coins PRO

API Specification

Version 0.3.0

13 September, 2018

Changelog

Version 0.3.0 (13 September) - added GetDepositTickets, GetWithdrawTickets methods. Added API keys management methods.

Table of contents

URLs

You can access the Coins PRO API using the following websocket addresses:

Production	wss://api-cx.coins.asia/ws-api/
Staging	wss://api-cx.staging.coins.technology/ws-api/

Message Frame Format

All WebSocket Calls and Replies are embedded into a JSON-Formatted Frame object containing the relevant information about the call or reply, as well as the payload. Take attention, that all sections about functions, **contains format only of response payload** ("o" in the frame).

Example

```
{
    "m":0,
    "i":0,
    "n":"GetProducts",
    "o":"{\"OMSId\":1\"}"
}
```

Frame Payload Parameters

Attribute Name	Example	Description
m: Message Type	"m":0,	Describes the type of call the message relates to. Supported types: **0 = Request** 1 = Reply **2 = Subscribe To Event** 3 = Event 4 = Unsubscribe from Event 5 = Error Client should only use 0 (Request) and 2 - to subscribe to data feed.
i: Sequence Number	"i":0,	This is the sequence number of the message. The Client-Side sequence number should always be an Even Number, such that your sequence number variable should

		always be incremented by 2 (i += 2). All messages will contain either an original sequence number (message types 0, 2, 3, 4), or will contain the ID of the request message the message is in reply to (1, 5 types). For example, if you requested something using i=0, you'll get reply, with i=0 as well, but with type 1.
n: Function Name	"n": "GetProducts"	This is the name of the Remote Function that the message is a Request or Reply to. For instance, if you wish to make a call to the **"GetProducts"** function, the Request should contain **"GetProducts"** in this field, and the Reply will also contain **"GetProducts"** in this field.
o: Payload	"o": "{"OMSId":1}"	This is a JSON-formatted string containing the data being sent with the message, either as Parameters in a Request, or as the data in a Reply to a Request. **Take attention:** the rest of functions described below, contain only payload description, so we do not include whole frame to in the rest of the documentation. *Note:* OMSId refers to an internal ID of the order management system, and should be always 1.

How to use the frame.

When sending a Request to our API using javascript, a call would look as follows:

```
var frame =
{
    "m":0,
    "i":0,
    "n":"GetProducts",
    "o":""
};

var requestPayload =
{
    "OMSId":1,
};
```

```
frame.o = json.Stringify(requestPayload);
WS.Send(json.Stringify(frame));
```

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

```
var frame = json.Parse(wsMessage);

if (frame.i == 1)
{
    //This is a Reply
    if (frame.n == "GetProducts")
    {
        var LoginReply = json.Parse(frame.o);
        if (LoginReply.Authenticated)
        {
            var user = LoginReply.User;
        }
    }
}
```

Standard API Responses

Response payloads not returning object data and common error responses will follow pattern below. Please note, that function can have extra error codes, which are described below under the function section.

Common error codes

Error Code	Description
20	Not Authorized
100	Invalid Request
101	Operation Failed
102	Server Error
104	Resource Not Found

Successful call with no return object

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

Unsuccessful call due to an issue with authorization

```
{
    "result":false,
    "errormsg":"Not Authorized",
    "errorcode":20,
    "detail":null
}
```

Unsuccessful call due to invalid request params

```
{
    "result":false,
    "errormsg":"Invalid Request",
    "errorcode":100,
    "detail":"OMSId must be an Integer"
}
```

Unsuccessful call due to operation not completing successfully

```
{
    "result":false,
    "errormsg":"Operation Failed",
    "errorcode":101,
    "detail":null
}
```

Unsuccessful call due to unexpected server error

```
{
    "result":false,
    "errormsg":"Server Error",
    "errorcode":102,
    "detail":null
}
```

Unsuccessful call due to missing resource (user id not in system, etc)

```
{
    "result":false,
    "errormsg":"Resource Not Found",
    "errorcode":104,
    "detail":null
}
```

Unauthenticated Calls

For all calls requiring OMSId, set OMSId = 1, also, all sections here represent separate function names. Please read more about OMS and function names in the Message Frame Format section.

GetProducts

Requests a list of available Products from the API.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"GetProducts",
    "o":"{\"OMSId\": 1}"
}

Response <- (full frame omitted, please read Message Frame Format)

[
    {
        "ProductId": 1,
        "Product": "BTC",
        "ProductFullName": "Bitcoin",
        "ProductType": "CryptoCurrency",
        "DecimalPlaces": 9
    },
    ...
]
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.

Response Parameters

Attribute Name	Example	Description
ProductId	1	Product's ID Number [Integer]
Product	"BTC"	Product's Symbol [String]
ProductFullName	"Bitcoin"	Product's Full Name [String]
ProductType	"CryptoCurrency"	Product's Type [String]

		Supported values: - "NationalCurrency" - "CryptoCurrency"
DecimalPlaces	9	Product's Precision [Integer]

GetInstruments

Requests a list of available instruments from the API.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"GetInstruments",
    "o":"{\"OMSId\": 1}"
}

Response <- (full frame omitted, please read Message Frame Format)

[
    {
        "InstrumentId":1,
        "Symbol":"BTCUSD",
        "Product1":1,
        "Product1Symbol":"BTC",
        "Product2":2,
        "Product2Symbol":"USD",
        "InstrumentType": "Standard",
    },
    ...
]
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.

Response Parameters

Attribute Name	Example	Description

InstrumentId	1	Instrument's Identifier [Integer]
Symbol	"BTCUSD"	Instrument's Symbol [String]
Product1	1	Instrument's Product 1 Identifier [Integer]
Product1Symbol	"BTC"	Instrument's Product 1 Symbol [String]
Product2	2	Instrument's Product 2 Identifier [Integer]
Product2Symbol	"USD"	Instrument's Product 2 Symbol [String]
InstrumentType	"Standard"	Type of Instrument [String] 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.

Authentication

WebAuthenticateUser

You must call this in order to use any of the authenticated calls below.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"WebAuthenticateUser",
    "o":"{\"UserName\":\"User1\",\"Password\":\"Password\"}"
}

Response <- (full frame omitted, please read Message Frame Format)

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

Request Parameters

Attribute Name	Example	Description
UserName	"User1"	Username [String]
Password	"Password"	Password [String]

Response Parameters

Attribute Name	Example	Description
Authenticated	true	Is session authenticated now [Bool] *Note*: this will be false if auth failed.
SessionToken	"7d0ccf3a-ae63-44f5-a409-2301d80228bc"	Session token [String]
UserId	1	Authenticated user ID [Integer]

GetUserAPIKeys

The endpoint will return existing APIKeys with assigned permissions. APISecret field will be blank.

