LMAX PROTOCOL REFERENCE GUIDE

Effective Date: 12 September 2011

Version: 1.8

LMAX TRADER

Table of contents

Revision History

Version	Date	Changes
1.0 - Draft	10/09/2010	Initial Draft
1.1	08/11/2010	Version 1.1 – Beta of LMAX Trader API for Beta API program.
1.1	30/11/2010	**Corrected version of Version 1.1 – Beta** of LMAX Trader API for Beta API program. **Section 2.1** - **Added** the URL link to the test environment. **Sections 1.8, 2.3.1, 2.3.6, 2.4.1, 2.6.1** – **Corrected** Request example to include <body> element **Section 2.7.1** – **Corrected** <to> and <from> elements to be small script **Section 2.7.2.5** – **Corrected** event example to contain <type> element **Section 2.3.3** – **Corrected** Description field for the longPollKey element.
1.2	15/12/2010	**Section 1.7 – Changed** events parent <batch> element to <events>; **Section 1.7 – Changed** the structure of <PushEventType> element referenced in the <events> element; **Section 1.7 – Updated** events description to say: *"No events will be resent if the connection was lost and re-established. In case of disconnection the sequence number will be reset to 1. "The batch body contains one or more events occurred between the last batch sent and the current batch. The element name identifies the nature of the event (for example orderCancelled, orderBookStatus, etc). Only events that user has subscribed to will be listed*"; **Section 1.7 – Updated** <events> example to comply with the current structure of the schema; **Section 2.3.1 – Added** product type parameter for the test environment login; **Section 2.4.1 – Removed** <side> element from the schema; **Section 2.4.1 – Added** text to the Supported Values column of the <quantity> element; **Section 2.4.1 – Updated** description to say: **"***When Execution occurs in the event of matching – event of type orderState is generated following the structure of the orderState message with the execution element ;When Execution occurs in the event of order cancelation – event of type orderState is generated following the structure of the orderState message with the orderCancelled element* " **Section 2.4.1 –Updated Place market Sell Order request example** to comply with the current structure of the schema; **Section 2.4.1.1 –Updated GoodUntill Cancelled Limit Buy Order request Example** to comply with the current structure of the schema; **Section 2.6.1 – Added** `orderBookStatus` subscription **Section 2.7.1–Removed <**sequenceNumber> element from the exchangeRate event message schema; **Section 2.7.1–Updated Message Example** to comply with the current structure of the exchangeRate event message schema; **Section 2.7.2.1–Removed <**sequenceNumber> element from the accountState event message schema; **Section 2.7.2.1–Updated Message Example** to comply with the current structure of the accountState event schema; **Section 2.7.2.2–Replaced** parent **<**orders> element from the orderState event message schema with **<**order> element; **Section 2.7.2.2–Removed <**side>, <stopLossCost>, <stopBuffer> , < closedCost>, < closedQuantity >, < realisedProfit >, <volumeWeightedAvgPrice> elements from the orderState event message schema;

		Section 2.7.2.2–Added < cumulativeCost> element to the orderState event message schema; **Section 2.7.2.2–renamed** <matched>, <openedCost>, <openedQuantity>, <cancel>, element to <matchedQuantity<, <openCost >, < openQuantity>, <cancelledQuantity >in the orderState event message schema; **Section 2.7.2.2–Updated Message Example** to comply with the current structure of the orderState event schema; **Section 2.7.2.3–Replaced** parent **<**positions> element from the position event message schema with **<**position> element; **Section 2.7.2.3–Removed <**sequenceNumber> element from the position event message schema; **Section 2.7.1.3–Updated Message Example** to comply with the current structure of the position event message schema; **Section 2.7.1.4–Updated Message Example** to comply with the current structure of the instructionRejected event schema; **Section 2.7.1.5–Updated Message Example** to comply with the current structure of the heartbeat event message schema; **Section 2.8.1–Updated Message Example** to comply with the current structure of the orderBookStatus event message schema; **Section 2.8.2–Updated Message Example** to comply with the current structure of the orderBook event message schema; **Section 3.3–Removed** UNKNOWN code from the list of order rejection reasons **Section 4 – Added** sections 4.1-4.31 (API Schemas)
1.2	07/01/2011	**Corrected** Effective Date to 19 January 2011
1.3	26/02/2011	**No Structural Updates**
1.4	08/03/2011	**Section 1 – Changed** the version to 1.4 **Section 2.3.1 – Added** “protocolVersion” element to the message structure and example **Section 2.4.1 – Added** Stop Market to supported order types **Section 2.5 – Added** Place Stop Market Order Response/Request examples **Section 2.8.2 – Added** exchangeTimestamp definition and Updated Message Example for same **Section 2.9.2.2 – Added** “Signed” to “quantity” element description **Section 2.9.2.2 – Added** an example of “openQuantity” calculation **Section 2.9.2.2 – Corrected** example from </openedCost> to </openCost> **Section 2.9.2.2 – Corrected** example from </openedQuantity> to <openQuantity> **Section 2.9.2.5 – Corrected** “accountposition” to read “position” **Section 2.9.2.5 – Removed** <page> element from the Example **Section 3.2– Added** “PROTOCOL_VERSION_INVALID” error message **Section 4.1 – Added** “protocolVersion” element to the schema **Section 4.24 –Added** < exchangeTimestamp > element to the orderBook-event message schema (API Schemas); **Section 4.26 – Updated** “PasswordType” from 6 characters to 8 **Section 4.9 – Added** PlaceOrderPayload element to the schema **Section 4.10 – Added** “placeStop-request” schema **Section 4.11 – Added** “placeStop-response”schema **Section 4.27 – Changed** “public-datatypes” to published-datatypes **Section 4.27 – Added** “PlaceOrderPayload” element to schema **Section 4.27 – Added** “PlaceStopPayload“ element to schema **Section 4.27 – Added** “PasswordType“ element to schema **Section 4.27 – Added** “LoginAttemptPasswordType“ element to schema **Section 4.27 – Added** “CurrencyType“ element to schema **Section 4.27 – Added** “LocaleType“ element to schema

