16T14:57:19.785823Z",
  "data": [{
      "Timestamp":"2023-02-27T10:52:00.000000Z",
      "Symbol":"BTC-USD",
      "Open":"23413.30000",
      "High":"23418.24",
      "Low":"23399.46",
      "Close":"23407.10000",
      "Volume":"0.49309984",
      "Ticks":74
        }
      ]
    }
  ]
}
```


Response

Key	Type	Description
`Timestamp`	datetime	The start time for this interval
`Symbol`	string	A unique symbol for this security.
`Volume`	decimal	Total base-currency volume traded (if any) for the symbol in the time interval.
`Open`	decimal	First trade price (if any) for the symbol in the time interval.
`Close`	decimal	Last trade price (if any) for the symbol in the time interval.
`High`	decimal	Highest traded price (if any) for the symbol in the time interval.
`Low`	decimal	Lowest traded price (if any) for the symbol in the time interval.
`Ticks`	decimal	Total number of ticks for the symbol in the time interval

Changelog

v1.4.0

- Adds new REST API endpoint OHLCV

v1.3.0

- Adds new subscription BalanceTransaction
- Adds new command NewWithdrawRequest

v1.2.0

- Change `TransactTime` on QuoteRequest and <a href=#executionreport">ExecutionReport to be optional. Please note that this documentation change reflects existing behavior in the API.
- Change `Currency` on QuoteRequest to be required. Please note that this documentation change reflects existing behavior in the API.
- Add `MinAmtIncrement` field to Security.

v1.1.0

- Change `MaximumSize` for Security to be optional field.