Request Parameters

Attribute Name	Example	Description
UserId	4	User ID [Integer]

Response Parameters

Attribute Name	Example	Description
UserId	4	User ID [Integer]
ApiKey	“ab1eb50a1b0cb56a9130cdccba995a47da77a22e”	Key that you will need to authenticate with [String]
ApiSecret	“”	Secret that you will use for signature generation [String]
Permissions	[‘Deposit’, ‘Withdraw’]	Permissions enabled for API Keys [Array of Strings]

AddUserAPIKey

In order to authenticate using an API key, you will need to create an ApiKey and ApiSecret using this endpoint. Your UserId will need to be provided along with the permissions you wish to enable in payload.

The endpoint will return a UserAPIKey classtype json string containing the APIKey and APISecret.

Note: Please save the APIKey and APISecret values returned in a secure location. Both of these values are needed for authentication and signature generation.

Request Parameters

Attribute Name	Example	Description
UserId	4	User ID [Integer]
Permissions	[‘Deposit’, ‘Withdraw’]	Permissions you wish to enable for this API Key [Array of Strings]

		Possible values: Deposit, Withdraw, Trading

Response Parameters

Attribute Name	Example	Description
UserId	4	User ID [Integer]
ApiKey	“ab1eb50a1b0cb56a9130cdccba995a47da77a22e”	Key that you will need to authenticate with [String]
ApiSecret	“1b7f480232b6fc2844bf778d50b97c18474c67ec”	Secret that you will use for signature generation [String]
Permissions	[‘Deposit’, ‘Withdraw’]	Permissions you wish to enable for this API Key [Array of Strings] Possible values: Deposit, Withdraw, Trading

RemoveUserAPIKey

Request Parameters

Attribute Name	Example	Description
UserId	4	User ID [Integer]
ApiKey	“ab1eb50a1b0cb56a9130cdccba995a47da77a22e”	String returned from AddUserAPIKey call [String]

AuthenticateUser

In order to authenticate with a key you need to use AuthenticateUser call with nonce and generated signature.

Request Parameters

Attribute Name	Example	Description
UserId	“4”	User ID [String]
APIKey	“ab1eb50a1b0cb56a9130cdccba995a	String returned from AddUserAPIKey call [String]

	47da77a22e”	
Nonce	“1534750973”	Current timestamp as a string [String]
Signature	“6a8de403af40e227ce00fd1d5c03e9eeca0d5ced5227d273122c398143be2499”	This is a HMAC-SHA256 signature with ApiSecret as a key and `nonce + UserId + ApiKey` as a message.

Response Parameters

Attribute Name	Example	Description
User	{"UserId":1,"UserName":"email@email.com","Email":"email@email.com","EmailVerified":true,"AccountId":3,"OMSId":1,"Use2FA":false}	User info [Object]
Authenticated	true	If authentication was successful [Bool]

User Account Calls

GetUserAccounts

Retrieves a list of account IDs for the current user. The Request should have an empty string as the payload. Typically, each user is assigned one account.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"GetUserAccounts",
    "o":""
}

Response <- (full frame omitted, please read Message Frame Format)

[4]
```

Response Parameters

Attribute Name	Example	Description
An array of account IDs	[4]	Array of Account IDs [Integer]

GetAccountTransactions

Retrieves a list of recent transactions from your account.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"GetAccountTransactions",
    "o":"{\"OMSId\":1,\"AccountId\": 1,\"Depth\": 200}"
}

Response <- (full frame omitted, please read Message Frame Format)
[
    {
        "TransactionId": 945,
        "AccountId": 4,
        "CR": 76,
        "DR": 0,
        "TransactionType": "Trade",
        "ReferenceId": 232,
        "ReferenceType": "Trade",
        "ProductId": 2,
        "Balance": 101111.1,
        "TimeStamp": 1501354796418
    },
    ...
]
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
AccountId	1	Account ID of user. Should be populated using GetUserAccounts.

Depth	200	Number of transactions API should retrieve, starting with the most recent one.

Response Parameters

Attribute Name	Example	Description
TransactionId	945	The Transaction Id [64 Bit Integer]
AccountId	4	Your Account Id [Integer]
CR	76	The amount credited to your account. [Decimal]
DR	0	The amount debited from your account. [Decimal]]
TransactionType	"Trade"	The nature/type of the transaction. [String] Values: - "Other" - "Trade" - "Fee" **TransactionType** / **ReferenceType** "Other" / Deposit "Other" / Withdraw "Trade" / Trade "Fee" / Trade
ReferenceId	232	The Id Number of the action/event object that initiated this transaction. For instance, if this was the result of a trade, this would be the TradeId. [64 bit Integer]
ProductId	2	Product Id [Integer]
Balance	101111.1	The resulting balance after this transaction for the specified product. [Decimal]
ReferenceType	"Trade"	The type of action or event that triggered this transaction. [String]

		Values: - Trade - Deposit - Withdraw - Transfer - OrderHold - WithdrawHold - DepositHold - MarginHold
TimeStamp	1501354796418	[Long Integer] Time at which the transaction took place, in POSIX format and UTC time zone.

GetAccountPositions

Retrieves a list of Positions(Balances) on a specific account.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"GetAccountPositions",
    "o":"{\"OMSId\":1,\"AccountId\":4}"
}

Response <- (full frame omitted, please read Message Frame Format)

[{
    "AccountId": 4,
    "ProductSymbol": "BTC",
    "ProductId": 1,
    "Amount": 10497.3,
    "Hold": 3.19,
    "PendingDeposits": 0,
    "PendingWithdraws": 0,
    "TotalDayDeposits": 10500,
    "TotalDayWithdraws": 0,
    "TotalMonthWithdraws": 0,
}]
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
AccountId	4	Account ID [Integer]

Response Parameters

Attribute Name	Example	Description
AccountId	4	Your account id. [Integer]
ProductId	5	Product's ID Number [Integer]
ProductSymbol	"BTC"	The product symbol for this record. [String]
Amount	10497.3	The Total Balance for the specified product. [Decimal]
Hold	3.19	The total amount of your balance that is on hold. Your Available balance for trading and withdraw is (Amount - Hold). [Decimal]
PendingDeposits	0	Total Deposits Pending for the specified product. [Decimal]
PendingWithdraws	0	Total Withdraws Pending for the specified product. [Decimal]
TotalDayDeposits	10500	The total 24 hour deposits for the specified product. [Decimal]
TotalDayWithdraws	0	The total 24 hour withdraws for the specified product. [Decimal]
TotalMonthWithdraws	0	The total month withdraws for the specified product. [Decimal]

GetAccountTrades

Retrieves Trade History for a specific account.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"GetAccountTrades",
    "o":"{\"OMSId\":1,\"AccountId\":4,\"Count\":50,\"StartIndex\":0}"
}
```

```
Response <- (full frame omitted, please read Message Frame Format)

[
    {
        "TradeId": 230,
        "OrderId": 9861,
        "AccountId": 4,
        "ClientOrderId": 0, //Your client order id. [64 bit Integer]
        "InstrumentId": 1, //Instrument Id [Integer]
        "Side": "Buy", //Side [String] Values are "Buy", "Sell"
        "Quantity": 2, //Quantity [Decimal]
        "Price": 95, //Price [Decimal]
        "Value": 190, //Value [Decimal]
        "TradeTime": 1501354796418,
    },
    ...
]
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
AccountId	4	Account ID [Integer]
Count	50	Number of trades - defaults to 100, and max is 200 [Integer]
StartIndex	0	Start Index - for pagination. 0 is the most recent trade. [Integer]

Response Parameters

Attribute Name	Example	Description
TradeId	230	Trade Id [64 bit Integer]
OrderId	9861	Order Id [64 bit Integer]
AccountId	4	Your account id. [Integer]
ClientOrderId	0	Your client order id. [64 bit Integer]
InstrumentId	1	Instrument's Identifier [Integer]
Side	"Buy"	Side of the trade [String] Values: - "Buy" - "Sell"

Quantity	2	Quantity [Decimal]
Price	95	Price [Decimal]
Value	190	Value = price * quantity [Decimal]
TradeTime	1501354796418	Timestamp [Integer] The time at which the trade took place, in POSIX format and UTC time zone.

Order Handling Calls

Description of Order Types used in various calls

Market - An order used to buy or sell at the best available price. No restrictions on price or time frame

Limit - An order to buy or sell at a specified price or better. A limit order may not be executed if the price set is not met during the time that the order is open

StopMarket - An order to buy or sell only when an instrument reaches a set price. Once the instrument reaches this price, the order becomes a market order.

StopLimit - An order to buy or sell only when a set price is reached. Once this price is reached, the order becomes a limit order to buy or sell at the limit price or better.

TrailingStopMarket - An order that sets the stop price at a price with a fixed offset relative to the market price. If the market moves and the stop price is reached, the order becomes a market order.

TrailingStopLimit - An order that recalculates the stop price at a fixed offset relative to the market price. It also recalculates a limit price based on a different fixed offset. If the market reaches the stop price, the order becomes a limit order.

SendOrder

Sends a new order into the API. It is important that you are subscribed to Account Actions (SubscribeAccountEvents) in order to see updated status events for entered orders. Alternatively, you can also call GetOpenOrders and/or GetOrderHistory to check for the status of your order.

Example

