REST API

Authentication

https://api.bittxn.com/api/v1/b2trade/auth [POST]

Request params

```
   email: string [required]
   password: string [required]
```

Response data: array

```
        id: int
        email: string
        country_code: int
        settings: array
        status: int
        logged_at: string
        created_at: string
        updated_at: string
        nickname: int
        photo: string
        token: string
        user_id: int
        session_token: string
```

WebSocket API

wss://wss.bittxn.com/WSGateway/?session_token=session_token

Message Frame

Wrap all calls in a JSON-formatted frame object. Responses are similarly wrapped.
{ “m”:0, “i”:0, “n”:”function_name”, “o”:”payload” }

Where:

m message type	integer. The type of the message: 0 request 1 reply 2 subscribe to event 3 event 4 unsubscribe from event 5 error
m message type	long integer. The sequence number identifies an individual request, or requestand-response pair, to your application. A non-zero sequence number is required, but the numbering scheme you use is up to you. No arbitrary sequence numbering scheme is enforced. Best practices: A client-generated API call (of message types 0, 2, and 4) should: Carry an even sequence number. Begin at the start of each user session. Be unique within each user session. Begin with 2 (2, 4, 6, 8). Message types 1 (reply), 3 (event), and 5 (error) are generated by the server. These messages echo the sequence number of the message to which they respond.
n function name	string. The function name is the name of the function that you are calling or that the server responds to. The server echoes your call.
o payload	Payload is a JSON-formatted string containing the data being sent with the message. Payload may consist of request parameters (string-value pairs) or response parameters.

Example

When sending a request in the frame to the software using JavaScript, a call looks like:

```
    var frame = { “m”:0, “i”:0, “n”:”function name”, “o”:”” };
    var requestPayload = { “Parameter1”:”Value”, “Parameter2”:0 };
    frame.o = json.Stringify(requestPayload);
    WS.Send(json.Stringify(frame));
```

When receiving a frame from the software, use the frame to determine the context, and then unwrap the content:

```
    var frame = json.Parse(wsMessage);
    if (frame.m == 1) //message of type reply
    { //This is a Reply
        if (frame.n == “WebAuthenticateUser”) {
            var LoginReply = json.Parse(frame.o);
            if (LoginReply.Authenticated) {
                var user = LoginReply.User;
            }
        }
    }
```

Bittxn

WebAuthenticateUser

WebAuthenticateUser authenticates a user (logs in a user) for the current websocket session. You must call WebAuthenticateUser in order to use the calls in this document not otherwise shown as "No authentication required."

Request

```
{ "SessionToken": "token" }
```

Where:

SessionToken	string. token returned by REST API b2trade/auth method.

Response
Unsuccessful response:

```
{ "Authenticated": false }
```

Where:

Authenticated	Boolean. The default response is false for an unsuccessful authentication

A successful response returns the following (with UserId and SessionToken simulated):

```
{
    "Authenticated": true,
    "SessionToken":"7d0ccf3a-ae63-44f5-a409-2301d80228bc",
    "UserId": 1
}
```

Where:

Authenticated	Boolean. The response is true for a successful authentication.
SessionToken	string. SessionToken uniquely identifies the session on the OMS. By returning the SessionToken in the response, the user can log in again if the session is interrupted without going through two-factor authentication.
UserId	integer. Returns the user ID of the authenticated user

Bittxn

GetInstruments

No authentication required
Retrieves an array of instrument objects describing all instruments available on a trading venue to the user. An instrument is a pair of exchanged products (or fractions of them) such as US dollars and ounces of gold. See "Products and Instruments" on page 4 for more information about how products and instruments differ.

Request
{ "OMSId": 1 }
Where:

OMSId	1

Response
The response for GetInstruments is an array of objects describing all the instruments available to the authenticated user on the Order Management System.