		Section 4.27 – Added "CountryType" element to schema **Section 4.27 – Added** "UsernameType" element to schema **Section 4.27 – Added** "QuantityType" element to schema **Section 4.27 – Added** "ExchangeRateReferenceType" element to schema **Section 4.27 – Added** "AccountIdType" element to schema **Section 4.27 – Added** "PriceType" element to schema **Section 4.27 – Added** "LongPollKeyType" element to schema **Section 4.27 – Added** "InstrumentIdType" element to schema **Section 4.27 – Added** "PricePointType" element to schema **Section 4.27 – Added** "ExecutionIdType" element to schema **Section 4.27 – Added** "OrderIdType" element to schema **Section 4.27 – Added** "AccountStateType" element to schema **Section 4.27 – Added** "InstructionIdType" element to schema **Section 4.27 – Added** "MarketClosePriceType" element to schema Section 4.27 – **Added** "orderBook2" element to schema **Section 4.28 – Removed** "tfx-datatypes" schema **Section 4.32 – Added** "PROTOCOL_VERSION_INVALID" error message to "LoginFailureEnum" **Section 4.32 – Added** "STOP_MARKET_ORDER" and "STOP_MARKET_OPENING_ORDER" to OrderTypeEnum
1.5	24/03/2011	**Updated Effective Date to 04/04/2011** **Corrected the title** of this document from "LMAX Trader API Reference Guide" to "LMAX Protocol Reference Guide" **Changed** all the refererences to API to Protocol **Section 2 – Changed** the name of the section from "API Service Reference" to "Protocol Services Reference" **Section 2.3.1 – Changed "**`protocolVersion"`to 1.5 **Section 2.3.1 – Added "**`minProtocolVersion"` and `"maxProtocolVersion"` to the Response message definition **Section 4 – Changed** the name of the section from "API Schemas" to "Protocol Schemas" **Section 4.2 – Removed** reference to "public-datatypes.rng" **Section 4.2** – **Added** reference to "`published-datatypes.rng"` **Section 4.2 - Added "**`minProtocolVersion"` and `"maxProtocolVersion"` elements to the Response message definition **Section 4.20 - Removed** reference to "`public-datatypes.rng`" **Section 4.20** – **Added** reference to "`published-datatypes.rng"` **Section 4.23 – Added** reference to "`errors.rng"` **Section 4.23 – Removed** `"InstructionRejectionEnum"` **Section 4.24 - Removed** reference to "`public-datatypes.rng`" **Section 4.24 – Added** reference to "`published-datatypes.rng"` **`Section 4.25`** **- Removed** reference to "`public-datatypes.rng`" **Section 4.25 – Added** reference to "`published-datatypes.rng"` **Section 4.27 - Removed** reference to "`public-datatypes.rng`" **Section 4.27 – Added** reference to "`published-datatypes.rng"` **Section 4.27 – Removed** `"pattern"` parameter from `"LoginAttemptPasswordType"` datatype **Section 4.27 – Changed** "`minInclusive`" element to "`minExclusive`" in "`StopOffsetType`" **Section 4.28 – Added** "`errors`" schema **Section 4.30 – Added** reference to "`errors.rng"` **Section 4.30 – Added** references to "`InstructionRejectionEnum`" and "`WarningEnum`" in "`message`" element of the "`WarningsType`"

1.6	11/05/2011	**Updated Effective Date to 15 May 2011** **Updated the version of the document to 1.6**
1.7	22/06/2011	**Section 1.7 Removed** <orderCancelled> in <accountState> **Section 1.7 Changed** <position> tin <accountState> to <positions> **Section 2.3.1 Changed "**`protocolVersion`**"**to 1.7 **Section 2.4.2 Added** text and element description to describe optional <instructionId> field **Section 2.4.2 Added** <instructionId> to Place Market Sell Order with supplied InstructionID Request and Response **Section 2.4.2 Added** GTC order Request & Response example without supplied <instructionId> **Section 2.5 Added** <instructionId> element to Place Stop Market Order **Section 2.6 Added** text and element description to Cancel Order **Section 2.8 Added** type Order and type Position to Subscribe Request Push Events **Section 2.9.2.1 Added** <unrealizedProfitAndLoss> element to the accountState event **Section 2.9.2.1 Added** <margin> element to the accountState event **Section 2.9.2.2 Added** <orders> element to orderState event **Section 2.9.2.2 Removed** <page> element **Section 2.9.2.2 Changed** definition of <order> element **Section 2.9.2.2 Added** <order element to <orders> in message example **Section 2.9.2.5 Added** <positions> element to bracket the <position> element **Section 2.9.2.5 Added** <position> element to <positions> element in message example **Section 2.10.3. Added** orderBook2 event to message example **Section 2.10.4. Added** Historical Market Data event and message example **Section 4.9 Added** optional <instructionId> element to PlaceOrder-request schema **Section 4.10 Added** optional <instructionId> element to PlaceStop-request schema **Section 4.13 Added** optional <instructionId> element to cancelOrder-request schema **Section 4.21 Removed** <order> element from schema **Section 4.21 Added** <orders> element to schema **Section 4.21 Modified** <instructionId> element placement in schema **Section 4.22 Added** <positions> element to position-event schema **Section 4.22 Removed** <page> element from position-event schema **Section 4.27 Added** <unrealizedProfitAndLoss> element to published-datatypes schema **Section 4.27 Added** <margin> element to published-datatypes schema **Section 4.27 Added** <"SuppliedInstructionIdType"> to published-datatypes schema **Section 4.27 Changed** <"PositiveNonZeroLongType"> element to <"long"> element in published-datatypes schema **Section 4.32 Added** <value>order</value> and <value>position</value> to <"SubscriptionType"> in public-enumerations schema **Section 4.32 Added** <value>NETHERLANDS</value> to <RegistrationLegalEntityType>
1.7	22/07/2011	**Section 2.10.4.1 Added** Historical Market Data Request and Response Examples **Section 2.3.4 Added** better explanation to POST longPoll
1.7	22/08/2011	**Section 2.4.1 Added Order Types** TimeInForceEnum to Public

		Enumerations with notes on Deprecation of GoodUntilEnum **Section 4.32 Added** TimeInForceEnum Element to Public Enumerations
1.8	09/09/2011	**Changed Version Number from 1.7 to 1.8** **Section 2.10.1 Removed** orderBook as it has been deprecated in favour of orderBook2 **Section 2.10.2 DEPRECATING** note on **historic market data** **Section 2.4.2 Removed** GoodUntil and AllowUnmatched references and replaced with TimeInForce **Section 2.10.2 DEPRECATING** orderBook and replaced by orderBook2 **Section 4.26 Removed** orderBook schema due to deprecation and replaced by orderBook2 **Section 4.32 Removed** <GoodUntil> from the Public Enumerations schema replaced by <TimeInForce>

LMAX Protocol Reference Guide

1. OVERVIEW

This document outlines the services that are available via the LMAX Protocol 1.5 This Protocol is an easy-to-use, standards-based, programming-language-independent interface to Web Services that enables access to the functionality of LMAX Trader by programs and allows a program to perform transactions on the LMAX Multi-lateral Trading Facility (MTF).