```
Request -> (full frame omitted, please read Message Frame Format)

{
    "OMSId": 1,
    "AccountId": 4,
    "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
}


Response <- (full frame omitted, please read Message Frame Format)

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

Request Parameters

Attribute Name	Example	Description
AccountId	4	Your account id. [Integer]
ClientOrderId	99	Set this to your own id if you wish to use one. It will be useful for recognizing future order states related to this call. [64 bit Integer]
OMSId	1	Order Management System ID. Always 1.
UseDisplayQuantity	true	[Boolean] If you enter a Limit order with a reserve, you must set UseDisplayQuantity to true.
Quantity	1	Quantity of the Order [Decimal]
DisplayQuantity	0	Quantity to Display on the Market. If your order is for 1000, and you only want to show 100 at a time in market data, set this to 100. Set to 0 to display all. [Decimal]
LimitPrice optional	95	The limit price for this order. [Decimal]
OrderIdOCO optional	0	If you would like to have this order cancel another on execution, set this field to the other order's server order id. Omit or set to 0 if no OCO is required. [64 Bit Integer]
OrderType	2	The type of order. [Integer]

		Values: - 1 (Market) - 2 (Limit) - 3 (StopMarket) - 4 (StopLimit) - 5 (TrailingStopMarket) - 6 (TrailingStopLimit)
PegPriceType optional	1	When entering a Stop/Trailing order, set this to the type of price you would like to peg the Stop to. [Integer] Values: - 1 (Last) - 2 (Bid) - 3 (Ask)
InstrumentId	1	Instrument's Identifier [Integer]
TrailingAmount optional	1.0	When entering a Trailing order, set this to the amount that you'd like to have your trigger price trail the market by. [Decimal]
LimitOffset optional	2.0	When entering a Trailing Limit order, this specifies the distance from activation price to enter your limit order. [Decimal]
Side	0	0 (Buy) or 1(Sell)
StopPrice optional	96	The Stop Price for this order, if it is a stop order. Otherwise you may omit this field. [Decimal]
TimeInForce	1	Values: - 1 (Good 'til Canceled), - 3 (Immediate or Cancel), - 4 (Fill or Kill)

Response Parameters

Attribute Name	Example	Description
status	"Accepted"	Status of the order [String]
errormsg	""	Error message in case of error [String]
OrderId	123	Order Identifier [Integer]

CancelOrder

Cancels an open order - Can cancel by either specifying the OrderId returned when the order was created, or by specifying both the ClientOrderId and AccountId of the order. If AccountId is not specified, the default user account will be used

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"CancelOrder",
    "o":"{\"OMSId\":1,\"AccountId\":4,\"OrderId\":12}"
}

Response <- (full frame omitted, please read Message Frame Format)

{
    "status":"Processing",
    "errormsg":""
     "errorcode": 104, //An error code
     "detail": ""
}
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
AccountId	4	Account ID [Integer]
ClientOrderId	99	The client order id of the order to cancel. [64 bit Integer]
OrderId	12	The server order id of the order to cancel. [64 bit Integer]

Response Parameters

Attribute Name	Example	Description
status	"Processing"	Status [String] "Processing" status means system got request for cancellation this order. To check actual status of the order please use GetOrderStatus, GetOpenOrders or

		OrderStateEvent (read more in SubscribeAccountEvents section) functions, which contain order status in payload. In case if order was canceled, state of order will be "Canceled".
errormsg	""	Error message [String]
detail	“”	Detailed error description [String]
errorcode	104	Error code [Integer]

GetOrderStatus

Gets the current operating status of an order submitted to Order Management System.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"GetOrderStatus",
    "o":"{\"OMSId\":1,\"AccountId\":4,\"OrderId\":12}"
}

Response <- (full frame omitted, please read Message Frame Format)
{
    "Side": "Sell",
    "OrderId": 9849,
    "Price": 97,
    "Quantity": 0.29,
    "Instrument": 1,
    "Account": 4,
    "OrderType": "Limit",
    "ClientOrderId": 0,
    "OrderState": "FullyExecuted",
    "ReceiveTime": 1532346260831,
    "OrigQuantity": 1,
    "QuantityExecuted": 0.71,
    "OrigOrderId": 9849,
    "OrigClOrdId": 0,
    "RejectReason": ""
}
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
AccountId	4	Account ID [Integer]
OrderId	12	The server order id of the order to cancel (returned in SendOrder). [64 bit Integer]