Where:

OMSId	1
InstrumentId	long integer. The ID of the instrument.
Symbol	string. Trading symbol of the instrument
Product1	integer. The first product comprising the instrument. For example, USD in a USD/ BitCoin instrument.
Product1Symbol	string. The symbol for Product 1 on the trading venue. For example, USD.
Product2	integer. The second product comprising the instrument. For example, BitCoin in a USD/BitCoin instrument.
Product2Symbol	string. The symbol for Product 2 on the trading venue. For example, BTC.
InstrumentType	string. The type of the instrument. All instrument types currently are standard, an exchange of one product for another (or unknown, an error condition), but this may expand to new types in the future. Unknown Standard
VenueInstrumentId	long integer. If the instrument trades on another trading venue to which the user has access, this value is the instrument ID on that other venue. See VenueId.
VenueId	integer. The ID of the trading venue on which the instrument trades, if not this venue. See VenueInstrumentId.
SortIndex	integer. The numerical position in which to sort the returned list of instruments on a visual display
SessionStatus	string. Is the market for this instrument currently open and operational? Returns one of: Unknown Running Paused Stopped Starting
PreviousSessionStatus	string. What was the previous session status for this instrument? One of: Unknown Running Paused Stopped Starting
SessionStatusDateTime	string. The time and date at which the session status was reported, in ISO 8601 format.
SelfTradePrevention	Boolean. An account trading with itself still incurs fees. If this instrument prevents an account from trading the instrument with itself, the value returns true; otherwise defaults to false.
QuantityIncrement	integer. The number of decimal places for the smallest quantity of the instrument that can trade (analogous to smallest lot size). For example, the smallest increment of a US Dollar that can trade i s 0.01 (one cent, or 2 decimal places). Current maximum is 8 decimal places. The default is 0.

GetProducts

No authentication required
Returns an array of products available on the trading venue. A product is an asset that is tradable or paid out.

Request
```
{ “OMSId”: 1 }
```
Where:

OMSId	1

Response
The response returns an array of objects, one object for each product available on the Order Management System

Where:

OMSId	integer. The ID of the Order Management System that offers the product.
ProductId	long integer. The ID of the product.
Product	string. “Nickname” or shortened name of the product. For example, NZD (New Zealand Dollar).
ProductType	string. The nature of the product. One of: 0 Unknown (an error condition) 1 NationalCurrency 2 CryptoCurrency 3 Contract
DecimalPlaces	integer. The number of decimal places in which the product is divided. For example, US Dollars are divided into 100 units, or 2 decimal places. Other products may be different. Burundi Francs use 0 decimal places and the Rial Omani uses 3.
TickSize	integer. Minimum tradable quantity of the product. See also GetInstrument, where this value is called QuantityIncrement. For example, with a US Dollar, the minimal tradable quantity is $0.01.
NoFees	Boolean. Shows whether trading the product incurs fees. The default is false; that is, if NoFees is false, fees will be incurred. If NoFees is true, no fees are incurred

SendOrder

Creates an order.

Request

```
{
    "AccountId": 5,
    "ClientOrderId": 99,
    "Quantity": 1,
    "DisplayQuantity": 0,
    "UseDisplayQuantity": true,
    "LimitPrice": 95,
    "OrderIdOCO": 0,
    "OrderType": 2,
    "PegPriceType": 1,
    "InstrumentId": 1,
    "TrailingAmount": 1.0,
    "LimitOffset": 2.0,
    "Side": 0,
    "StopPrice": 96,
    "TimeInForce": 1,
    "OMSId": 1
}
```

Where:

AccountId	integer. The ID of the account placing the order.
ClientOrderId	long integer. A user-assigned ID for the order (like a purchase-order number assigned by a company). This ID is useful for recognizing future states related to this order. ClientOrderId defaults to 0.
Quantity	real. The quantity of the instrument being ordered.
DisplayQuantity	real. The quantity available to buy or sell that is publicly displayed to the market. To display a DisplayQuantity value, an order must be a Limit order with a reserve.
UseDisplayQuantity	Boolean. If you enter a Limit order with a reserve, you must set UseDisplayQuantity to true.
LimitPrice	real. The price at which to execute the order, if the order is a Limit order.
OrderIdOCO	integer. One Cancels the Other — If this order is order A, OrderIdOCO refers to the order ID of an order B (which is not the order being created by this call). If order B executes, then order A created by this call is canceled. You can also set up order B to watch order A in the same way, but that may require an update to order B to make it watch this one, which could have implications for priority in the order book.
OrderType	integer. The type of this order, as expressed in integer format. You'll need Market order. 1 Market 2 Limit 3 StopMarket 4 StopLimit 5 TrailingStopMarket 6 TrailingStopLimit
PegPriceType	integer. When entering a stop/trailing order, set PegPriceType to an integer that corresponds to the type of price that pegs the stop: 1 Last 2 Bid 3 Ask 4 Midpoint
InstrumentId	long integer. The ID of the instrument being traded in the order
TrailingAmount	real. The offset by which to trail the market in one of the trailing order types. Set this to the current price of the market to ensure that the trailing offset is the amount intended in a fast-moving market.
LimitOffset	real. The amount by which a trailing limit order is offset from the activation price.
Side	integer. The side of the trade represented by this order. One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
StopPrice	real. The price at which to execute the order, if the order is a Stop order (either buy or sell).
TimeInForce	integer. The period during which the order is executable. You'll need FOK. 4 FOK fill or kill — fill the order immediately, or cancel it immediately There may be other settings for TimeInForce depending on the trading venue.
OMSId	1