Please note that currently LMAX Protocol is subject to enhan cements and changes. This document provides a description of the current Protocol functionality and is likely to be changed/enhanced to reflect any changes applied to the Protocol. Any changes or additional functionality due to be applied to the Protocol will be published on LMAX Broker website in advance. LMAX does not guarantee backward compatibility.

1.1. INTERACTION MODEL

The LMAX Protocol supports asynchronous event push behaviour where your application is notified of relevant events as they occur.

If a user would like to get updates for the particular events, he has to make a subscription request for event updates. A request will result in a response, and may be followed by one or more events which occur asynchronously as a result of the original request, for example an order being placed may be followed by a number of order state events as the unmatched order becomes matched.

A specific set of subscription requests is also provided to allow you to register interest in a number of topics, such as market prices, which may generate events without any activity on the user's part. Subscriptions remain enabled for the session or until explicitly disabled or session logout.

A typical client dialogue will progress through the following steps:

- Login
- Request a subscription: market data information, order book information, account state, executions
- Interact (e.g. place , cancel orders)
- Receive and respond to the events
- Logout

1.2. COMMUNICATION MECHANICS

The programming interface to the LMAX Protocol is implemented as a collection of messages sent to a secure web server over HTTP. Messages can be sent in JSON and XML formats.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all gratis open standards.

JSON (JavaScript Object Notation) is a lightweight data interchange format. JSON is easy to read and write; you can parse it using any programming language, and its structures map directly to data structures used in most programming languages.

The example mapping between XML and JSON is provided in this chapter, however all the further message examples in this document are given in XML format.

The LMAX Trader exposes a collection of operations that may be invoked upon it, and publishes a collection of events that will be sent to clients to keep them informed of changes in their position and in the state of their account.

The Protocol is designed to be minimal, and to present concise and timely information in a clear, and easily interpretable form.

The MTF is asynchronous in nature and so many of the calls do not return the data or result requested. The response will come as an event on the asynchronous channel.

The asynchronous channel can work in one of two ways.

- Long Poll: suitable for JavaScript in browsers drawing down a batch of events as soon as they are available.
- Streaming: uses chunked transfer encoding to send up batches of events to the client as soon as available.

1.3. REQUEST HEADER PARAMETERS

The content type of the message sent or requested must be set in the Content-Type HTTP Header.

- XML: Content-Type: text/xml - sending, and Accept: text/xml - accepting
- JSON: Content-Type: application/json – sending, Accept: application/json - accepting

1.4. MESSAGE STRUCTURE

Messages take one of three forms. Each message is one of a Request (<req>), a Response (<res>), or an Event (<event>).

Requests are instructions sent to the MTF, and are intended to invoke some behaviour within it. Responses are messages sent in response to specific Requests. Every Request will have a matching Response.

Events are messages sent directly to a client from the LMAX Trader. These messages will be sent in response to some internal event within the MTF rather than directly in response to any specific request. For example if a match occurs for your account within the MTF, due to some other account holder placing orders, you will be informed of the change to your position asynchronously. If the status of an instrument in which you have registered an interest changes, you will be informed.
To maximise throughput, events are pushed back to the client in batches. Each batch may contain multiple events of more than one type.

1.5. MAKING REQUESTS

The activity of the LMAX Trader that may be invoked by the clients is represented as a separate URI which ends in a name that represents the action being invoked.

```
e.g.        https://.... /public/security/login
```

Messages are based on a simple structure; requests are identified by a <req> block containing a <body> which in-turn contains any relevant parameters for the message. This means that an empty message will take the following form:

```
<req>
    <body/>
</req>
```

1.6. MESSAGE RESPONSES

Message responses also have a common simple structure. They consist of a <res> block containing a <header> and a <body>.

The <header> will contain the status of the message, whatever the nature of the request that invoked this Response. This will be one of ‘OK’, ‘WARN’ or ‘ERROR’.

The <body> will contain data pertinent to the Requesting message, if there is any. If not, an empty <body> will be returned. This means that a minimal response message will take the form:

```
<res>
    <header>
          <status>OK</status>
    </header>
    <body/>
</res>
```

Example Response: Login (successful)

```
<res>
    <header>
          <status>OK</status>
    </header>
    <body>
          <username>user9001</username>
          <currency>GBP</currency>
          <accountId>9001</accountId>
          <AccountType>STANDARD_TRADER</AccountType>
          <productType>CFD_LIVE</productType>
          <fundingDisallowed>False</fundingDisallowed>
    </body>
</res>
```

1.7. EVENT BATCHES

Each event batch has the same overall structure:

```
<events>
    <header>
          <seq/>
    </header>
    </body>
</events>
```

The events batch header contains a contiguous sequence number which is incremented for every event batch sent from LMAX to your application during a single session. Any gap in the sequence indicates a missed message, and it is your responsibility to reinitiate any local state and subscriptions if this occurs. No events will be resent if the connection was lost and then re-established. In case of disconnection the sequence number will be reset to 0.

The batch body contains one or more events occurred between the last batch sent and the current batch. The element name identifies the nature of the event (for example orderCancelled, orderBookStatus, etc). Only events that user has subscribed to will be listed.

