xAPI
Protocol Documentation

Version 2.5.0

1. Introduction

This document presents information on API communication protocol. The communication protocol of the API uses JSON format.

JSON format used by the server doesn't allow extensions (e.g. comments, other flags). JSON format standardization document is available under the following link: http://tools.ietf.org/html/rfc4627 (http://tools.ietf.org/html/rfc4627).

The connection is performed by clean socket connection. For real trading SSL connection will be used.

2. Definitions

The following definitions will be used in this specification document:

- **simple type:** type, which value is itself: int, int64, string, double, bool;
- **j-value (JSON value):** any simple type, j-object or j-array;
- **j-object (JSON object):** a record containing any number of named j-values (pair);
- **j-array (JSON array):** an array where each element is j-value;
- **j-subvalue:** j-value which is a component of a j-object.

A j-object can contain zero elements. A j-array can have zero length. The name of j-value can be an empty string.

Encoding of strings is set to UTF-8. In this format the server sends and receives data.

Definition of unix-time:

```
Unix time, or POSIX time, is a system for describing points in time, defined as
the number of milliseconds elapsed since midnight Coordinated Universal Time (UT
C) of January 1, 1970.
```

3. General data format

Each packet consists of exactly one main, unnamed j-value. The data stream consists of consecutive j–values, with no punctuation.

The main j-value is a j-object containing exactly two j-subvalues which are j-objects. The first j-subobject is named **header** and consists of at least a field type as a simple type string. This is a packet type. The second j-subobject of the main packet j-object is named **data** and its content is specific to a given packet type. The specifications for different types of packages are described in the next chapter.

A sample of properly defined packet:

```
{
        "command" : "login",
        "arguments" : {
                "userId" : "1000",
                "password": "PASSWORD"
        }
}
```

Every single command sent to the API is allowed to contain an optional field called **customTag**. The API guarantees to return the very same customTag in the command’s response. For example, the following command:

```
{
        "command" : "login",
        "arguments" : {
                "userId" : "1000",
                "password": "PASSWORD"
        },
        "customTag": "my_login_command_id"
}
```

returns (in case of a successful login):

```
{
        "status": true,
        "customTag": "my_login_command_id"
}
```

4. Communication with the API

XTB Clients:

Host:

- xapi.xtb.com

Here are the details of DEMO and REAL servers hosted on each of the addresses above:

- DEMO: main port: 5124, streaming port: 5125,
- REAL: main port: 5112, streaming port: 5113.

Furthermore, WebSockets can be used to connect to the API using the following addresses:

- wss://ws.xtb.com/demo
- wss://ws.xtb.com/demoStream
- wss://ws.xtb.com/real
- wss://ws.xtb.com/realStream

X Open Hub Clients:

Host:
There are two addresses (that can be used interchangeably)

- xapia.x-station.eu
- xapib.x-station.eu

Here are the details of DEMO and REAL servers hosted on each of the addresses above:

- DEMO: main port: 5124, streaming port: 5125,
- REAL: main port: 5112, streaming port: 5113.

Furthermore, WebSockets can be used to connect to the API using the following addresses:

- wss://ws.xapi.pro/demo
- wss://ws.xapi.pro/demoStream
- wss://ws.xapi.pro/real
- wss://ws.xapi.pro/realStream

All servers use **SSL** connection.

Communication is established as long as both server and client have opened and connected sockets.

For convenience server guarantees that every separate reply to client command returned by server will be separated by two new line characters ("\n").

1. Connection validation

In order to provide best service for all users API set rules on connection and data send process. If any of the following rules is breached, then connection is closed immediately without server notification.

List of rules:

- At most **50** simultaneous connections from the same client address are allowed (an attempt to obtain the 51st connection returns the error EX008). If you need this rule can be lenified please contact the xStore Support Team (mailto:support@xstore.pro).
- Every new connection that fails to deliver data within **one second** from when it is established may be forced to close with no notification.
- Each command invocation should not contain more than **1kB** of data.
- User should send requests in **200 ms** intervals. This rule can be broken, but if it happens **6 times** in a row the connection is dropped.
- Each command should be a **proper JSON object**.

Exception:

If the client sends a request that is a valid JSON object, but does not conform to the published API (incorrect command, missing fields, etc.), the response is sent back with the error description but the connection is not closed.

This rule prevents incorrect messages from reaching further down the processing chain and allows clients to analyze and understand the source of problem.

2. Default login credentials

Default login credentials can be obtained at: http://developers.xstore.pro/panel/ (http://developers.xstore.pro/panel).

3. Input data format

The input data format is a JSON object that consists of service name and command name. Some commands also require an object of command's arguments.

If optional `prettyPrint` field is set to true, an output JSON is printed in human-readable format. `prettyPrint` field can be omitted.

```
{
        "command": "commandName",
        "arguments": {
                "arg1Name": 10,
                "arg2Name": "Some text",
                ...
        },
        "prettyPrint": true
}
```

4. Output data format

The output data format is a JSON object that consists of `status` and `returnData` fields if command succeeded, or `status`, `errorCode` and `errorDescr` fields if an error occurred.

```
{
        "status": true,
        "returnData": JSON value
}
```

or, in case of an error:

```
{
        "status": false,
        "errorCode": "E123",
        "errorDescr": "Error description"
}
```

5. Time format

Time is number of milliseconds from 01.01.1970, 00:00 GMT.

6. Floating number format

In all Floating numbers '.' (period) is used as a decimal separator.

5. Available commands

Request-Reply commands are performed on main connection socket. The reply is sent by main connection socket.

1. Login

In order to perform any action client application have to perform login process. No functionality is available before proper login process.

After initial login, a new session is created and all commands are executed for a logged user until he/she logs out or drops the connection.

Request:

Parameters:

Name	Type	Desc
userId	String	userId
password	String	password
appId	String	(optional, deprecated)
appName	String	(optional) application name

Example:

```
{
        "command": "login",
        "arguments": {
                "userId": "1000",
                "password": "PASSWORD",
                "appId": "test",
                "appName": "test"
        }
}
```

After successful login the system responds with the `status` message that can contain the String representing `streamSessionId` field:

```
{
        "status": true,
        "streamSessionId": "8469308861804289383"
}
```

The `streamSessionId` field of the string type, if present, is a token that can be used to establish a streaming subscription on a separate network connection. `streamSessionId` is used in streaming subscription commands.

`streamSessionId` is unique for the given main session and will change between login sessions.

2. Logout

Format of input:

```
{
        "command": "logout"
}
```

No `returnData` field in output. Only `status` message is sent.

3. Retrieving trading data

1. Command: getAllSymbols

Description: Returns array of all symbols available for the user.

Request:

Example:

```
{
        "command": "getAllSymbols"
}
```

Response:

Parameters:

name	type	description
	array	Array of `SYMBOL_RECORD`

Example:

```
{
        "status": true,
        "returnData": [SYMBOL_RECORD, SYMBOL_RECORD, ...]
}
```

Format of `SYMBOL_RECORD`:

Please be advised that result values for profit and margin calculation can be used optionally, because server is able to perform all profit/margin calculations for Client application by commands described later in this document.

name	type	description
ask	Floating number	Ask price in base currency
bid	Floating number	Bid price in base currency
categoryName	String	Category name
contractSize	Number	Size of 1 lot
currency	String	Currency
currencyPair	Boolean	Indicates whether the symbol represents a currency pair

name	type	description
currencyProfit	String	The currency of calculated profit
description	String	Description
expiration	Time	Null if not applicable
groupName	String	Symbol group name
high	Floating number	The highest price of the day in base currency
initialMargin	Number	Initial margin for 1 lot order, used for profit/margin calculation
instantMaxVolume	Number	Maximum instant volume multiplied by 100 (in lots)
leverage	Floating number	Symbol leverage
longOnly	Boolean	Long only
lotMax	Floating number	Maximum size of trade
lotMin	Floating number	Minimum size of trade
lotStep	Floating number	A value of minimum step by which the size of trade can be changed (within `lotMin` - `lotMax` range)
low	Floating number	The lowest price of the day in base currency
marginHedged	Number	Used for profit calculation
marginHedgedStrong	Boolean	For margin calculation
marginMaintenance	Number	For margin calculation, null if not applicable
marginMode	Number	For margin calculation
percentage	Floating number	Percentage
pipsPrecision	Number	Number of symbol's pip decimal places
precision	Number	Number of symbol's price decimal places
profitMode	Number	For profit calculation
quoteId	Number	Source of price
shortSelling	Boolean	Indicates whether short selling is allowed on the instrument

name	type	description
spreadRaw	Floating number	The difference between raw ask and bid prices
spreadTable	Floating number	Spread representation
starting	Time	Null if not applicable
stepRuleId	Number	Appropriate step rule ID from `getStepRules` command response
stopsLevel	Number	Minimal distance (in pips) from the current price where the stopLoss/takeProfit can be set
swap_rollover3days	Number	Time when additional swap is accounted for weekend
swapEnable	Boolean	Indicates whether swap value is added to position on end of day
swapLong	Floating number	Swap value for long positions in pips
swapShort	Floating number	Swap value for short positions in pips
swapType	Number	Type of swap calculated
symbol	String	Symbol name
tickSize	Floating number	Smallest possible price change, used for profit/margin calculation, null if not applicable
tickValue	Floating number	Value of smallest possible price change (in base currency), used for profit/margin calculation, null if not applicable
time	Time	Ask & bid tick time
timeString	String	Time in String
trailingEnabled	Boolean	Indicates whether trailing stop (offset) is applicable to the instrument.
type	Number	Instrument class number