Response

```
{ "status":"Accepted", "errormsg":"", "OrderId": 123 // Server order id }
```

Where:

status	string. If the order is accepted by the system, it returns 0. 0 Accepted 1 Rejected
errormsg	string. Any error message the server returns.
OrderId	long integer. The ID assigned to the order by the server. This allows you to track the order.

SubscribeLevel1

No authentication required
Retrieves the latest Level 1 Ticker information and then subscribes the user to ongoing Level 1 market data event updates for one specific instrument. The SubscribeLevel1 call responds with the Level 1 response shown below. The OMS then periodically sends Leve1UpdateEvent information when best bid/best offer issues in the same format as this response, until you send the UnsubscribeLevel1 call.

Request
You can identify the instrument with its ID or with its market symbol (string).

```
{ “OMSId”: 1, “InstrumentId”: 0 }
```

Or

```
{ “OMSId”: 1, “Symbol”: “BTCUSD”}
```

Where:

OMSId	integer. The ID of the Order Management System that offers the product.
InstrumentId	integer. The ID of the instrument you’re tracking. Conditionally optional.
Symbol	string. The symbol of the instrument you’re tracking. Conditionally optional.

Response
The SubscribeLevel1 response and Level1UpdateEvent both provide the same information.

Where:

OMSId	1
InstrumentId	integer. The ID of the instrument being tracked.
BestBid	real. The current best bid for the instrument.
BestOffer	real. The current best offer for the instrument.
LastTradedPx	real. The last-traded price for the instrument.
LastTradedQty	real. The last-traded quantity for the instrument.
LastTradeTime	long integer. The time of the last trade in POSIX format X 1000 (milliseconds since 1 January 1970).
SessionOpen	real. Opening price. In markets with openings and closings, this is the opening price for the current session; in 24-hour markets, it is the price as of UTC Midnight.
SessionHigh	real. Highest price during the trading day, either during a true session (with opening and closing prices) or UTC midnight to UTC midnight.
SessionLow	real. Lowest price during the trading day, either during a true session (with opening and closing prices) or UTC midnight to UTC midnight.
SessionClose	real. The closing price. In markets with openings and closings, this is the closing price for the current session; in 24-hour markets, it is the price as of UTC Midnight.
Volume	real. The unit volume of the instrument traded, either during a true session (with openings and closings) or in 24-hour markets, the period from UTC Midnight to UTC Midnight.
CurrentDayVolume	real. The unit volume of the instrument traded either during a true session (with openings and closings) or in 24-hour markets, the period from UTC Midnight to UTC Midnight.
CurrentDayNumTrades	integer. The number of trades during the current day, either during a true session (with openings and closings) or in 24-hour markets, the period from UTC Midnight to UTC Midnight.
CurrentDayPxChange	real. Current day price change, either during a true trading session or UTC Midnight to UTC midnight.
Rolling24HrVolume	real. Unit volume of the instrument during the past 24 hours, regardless of time zone. Recalculates continuously.
Rolling24HrNumTrades	integer. Number of trades during the past 24 hours, regardless of time zone. Recalculates continuously.
Rolling24HrPxChange	real. Price change during the past 24 hours, regardless of time zone. Recalculates continuously
TimeStamp	long integer. The time this information was provided, in POSIX format X 1000 (milliseconds since 1 January 1970).