Each event body within the events batch contains the full details of the event, which are documented in the following sections. An example of a complete events batch with number of events listed is shown below.

```
<events>
   <header>
      <seq>3</seq>
   </header>
   <body>
        <accountState>
            <accountId>10000</accountId>
            <balance>1000.00000</balance>
            <exposure>-50.00000</exposure>
            <availableFunds>950.00000</availableFunds>
            <availableToWithdraw>950.00000</availableToWithdraw>
            <wallets>
                <wallet>
                    <currency>USD</currency>
                    <balance> 1842080.1</balance>
                </wallet>
                <wallet>
                    <currency>JPY</currency>
                    <balance> 1042080.0</balance>
                </wallet>
                <wallet>
                    <currency>GBP</currency>
                    <balance> 142080.5</balance>
                </wallet>
            </wallets>
        </accountState>

        <orderBookStatus>
           <id>4001</id>
           <status>Opened</status>
        </orderBookStatus>

        <orderBookStatus>
           <id>4008</id>
           <status>Opened</status>
        </orderBookStatus>



        <positions>
                <position>
```

```
                <accountId>100000</accountId>
                <instrumentId>4008</instrumentId>
                <valuation>-30366480</valuation>
                <shortUnfilledCost>0</shortUnfilledCost>
                <longUnfilledCost>0</longUnfilledCost>
                <openQuantity>-603</openQuantity>
                <cumulativeCost>-340039750</cumulativeCost>
                <openCost>-465101250</openCost>
            </position>
            <position>
                <accountId>100000</accountId>
                <instrumentId>4001</instrumentId>
                <valuation>-228558</valuation>
                <shortUnfilledCost>0</shortUnfilledCost>
                <longUnfilledCost>0</longUnfilledCost>
                <openQuantity>-105</openQuantity>
                <cumulativeCost>-1259650</cumulativeCost>
                <openCost>-1259650</openCost>
            </position>
        <hasMoreResults>false</hasMoreResults>
        <correlationId>0-0</correlationId>
    </positions>

  </body>
</events>
```

1.8. MAPPING BETWEEN JSON AND XML CONTENT

The JSON message structure will have exactly the same content as the XML messages described above. Below are the examples of the request and response message formatted using JSON.

Request Example

```
{
      "req":[
            {
            "subscription":[
                  {
                        "orderBook":"100101"
                  }
            ],
            "subscription":[
                  {
                        "exchangeRate":[
                              {
                                    "from":"EUR",
                                    "to":"GBP"
                              }
                        ]
                  }
            ],
            "longPollKey":"1"
            }
      ]
```

```
}
```

Response Example

```
{
"res":      [
      {
            "$":"",
            "header":[
                  {
                        "$":"",
                        "status":[
                              "OK"
                        ]
                  }
            ],
            "body":[
                  ""
            ]
      }
]
}
```

Examples of the same messages formatted in XML.

Request Example 1

```
<req>
    <body>
        <subscription>
            <orderBook>4008</orderBook>
        </subscription>
        <subscription>
            <orderBook>4007</orderBook>
        </subscription>
    </body>
</req>
```

Response Example 1

```
<res>
    <header>
        <status>OK</status>
    </header>
    <body/>
</res>
```

Request Example 2

```
<req>
    <body>
        <subscription>
            <exchangeRate>
                <from>JPY</From>
                <to>GBP</To>
            </exchangeRate>
```

```
        </subscription>
        <subscription>
            <exchangeRate>
                <from>USD</From>
                <to>GBP</To>
            </exchangeRate>
        </subscription>
    </body>
</req>
```

Response Example 2

```
<res>
    <header>
        <status>OK</status>
    </header>
    <body/>
</res>
```

2. Protocol SERVICES REFERENCE

This chapter contains descriptions of each of the methods supported by the LMAX Protocol.

2.1. ENDPOINT URL

The access point to LMAX Protocol over HTTPS interface is:

Production environment: https://api.lmaxtrader.com/

Test environment: https://testapi.lmaxtrader.com/

2.2. AUTHORISATION INFORMATION

`/public/*`	accessible without logging in
`/secure/*`	must be authenticated to access
`/secure/read/*`	read only access
`/secure/trade/*`	access to trading functionality

2.3. SESSION MANAGEMENT

2.3.1. Login

The login establishes a session-level login that will remain active while messages are being exchanged and that will timeout if traffic ceases for a prolonged period.

Responses to the Request message will either indicate a successful login or failure to login with no details of the nature of the failure.

A successful login will authenticate the session so that subsequent requests with the JSESSIONID cookie set will be identified as this user. The client must submit the session id with each request.

URI: *public/security/login*

Request

Element	Required	Supported values	Description
username	Yes		Valid user name
password	Yes		Valid password
productType	Yes	**CFD_LIVE**(live environment only) **CFD_DEMO**(test environment only)	Type of the product for the user
protocolVersion	No	**1.7**	Optional parameter, helps to ensure that the user uses currently supported version of LMAX protocol

Request Example

```
<req>
    <body>
        <username>user9001</username>
        <productType>CFD_LIVE</productType>
        <password>password</password>
       < protocolVersion>1.7</ protocolVersion>
    </body>
</req>
```

Please note: The order of the elements in the Login call is not important.

Response

Element	Description
status	The status of the request, eg. OK (successful) , WARN (Unsuccessful), ERROR(System error occurred)
username	Logged in user's username.
currency	Logged in user's currency
accountID	Logged in user's account id.
accountType	Type of the account
legalEntity	Type of entity (internal use only)
locale	Users local timezone information
FundingDisallowed	true or false
productType	Type of the product
failureType	Sent in case when login fails. Please see Login Failure Codes section for more details.
minProtocolVersion	Minimum supported version of the protocol

maxProtocolVersion	Maximum supported version of the protocol

Response Example

```
<res>
    <header>
        <status>OK</status>
    </header>
    <body>
        <username>user9001</username>
        <currency>GBP</currency>
        <accountId>9001</accountId>
        <AccountType>STANDARD_TRADER</AccountType>
        <productType>CFD_LIVE</productType>
        <fundingDisallowed>False</fundingDisallowed>
    </body>
</res>
```

2.3.2. Logout

Ending a session logs the user out of the LMAX Trader. It is recommended that you explicitly call the **Logout** service when your application shuts down.

Logout takes no parameters and will deactivate the current session.

URI: *public/security/logout*

Request

Element	Required	Description
none		

Request Example

```
<req>
   <body/>
</req>
```

Response

Element	Description
status	The status of the request, eg. OK (successful) , WARN (Unsuccessful), ERROR(System error occurred)

Response Example

```
<res>
   <header>
        <status>OK</status>
   </header>
```

```
    <body/>
</res>
```

2.3.3. GET longPollKey

URI: *secure/longPollKey*

Request

Element	Required	Description
none		

Response

Element	Description
`longPollKey`	longPollkey that must be submitted in the Request header of all the longPoll calls via the protocol throughout current session.