Example:

```
{
        "ask": 4000.0,
        "bid": 4000.0,
        "categoryName": "Forex",
        "contractSize": 100000,
        "currency": "USD",
        "currencyPair": true,
        "currencyProfit": "SEK",
        "description": "USD/PLN",
        "expiration": null,
        "groupName": "Minor",
        "high": 4000.0,
        "initialMargin": 0,
        "instantMaxVolume": 0,
        "leverage": 1.5,
        "longOnly": false,
        "lotMax": 10.0,
        "lotMin": 0.1,
        "lotStep": 0.1,
        "low": 3500.0,
        "marginHedged": 0,
        "marginHedgedStrong": false,
        "marginMaintenance": null,
        "marginMode": 101,
        "percentage": 100.0,
        "precision": 2,
        "profitMode": 5,
        "quoteId": 1,
        "shortSelling": true,
        "spreadRaw": 0.000003,
        "spreadTable": 0.00042,
        "starting": null,
        "stepRuleId": 1,
        "stopsLevel": 0,
        "swap_rollover3days": 0,
        "swapEnable": true,
        "swapLong": -2.55929,
        "swapShort": 0.131,
        "swapType": 0,
        "symbol": "USDPLN",
        "tickSize": 1.0,
        "tickValue": 1.0,
        "time": 1272446136891,
        "timeString": "Thu May 23 12:23:44 EDT 2013",
        "trailingEnabled": true,
        "type": 21
}
```

Possible values of `quoteId` field:

name	value	description
fixed	1	fixed

name	value	description
float	2	float
depth	3	depth
cross	4	cross

Possible values of `marginMode` field:

name	value	description
Forex	101	Forex
CFD leveraged	102	CFD leveraged
CFD	103	CFD

Possible values of `profitMode` field:

name	value	description
FOREX	5	FOREX
CFD	6	CFD

2. Command: getCalendar

Description: Returns calendar with market events.

Request:

Example:

```
{
	"command": "getCalendar"
}
```

Response:

Parameters:

name	type	description
	array	Array of `CALENDAR_RECORD`

Example:

```
{
	"status": true,
	"returnData": [CALENDAR_RECORD, CALENDAR_RECORD, ...]
}
```

Format of `CALENDAR_RECORD`:

name	type	description
country	String	Two letter country code
current	String	Market value (current), empty before time of release of this value (time from "time" record)
forecast	String	Forecasted value
impact	String	Impact on market
period	String	Information period
previous	String	Value from previous information release
time	Time	Time, when the information will be released (in this time empty "current" value should be changed with exact released value)
title	String	Name of the indicator for which values will be released

Example:

```
{
        "country": "CA",
        "current": "",
        "forecast": "",
        "impact": "3",
        "period": "(FEB)",
        "previous": "58.3",
        "time": 1374846900000,
        "title": "Ivey Purchasing Managers Index"
}
```

Possible values of `impact` field:

name	value	description
low	1	low
medium	2	medium
high	3	high

3. Command: getChartLastRequest

Description: **Please note that this function can be usually replaced by its streaming equivalent `getCandles` which is the preferred way of retrieving current candle data.** Returns chart info, from start date to the current time. If the chosen period of `CHART_LAST_INFO_RECORD` is greater than 1 minute, the last candle returned by the API can change until the end of the period (the candle is being automatically updated every minute).

Limitations: there are limitations in charts data availability. Detailed ranges for charts data, what can be accessed with specific period, are as follows:

PERIOD_M1 --- <0-1) month, i.e. one month time
PERIOD_M30 --- <1-7) month, six months time
PERIOD_H4 --- <7-13) month, six months time
PERIOD_D1 --- 13 month, and earlier on

Note, that specific PERIOD_ is the lowest (i.e. the most detailed) period, accessible in listed range. For instance, in months range <1-7) you can access periods: PERIOD_M30, PERIOD_H1, PERIOD_H4, PERIOD_D1, PERIOD_W1, PERIOD_MN1. Specific data ranges availability is guaranteed, however those ranges may be wider, e.g.: PERIOD_M1 may be accessible for 1.5 months back from now, where 1.0 months is guaranteed.

Example scenario:

- request charts of 5 minutes period, for 3 months time span, back from now;
- response: you are guaranteed to get 1 month of 5 minutes charts; because, 5 minutes period charts are not accessible 2 months and 3 months back from now.

Request:

Parameters:

name	type	description
info	CHART_LAST_INFO_RECORD	info

Example:

```
{
        "command": "getChartLastRequest",
        "arguments": {
                "info": CHART_LAST_INFO_RECORD
        }
}
```

Format of `CHART_LAST_INFO_RECORD`:

name	type	description
period	Number	Period code
start	Time	Start of chart block (rounded down to the nearest interval and excluding)
symbol	String	Symbol

Example:

```
{
        "period": 5,
        "start": 1262944112000,
        "symbol": "PKN.PL"
}
```

Possible values of `period` field:

name	value	description

name	value	description
PERIOD_M1	1	1 minute
PERIOD_M5	5	5 minutes
PERIOD_M15	15	15 minutes
PERIOD_M30	30	30 minutes
PERIOD_H1	60	60 minutes (1 hour)
PERIOD_H4	240	240 minutes (4 hours)
PERIOD_D1	1440	1440 minutes (1 day)
PERIOD_W1	10080	10080 minutes (1 week)
PERIOD_MN1	43200	43200 minutes (30 days)

Response:

Parameters:

name	type	description
digits	Number	Number of decimal places
rateInfos	array	Array of `RATE_INFO_RECORD`

Example:

```
{
        "status": true,
        "returnData": {
                "digits": 4,
                "rateInfos": [RATE_INFO_RECORD, RATE_INFO_RECORD, ...]
        }
}
```

Format of `RATE_INFO_RECORD`:

Price values must be divided by 10 to the power of digits in order to obtain exact prices.

name	type	description
close	Floating number	Value of close price (shift from open price)
ctm	Time	Candle start time in CET / CEST time zone (see Daylight Saving Time, DST)
ctmString	String	String representation of the 'ctm' field

name	type	description
high	Floating number	Highest value in the given period (shift from open price)
low	Floating number	Lowest value in the given period (shift from open price)
open	Floating number	Open price (in base currency * 10 to the power of digits)
vol	Floating number	Volume in lots

Example:

```
{
        "close": 1.0,
        "ctm": 1389362640000,
        "ctmString": "Jan 10, 2014 3:04:00 PM",
        "high": 6.0,
        "low": 0.0,
        "open": 41848.0,
        "vol": 0.0
}
```

4. Command: getChartRangeRequest

Description: **Please note that this function can be usually replaced by its streaming equivalent `getCandles` which is the preferred way of retrieving current candle data.** Returns chart info with data between given start and end dates.

Limitations: there are limitations in charts data availability. Detailed ranges for charts data, what can be accessed with specific period, are as follows:

PERIOD_M1 --- <0-1) month, i.e. one month time
PERIOD_M30 --- <1-7) month, six months time
PERIOD_H4 --- <7-13) month, six months time
PERIOD_D1 --- 13 month, and earlier on

Note, that specific PERIOD_ is the lowest (i.e. the most detailed) period, accessible in listed range. For instance, in months range <1-7) you can access periods: PERIOD_M30, PERIOD_H1, PERIOD_H4, PERIOD_D1, PERIOD_W1, PERIOD_MN1. Specific data ranges availability is guaranteed, however those ranges may be wider, e.g.: PERIOD_M1 may be accessible for 1.5 months back from now, where 1.0 months is guaranteed.