UnsubscribeLevel1

No authentication required
Unsubscribes the user from a Level 1 Market Data Feed subscription.
Request

```
{ "OMSId": 1, "InstrumentId": 1 }
```

Where:

OMSId	1
InstrumentId	long integer. The ID of the instrument being tracked by the Level 1 market data feed.

Response

```
{ "result": true, "errormsg": null, "errorcode":0, "detail": null }
```

Where:

result	Boolean. A successful receipt of the unsubscribe request returns true; and unsuccessful receipt (an error condition) returns false.
errormsg	string. A successful receipt of the unsubscribe request returns null; the errormsg parameter for an unsuccessful request returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	integer. A successful receipt of the unsubscribe request returns 0. An unsuccessful receipt returns one of the errorcodes shown in the errormsg list.
detail	string. Message text that the system may send. Usually null

SubscribeLevel2

No authentication required
Retrieves the latest Level 2 Ticker information and then subscribes the user to Level 2 market data event updates for one specific instrument. Level 2 allows the user to specify the level of market depth information on either side of the bid and ask. The SubscribeLevel2 call responds with the Level 2 response shown below. The OMS then periodically sends Level2UpdateEvent information in the same format as this response until you send the UnsubscribeLevel2 call.

Request
You can identify the instrument either by ID or by market symbol.

```
    {
        “OMSId”: 1,
        “InstrumentId”: 0
        “Depth”: 10
    }
```

or

```
    {
        “OMSId”: 1,
        “Symbol”: “BTCUSD”
        “Depth”: 10
    }
```

Where:

OMSId	integer. The ID of the Order Management System on which the instrument trades.
InstrumentId	integer. The ID of the instrument you’re tracking. Conditionally optional.
Symbol	string. The symbol of the instrument you’re tracking. Conditionally optional.
Depth	integer. The depth of the order book. The example request returns 10 price levels on each side of the market.

Response
The response is a single object (for one specific instrument). The Level2UpdateEvent contains the same data, but is sent by the OMS when trades occur.

```
    {
        “MDUpdateID”: 0,
        “Accounts”: 0,
        “ActionDateTime”: 635872032000000000,
        “ActionType”: {
        “Options”: [
        “New”,
        “Update”, ] “Delete”
        “LastTradePrice”: 0,
        “Orders”: 0,
        “Price”: 0,
        “ProductPairCode”: 0,
        “Quantity”: 0,
        “Side”: 0,
    }
```

Where:

MDUpdateID	integer. Market Data Update ID. This sequential ID identifies the order in which the update was created.
Accounts	integer. The number of accounts that have orders at this price level.
ActionDateTime	string. ActionDateTime identifies the time and date that the snapshot was taken or the event occurred, in POSIX format X 1000 (milliseconds since 1 January 1970)
ActionType	string. L2 information provides price data. This value shows whether this data is new, an update, or a deletion. One of: New Update Delete
LastTradePrice	real. The price at which the instrument was last traded.
Orders	integer. The number of orders at this price point. Whether it is a Buy or Sell order is shown by Side, below.
Price	real. Bid or Ask price for the Quantity (see Quantity below).
ProductPairCode	integer. ProductPairCode is the same number and used for the same purpose as InstrumentID. The two are completely equivalent in value. InstrumentId 47 = ProductPairCode 47.
Quantity	real. Quantity available at a given Bid or Ask price (see Price above).
Side	integer. One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)

UnsubscribeLevel2

No authentication required
Unsubscribes the user from a Level 2 Market Data Feed subscription.
Request

```
    { “OMSId”: 1, “InstrumentId”: 1 }
```

Where:

OMSId	integer. The ID of the Order Management System on which the user has subscribed to a Level 2 market data feed.
InstrumentId	long integer. The ID of the instrument being tracked by the Level 2 market data feed.

Response

```
    { “result”: true, “errormsg”: null, “errorcode”:0, “detail”: null }
```

Where:

result	Boolean. A successful receipt of the unsubscribe request returns true; and unsuccessful receipt (an error condition) returns false.
errormsg	string. A successful receipt of the unsubscribe request returns null; the errormsg parameter for an unsuccessful request returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	integer. A successful receipt of the unsubscribe request returns 0. An unsuccessful receipt returns one of the errorcodes shown in the errormsg list.
detail	string. Message text that the system may send. Usually null.