Response Parameters

Attribute Name	Example	Description
Account	4	Your account id. [Integer]
ClientOrderId	99	Set this to your own id if you wish to use one. It will be useful for recognizing future order states related to this call. [64 bit Integer]
Quantity	1	The Quantity (Remaining if partially or fully executed) of your order. [Decimal]
OrderType	2	The type of order. [Integer] Values: - 1 (Market) - 2 (Limit) - 3 (StopMarket) - 4 (StopLimit) - 5 (TrailingStopMarket) - 6 (TrailingStopLimit)
Instrument	1	Instrument's Identifier [Integer]
Side	"Sell"	The side of your order. [String] Values are "Sell", "Buy" Values: - "Buy" - "Sell"
OrderId	123	The Server-Assigned Order Id. [64 bit Integer]
Price	96	The Price of your order. [Decimal]
OrderState	"Working"	The current state of the order. [String] Values: - "Working" - "Rejected" - "FullyExecuted" - "Canceled"

		- "Expired"
OrigQuantity	1	The original quantity of your order. [Decimal]
QuantityExecuted	0.71	The total executed quantity. [Decimal]
RejectReason	""	If the trade was rejected, this string holds the reason [String]
OrigOrderId	9849	The Original Server-Assigned Order Id, if the order has changed. [64 bit Integer]
OrigClOrdId	1	The Original Client-Designate Order Id. [64 bit Integer]
ReceiveTime	1532346260831	Timestamp

GetOrderFee

Returns an estimate of the fee for a specific order and order type.

Example

```
Request -> (full frame omitted, please read Message Frame Format)

{
    "OMSId": 1,
    "AccountId": 1,
    "InstrumentId": 1,
    "ProductId": 1,
    "Amount": 500,
    "OrderType": "Market",
    "MakerTaker": "Maker"
}


Response <- (full frame omitted, please read Message Frame Format)

{
    "OrderFee": 0,
    "ProductId": 1
}
```

Request Parameters

Attribute Name	Example	Description

OMSId	1	Order Management System ID. Always 1.
AccountId	1	Your account ID [Integer]
InstrumentId	1	Instrument's Identifier [Integer]
ProductId	1	Product's Identifier [Integer]
Amount	500	Amount [Decimal]
OrderType	"Market"	The type of order. [String] Values: 1 (Market) 2 (Limit) 3 (StopMarket) 4 (StopLimit) 5 (TrailingStopMarket) 6 (TrailingStopLimit) 7 (BlockTrade)
MakerTaker	"Maker"	Whether the order is expected to execute instantly against existing orders in the order book (“Taker”) or rest in the order book as an open order (“Maker”)

Response Parameters

Attribute Name	Example	Description
OrderFee	10	Fee amount [Decimal]
ProductId	2	ID of the product in which the fee is denominated [Integer]

GetOrderHistory

Retrieves a list of the last 100 orders placed on your account.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"GetOrderHistory",
```

```
    "o":"{\"OMSId\": 1,\"AccountId\": 1}"
}

Response <- (full frame omitted, please read Message Frame Format)

[
    {
        "Side": "Sell",
        "OrderId": 9849,
        "Price": 97,
        "Quantity": 0.29,
        "Instrument": 1,
        "Account": 4,
        "OrderType": "Limit",
        "ClientOrderId": 0,
        "OrderState": "FullyExecuted",
        "ReceiveTime": 1532346260831,
        "OrigQuantity": 1,
        "QuantityExecuted": 0.71,
        "OrigOrderId": 9849,
        "OrigClOrdId": 0,
        "RejectReason": ""
    },
    ...
]
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
AccountId	4	Account ID [Integer]

Response Parameters

Attribute Name	Example	Description
Account	4	Your account id. [Integer]
ClientOrderId	99	Set this to your own id if you wish to use one. It will be useful for recognizing future order states related to this call. [64 bit Integer]
Quantity	1	The Quantity (Remaining if partially or fully executed) of your order. [Decimal]
OrderType	2	The type of order. [Integer] Values:

		- 1 (Market) - 2 (Limit) - 3 (StopMarket) - 4 (StopLimit) - 5 (TrailingStopMarket) - 6 (TrailingStopLimit)
Instrument	1	Instrument's Identifier [Integer]
Side	"Sell"	The side of your order. [String] Values are "Sell", "Buy" Values: - "Buy" - "Sell"
OrderId	123	The Server-Assigned Order Id. [64 bit Integer]
Price	96	The Price of your order. [Decimal]
OrderState	"Working"	The current state of the order. [String] Values: - "Working" - "Rejected" - "FullyExecuted" - "Canceled" - "Expired"
OrigQuantity	1	The original quantity of your order. [Decimal]
QuantityExecuted	0.71	The total executed quantity. [Decimal]
RejectReason	""	If the trade was rejected, this string holds the reason [String]
OrigOrderId	9849	The Original Server-Assigned Order Id if the order has been changed. [64 bit Integer]
OrigClOrdId	1	The Original Client-Designated Order Id if the order has been changed. [64 bit Integer]
ReceiveTime	1532346260831	Timestamp

GetOpenOrders

Retrieves the Open Orders for specified account of current user. Keep in mind that if your order is no longer in a working state, you will not find it using GetOpenOrders.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"GetOpenOrders",
    "o":"{\"OMSId\": 1,\"AccountId\": 1}"
}

Response <- (full frame omitted, please read Message Frame Format)

[
    {
        "Side": "Sell",
        "OrderId": 9849,
        "Price": 97,
        "Quantity": 0.29,
        "Instrument": 1,
        "Account": 4,
        "OrderType": "Limit",
        "ClientOrderId": 0,
        "OrderState": "FullyExecuted",
        "ReceiveTime": 1532346260831,
        "OrigQuantity": 1,
        "QuantityExecuted": 0.71,
        "AvgPrice": 0,
        "OrigOrderId": 9849,
        "OrigClOrdId": 0,
        "RejectReason": ""
    },
    ...
]
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
AccountId	4	Account ID [Integer]

Response Parameters

Attribute Name	Example	Description
Account	4	Your account id. [Integer]