Request:

Parameters:

name	type	description
info	CHART_RANGE_INFO_RECORD	info

Example:

```
{
	"command": "getChartRangeRequest",
	"arguments": {
		"info": CHART_RANGE_INFO_RECORD
	}
}
```

Format of `CHART_RANGE_INFO_RECORD`:

Ticks field - if ticks is not set or value is 0, `getChartRangeRequest` works as before (you must send valid `start` and `end` time fields).
If ticks value is not equal to 0, field `end` is ignored.
If ticks >0 (e.g. N) then API returns N candles from time start.
If ticks <0 then API returns N candles to time start.
It is possible for API to return fewer chart candles than set in tick field.

name	type	description
end	Time	End of chart block (rounded down to the nearest interval and excluding)
period	Number	Period code
start	Time	Start of chart block (rounded down to the nearest interval and excluding)
symbol	String	Symbol
ticks	Number	Number of ticks needed, this field is optional, please read the description above

Example:

```
{
	"end": 1262944412000,
	"period": 5,
	"start": 1262944112000,
	"symbol": "PKN.PL",
	"ticks": 0
}
```

Possible values of `period` field:

name	value	description
PERIOD_M1	1	1 minute
PERIOD_M5	5	5 minutes
PERIOD_M15	15	15 minutes
PERIOD_M30	30	30 minutes
PERIOD_H1	60	60 minutes (1 hour)
PERIOD_H4	240	240 minutes (4 hours)

name	value	description
PERIOD_D1	1440	1440 minutes (1 day)
PERIOD_W1	10080	10080 minutes (1 week)
PERIOD_MN1	43200	43200 minutes (30 days)

Response:

Parameters:

name	type	description
digits	Number	Number of decimal places
rateInfos	array	Array of `RATE_INFO_RECORD`

Example:

```
{
	"status": true,
	"returnData": {
		"digits": 4,
		"rateInfos": [RATE_INFO_RECORD, RATE_INFO_RECORD, ...]
	}
}
```

Format of `RATE_INFO_RECORD`:

Price values must be divided by 10 to the power of digits in order to obtain exact prices.

name	type	description
close	Floating number	Value of close price (shift from open price)
ctm	Time	Candle start time in CET / CEST time zone (see Daylight Saving Time, DST)
ctmString	String	String representation of the 'ctm' field
high	Floating number	Highest value in the given period (shift from open price)
low	Floating number	Lowest value in the given period (shift from open price)
open	Floating number	Open price (in base currency * 10 to the power of digits)
vol	Floating number	Volume in lots

Example:

```
{
	"close": 1.0,
	"ctm": 1389362640000,
	"ctmString": "Jan 10, 2014 3:04:00 PM",
	"high": 6.0,
	"low": 0.0,
	"open": 41848.0,
	"vol": 0.0
}
```

5. Command: getCommissionDef

Description: Returns calculation of commission and rate of exchange. The value is calculated as expected value, and therefore might not be perfectly accurate.

Request:

Parameters:

name	type	description
symbol	String	symbol
volume	Floating number	volume

Example:

```
{
	"command": "getCommissionDef",
	"arguments": {
		"symbol": "T.US",
		"volume": 1.0
	}
}
```

Response:

Parameters:

name	type	description
commission	Floating number	calculated commission in account currency, could be null if not applicable
rateOfExchange	Floating number	rate of exchange between account currency and instrument base currency, could be null if not applicable

Example:

```
{
	"status": true,
	"returnData": {
		"commission": 0.51,
		"rateOfExchange": 0.1609
	}
}
```

6. Command: getCurrentUserData

Description: Returns information about account currency, and account leverage.

Request:

Example:

```
{
	"command": "getCurrentUserData"
}
```

Response:

Parameters:

name	type	description
companyUnit	Number	Unit the account is assigned to.
currency	String	account currency
group	String	group
ibAccount	Boolean	Indicates whether this account is an IB account.
leverage	Number	This field should not be used. It is inactive and its value is always 1.
leverageMultiplier	Floating number	The factor used for margin calculations. The actual value of leverage can be calculated by dividing this value by 100.
spreadType	String	spreadType, null if not applicable
trailingStop	Boolean	Indicates whether this account is enabled to use trailing stop.

Example:

```
{
	"status": true,
	"returnData": {
		"companyUnit": 8,
		"currency": "PLN",
		"group": "demoPLeurSTANDARD200",
		"ibAccount": false,
		"leverage": 1,
		"leverageMultiplier": 0.25,
		"spreadType": "FLOAT",
		"trailingStop": false
	}
}
```

7. Command: getIbsHistory

Description: Returns IBs data from the given time range.

Request:

Parameters:

name	type	description
end	Time	End of IBs history block
start	Time	Start of IBs history block

Example:

```
{
	"command": "getIbsHistory",
	"arguments": {
		"end": 1395053810991,
		"start": 1394449010991
	}
}
```

Response:

Parameters:

name	type	description
	array	Array of `IB_RECORD`

Example:

```
{
	"status": true,
	"returnData": [IB_RECORD, IB_RECORD, ...]
}
```

Format of `IB_RECORD`:

name	type	description
closePrice	Floating number	IB close price or null if not allowed to view
login	String	IB user login or null if not allowed to view
nominal	Floating number	IB nominal or null if not allowed to view
openPrice	Floating number	IB open price or null if not allowed to view
side	Number	Operation code or null if not allowed to view
surname	String	IB user surname or null if not allowed to view
symbol	String	Symbol or null if not allowed to view
timestamp	Time	Time the record was created or null if not allowed to view
volume	Floating number	Volume in lots or null if not allowed to view

Example:

```
{
	"closePrice": 1.39302,
	"login": "12345",
	"nominal": 6.00,
	"openPrice": 1.39376,
	"side": 0,
	"surname": "IB_Client_1",
	"symbol": "EURUSD",
	"timestamp": 1395755870000,
	"volume": 1.0
}
```

Possible values of `side` field:

name	value	description
BUY	0	buy
SELL	1	sell

8. Command: getMarginLevel

Description: **Please note that this function can be usually replaced by its streaming equivalent `getBalance` which is the preferred way of retrieving account indicators.** Returns various account indicators.

Request:

Example:

```
{
	"command": "getMarginLevel"
}
```

Response:

Parameters:

name	type	description
balance	Floating number	balance in account currency
credit	Floating number	credit
currency	String	user currency
equity	Floating number	sum of balance and all profits in account currency
margin	Floating number	margin requirements in account currency
margin_free	Floating number	free margin in account currency
margin_level	Floating number	margin level percentage

Example:

```
{
        "status": true,
        "returnData": {
                "balance": 995800269.43,
                "credit": 1000.00,
                "currency": "PLN",
                "equity": 995985397.56,
                "margin": 572634.43,
                "margin_free": 995227635.00,
                "margin_level": 173930.41
        }
}
```

9. Command: getMarginTrade

Description: Returns expected margin for given instrument and volume. The value is calculated as expected margin value, and therefore might not be perfectly accurate.

Request:

Parameters:

name	type	description
symbol	String	symbol
volume	Floating number	volume

Example:

```
{
	"command": "getMarginTrade",
	"arguments": {
		"symbol": "EURPLN",
		"volume": 1.0
	}
}
```

Response:

Parameters:

name	type	description
margin	Floating number	calculated margin in account currency

Example:

```
{
	"status": true,
	"returnData": {
		"margin": 4399.350
	}
}
```

10. Command: getNews

Description: **Please note that this function can be usually replaced by its streaming equivalent `getNews` which is the preferred way of retrieving news data.** Returns news from trading server which were sent within specified period of time.