SubscribeTrades

Subscribes an authenticated user to the Trades Market Data Feed for a specific instrument. Each trade has two sides: Buy and Sell.
SubscribeTrades returns the response documented here for your immediate information, then periodically sends the OrderTradeEvent documented in SubscribeAccountEvents.

Request

```
{
        “OMSId”: 1,
        “InstrumentId”: 1,
        “IncludeLastCount”: 100
}
```

Where:

OMSId	integer. The ID of the Order Management System on which the instrument is traded.
InstrumentId	long integer. The ID of the instrument whose trades will be reported.
IncludeLastCount	integer. Specifies the number of previous trades to retrieve in the immediate snapshot. Default is 100.

Response
The response returns an array of trades. Both sides of each trade are described.

```
[
        {
                {
                        “OMSId”: 0, “TradeID”: 0,
                        “ProductPairCode”: 0,
                        “Quantity”: 0,
                        “Price”: 0,
                        “Order1”: 0,
                        “Order2”: 0,
                        “TradeTime”: “0001-01-01T05:00:00Z”,
                        “Direction”: {
                                “Options”: [
                                        “NoChange”,
                                        “UpTick”,
                                        “DownTick”
                                ]
                        },
                “TakerSide”: 0,
                “Side1AccountId”: 0,
                “Side2AccountId”: 0,
                “Order1Side”: {
                        “Options”: [
                                “Buy”,
                                “Sell”,
                                “Short”,
                                “Unknown”
                                ]
                        },
                “Order2Side”: {
                “Options”: [
                        “Buy”,
                        “Sell”,
                        “Short”,
                        “Unknown”
                        ]
                },
                “BlockTrade”: false,
                “Order1ClientId”: 0,
                “Order2ClientId”: 0,
                },
        }
]
```

Where:

OMSId	integer. The ID of the Order Management System where the instrument to be tracked is traded.
TradeID	integer. The ID of this trade.
ProductPairCode	integer. ProductPairCode is the same number and used for the same purpose as InstrumentID. The two are completely equivalent in value. InstrumentId 47 = ProductPairCode 47.
Quantity	real. The quantity of the instrument traded.
Price	real. The price at which the instrument traded.
Order1	integer. The ID of one of the orders that resulted in the trade.
Order2	integer. The ID of the other order that resulted in the trade.
TradeTime	long integer. The time at which the trade took place. UTC time.
Direction	string. Effect of the trade on the instrument’s market price. One of: 0 NoChange 1 UpTick 2 DownTick
TakerSide	integer. Which side of the trade took liquidity? One of: 0 Buy 1 Sell The maker side of the trade provides liquidity by placing the order on the book (this can be a buy or a sell order). The other side takes the liquidity. It, too, can be buy-side or sell-side.
Side1AccountId	integer. The account ID of the 1-side of the trade.
Side2AccountId	integer. The account ID of the 2-side of the trade.
Order1Side	string. The side taken by order 1 of the trade. One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
Order2Side	string. The side taken by order 2 of the trade. One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
BlockTrade	Boolean. Was this a privately negotiated trade that was reported to the OMS? A private trade returns true; otherwise false. Default is false.
Order1ClientId	long integer. The client-supplied order ID of the 1-side client.
Order2ClientId	long integer. The client-supplied order ID of the 2-side client.

UnsubscribeTrades

No authentication required
Unsubscribes a user from the Trades Market Data Feed.
Request

```
{ “OMSId”: 1, “InstrumentId”: 1 }
```

Where:

OMSId	integer. The ID of the Order Management System on which the user has subscribed to a trades market data feed.
InstrumentId	long integer. The ID of the instrument being tracked by the trades market data feed.

Response

```
{ “result”: true, “errormsg”: null, “errorcode”:0, “detail”: null }
```

Where:

result	Boolean. A successful receipt of the unsubscribe request returns true; and unsuccessful receipt (an error condition) returns false.
errormsg	string. A successful receipt of the unsubscribe request returns null; the errormsg parameter for an unsuccessful request returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	integer. A successful receipt of the unsubscribe request returns 0. An unsuccessful receipt returns one of the errorcodes shown in the errormsg list.
detail	string. Message text that the system may send. Usually null.