ClientOrderId	99	Set this to your own id if you wish to use one. It will be useful for recognizing future order states related to this call. [64 bit Integer]
Quantity	1	The Quantity (Remaining if partially or fully executed) of your order. [Decimal]
OrderType	2	The type of order. [Integer] Values: - 1 (Market) - 2 (Limit) - 3 (StopMarket) - 4 (StopLimit) - 5 (TrailingStopMarket) - 6 (TrailingStopLimit)
Instrument	1	Instrument's Identifier [Integer]
Side	"Sell"	The side of your order. [String] Values are "Sell", "Buy", "Short" Values: - "Buy" - "Sell"
OrderId	123	The Server-Assigned Order Id. [64 bit Integer]
Price	96	The Price of your order. [Decimal]
OrderState	"Working"	The current state of the order. [String] Values: - "Working" - "Rejected" - "FullyExecuted" - "Canceled" - "Expired"
OrigQuantity	1	The original quantity of your order. [Decimal]
QuantityExecuted	0.71	The total executed quantity. [Decimal]
OrigOrderId	9849	The Original Server-Assigned Order Id if the order has been changed. [64 bit Integer]
OrigClOrdId	1	The Original Client-Designated Order Id if the order has been changed. [64 bit Integer]
ReceiveTime	1532346260831	Timestamp

RejectReason	""	If the trade was rejected, this string holds the reason [String]

Deposits

Deposits of all currencies to Coins Pro are done through the user's Coins.ph wallet. These transactions are initiated using the Coins.ph mobile / web clients or API.

For further details on initiating a deposit using the Coins.ph API, please refer to the `POST /v2/sellorder` endpoint in the Coins.ph API documentation.

GetDepositTickets

Get list of deposits for account.

Example

```
Request -> (full frame omitted, please read Message Frame Format)

{
    "OMSId":1,
    "AccountId":4,
    "Limit":100,
    "OperatorId":1
}


Response <- (full frame omitted, please read Message Frame Format)

{
    "AccountId":4,
    "AssetId":4,
    "AssetName":"Philippine Peso",
    "Amount":23,
    "OMSId":1,
    "RequestCode":"178ab0fb-bbe1-4095-abf8-dca5ea4b6624",
    "Status":"FullyProcessed",
    "DepositInfo":"{\\"Bank Name\\": \\"Coins.ph\\",
\\"Comment\\": \\"Cash in from Coins.ph\\", \\"OrderId\\":
\\"3675137698bd4f08b4b608a398176e26\\"}",
    "CreatedTimestamp":"2018-07-12T11:20:16Z",
    "LastUpdateTimeStamp":"2018-07-12T11:20:16Z"
}
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.

Limit	100	Number of last deposits to fetch [Integer]
AccountId	4	Your account ID [Integer]
OperatorId	1	Always 1 [Integer]
StartIndex	0	Offset in deposits list [Integer]

Response Parameters

Attribute Name	Example	Description
AccountId	4	Your account ID [Integer]
AssetId	1	The ProductId of the deposit. [Integer]
AssetName	"Philippine Peso"	Name of the asset [String]
Amount	23	Deposit amount [Decimal]
OMSId	1	Order Management System ID. Always 1.
RequestCode	"178ab0fb-bbe1-4095-abf8-dca5ea4b6624"	Request code of deposit, this ID is a common reference between Coins.ph and Coins Pro APIs and can be used to track transactions between the two systems [String] Use **settlement_detail.confirmation_code** field in Coins.ph SellOrder API to match transactions.
Status	"FullyProcessed"	Status of deposit [String]. Possible values: - New - AdminProcessing - Accepted - Rejected - SystemProcessing - **FullyProcessed** - Failed - Pending
DepositInfo	"{}"	JSON-format meta information about this deposit [String].
CreatedTimestamp	"2018-07-12T11:20:16Z"	Timestamp of creation [ISO datetime]

LastUpdateTimeStamp	"2018-07-12T11:20:16Z"	Timestamp of last update [ISO datetime]

Withdrawals

CreateWithdrawTicket

Creates a withdrawal ticket to send funds from Coins Pro to the user's Coins wallet.

Example

```
Request -> (full frame omitted, please read Message Frame Format)

{
    "OMSId":1,
    "ProductId":4,
    "AccountId":146,
    "Amount":"100"
}


Response <- (full frame omitted, please read Message Frame Format)

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

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
ProductId	4	ID of the product which will be withdrawn [Integer]
AccountId	4	Your account ID [Integer]
Amount	100	Amount for withdrawal [Decimal]

Response Parameters

Attribute Name	Example	Description

result	true	Result of operation, true if withdrawal ticket was created successfully [Bool]
errormsg	""	Error message if request was unsuccessful [String]
errorcode	0	Error code [Integer]. Zero in case of no errors.

GetWithdrawTickets

Get list of withdrawals for account.

Example

```
Request -> (full frame omitted, please read Message Frame Format)

{
    "OMSId":1,
    "AccountId":4,
    "Limit":100,
}


Response <- (full frame omitted, please read Message Frame Format)

{
    "AccountId":152,
    "AssetId":5,
    "AssetName":"BitcoinCash",
    "Amount":0.0001,
    "OMSId":1,
    "RequestCode":"0d390b87-e482-48ce-81de-ac208f66817e",
    "Status":"FullyProcessed",
    "CreatedTimestamp":"2018-06-07T06:26:07Z",
    "LastUpdateTimestamp":"2018-06-07T06:27:20Z",
}
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
Limit	100	Number of last deposits to fetch [Integer]
AccountId	4	Your account ID [Integer]
StartIndex	0	Offset in withdrawals list [Integer]

Response Parameters

Attribute Name	Example	Description
AccountId	4	Your account ID [Integer]
AssetId	1	The ProductId of the withdrawal. [Integer]
AssetName	"Philippine Peso"	Name of the asset [String]
Amount	23	Withdraw amount [Decimal]
OMSId	1	Order Management System ID. Always 1.
RequestCode	"178ab0fb-bbe1-4095-abf8-dca5ea4b6624"	Request code of withdrawal, this ID is a common reference between Coins.ph and Coins Pro APIs and can be used to track transactions between the two systems [String] Use **payment_validation_code** field in Coins.ph BuyOrder API to match transactions.
Status	"FullyProcessed"	Status of withdrawal [String]. Possible values: - New - AdminProcessing - Accepted - Rejected - SystemProcessing - **FullyProcessed** - Failed - Pending - Pending2Fa - AutoAccepted - Delayed
CreatedTimestamp	"2018-07-12T11:20:16Z"	Timestamp of creation [ISO datetime]
LastUpdateTimeStamp	"2018-07-12T11:20:16Z"	Timestamp of last update [ISO datetime]

Market Data

Overview of Level 1 and Level 2 Market Information

Level 1 information
Consists of real-time bid and ask quotes for a specific instrument ID or instrument symbol. Level 1 provides last trade information; session open, high, low, and close; current, and rolling information.

Level 2 information
Also consists of real-time bid and ask quotes for a specific instrument ID or instrument symbol as does Level 1, but allows the user to specify the level of market depth information on either side of the bid and ask, and to see quantities.

Level 1 Data

SubscribeLevel1

Retrieves the latest Level 1 Ticker information and Subscribes User to Level 1 Market Data updates for the specified Instrument. After subscribing, user will receive periodic Level1UpdateEvent event information until they call UnsubscribeLevel1.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"SubscribeLevel1",
    "o":"{\"OMSId\":1,\"InstrumentId\":1}"
}

Response <- (full frame omitted, please read Message Frame Format)

{
    "InstrumentId": 1,
    "BestBid": 0.00,
    "BestOffer": 0.00,
    "LastTradedPx": 0.00,
    "LastTradedQty": 0.00,
    "LastTradeTime": 635872032000000000,
    "SessionOpen": 0.00,
    "SessionHigh": 0.00,
    "SessionLow": 0.00,
    "SessionClose": 0.00,
    "Volume": 0.00,
    "CurrentDayVolume": 0.00,
```

```
    "CurrentDayNumTrades": 0,
    "CurrentDayPxChange": 0.0,
    "Rolling24HrVolume": 0.0,
    "Rolling24NumTrades": 0.0,
    "Rolling24HrPxChange": 0.0,
    "TimeStamp": 635872032000000000
}
```


Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
InstrumentId	1	Instrument's Identifier [Integer]

Response Parameters

Attribute Name	Example	Description
InstrumentId	1	Instrument's Identifier [Integer]
BestBid	0.00	The current Best Bid for the Instrument. [Decimal]]
BestOffer	0.00	The current Best Offer for the Instrument. [Decimal]
LastTradedPx	0.00	The last traded price for the Instrument. [Decimal]
LastTradedQty	0.00	The last traded quantity for the Instrument. [Decimal]
LastTradeTime	635872032000000000	Timestamp in .NET UTC Ticks.
SessionOpen	0.00	The Opening price. [Decimal]
SessionHigh	0.00	The Highest price. [Decimal]
SessionLow	0.00	The Lowest price. [Decimal]
SessionClose	0.00	The Closing price. [Decimal]
Volume	0.00	Volume traded. [Decimal]
CurrentDayVolume	0.00	Current day volume traded. [Decimal]
CurrentDayNumTrades	0	Number of trades for current day. [Integer]

CurrentDayPxChange	0.00	Current Day price change. [Decimal]
Rolling24HrVolume	0.00	Last 24 hr volume. [Decimal]
Rolling24NumTrades	0.00	Last 24 number of trades. [Integer]
Rolling24HrPxChange	0.00	Last 24 hr price change. [Decimal]
TimeStamp	635872032000000000	Timestamp in .NET UTC Ticks.

Level1UpdateEvent

When subscribed to Level1 Market Data (after calling SubscribeLevel1), you will receive Level1UpdateEvent messages from the server.

Example

```
Response <- (full frame omitted, please read Message Frame Format)

{
    "InstrumentId": 1,
    "BestBid": 0.00,
    "BestOffer": 0.00,
    "LastTradedPx": 0.00,
    "LastTradedQty": 0.00,
    "LastTradeTime": 635872032000000000,
    "SessionOpen": 0.00,
    "SessionHigh": 0.00,
    "SessionLow": 0.00,
    "SessionClose": 0.00,
    "Volume": 0.00,
    "CurrentDayVolume": 0.00,
    "CurrentDayNumTrades": 0,
    "CurrentDayPxChange": 0.0,
    "Rolling24HrVolume": 0.0,
    "Rolling24NumTrades": 0.0,
    "Rolling24HrPxChange": 0.0,
    "TimeStamp": 635872032000000000
}
```

Response Parameters

Attribute Name	Example	Description
InstrumentId	1	Instrument's Identifier [Integer]

BestBid	0.00	The current Best Bid for the Instrument. [Decimal]]
BestOffer	0.00	The current Best Offer for the Instrument. [Decimal]
LastTradedPx	0.00	The last traded price for the Instrument. [Decimal]
LastTradedQty	0.00	The last traded quantity for the Instrument. [Decimal]
LastTradeTime	635872032000000000	Timestamp in .NET UTC Ticks.
SessionOpen	0.00	The Opening price. [Decimal]
SessionHigh	0.00	The Highest price. [Decimal]
SessionLow	0.00	The Lowest price. [Decimal]
SessionClose	0.00	The Closing price. [Decimal]
Volume	0.00	Volume traded. [Decimal]
CurrentDayVolume	0.00	Current day volume traded. [Decimal]
CurrentDayNumTrades	0	Number of trades for current day. [Integer]
CurrentDayPxChange	0.00	Current Day price change. [Decimal]
Rolling24HrVolume	0.00	Last 24 hr volume. [Decimal]
Rolling24NumTrades	0.00	Last 24 number of trades. [Integer]
Rolling24HrPxChange	0.00	Last 24 hr price change. [Decimal]
TimeStamp	635872032000000000	Timestamp in .NET UTC Ticks.

UnsubscribeLevel1

Unsubscribes user from Level 1 Market Data updates for the specified Instrument.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"UnsubscribeLevel1",
```

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

Response <- (full frame omitted, please read Message Frame Format)

{
    "result": true // [Boolean]
}
//on error:
{
    "result": false, // [Boolean]
    "errormsg": "Error", //A message describing the error.
[String]
    "errorcode": 104, //An error code
    "detail": "Instrument not Found"
}
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
InstrumentId	1	Instrument's Identifier [Integer]

Response Parameters

Attribute Name	Example	Description
result	true	Result of operation, true if unsubscribed [Bool]
errormsg	""	Error message [String]
detail	“Instrument not Found”	Detailed error description [String]
errorcode	104	Error code [Integer]

Level 2 data

SubscribeLevel2

Retrieves the latest Level 2 Snapshot and Subscribes User to Level 2 Market Data updates for the specified Instrument.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"SubscribeLevel2",
    "o":"{\"OMSId\":1,\"InstrumentId\":1,\"Depth\":10}"
}

Response <- (full frame omitted, please read Message Frame Format)

[
    [18981,0,635872032000000000,2,300000,0,300000,1,0,1],
    [18982,1,635872032000000000,0,300000,1,300000,1,2,0],
    ...
]
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
InstrumentId	1	Instrument's Identifier [Integer]
Depth	10	The Depth of the book to subscribe to updates for. In this example, you would receive 10 price levels on each side of the market. [Integer]

Response Parameters

Attribute	Example	Description
1	18981	Market Data Sequence Number [64 bit Integer]
2	0	Number of Traders at this price level [Integer]

3	635872032000000000	Timestamp [64 bit Integer] .NET UTC Ticks. See code snippets for examples on converting this to other formats.
4	2	Change Type [Integer] Values: - 0=New - 1=Update - 2=Delete
5	300000	Last Traded Price [Decimal]
6	0	Number of Orders at this price level [Integer]
7	300000	Price [Decimal]
8	1	InstrumentId [Integer]
9	0	Quantity [Decimal]
10	1	Side [Integer] Values: - 0=Buy - 1=Sell

Level2UpdateEventB

When subscribed to Level 2 Market Data, you will receive Level2UpdateEvent messages from the server.

Example