Request:

Parameters:

name	type	description
end	Time	Time, 0 means current time for simplicity
start	Time	Time

Example:

```
{
	"command": "getNews",
	"arguments": {
		"end": 0,
		"start": 1275993488000
	}
}
```

Response:

Parameters:

name	type	description
	array	Array of `NEWS_TOPIC_RECORD`

Example:

```
{
        "status": true,
        "returnData": [NEWS_TOPIC_RECORD, NEWS_TOPIC_RECORD, ...]
}
```

Format of `NEWS_TOPIC_RECORD`:

name	type	description
body	String	Body
bodylen	Number	Body length
key	String	News key
time	Time	Time
timeString	String	Time string
title	String	News title

Example:

```
{
        "body": "<html>...</html>",
        "bodylen": 110,
        "key": "1f6da766abd29927aa854823f0105c23",
        "time": 1262944112000,
        "timeString": "May 17, 2013 4:30:00 PM",
        "title": "Breaking trend"
}
```

11. Command: getProfitCalculation

Description: Calculates estimated profit for given deal data Should be used for calculator-like apps only. Profit for opened transactions should be taken from server, due to higher precision of server calculation.

Request:

Parameters:

name	type	description
closePrice	Floating number	theoretical close price of order
cmd	Number	Operation code
openPrice	Floating number	theoretical open price of order

name	type	description
symbol	String	symbol
volume	Floating number	volume

Example:

```
{
        "command": "getProfitCalculation",
        "arguments": {
                "closePrice": 1.3000,
                "cmd": 0,
                "openPrice": 1.2233,
                "symbol": "EURPLN",
                "volume": 1.0
        }
}
```

Possible values of `cmd` field:

name	value	description
BUY	0	buy
SELL	1	sell
BUY_LIMIT	2	buy limit
SELL_LIMIT	3	sell limit
BUY_STOP	4	buy stop
SELL_STOP	5	sell stop
BALANCE	6	Read only. Used in `getTradesHistory` for manager's deposit/withdrawal operations (profit>0 for deposit, profit<0 for withdrawal).
CREDIT	7	Read only

Response:

Parameters:

name	type	description
profit	Floating number	Profit in account currency

Example:

```
{
        "status": true,
        "returnData": {
                "profit": 714.303
        }
}
```

12. Command: getServerTime

Description: Returns current time on trading server.

Request:

Example:

```
{
        "command": "getServerTime"
}
```

Response:

Parameters:

name	type	description
time	Time	Time
timeString	String	Time described in form set on server (local time of server)

Example:

```
{
        "status": true,
        "returnData": {
                "time": 1392211379731,
                "timeString": "Feb 12, 2014 2:22:59 PM"
        }
}
```

13. Command: getStepRules

Description: Returns a list of step rules for DMAs.

Request:

Example:

```
{
        "command": "getStepRules"
}
```

Response:

Parameters:

name	type	description

name	type	description
	array	Array of `STEP_RULE_RECORD`

Example:

```
{
	"status": true,
	"returnData": [STEP_RULE_RECORD, STEP_RULE_RECORD, ...]
}
```

Format of `STEP_RULE_RECORD`:

name	type	description
id	Number	Step rule ID
name	String	Step rule name
steps	array	Array of `STEP_RECORD`

Example:

```
{
	"id": 1,
	"name": "Forex",
	"steps": [STEP_RECORD, STEP_RECORD, ...]
}
```

Format of `STEP_RECORD`:

name	type	description
fromValue	Floating number	Lower border of the volume range
step	Floating number	lotStep value in the given volume range

Example:

```
{
	"fromValue": 0.1,
	"step": 0.0025
}
```

14. Command: getSymbol

Description: Returns information about symbol available for the user.

Request:

Parameters:

name	type	description

name	type	description
symbol	String	symbol

Example:

```
{
        "command": "getSymbol",
        "arguments": {
                "symbol": "EURPLN"
        }
}
```

Response:

Parameters:

name	type	description
	SYMBOL_RECORD	`SYMBOL_RECORD`

Example:

```
{
        "status": true,
        "returnData": SYMBOL_RECORD
}
```

Format of `SYMBOL_RECORD`:

Please be advised that result values for profit and margin calculation can be used optionally, because server is able to perform all profit/margin calculations for Client application by commands described later in this document.

name	type	description
ask	Floating number	Ask price in base currency
bid	Floating number	Bid price in base currency
categoryName	String	Category name
contractSize	Number	Size of 1 lot
currency	String	Currency
currencyPair	Boolean	Indicates whether the symbol represents a currency pair
currencyProfit	String	The currency of calculated profit
description	String	Description
expiration	Time	Null if not applicable

name	type	description
groupName	String	Symbol group name
high	Floating number	The highest price of the day in base currency
initialMargin	Number	Initial margin for 1 lot order, used for profit/margin calculation
instantMaxVolume	Number	Maximum instant volume multiplied by 100 (in lots)
leverage	Floating number	Symbol leverage
longOnly	Boolean	Long only
lotMax	Floating number	Maximum size of trade
lotMin	Floating number	Minimum size of trade
lotStep	Floating number	A value of minimum step by which the size of trade can be changed (within `lotMin` - `lotMax` range)
low	Floating number	The lowest price of the day in base currency
marginHedged	Number	Used for profit calculation
marginHedgedStrong	Boolean	For margin calculation
marginMaintenance	Number	For margin calculation, null if not applicable
marginMode	Number	For margin calculation
percentage	Floating number	Percentage
pipsPrecision	Number	Number of symbol's pip decimal places
precision	Number	Number of symbol's price decimal places
profitMode	Number	For profit calculation
quoteId	Number	Source of price
shortSelling	Boolean	Indicates whether short selling is allowed on the instrument
spreadRaw	Floating number	The difference between raw ask and bid prices
spreadTable	Floating number	Spread representation

name	type	description
starting	Time	Null if not applicable
stepRuleId	Number	Appropriate step rule ID from `getStepRules` command response
stopsLevel	Number	Minimal distance (in pips) from the current price where the stopLoss/takeProfit can be set
swap_rollover3days	Number	Time when additional swap is accounted for weekend
swapEnable	Boolean	Indicates whether swap value is added to position on end of day
swapLong	Floating number	Swap value for long positions in pips
swapShort	Floating number	Swap value for short positions in pips
swapType	Number	Type of swap calculated
symbol	String	Symbol name
tickSize	Floating number	Smallest possible price change, used for profit/margin calculation, null if not applicable
tickValue	Floating number	Value of smallest possible price change (in base currency), used for profit/margin calculation, null if not applicable
time	Time	Ask & bid tick time
timeString	String	Time in String
trailingEnabled	Boolean	Indicates whether trailing stop (offset) is applicable to the instrument.
type	Number	Instrument class number

Example:

```
{
	"ask": 4000.0,
	"bid": 4000.0,
	"categoryName": "Forex",
	"contractSize": 100000,
	"currency": "USD",
	"currencyPair": true,
	"currencyProfit": "SEK",
	"description": "USD/PLN",
	"expiration": null,
	"groupName": "Minor",
	"high": 4000.0,
	"initialMargin": 0,
	"instantMaxVolume": 0,
	"leverage": 1.5,
	"longOnly": false,
	"lotMax": 10.0,
	"lotMin": 0.1,
	"lotStep": 0.1,
	"low": 3500.0,
	"marginHedged": 0,
	"marginHedgedStrong": false,
	"marginMaintenance": null,
	"marginMode": 101,
	"percentage": 100.0,
	"precision": 2,
	"profitMode": 5,
	"quoteId": 1,
	"shortSelling": true,
	"spreadRaw": 0.000003,
	"spreadTable": 0.00042,
	"starting": null,
	"stepRuleId": 1,
	"stopsLevel": 0,
	"swap_rollover3days": 0,
	"swapEnable": true,
	"swapLong": -2.55929,
	"swapShort": 0.131,
	"swapType": 0,
	"symbol": "USDPLN",
	"tickSize": 1.0,
	"tickValue": 1.0,
	"time": 1272446136891,
	"timeString": "Thu May 23 12:23:44 EDT 2013",
	"trailingEnabled": true,
	"type": 21
}
```

Possible values of `quoteId` field:

name	value	description
fixed	1	fixed

name	value	description
float	2	float
depth	3	depth
cross	4	cross

Possible values of `marginMode` field:

name	value	description
Forex	101	Forex
CFD leveraged	102	CFD leveraged
CFD	103	CFD

Possible values of `profitMode` field:

name	value	description
FOREX	5	FOREX
CFD	6	CFD

15. Command: getTickPrices

Description: **Please note that this function can be usually replaced by its streaming equivalent `getTickPrices` which is the preferred way of retrieving ticks data.** Returns array of current quotations for given symbols, only quotations that changed from given timestamp are returned. New timestamp obtained from output will be used as an argument of the next call of this command.