Response Example

```
<res>
    <header>
        <status>OK</status>
    </header>
    <body>
        <longPollKey>0</longPollKey>
    </body>
</res>
```

2.3.4. POST longPoll

longPoll call will result in an asynchronous batch response that contains the event updates for the period between the previous response to a longPoll call and the time when the last longPoll request received. Typically, a client will continuously make longPoll requests in order to receive all their events. Any longPoll request must contain the longPollKey, originally retrieved by the getLongPollKey call, as a HTTP request header

URI: *push/longPoll*

Request

Element	Required	Description
none		

Response Example

```
<res>
    <header>
        <status>OK</status>
```

```
    </header>
    <body>
        <heartbeat>
            <accountId>10209</accountId>
            <token>ABC123</token>
        </heartbeat>
    </body>
</res>
```

2.3.5. POST stream

Stream call will enable a single persistent connection that allows a server to push all real time updates to a client, without the client explicitly requesting it. These events are incrementally handled and interpreted on the client side every time the server sends a new event, with neither side closing the connection.

URI: *push/stream*

Request

Element	Required	Description
none		

2.3.6. Heartbeat

If a user wishes to maintain a logged-in session his software should send messages regularly to keep the session alive. Whenever the application calls the service the timer is reset. Any message activity is sufficient, and the heartbeat request is provided specifically to be used when your application generates no 'natural' activity.

The system does not send heartbeat events automatically; it will only respond to the users heartbeat requests. The heartbeat returns a synchronous response and echoes a heartbeat event, which serves to maintain the socket connection on the asynchronous push channel. The heartbeat also includes a client token which allows the client application to match responses to requests in an asynchronous fashion.

Users need to subscribe to the heartbeat events prior to sending a HTTP heartbeat request with the string token. If a user is subscribed to heartbeat events, the same string token will be returned via a heartbeat event in the event channel (i.e. event pushed to the user using either long poll or stream).

The LMAX Trader does not perform any uniqueness checking or de-duplication of tokens.
An event of type heartbeat will be emitted every time the system registers a heartbeat request as valid account activity. The same message structure applies to the body of the event.

URI: *secure/read/heartbeat*

Request

Element	Required	Supported values	Type	Description
token	Yes			

Request Example

```
<req>
    <body>
        <token>ABC123</token>
    </body>
</req>
```

Response

Element	Description
status	The status of the request, eg. OK (successful) , WARN (Unsuccessful), ERROR(System error occurred)

Response Example

```
<res>
    <header>
        <status>OK</status>
    </header>
    <body>
        <heartbeat>
            <accountId>10209</accountId>
            <token>ABC123</token>
        </heartbeat>
    </body>
</res>
```

2.4. ORDER MANAGEMENT

2.4.1. Supported Order Types

GoodUntil has been removed and replaced by the TimeInForce Enum

The use of Good Until Cancelled and Good For Day in the TimeInForce enum will result in the same behavior until we have completed the final behavior for Good Until Cancelled shortly.
When you use the TimeInForce Enum then the 'AllowUnmatched' option becomes irrelevant

Order Types:

- Market Order
- Stop Market
- Limit order

Time In Force Policies:

- Good Until Cancelled
- Good For Day
- Fill or Kill
- Immediate or Cancel

Order Type	Good Until	Description
Good Until Cancelled	`Cancelled`	The order is good until cancelled. This is the default if good until and allow unmatched are not specified.
Good For Day	`Day`	The order is good for the day only and will be closed at the end of the trading day
Fill or Kill	`Immediate`	If the order is not filled completely then the order is rejected with a reason code of INSUFFICIENT_LIQUIDITY
Immediate or Cancel	`Immediate`	If the order is not filled immediately the remaining unmatched portion is cancelled.
Stop Market	`Cancelled`	Instruction to place a Market Order to buy or Sell when the best price in the Market reaches your level

2.4.2. Place Order

Clients can request new orders for execution on the LMAX MTF. In response the LMAX Trader will indicate that that order has been successfully received, and will return an identifier unique to that order. Clients can also optionally supply their own unique identifier for an order by using <instructionId> field. If this value is not supplied by the Client LMAX will assign the instruction Id to an order on its entry to the LMAX system. Please note, LMAX will guarantee the uniqueness of the instruction ID only during the lifetime of the order and it's subsequent position.

Order placement will result in a sequence of one or more of the following asynchronous events being sent from the MTF over the lifetime of the order. Not every order will generate the same events – for example an unmatched order that would (but is yet to) reduce the exposure of a matched position will not generate an `accountState` event.

- When position change occurs – event of type position is generated following the structure of the `positions` message
- When Execution occurs in the event of matching – event of type `orderState` is generated following the structure of the `orderState` message with the `execution` element
- When Execution occurs in the event of order cancelation – event of type `orderState` is generated following the structure of the `orderState` message with the `orderCancelled` element
- When account state change occurs – event of type `accountState` is generated following the structure of the `accountState` message
- When order book change for an instrument occurs – event of type `orderBook` is generated following the structure of the `orderBook` message

An event of type order will be emitted every time a match occurs. The message structure that applies to the body of the event is described below.

URI: *secure/trade/placeOrder*

Request

Element	Required	Supported Values	Description
order			Request type
instructionId	No	Long Positive, non-zero	Unique identifier optionally supplied by the client for this order.
instrumentId	Yes		The LMAX instrument in which the order is to be placed
price	Yes Not if Market		The least favourable per-unit price at which the order should match.
quantity	Yes	If quantity is negative number – sell order If quantity is positive number – buy order	Requested quantity for the order.
stopLossOffset	No	must be non-zero positive integer	Stop loss offset
stopProfitOffset	No	must be non-zero positive integer	Stop profit offset
timeInForce	Yes	GoodTilCancelled, GoodForDay, FillOrKill, ImmediateOrCancel	Valid values are as follows: ● GoodTilCancelled ● GoodForDay ● ImmediateOrCancel – the order completes immediately and can be partially filled ● FillOrKill – the order completes immediately can cannot be partially filled

Response

Element	Description
status	The status of the request, eg. OK (successful) , WARN (Unsuccessful), ERROR(System error occurred)
instructionId	The unique identifier of the instruction for the order placed.