```
Response <- (full frame omitted, please read Message Frame Format)

[
    [18981,0,635872032000000000,2,300000,0,95,1,0,1],
    [18982,1,635872032000000000,0,300000,1,95,1,2,0],
    ...
]
```

Response Parameters

Attribute #	Example	Description
1	18981	Market Data Sequence Number [64 bit Integer]
2	0	Number of Traders at this price level [Integer]
3	635872032000000000	Timestamp [64 bit Integer] .NET UTC Ticks.
4	2	Change Type [Integer] Values: - 0=New - 1=Update - 2=Delete
5	300000	Last Traded Price [Decimal]
6	0	Number of Orders at this price level [Integer]
7	300000	Price [Decimal]
8	1	InstrumentId [Integer]
9	0	Quantity [Decimal]
10	1	Side [Integer] Values: - 0=Buy - 1=Sell

UnsubscribeLevel2

Unsubscribes from Level 2 Market Data updates for the specified Instrument.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"UnsubscribeLevel2",
    "o":"{\"OMSId\":1,\"InstrumentId\":1}"
}
```

```
Response <- (full frame omitted, please read Message Frame Format)

{
    "result": true // [Boolean]
}
//on error:
{
    "result": false, // [Boolean]
    "errormsg": "Error" //A message describing the error. [String]
    "errorcode": 104, //An error code
    "detail": "Instrument not Found"
}
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
InstrumentId	1	Instrument's Identifier [Integer]

Response Parameters

Attribute Name	Example	Description
result	true	Result of operation, true if unsubscribed [Bool]
errormsg	""	Error message [String]
detail	“Instrument not Found”	Detailed error description [String]
errorcode	104	Error code [Integer]

Trades feed

SubscribeTrades

Retrieves the latest public market trades and Subscribes User to Trade updates for the specified Instrument.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"SubscribeTrades",
    "o":"{\"OMSId\":1,\"InstrumentId\":1,\"IncludeLastCount\":100}"
}

Response <- (full frame omitted, please read Message Frame Format)

[
    [87,1,0.01,450.98,635872032000000000,0,1],
    [88,1,0.01,450.98,635872032000000000,0,1],
    [89,1,0.0368,450.98,635872032000000000,0,1],
    ...
]
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
InstrumentId	1	Instrument's Identifier [Integer]
IncludeLastCount	10	Specifies the number of previous trades to retrieve in the immediate snapshot. Default is 100. [Integer]

Response Parameters

Attribute #	Example	Description
1	87	Trade Number [64 bit Integer]
2	1	InstrumentId [Integer]
3	0.01	Quantity [Decimal]
4	450.98	Price [Decimal]
7	635872032000000000	Timestamp

8	0	Direction [Integer] Shows if this trade represents a movement of the book price up, down, or no change. Values: - 0=NoChange - 1=Uptick - 2=Downtick
9	1	TakerSide [Integer] Side of trade representing the Taker

TradeDataUpdateEvent

When subscribed to Trades, you will receive TradeDataUpdateEvent messages from the server.

Example

```
Response <- (full frame omitted, please read Message Frame Format)

[
    [87,1,0.01,450.98,635872032000000000,0,1],
    [88,1,0.01,450.98,635872032000000000,0,1],
    [89,1,0.0368,450.98,635872032000000000,0,1],
    ...
]
```

Response Parameters

Attribute #	Example	Description
1	87	Trade Number [64 bit Integer]
2	1	InstrumentId [Integer]
3	0.01	Quantity [Decimal]
4	450.98	Price [Decimal]
7	635872032000000000	Timestamp
8	0	Direction [Integer] Shows if this trade represents a movement of the book price up, down, or no change.

		Values: - 0=NoChange - 1=Uptick - 2=Downtick
9	1	TakerSide [Integer] Side of trade representing the Taker

UnsubscribeTrades

Unsubscribes user from Trades Market Data Feed.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"UnsubscribeTrades",
    "o":"{\"OMSId\":1,\"InstrumentId\":1}"
}

Response <- (full frame omitted, please read Message Frame Format)

{
    "result": true // [Boolean]
}
//on error:
{
    "result": false, // [Boolean]
    "errormsg": "Error" //A message describing the error. [String]
    "errorcode": 104, //An error code
    "detail": "Instrument not Found"
}
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
InstrumentId	1	Instrument's Identifier [Integer]

Response Parameters

Attribute Name	Example	Description
result	true	Result of operation, true if unsubscribed [Bool]
errormsg	""	Error message [String]
detail	“Instrument not Found”	Detailed error description [String]
errorcode	104	Error code [Integer]

Account events feed

SubscribeAccountEvents

Subscribe to account-level events, such as orders, trades, deposits and withdraws. We highly recommend that you use this subscription to track your order states.

Example

```
Request ->
{
    "m":0,
    "i":0,
    "n":"SubscribeAccountEvents",
    "o":"{\"OMSId\":1,\"AccountId\":1}"
}

Response <- (full frame omitted, please read Message Frame Format)

{
    "result": true
}
```

Request Parameters

Attribute Name	Example	Description
OMSId	1	Order Management System ID. Always 1.
InstrumentId	1	Instrument's Identifier [Integer]

Response Parameters

Attribute Name	Example	Description
result	true	Result of operation, true if unsubscribed [Bool]
errormsg	""	Error message [String]

PendingDepositUpdate

When subscribed to Account Events, you will receive a PendingDepositUpdate message when a deposit is pending on your account.

Example

```
Response <- (full frame omitted, please read Message Frame Format)

{
    "AccountId": 4,
    "AssetId": 1,
    "TotalPendingDepositValue": 0.01
}
```

Response Parameters

Attribute Name	Example	Description
AccountId	4	Your account id number. [Integer]
AssetId	1	The ProductId of the pending deposit. [Integer]
TotalPendingDeposit Value	0.01	The value of the pending deposit. [Decimal]

AccountPositionEvent

When subscribed to Account Events, you will receive an AccountPositionEvent any time the balance of your account changes.

Example

```
Response <- (full frame omitted, please read Message Frame Format)