Request:

Parameters:

name	type	description
level	Number	price level
symbols	array	Array of symbol names (Strings)
timestamp	Time	The time from which the most recent tick should be looked for. Historical prices cannot be obtained using this parameter. It can only be used to verify whether a price has changed since the given time.

Example:

```
{
	"command": "getTickPrices",
	"arguments": {
		"level": 0,
		"symbols": ["EURPLN", "AGO.PL", ...],
		"timestamp": 1262944112000
	}
}
```

Possible values of `level` field:

name	value	description
	-1	all available levels
	0	base level bid and ask price for instrument
	>0	specified level

Response:

Parameters:

name	type	description
quotations	array	Array of `TICK_RECORD`

Example:

```
{
	"status": true,
	"returnData": {
		"quotations": [TICK_RECORD, TICK_RECORD, ...]
	}
}
```

Format of `TICK_RECORD`:

name	type	description
ask	Floating number	Ask price in base currency
askVolume	Number	Number of available lots to buy at given price or null if not applicable
bid	Floating number	Bid price in base currency
bidVolume	Number	Number of available lots to buy at given price or null if not applicable
high	Floating number	The highest price of the day in base currency

name	type	description
level	Number	Price level
low	Floating number	The lowest price of the day in base currency
spreadRaw	Floating number	The difference between raw ask and bid prices
spreadTable	Floating number	Spread representation
symbol	String	Symbol
timestamp	Time	Timestamp

Example:

```
{
	"ask": 4000.0,
	"askVolume": 15000,
	"bid": 4000.0,
	"bidVolume": 16000,
	"high": 4000.0,
	"level": 0,
	"low": 3500.0,
	"spreadRaw": 0.000003,
	"spreadTable": 0.00042,
	"symbol": "KOMB.CZ",
	"timestamp": 1272529161605
}
```

Possible values of `level` field:

name	value	description
	-1	all available levels
	0	base level bid and ask price for instrument
	>0	specified level

16. Command: getTradeRecords

Description: Returns array of trades listed in `orders` argument.

Request:

Parameters:

name	type	description
orders	array	Array of orders (position numbers)

Example:

```
{
	"command": "getTradeRecords",
	"arguments": {
		"orders": [7489839, 7489841, ...]
	}
}
```

Response:

Parameters:

name	type	description
	array	Array of `TRADE_RECORD`

Example:

```
{
	"status": true,
	"returnData": [TRADE_RECORD, TRADE_RECORD, ...]
}
```

Format of `TRADE_RECORD`:

`cmd` is the operation code, for user's trade operations it equals to `cmd` from `TRADE_TRANS_INFO` record used as an argument in `tradeTransaction` command

name	type	description
close_price	Floating number	Close price in base currency
close_time	Time	Null if order is not closed
close_timeString	String	Null if order is not closed
closed	Boolean	Closed
cmd	Number	Operation code
comment	String	Comment
commission	Floating number	Commission in account currency, null if not applicable
customComment	String	The value the customer may provide in order to retrieve it later.
digits	Number	Number of decimal places
expiration	Time	Null if order is not closed
expirationString	String	Null if order is not closed

name	type	description
margin_rate	Floating number	Margin rate
offset	Number	Trailing offset
open_price	Floating number	Open price in base currency
open_time	Time	Open time
open_timeString	String	Open time string
order	Number	Order number for opened transaction
order2	Number	Order number for closed transaction
position	Number	Order number common both for opened and closed transaction
profit	Floating number	Profit in account currency
sl	Floating number	Zero if stop loss is not set (in base currency)
storage	Floating number	order swaps in account currency
symbol	String	symbol name or null for deposit/withdrawal operations
timestamp	Time	Timestamp
tp	Floating number	Zero if take profit is not set (in base currency)
volume	Floating number	Volume in lots

Example:

```
{
	"close_price": 1.3256,
	"close_time": null,
	"close_timeString": null,
	"closed": false,
	"cmd": 0,
	"comment": "Web Trader",
	"commission": 0.0,
	"customComment": "Some text",
	"digits": 4,
	"expiration": null,
	"expirationString": null,
	"margin_rate": 0.0,
	"offset": 0,
	"open_price": 1.4,
	"open_time": 1272380927000,
	"open_timeString": "Fri Jan 11 10:03:36 CET 2013",
	"order": 7497776,
	"order2": 1234567,
	"position": 1234567,
	"profit": -2196.44,
	"sl": 0.0,
	"storage": -4.46,
	"symbol": "EURUSD",
	"timestamp": 1272540251000,
	"tp": 0.0,
	"volume": 0.10
}
```

Possible values of `cmd` field:

name	value	description
BUY	0	buy
SELL	1	sell
BUY_LIMIT	2	buy limit
SELL_LIMIT	3	sell limit
BUY_STOP	4	buy stop
SELL_STOP	5	sell stop
BALANCE	6	Read only. Used in `getTradesHistory` for manager's deposit/withdrawal operations (profit>0 for deposit, profit<0 for withdrawal).
CREDIT	7	Read only

17. Command: getTrades

Description: **Please note that this function can be usually replaced by its streaming equivalent `getTrades` which is the preferred way of retrieving trades data.** Returns array of user's trades.

Request:

Parameters:

name	type	description
openedOnly	boolean	if true then only opened trades will be returned

Example:

```
{
	"command": "getTrades",
	"arguments": {
		"openedOnly": true
	}
}
```

Response:

Parameters:

name	type	description
	array	Array of `TRADE_RECORD`

Example:

```
{
	"status": true,
	"returnData": [TRADE_RECORD, TRADE_RECORD, ...]
}
```

Format of `TRADE_RECORD`:

`cmd` is the operation code, for user's trade operations it equals to `cmd` from `TRADE_TRANS_INFO` record used as an argument in `tradeTransaction` command

name	type	description
close_price	Floating number	Close price in base currency
close_time	Time	Null if order is not closed
close_timeString	String	Null if order is not closed
closed	Boolean	Closed
cmd	Number	Operation code
comment	String	Comment

name	type	description
commission	Floating number	Commission in account currency, null if not applicable
customComment	String	The value the customer may provide in order to retrieve it later.
digits	Number	Number of decimal places
expiration	Time	Null if order is not closed
expirationString	String	Null if order is not closed
margin_rate	Floating number	Margin rate
offset	Number	Trailing offset
open_price	Floating number	Open price in base currency
open_time	Time	Open time
open_timeString	String	Open time string
order	Number	Order number for opened transaction
order2	Number	Order number for closed transaction
position	Number	Order number common both for opened and closed transaction
profit	Floating number	Profit in account currency
sl	Floating number	Zero if stop loss is not set (in base currency)
storage	Floating number	order swaps in account currency
symbol	String	symbol name or null for deposit/withdrawal operations
timestamp	Time	Timestamp
tp	Floating number	Zero if take profit is not set (in base currency)
volume	Floating number	Volume in lots

Example:

```
{
        "close_price": 1.3256,
        "close_time": null,
        "close_timeString": null,
        "closed": false,
        "cmd": 0,
        "comment": "Web Trader",
        "commission": 0.0,
        "customComment": "Some text",
        "digits": 4,
        "expiration": null,
        "expirationString": null,
        "margin_rate": 0.0,
        "offset": 0,
        "open_price": 1.4,
        "open_time": 1272380927000,
        "open_timeString": "Fri Jan 11 10:03:36 CET 2013",
        "order": 7497776,
        "order2": 1234567,
        "position": 1234567,
        "profit": -2196.44,
        "sl": 0.0,
        "storage": -4.46,
        "symbol": "EURUSD",
        "timestamp": 1272540251000,
        "tp": 0.0,
        "volume": 0.10
}
```

Possible values of `cmd` field:

name	value	description
BUY	0	buy
SELL	1	sell
BUY_LIMIT	2	buy limit
SELL_LIMIT	3	sell limit
BUY_STOP	4	buy stop
SELL_STOP	5	sell stop
BALANCE	6	Read only. Used in `getTradesHistory` for manager's deposit/withdrawal operations (profit>0 for deposit, profit<0 for withdrawal).
CREDIT	7	Read only

18. Command: getTradesHistory

Description: **Please note that this function can be usually replaced by its streaming equivalent `getTrades` which is the preferred way of retrieving trades data.** Returns array of user's trades which were closed within specified period of time.