Place Market Sell Order request example – with supplied instructionId

```
<req>
    <body>
        <order>
```

```
            <instructionId>1</instructionId>
            <instrumentId>4003</instrumentId>
            <quantity>-1</quantity>
           <timeInForce>FillOrKill</timeInForce>
      </order>
    </body>
</req>
```

Place Market Order response example

```
<res>
    <header>
        <status>OK</status>
    </header>
    <body>
        <instructionId>1</instructionId>
    </body>
</res>
```

Good Until Cancelled Limit Buy Order Request Example – without supplied instructionId

```
<req>
    <body>
        <order>
            <instrumentId>4008</instrumentId>
            <price>1.0</price>
            <quantity>100</quantity>
            <stopLossOffset>0.1</stopLossOffset>
            <stopProfitOffset>0.1</stopProfitOffset>
           <timeInForce>GoodTilCancelled</timeInForce>
        </order>
    </body>
</req>
```

Good Until Cancelled Limit Order Response Example

```
<res>
      <header>
            <status>OK</status>
      </header>
      <body>
            <instructionId>-260048</instructionId>
      </body>
</res>
```

2.5. Place Stop Market Order

URI: *secure/trade/placeStop*

Request

Element	Required	Supported	Description

		Values	
order			Request type
instructionId	No	Long Positive, non-zero	Unique identifier optionally supplied by the client for this order.
instrumentId	Yes		The LMAX instrument in which the order is to be placed
stopCondition	Yes		The condition for the stop trigger
price	Yes Not if Market		The least favourable per-unit price at which the order should match.
quantity	Yes	If quantity is negative number – sell order If quantity is positive number – buy order	Requested quantity for the order.
goodUntil	No	Immediate Cancelled	Specifies the lifespan of the order.
allowUnmatched	No	false true	Specifies whether the order must be matched in full (false) or can be partially matched (true). Must be specified if goodUntil is specified.

Response

Element	Description
status	The status of the request, eg. OK (successful) , WARN (Unsuccessful), ERROR(System error occurred)
instructionId	The unique identifier of the instruction for the order placed.

Place Stop Market Order request example

```
<req>
     <body>
         <order>
             <instrumentId>179360</instrumentId>
```

```
            <stopCondition>
                <price>165</price>
            </stopCondition>
            <quantity>20</quantity>
            <goodUntil>Cancelled</goodUntil>
            <allowUnmatched>true</allowUnmatched>
        </order>
    </body>
</req>
```

Place Stop Market Order response example

```
<res>
      <header>
            <status>OK</status>
      </header>
      <body>
            <instructionId>-260050</instructionId>
      </body>
</res>
```

2.6. Cancel Order

This request allows the unmatched portion of one or more previously placed orders to be cancelled either individually, by instrument or across all positions for the account.

Clients can also optionally supply their own unique identifier for an order cancel request in <instructionId> field. If this value is not supplied by the Client LMAX will assign the instruction Id to a request on its entry to the LMAX system.

Fully cancelled orders (i.e. orders cancelled before any matches occurred) are NOT returned by the `cancel` request.

Cancelling an order will cause a position change, indicated by an event of type PositionEvent following the structure of the `positions` message and may cause a change to available balance, indicated by an event of type `accountState` following the structure of the `accountState` message

Also, `orderBook` event update will be generated if the cancelation of the order resulted in any changes to the order book.

URL: *secure/trade/cancel*

Request

Element	Required	Description

instrumentId	Yes	The id of the LMAX instrument for which to cancel orders.
instructionId	Yes	Long Positive, non-zero unique identifier optionally supplied by the client for this order.
originalInstructionId	Yes	The id of the original order to be cancelled.
Note: If instrumentId, originalInstructionId are not specified ALL unmatched order quantities for the account will be cancelled.		

Request Example

```
<req>
        <instrumentId>4008</instrumentId>
        <originalInstructionId>260048</originalInstructionId>
</req>
```

Request Example all orders

```
<req>
    <body/>
</req>
```

Response

Element	Description
status	The status of the request, eg. OK (successful) , WARN (Unsuccessful), ERROR(System error occurred)
instructionId	The ID of the cancelled order

Response Example

```
<res>
    <header>
        <status>OK</status>
    </header>
    <body>
        <instructionID>-260048</instructionID>
    </body>
</res>
```

2.7. MARKET DATA MANAGEMENT

In order to actively monitor market events user can subscribe to the state and/or price changes for any instrument meeting their criteria.

Please see the Subscription Management and Push Events chapters for the details of different subscription options and corresponding events.

2.8. SUBSCRIPTION MANAGEMENT

The Subscribe message allows the client to subscribe to a variety of events. Each request contains one or more subscription elements. Subscriptions are NOT hierarchical, so subscriptions to an underlying category are to changes to that category node only, not to the addition, removal or amendments of its children.

You can subscribe to the following events:

- Account State Changes
- Position Changes
- Order Executions
- MTF Rates
- Order Book events (e.g. price updates)
- Order Book Status (e.g. New, Closed, Suspended)

2.8.1. Subscribe request

The generic subscription request is described below. In order to subscribe for a particular update the user must include the corresponding elements or specify the type of the subscription.

URI: *secure/subscribe*

Request

Element	Required	Supported Values	Description

`longPollKey`	Yes- if using longPoll
	
	No – if using Streaming
		`minLength = 1`
		
		`maxLength = 20`
			Long Poll Key

`subscription`	Yes		One or more subscriptions.
`orderBook`	No	Positive non-zero long type unique identifier of the instrument whose price events are required.	Subscribe to the price changes for the instrument(s) specified.
`orderBookStatus`	No	Positive non-zero long type unique identifier of the instrument whose price events are required.	Subscribe to order book status changes
`exchangeRate`	No	The “From” and “To”	Subscribe for the Exchange Rates for particular “From” and

		currencies. "pattern">[A-Z]{3}	“To” currency.
type	No	account	Subscribe to all events of the specified type across all instruments: account state
type	No	order	Subscribe to all events reflecting orders state changes
type	No	position	Subscribe to all events reflecting positions state
*Note: one and only one of each presented above elements must be specified per **subscription** element.*			

Request OrderBook Example

```
<req>
    <body>
        <subscription>
            <orderBook>4001</orderBook>
        </subscription>
        <subscription>
            <orderBook>4008</orderBook>
        </subscription>
    </body>
</req>
```

Request Account Changes Example

```
<req>
    <body>
        <subscription>
            <type>account</type>
        </subscription>
    </body>
</req>
```

Request Exchange Rates Example

```
<req>
    <body>
        <subscription>
            <exchangeRate>
                <from>USD</from>
                <to>GBP</to>
            </exchangeRate>
        </subscription>
    </body>
</req>
```

Response

Element	Description
status	The status of the request, eg. OK (successful) , WARN (Unsuccessful), ERROR(System error occurred)

Response Example

```
<res>
    <header>
        <status>OK</status>
    </header>
    <body/>
</res>
```

2.8.2. Unsubscribe Request

The unsubscribe request is the inverse of a subscribe request. Once received LMAX Trader will stop sending events for the `subscription` topic. Request and response messages are identical to the subscribe request.