{
    "AccountId":4, //Your account id number. [Integer]
    "ProductSymbol":"BTC", //The Product Symbol for this balance
message. [String]
    "ProductId":1, //The Product Id for this balance message.
[Integer]
    "Amount":10499.1, //
    "Hold":2.1, //
    "PendingDeposits":0, //
    "PendingWithdraws":0, //
    "TotalDayDeposits":0, //
    "TotalDayWithdraws":0 //
}
```

Response Parameters

Attribute Name	Example	Description
AccountId	4	Your account id number. [Integer]
ProductId	1	The Product Id for this balance message. [Integer]
ProductSymbol	1	The Product Symbol for this balance message. [String]
Amount	10499.1	The total balance for the specified product. [Decimal]
Hold	2.1	The total amount of your balance that is on hold. Your Available balance for trading and withdraw is (Amount - Hold). [Decimal]
PendingDeposits	0	Total Deposits Pending for the specified product. [Decimal]
PendingWithdraws	0	Total Withdraws Pending for the specified product. [Decimal]
TotalDayDeposits	0	The total 24 hour deposits for the specified product. [Decimal]
TotalDayWithdraws	0	The total 24 hour withdraws for the specified product. [Decimal]

OrderStateEvent

When subscribed to Account Events, you will receive OrderState events any time the status of an order on your account changes.

Example

```
Response <- (full frame omitted, please read Message Frame Format)

{
    "Side":"Sell",
    "OrderId":9849,
    "Price":97,
    "Quantity":1,
    "Instrument":1,
    "Account":4,
```

```
    "OrderType":"Limit",
    "ClientOrderId":0,
    "OrderState":"Working",
    "ReceiveTime":0,
    "OrigQuantity":1,
    "QuantityExecuted":0,
    "ChangeReason": ""
}
```


Response Parameters

Attribute Name	Example	Description
Account	4	Your account id. [Integer]
ClientOrderId	99	Set this to your own id if you wish to use one. It will be useful for recognizing future order states related to this call. [64 bit Integer]
Quantity	1	The Quantity (Remaining if partially or fully executed) of your order. [Decimal]
OrderType	2	The type of order. [Integer] Values: - 1 (Market) - 2 (Limit) - 3 (StopMarket) - 4 (StopLimit) - 5 (TrailingStopMarket) - 6 (TrailingStopLimit)
Instrument	1	Instrument's Identifier [Integer]
Side	"Sell"	The side of your order. [String] Values are "Sell", "Buy" Values: - "Buy" - "Sell"
OrderId	123	The Server-Assigned Order Id. [64 bit Integer]
Price	96	The Price of your order. [Decimal]
OrderState	"Working"	The current state of the order. [String] Values: - "Working"

		- "Rejected" - "FullyExecuted" - "Canceled" - "Expired"
OrigQuantity	1	The original quantity of your order. [Decimal]
QuantityExecuted	0.71	The total executed quantity. [Decimal]
ChangeReason	""	The reason for the order state change. [String] Values are “NewInputAccepted”, “NewInputRejected”, “OtherRejected”, “Expired”, “Trade”, SystemCanceledBelowMinimum”, “SystemCanceledNoMoreMarket”, “UserModided”
ReceiveTime	1532346260831	Timestamp

OrderTradeEvent

When subscribed to Account Events, you will receive OrderTradeEvent events any time one of your orders results in a trade.

Example

```
Response <- (full frame omitted, please read Message Frame Format)

{
    "TradeId":213,
    "OrderId":9848,
    "AccountId":4,
    "ClientOrderId":0,
    "InstrumentId":1,
    "Side":"Buy",
    "Quantity":0.01,
    "Price":95,
    "Value":0.95,
    "TradeTime":635978008210426109
}
```

Response Parameters

Attribute Name	Example	Description

TradeId	230	Trade Id [64 bit Integer]
OrderId	9861	Order Id [64 bit Integer]
AccountId	4	Your account id. [Integer]
ClientOrderId	0	Your client order id. [64 bit Integer]
InstrumentId	1	Instrument's Identifier [Integer]
Side	"Buy"	Side of the trade [String] Values: - "Buy" - "Sell"
Quantity	2	Quantity [Decimal]
Price	95	Price [Decimal]
Value	190	Value = price * quantity [Decimal]
TradeTime	1501354796418	Timestamp [Integer] The time at which the trade took place, in POSIX format and UTC time zone.

NewOrderRejectEvent

When subscribed to Account Events, you will receive NewOrderRejectEvent event if your order is rejected.

Example

```
Response <- (full frame omitted, please read Message Frame Format)

{
    "AccountId": 4,
    "ClientOrderId": 1234,
    "Status": "Rejected",
    "RejectReason": "No More Market"
}
```

Response Parameters

Attribute Name	Example	Description
AccountId	4	Your account id number. [Integer]
ClientOrderId	1234	Your Client Order Id [64 Bit Integer]
Status	"Rejected"	Always "Rejected" [String]
RejectReason	"No More Market"	A message describing the reason for the reject [String]

CancelOrderRejectEvent

When subscribed to Account Events, you will receive CancelOrderRejectEvent event if your attempt to cancel an order is unsuccessful.

Example

```
Response <- (full frame omitted, please read Message Frame Format)

{
    "AccountId": 4,
    "OrderId": 1,
    "OrderRevision": 0,
    "OrderType": "Unknown",
    "InstrumentId": 1,
    "Status": "Rejected",
    "RejectReason": "Order Not Found"
}
```

Response Parameters

Attribute Name	Example	Description
AccountId	4	Your account id number. [Integer]
OrderId	1234	The Order Id from your Cancel request. [64 Bit Integer]
Status	"Rejected"	Always "Rejected" [String]
RejectReason	"Order Not Found"	A message describing the reason for the reject. [String]

InstrumentId	4	The InstrumentId from your Cancel request. [Integer]
OrderType	"Unknown"	The Order Type of the order, if it was found. Otherwise, . [String]
OrderRevision	0	The Revision of the Order, if it was found. [64 Bit Integer]

MarketStateUpdate

When subscribed to Account Events, you will receive MarketStateUpdate event if the market state is administratively altered

Example

```
Response <- (full frame omitted, please read Message Frame Format)

{
    "ExchangeId":1, //
    "VenueAdapterId":1, //
    "VenueInstrumentId":1, //
    "Action":"ReOpen", //
    "PreviousStatus":"Stopped", //
    "NewStatus":"Running", //
    "ExchangeDateTime":"2016-04-21T21:48:22Z"
}
```

Response Parameters

Attribute Name	Example	Description
ExchangeId	1	Exchange Id [Integer]
VenueAdapterId	1	Internal [Integer]
VenueInstrumentId	1	Instrument Id [Integer]
Action	"ReOpen"	Market State Action [String] Values: - "Pause" - "Resume" - "Halt" - "ReOpen"

PreviousStatus	"Stopped"	Previous Market Status for Instrument [String] Values: - "Running" - "Paused" - "Stopped" - "Starting"
NewStatus	"Running"	Market Status for Instrument [String] Values: - "Running" - "Paused" - "Stopped" - "Starting"
ExchangeDateTime	"2016-04-21T21:48:22Z"	Exchange datetime [ISO Datetime string]