Request:

Parameters:

name	type	description
end	Time	Time, 0 means current time for simplicity
start	Time	Time, 0 means last month interval

Example:

```
{
        "command": "getTradesHistory",
        "arguments": {
                "end": 0,
                "start": 1275993488000
        }
}
```

Response:

Parameters:

name	type	description
	array	Array of `TRADE_RECORD`

Example:

```
{
        "status": true,
        "returnData": [TRADE_RECORD, TRADE_RECORD, ...]
}
```

Format of `TRADE_RECORD`:

`cmd` is the operation code, for user's trade operations it equals to `cmd` from `TRADE_TRANS_INFO` record used as an argument in `tradeTransaction` command

name	type	description
close_price	Floating number	Close price in base currency
close_time	Time	Null if order is not closed
close_timeString	String	Null if order is not closed
closed	Boolean	Closed

name	type	description
cmd	Number	Operation code
comment	String	Comment
commission	Floating number	Commission in account currency, null if not applicable
customComment	String	The value the customer may provide in order to retrieve it later.
digits	Number	Number of decimal places
expiration	Time	Null if order is not closed
expirationString	String	Null if order is not closed
margin_rate	Floating number	Margin rate
offset	Number	Trailing offset
open_price	Floating number	Open price in base currency
open_time	Time	Open time
open_timeString	String	Open time string
order	Number	Order number for opened transaction
order2	Number	Order number for closed transaction
position	Number	Order number common both for opened and closed transaction
profit	Floating number	Profit in account currency
sl	Floating number	Zero if stop loss is not set (in base currency)
storage	Floating number	order swaps in account currency
symbol	String	symbol name or null for deposit/withdrawal operations
timestamp	Time	Timestamp
tp	Floating number	Zero if take profit is not set (in base currency)
volume	Floating number	Volume in lots

Example:

```
{
        "close_price": 1.3256,
        "close_time": null,
        "close_timeString": null,
        "closed": false,
        "cmd": 0,
        "comment": "Web Trader",
        "commission": 0.0,
        "customComment": "Some text",
        "digits": 4,
        "expiration": null,
        "expirationString": null,
        "margin_rate": 0.0,
        "offset": 0,
        "open_price": 1.4,
        "open_time": 1272380927000,
        "open_timeString": "Fri Jan 11 10:03:36 CET 2013",
        "order": 7497776,
        "order2": 1234567,
        "position": 1234567,
        "profit": -2196.44,
        "sl": 0.0,
        "storage": -4.46,
        "symbol": "EURUSD",
        "timestamp": 1272540251000,
        "tp": 0.0,
        "volume": 0.10
}
```

Possible values of `cmd` field:

name	value	description
BUY	0	buy
SELL	1	sell
BUY_LIMIT	2	buy limit
SELL_LIMIT	3	sell limit
BUY_STOP	4	buy stop
SELL_STOP	5	sell stop
BALANCE	6	Read only. Used in `getTradesHistory` for manager's deposit/withdrawal operations (profit>0 for deposit, profit<0 for withdrawal).
CREDIT	7	Read only

19. Command: getTradingHours

Description: Returns quotes and trading times.

Request:

Parameters:

name	type	description
symbols	array	Array of symbol names (Strings)

Example:

```
{
	"command": "getTradingHours",
	"arguments": {
		"symbols": ["EURPLN", "AGO.PL", ...]
	}
}
```

Response:

Parameters:

name	type	description
	array	Array of `TRADING_HOURS_RECORD`

Example:

```
{
	"status": true,
	"returnData": [TRADING_HOURS_RECORD, TRADING_HOURS_RECORD, ...]
}
```

Format of `TRADING_HOURS_RECORD`:

name	type	description
quotes	array	Array of `QUOTES_RECORD`
symbol	String	Symbol
trading	array	Array of `TRADING_RECORD`

Example:

```
{
	"quotes": [QUOTES_RECORD, QUOTES_RECORD, ...],
	"symbol": "USDPLN",
	"trading": [TRADING_RECORD, TRADING_RECORD, ...]
}
```

Format of `QUOTES_RECORD`:

name	type	description
day	Number	Day of week
fromT	Time	Start time in ms from 00:00 CET / CEST time zone (see Daylight Saving Time, DST)
toT	Time	End time in ms from 00:00 CET / CEST time zone (see Daylight Saving Time, DST)

Example:

```
{
        "day": 2,
        "fromT": 63000000,
        "toT": 63300000
}
```

Possible values of `day` field:

name	value	description
	1	Monday
	2	Tuesday
	3	Wednesday
	4	Thursday
	5	Friday
	6	Saturday
	7	Sunday

Format of `TRADING_RECORD`:

name	type	description
day	Number	Day of week
fromT	Time	Start time in ms from 00:00 CET / CEST time zone (see Daylight Saving Time, DST)
toT	Time	End time in ms from 00:00 CET / CEST time zone (see Daylight Saving Time, DST)

Example:

```
{
	"day": 2,
	"fromT": 63000000,
	"toT": 63300000
}
```

Possible values of `day` field:

name	value	description
	1	Monday
	2	Tuesday
	3	Wednesday
	4	Thursday
	5	Friday
	6	Saturday
	7	Sunday

20. Command: getVersion

Description: Returns the current API version.

Request:

Example:

```
{
	"command": "getVersion"
}
```

Response:

Parameters:

name	type	description
version	String	current API version

Example:

```
{
	"status": true,
	"returnData": {
		"version": "2.4.15"
	}
}
```

21. Command: ping

Description: Regularly calling this function is enough to refresh the internal state of all the components in the system. It is recommended that any application that does not execute other commands, should call this command at least once every 10 minutes. Please note that the streaming counterpart of this function is combination of `ping` and `getKeepAlive`.

Request:

Example:

```
{
	"command": "ping"
}
```

Response:

Example:

```
{
	"status": true
}
```

22. Command: tradeTransaction

Description: Starts trade transaction. tradeTransaction sends main transaction information to the server.

How to verify that the trade request was accepted?

The `status` field set to 'true' **does not** imply that the transaction was accepted. It only means, that the server acquired your request and began to process it. To analyse the status of the transaction (for example to verify if it was accepted or rejected) use the `tradeTransactionStatus` command with the `order` number, that came back with the response of the `tradeTransaction` command. You can find the example here: developers.xstore.pro/api/tutorials/opening_and_closing_trades2 (http://developers.xstore.pro/api/tutorials/opening_and_closing_trades2)

Request:

Parameters:

name	type	description
tradeTransInfo	TRADE_TRANS_INFO	tradeTransInfo

Example:

```
{
	"command": "tradeTransaction",
	"arguments": {
		"tradeTransInfo": TRADE_TRANS_INFO
	}
}
```

Format of `TRADE_TRANS_INFO`:

name	type	description
cmd	Number	Operation code
customComment	String	The value the customer may provide in order to retrieve it later.
expiration	Time	Pending order expiration time
offset	Number	Trailing offset
order	Number	0 or position number for closing/modifications
price	Floating number	Trade price
sl	Floating number	Stop loss
symbol	String	Trade symbol
tp	Floating number	Take profit
type	Number	Trade transaction type
volume	Floating number	Trade volume

Example:

```
{
	"cmd": 2,
	"customComment": "Some text",
	"expiration": 1462006335000,
	"offset": 0,
	"order": 82188055,
	"price": 1.12,
	"sl": 0.0,
	"symbol": "EURUSD",
	"tp": 0.0,
	"type": 0,
	"volume": 5.0
}
```

Possible values of `cmd` field:

name	value	description
BUY	0	buy
SELL	1	sell
BUY_LIMIT	2	buy limit

name	value	description
SELL_LIMIT	3	sell limit
BUY_STOP	4	buy stop
SELL_STOP	5	sell stop
BALANCE	6	Read only. Used in `getTradesHistory` for manager's deposit/withdrawal operations (profit>0 for deposit, profit<0 for withdrawal).
CREDIT	7	Read only

Possible values of `type` field:

name	value	description
OPEN	0	order open, used for opening orders
PENDING	1	order pending, only used in the streaming `getTrades` command
CLOSE	2	order close
MODIFY	3	order modify, only used in the `tradeTransaction` command
DELETE	4	order delete, only used in the `tradeTransaction` command

Response:

Parameters:

name	type	description
order	Number	order

Example:

```
{
        "status": true,
        "returnData": {
                "order": 43
        }
}
```

23. Command: tradeTransactionStatus

Description: **Please note that this function can be usually replaced by its streaming equivalent `getTradeStatus` which is the preferred way of retrieving transaction status data.** Returns current transaction status. At any time of transaction processing client might check the status of transaction on server side. In order to do that client must provide unique order taken from `tradeTransaction` invocation.