URL: secure/unsubscribe

2.9. PUSH EVENTS

Subscription	Event	Type	Description
type = account	accountState	Asynchronous Snapshot + Updates	This event occurs when there is a change to the trading partner's account current balances and exposure. It will generate an update on the current state of the account state.
orderBook	orderBook	Asynchronous Snapshot upon request+ Updates	It will generate a snapshot of the current prices available on a specified instrument This event signals the change in the current prices available on a specified instrument.
exchangeRate	exchangeRate	Asynchronous Snapshot + Updates	This event signals the change to the exchange rate for a particular currency. It will generate the buy and sell exchange rate update for the selected currencies.
type = position	position	Asynchronous Snapshot upon request+	As a result of the successful subscription request, initial snapshot of the positions will be generated, followed by the position update event every time any of the

		Updates	positions details change for a specified instrument, for example in response to a trading partner placing orders or matching with counter orders in the LMAX Trader.
`orderBookStatus`	`orderBookStatus`	Asynchronous Snapshot + Updates	An event signaling a change in an instrument's status. E.g. New, Close, suspended
`type = account`	`instructionRejected`	Asynchronous Updates	An event signalling that an order has been rejected.
`type = account`	`heartbeat`	Asynchronous Updates	Heartbeat event for streaming
`type = order`	`orderState`	Asynchronous Snapshot + Updates	As a result of the successful subscription request, initial snapshot of the open orders will be generated, followed by the orderState update event signaling an order changes resulting from order placement, order matching or order cancelation.

2.9.1. exchangeRate event

An initial snapshot message `exchangeRate` with current exchange rates will be submitted after the subscription request was made, followed by an event of type `exchangeRate` will be emitted every time exchange rate updated. The same message structure applies to the body of the event.

Message

Element	Description
`exchangeRate`	Name of the event
`from`	The currency to convert from
`to`	The currency to convert to
`buy`	Buy rate
`sell`	Sell rate

Message Example

```
<exchangeRate>
     <from>EUR</from>
     <to>GBP</to>
```

```
        <buy>1.24</buy>
        <sell>1.23</sell>
    </exchangeRate>
```

2.9.2. Account events

2.9.2.1. accountState event

This message will contain the trading partner's current balances and exposure.

If an account has open or working orders then the message with the initial snapshot of the account state will be published followed by an event of type `accountState` every time any of the balance details change.

The same message structure applies to the body of the event.

Message

Element	Description
accountState	The name of the event.
accountID	The unique identifier of the account.
balance	The current balance of the account
availableFunds	The funds in the account available for trading
availableToWithdraw	The funds in the account available to withdraw
unrealisedProfitAndLoss	Unrealised profit and loss value
margin	Margin value
wallets	Zero or more wallets
wallet	
currency	The currency of the wallet
balance	Current balance in the wallet

Message Example

```
<accountState>
    <accountId>10000</accountId>
    <balance>1000.00000</balance>
    <availableFunds>950.00000</availableFunds>
    <availableToWithdraw>950.00000</availableToWithdraw>
    <wallets>
        <wallet>
            <currency>USD</currency>
            <balance>1842080.1</balance>
        </wallet>
        <wallet>
```

```
            <currency>JPY</currency>
            <balance>1042080.0</balance>
        </wallet>
        <wallet>
            <currency>GBP</currency>
            <balance>142080.5</balance>
        </wallet>
    </wallets>
</accountState>
```

2.9.2.2. orderState event

`orderState` message is sent in the result of `orderState` event generated by any order changes resulting from order placement, order matching or order cancelation.

Element	Description
`orders`	0 or more orders
`order`	An order
`instructionId`	Instruction ID
`originalInstructionId`	Original Instruction ID
`orderId`	Order ID
`accountId`	The id of the account.
`instrumentId`	The unique LMAX id for the instrument associated with this execution report
`quantity`	Quantity of the order; Signed positive for Buy order, negative for Sell order
`price`	Price of the order
`matchedQuantity`	The portion of the order matched. Signed depending on the direction, positive for Buy, negative for Sell
`cancelledQuantity`	The portion of the order cancelled. Signed, positive for Buy, negative for Sell **Please note, LMAX only supports complete cancellations.**
`timestamp`	The time of the instruction
`orderType`	`STOP_COMPOUND_PRICE_LIMIT`, `STOP_MARKET_OPENING_ORDER`
`openQuantity`	Total position quantity as a result of this order. Signed, positive for Long, negative for Short.*
`openingOrderId`	Opening Order ID
`openCost`	Cumulative cost + signed profit accrued as a result of this trade. Negative values indicate a loss.
`cumulativeCost`	Signed opened cost + signed closed cost
`commission`	commission
`stopReferencePrice`	Stop reference type

stopLossOffset	Stop loss offset
stopProfitOffset	Stop profit offset
executions	Execution report for all matches if any occurred
executionId	An id for this execution, unique for the instrument, 0 or more executions.
execution	A record for each execution
price	The price at which the match was executed.
quantity	The quantity matched.
realisedProfit	The profit that this execution realized for the overall position on the instrument.
orderCancelled	A cancelled quantity at a price point
quantity	
hasMoreresults	Indicates whether there is another page with orders to come (true or fault)
correlationId	Correlation id for the position update event.

*** For example:**

If user has no position and places an order A to buy quantity of 10 and get filled for 3, we will have

order A
quantity = 10
matchedQuantity = 3
cancelledQuantity = 0
openQuantity = 3

So order A has an open position 3 long, and a working order with 7 left to fill. If the user then places Order B to Sell 2 and get filled for 2, we will see

order A
quantity = 10
matchedQuantity = 3
cancelledQuantity = 0
openQuantity = 1

order B
quantity = -2
matchedQuantity = -2
cancelledQuantity = 0
openQuantity = 0

So the openQuantity on B is zero because it didn't open a position - it closed part of the position opened by Order A.