Request:

Parameters:

name	type	description
order	Number	order

Example:

```
{
	"command": "tradeTransactionStatus",
	"arguments": {
		"order": 43
	}
}
```

Response:

Parameters:

name	type	description
ask	Floating number	Price in base currency
bid	Floating number	Price in base currency
customComment	String	The value the customer may provide in order to retrieve it later.
message	String	Can be null
order	Number	Unique order number
requestStatus	Number	Request status code, described below

Example:

```
{
	"status": true,
	"returnData": {
		"ask": 1.392,
		"bid": 1.392,
		"customComment": "Some text",
		"message": null,
		"order": 43,
		"requestStatus": 3
	}
}
```

Possible values of `requestStatus` field:

name	value	description

name	value	description
ERROR	0	error
PENDING	1	pending
ACCEPTED	3	The transaction has been executed successfully
REJECTED	4	The transaction has been rejected

6. Available streaming commands

Each streaming command takes as an argument `streamSessionId` which is sent in response message for login command performed in main connection. `streamSessionId` token allows to identify user in streaming connection. In one streaming connection multiple commands with different `streamSessionId` can be invoked. It will cause sending streaming data for multiple login sessions in one streaming connection. `streamSessionId` is valid until logout command is performed on main connection or main connection is disconnected.

1. Command: getBalance

Description: Allows to get actual account indicators values in real-time, as soon as they are available in the system.

Subscribe format:

Example:

```
{
        "command": "getBalance",
        "streamSessionId": "8469308861804289383"
}
```

Unsubscribe format:

Example:

```
{
        "command": "stopBalance"
}
```

Format of data in stream:

```
{
        "command": "balance",
        "data": STREAMING_BALANCE_RECORD
}
```

Format of `STREAMING_BALANCE_RECORD`:

name	type	description
balance	Floating number	balance in account currency
credit	Floating number	credit in account currency

name	type	description
equity	Floating number	sum of balance and all profits in account currency
margin	Floating number	margin requirements
marginFree	Floating number	free margin
marginLevel	Floating number	margin level percentage

Example:

```
{
	"balance": 995800269.43,
	"credit": 1000.00,
	"equity": 995985397.56,
	"margin": 572634.43,
	"marginFree": 995227635.00,
	"marginLevel": 173930.41
}
```

2. Command: getCandles

Description: Subscribes for and unsubscribes from API chart candles. The interval of every candle is 1 minute. A new candle arrives every minute.

Subscribe format:

Parameters:

name	type	description
symbol	String	Symbol

Example:

```
{
	"command": "getCandles",
	"streamSessionId": "8469308861804289383",
	"symbol": "EURUSD"
}
```

Unsubscribe format:

Parameters:

name	type	description
symbol	String	Symbol

Example:

```
{
	"command": "stopCandles",
	"symbol": "EURUSD"
}
```

Format of data in stream:

```
{
	"command": "candle",
	"data": STREAMING_CANDLE_RECORD
}
```

Format of `STREAMING_CANDLE_RECORD`:

name	type	description
close	Floating number	Close price in base currency
ctm	Time	Candle start time in CET time zone (Central European Time)
ctmString	String	String representation of the `ctm` field
high	Floating number	Highest value in the given period in base currency
low	Floating number	Lowest value in the given period in base currency
open	Floating number	Open price in base currency
quoteId	Number	Source of price
symbol	String	Symbol
vol	Floating number	Volume in lots

Example:

```
{
	"close": 4.1849,
	"ctm": 1378369375000,
	"ctmString": "Sep 05, 2013 10:22:55 AM",
	"high": 4.1854,
	"low": 4.1848,
	"open": 4.1848,
	"quoteId": 2,
	"symbol": "EURUSD",
	"vol": 0.0
}
```

Possible values of `quoteId` field:

name	value	description
fixed	1	fixed

name	value	description
float	2	float
depth	3	depth
cross	4	cross

3. Command: getKeepAlive

Description: Subscribes for and unsubscribes from 'keep alive' messages. A new 'keep alive' message is sent by the API every 3 seconds.

Subscribe format:

Example:

```
{
        "command": "getKeepAlive",
        "streamSessionId": "8469308861804289383"
}
```

Unsubscribe format:

Example:

```
{
        "command": "stopKeepAlive"
}
```

Format of data in stream:

```
{
        "command": "keepAlive",
        "data": STREAMING_KEEP_ALIVE_RECORD
}
```

Format of `STREAMING_KEEP_ALIVE_RECORD`:

name	type	description
timestamp	Time	Current timestamp

Example:

```
{
        "timestamp": 1362944112000
}
```

4. Command: getNews

Description: Subscribes for and unsubscribes from news.

Subscribe format:

Example:

```
{
	"command": "getNews",
	"streamSessionId": "8469308861804289383"
}
```

Unsubscribe format:

Example:

```
{
	"command": "stopNews"
}
```

Format of data in stream:

```
{
	"command": "news",
	"data": STREAMING_NEWS_RECORD
}
```

Format of `STREAMING_NEWS_RECORD`:

name	**type**	**description**
body	String	Body
key	String	News key
time	Time	Time
title	String	News title

Example:

```
{
	"body": "<html>...</html>",
	"key": "1f6da766abd29927aa854823f0105c23",
	"time": 1262944112000,
	"title": "Breaking trend"
}
```

5. Command: getProfits

Description: Subscribes for and unsubscribes from profits.

Subscribe format:

Example:

```
{
	"command": "getProfits",
	"streamSessionId": "8469308861804289383"
}
```

Unsubscribe format:

Example:

```
{
        "command": "stopProfits"
}
```

Format of data in stream:

```
{
        "command": "profit",
        "data": STREAMING_PROFIT_RECORD
}
```

Format of `STREAMING_PROFIT_RECORD`:

name	type	description
order	Number	Order number
order2	Number	Transaction ID
position	Number	Position number
profit	Floating number	Profit in account currency

Example:

```
{
        "order": 7497776,
        "order2": 7497777,
        "position": 7497776,
        "profit": 7076.52
}
```

6. Command: getTickPrices

Description: Establishes subscription for quotations and allows to obtain the relevant information in real-time, as soon as it is available in the system. The `getTickPrices` command can be invoked many times for the same symbol, but only one subscription for a given symbol will be created. Please beware that when multiple records are available, the order in which they are received is not guaranteed.

Subscribe format:

Parameters:

name	type	description
symbol	String	Symbol
minArrivalTime	Number	This field is optional and defines the minimal interval in milliseconds between any two consecutive updates. If this field is not present, or it is set to 0 (zero), ticks - if available - are sent to the client with interval equal to 200 milliseconds. In order to obtain ticks as frequently as server allows you, set it to 1 (one).

name	type	description
maxLevel	Number	This field is optional and specifies the maximum level of the quote that the user is interested in. If this field is not specified, the subscription is active for all levels that are managed in the system.

Example:

```
{
        "command": "getTickPrices",
        "streamSessionId": "8469308861804289383",
        "symbol": "EURUSD",
        "minArrivalTime": 5000,
        "maxLevel": 2
}
```

Unsubscribe format:

Parameters:

name	type	description
symbol	String	Symbol

Example:

```
{
        "command": "stopTickPrices",
        "symbol": "EURUSD"
}
```

Format of data in stream:

```
{
        "command": "tickPrices",
        "data": STREAMING_TICK_RECORD
}
```

Format of `STREAMING_TICK_RECORD`:

name	type	description
ask	Floating number	Ask price in base currency
askVolume	Number	Number of available lots to buy at given price or null if not applicable
bid	Floating number	Bid price in base currency
bidVolume	Number	Number of available lots to buy at given price or null if not applicable

name	type	description
high	Floating number	The highest price of the day in base currency
level	Number	Price level
low	Floating number	The lowest price of the day in base currency
quoteId	Number	Source of price, detailed description below
spreadRaw	Floating number	The difference between raw ask and bid prices
spreadTable	Floating number	Spread representation
symbol	String	Symbol
timestamp	Time	Timestamp

Example:

```
{
	"ask": 4000.0,
	"askVolume": 15000,
	"bid": 4000.0,
	"bidVolume": 16000,
	"high": 4000.0,
	"level": 0,
	"low": 3500.0,
	"quoteId": 0,
	"spreadRaw": 0.000003,
	"spreadTable": 0.00042,
	"symbol": "KOMB.CZ",
	"timestamp": 1272529161605
}
```

Possible values of `quoteId` field:

name	value	description
fixed	1	fixed
float	2	float
depth	3	depth
cross	4	cross

7. Command: getTrades

Description: Establishes subscription for user trade status data and allows to obtain the relevant information in real-time, as soon as it is available in the system. Please beware that when multiple records are available, the order in which they are received is not guaranteed.

Subscribe format:

Example:

```
{
        "command": "getTrades",
        "streamSessionId": "8469308861804289383"
}
```

Unsubscribe format:

Example:

```
{
        "command": "stopTrades"
}
```

Format of data in stream:

```
{
        "command": "trade",
        "data": STREAMING_TRADE_RECORD
}
```

Format of `STREAMING_TRADE_RECORD`:

New `STREAMING_TRADE_RECORD` are sent by streaming socket only in several cases:

- - Opening the trade
- - Closing the trade
- - Modification of trade parameters
- - Explicit trade update done by server system to synchronize data.

Situation that trade was closed can be checked by field `closed` set to true in `STREAMING_TRADE_RECORD` format. Also `close_time` field will NOT be set to null. Various reasons of trade close could be found out by information in `comment` field of `STREAMING_TRADE_RECORD` for closed order. If the `comment` remained unchanged from that of opened order, then the order was closed by user. If there is annotation in `comment` string like:

- - "[S/L]", then the trade was closed by stop loss
- - "[T/P]", then the trade was closed by take profit
- - "[S/O margin level% equity / margin (currency)]", then the trade was closed because of Stop Out (lack of money to maintain position). The example comment: [S/O -1968861.79% -24217.00 / 1.23 (USD)]

The annotation are in brackets (regular or square, depending on situation) with additional data about the closing action.

name	type	description
close_price	Floating number	Close price in base currency

name	type	description
close_time	Time	Null if order is not closed
closed	Boolean	Closed
cmd	Number	Operation code
comment	String	Comment
commission	Floating number	Commission in account currency, null if not applicable
customComment	String	The value the customer may provide in order to retrieve it later.
digits	Number	Number of decimal places
expiration	Time	Null if order is not closed
margin_rate	Floating number	Margin rate
offset	Number	Trailing offset
open_price	Floating number	Open price in base currency
open_time	Time	Open time
order	Number	Order number for opened transaction
order2	Number	Transaction id
position	Number	Position number (if type is 0 and 2) or transaction parameter (if type is 1)
profit	Floating number	null unless the trade is closed (type=2) or opened (type=0)
sl	Floating number	Zero if stop loss is not set (in base currency)
state	String	Trade state, should be used for detecting pending order's cancellation
storage	Floating number	Storage
symbol	String	Symbol
tp	Floating number	Zero if take profit is not set (in base currency)
type	Number	type

name	type	description
volume	Floating number	Volume in lots

Example:

```
{
        "close_price": 1.3256,
        "close_time": null,
        "closed": false,
        "cmd": 0,
        "comment": "Web Trader",
        "commission": 0.0,
        "customComment": "Some text",
        "digits": 4,
        "expiration": null,
        "margin_rate": 3.9149000,
        "offset": 0,
        "open_price": 1.4,
        "open_time": 1272380927000,
        "order": 7497776,
        "order2": 1234567,
        "position": 1234567,
        "profit": 68.392,
        "sl": 0.0,
        "state": "Modified",
        "storage": -4.46,
        "symbol": "EURUSD",
        "tp": 0.0,
        "type": 0,
        "volume": 0.10
}
```

Possible values of `cmd` field:

name	value	description
BUY	0	buy
SELL	1	sell
BUY_LIMIT	2	buy limit
SELL_LIMIT	3	sell limit
BUY_STOP	4	buy stop
SELL_STOP	5	sell stop
BALANCE	6	Read only. Used in `getTradesHistory` for manager's deposit/withdrawal operations (profit>0 for deposit, profit<0 for withdrawal).

name	value	description
CREDIT	7	Read only

Possible values of `state` field:

name	value	description
MODIFIED	"Modified"	modified
DELETED	"Deleted"	deleted

Possible values of `type` field:

name	value	description
OPEN	0	order open, used for opening orders
PENDING	1	order pending, only used in the streaming `getTrades` command
CLOSE	2	order close
MODIFY	3	order modify, only used in the `tradeTransaction` command
DELETE	4	order delete, only used in the `tradeTransaction` command

8. Command: getTradeStatus

Description: Allows to get status for sent trade requests in real-time, as soon as it is available in the system. Please beware that when multiple records are available, the order in which they are received is not guaranteed.

Subscribe format:

Example:

```
{
	"command": "getTradeStatus",
	"streamSessionId": "8469308861804289383"
}
```

Unsubscribe format:

Example:

```
{
	"command": "stopTradeStatus"
}
```

Format of data in stream:

```
{
	"command": "tradeStatus",
	"data": STREAMING_TRADE_STATUS_RECORD
}
```

Format of `STREAMING_TRADE_STATUS_RECORD`:

name	type	description
customComment	String	The value the customer may provide in order to retrieve it later.
message	String	Can be null
order	Number	Unique order number
price	Floating number	Price in base currency
requestStatus	Number	Request status code, described below

Example:

```
{
	"customComment": "Some text",
	"message": null,
	"order": 43,
	"price": 1.392,
	"requestStatus": 3
}
```

Possible values of `requestStatus` field:

name	value	description
ERROR	0	error
PENDING	1	pending
ACCEPTED	3	The transaction has been executed successfully
REJECTED	4	The transaction has been rejected

9. Command: ping

Description: Description: Regularly calling this function is enough to refresh the internal state of all the components in the system. Streaming connection, when any command is not sent by client in the session, generates only one way network traffic. It is recommended that any application that does not execute other commands, should call this command at least once every 10 minutes.

Note: There is no response in return to this command.

Subscribe format:

Example:

```
{
	"command": "ping",
	"streamSessionId": "8469308861804289383"
}
```

7. Error messages

Main reason of generated transaction server error messages is an error in the business logic of the application.

Errors list returned from transaction server:

Error code	Error description
BE001	Invalid price
BE002	Invalid StopLoss or TakeProfit
BE003	Invalid volume
BE004	Login disabled
BE005	userPasswordCheck: Invalid login or password.
BE006	Market for instrument is closed
BE007	Mismatched parameters
BE008	Modification is denied
BE009	Not enough money on account to perform trade
BE010	Off quotes
BE011	Opposite positions prohibited
BE012	Short positions prohibited
BE013	Price has changed
BE014	Request too frequent
BE016, BE017	Too many trade requests
BE018	Trading on instrument disabled
BE019	Trading timeout
BE020-BE037, BE099	Other error
BE094	Symbol does not exist for given account
BE095	Account cannot trade on given symbol

Error code	Error description
BE096	Pending order cannot be closed. Pending order must be deleted
BE097	Cannot close already closed order
BE098	No such transaction
BE101	Unknown instrument symbol
BE102	Unknown transaction type
BE103	User is not logged
BE104	Method does not exist
BE105	Incorrect period given
BE106	Missing data
BE110	Incorrect command format
BE115, BE116	Symbol does not exist
BE117	Invalid token
BE118	User already logged
BE200	Session timed out.
EX000	Invalid parameters
EX001, EX002, SExxx, BE000	Internal error, in case of such error, please contact support
EX003	Internal error, request timed out
EX004	Login credentials are incorrect or this login is not allowed to use an application with this appId
EX005	Internal error, system overloaded
EX006	No access
EX007	userPasswordCheck: Invalid login or password. This login/password is disabled for 10 minutes (the specific login and password pair is blocked after an unsuccessful login attempt).
EX008	You have reached the connection limit. For details see the Connection validation section.
EX009	Data limit potentially exceeded. Please narrow your request range. The potential data size is calculated by: (end_time - start_time) / interval. The limit is 50 000 candles

Error code	Error description
EX010	Your login is on the black list, perhaps due to previous misuse. For details please contact support.
EX011	You are not allowed to execute this command. For details please contact support.