Message Example

```
<orders>
   <order>
   <accountId>100000</accountId>
   <cancelledQuantity>0</cancelledQuantity>
   <commission>1</commission>
   <executions>
        <executionId>12200642</executionId>
        <execution>
            <price>1205.5</price>
            <quantity>-1</quantity>
        </execution>
    </executions>
    <instructionId>330043</instructionId>
    <instrumentId>1002</instrumentId>
    <matchedQuantity>-1</matchedQuantity>
    <openCost>-12055</openCost>
    <openQuantity>-1</openQuantity>
    <cumulativeCost>"-12055</cumulativeCost>
    <orderId>AAAPowAAAAAAXT38</orderId>
    <orderType>STOP_COMPOUND_MARKET</orderType>
    <quantity>-1</quantity>
    <stopLossOffset/>
    <stopProfitOffset/>
    <stopReferencePrice>1205.5</stopReferencePrice>
    <timestamp>2010-11-01T09:57:07</timestamp>
   </order>
</orders>
```

2.9.2.3. Order Execution Update

Each execution represents one or more matches on the LMAX MTF that is related to an order placed by the account holder. All matches of the same order at the same time and price are summarized in a single execution and sent as part of the Order Update Message.

Message Example

```
<executions>
    <executionId>12200642</executionId>
    <execution>
        <price>1.099</price>
        <quantity>-5</quantity>
    </execution>
</executions>
```

2.9.2.4. Order Cancellation Update

Each `orderCancelled` represents a cancelation of an order placed by the account holder.

Order Cancelled event example

```
<executions>
    <executionId>3</executionId>
    <orderCancelled>
        <quantity>4</quantity>
```

```
    </orderCancelled>
</executions>
```

2.9.2.5. position event

This message will contain the details of an account holder's current position in a specified instrument. Client will get the initial snapshot of the positions after the subscription request is made, followed by the position updates when the positions change

The subsequent events of type `positions` will be emitted every time any of the positions details change, for example in response to a trading partner placing orders or matching with counter orders in the MTF. The same message structure applies to the body of the event.

Message

Element	Description
positions	
position	The position for a specific instrument.
accountId	The account holding this position.
instrumentId	The id of the Instrument for which this is a report of position.
valuation	The current total valuation on this instrument (matched and unmatched).
shortUnfilledCost	The short unfilled cost
longUnfilledCost	The long unfilled cost
openQuantity	Open Quantity = Long Quantity - Short Quantity
cumulativeCost	Cumulative Cost = Long Cost - Short Cost
openCost	Open Cost = Cumulative Cost - Realised Profit
hasMoreResults	Indicates whether there will be more updates to come
correlationId	Correlation ID for the update

Message Example

```
<positions>
        <position>
            <accountId>100000</accountId>
            <instrumentId>4008</instrumentId>
            <valuation>-30366480</valuation>
            <shortUnfilledCost>0</shortUnfilledCost>
            <longUnfilledCost>0</longUnfilledCost>
            <openQuantity>-603</openQuantity>
            <cumulativeCost>-340039750</cumulativeCost>
            <openCost>-465101250</openCost>
        </position>
        <position>
            <accountId>100000</accountId>
            <instrumentId>4001</instrumentId>
            <valuation>-228558</valuation>
            <shortUnfilledCost>0</shortUnfilledCost>
```

```
            <longUnfilledCost>0</longUnfilledCost>
            <openQuantity>-105</openQuantity>
            <cumulativeCost>-1259650</cumulativeCost>
            <openCost>-1259650</openCost>
        </position>
    <hasMoreResults>false</hasMoreResults>
    <correlationId>0-0</correlationId>
</positions>
```

2.9.2.6. instructionRejected event

Response

Element	Description
instructionRejected	The status of the request, eg. OK (successful) , WARN (Unsuccessful), ERROR(System error occurred)
instructionId	The unique instruction id of the order.
accountId	The unique identifier of the account that placed the order.
instrumentId	The unique identifier of the instrument upon which the order was placed.
reason	The reason code for the rejection.

Example

```
<instructionRejected>
    <accountId>10000</accountId>
    <instrumentId>4008</instrumentId>
    <instructionId>260048</instructionId>
    <reason>TEMPORARY_SUSPENSION</reason>
</instructionRejected>
```

2.9.2.7. heartbeat event

Response

Element	Description
accountId	The unique identifier of the account that placed the order.
token	

Example

```
<heartbeat>
    <accountId>10000</accountId>
    <token>4008</token>
</heartbeat>
```

2.10. MARKET DATA EVENTS

2.10.1. orderBook2 Event

The <ob2> element is much more compact and designed to reduce the size of messages associated with the orderBook Event. Instead of an XML representation it is basically a String.

orderBook shall be deprecated as Ob2 is a much more efficient element.Please see the orderBook2 event message and examples below.

Message

Element	Description
ob2	Name of the element
instrumentId	The unique (long) identifier for the instrument the prices relate to
exchangeTimeStamp	Time on the current event as a Hexadecimal number
bids	A string of up to 5 semi-colon separated quantity@price pairs ordered from best to worst e.g. 10@50;10@49;10@48
offers	A string of up to 5 semi-colon separated quantity@price pairs ordered from best to worst e.g. 10@51;10@52;10@53
lastMarketClosePrices	
price	Market close price
timestamp	timestamp
dailyHighestTradedPrice	Highest traded price for the trading day
dailyLowestTradedPrice	Lowest traded price for the trading day
valuationBidPrice	Valuation bid price
valuationAskPrice	Valuation ask price
lastTradedPrice	Last traded price

Example

Format of the <ob2> element value is:

InstrumentId '|' Timestamp '|' Bids '|' Asks '|' MarketClose '|' DailyHigh '|' DailyLow '|' ValuationBid '|' ValuationAsk '|' LastTraded

For Example,
4001|1309cde347b|100@1.41969;300@1.41968;10@1.41967;50@1.41966;300@1.41965|200@1.41975;100@1.41978;300@1.41979;10@1.41991;50@1.41992||||1.41969|1.41975|

2.10.2. Historic Market Data – Deprecated, replaced by Top-of-Book only API to be added shortly

2.10.4.1. Historic Market Data Event

Historic Market Tick Data is available 5 levels deep from the beginning of our Exchange. Currently it will be available as a link that will return a compressed (.gz) file. You must be logged into your secure session in order to download this URL.

Currently, you will receive a full months worth of data to parse. If your request spans of multiple months then you will receive multiple URLs.

URL: secure/tickhistory

Message

Element	Description
instructionId	Name of the event
instrumentId	The unique identifier for the instrument the prices relate to
TickHistoryContentTypeEnum	Currently available as tick level only; <tick>
startDate	Start date
endDate	End date

Example Request

```
<body>
      <instructionId>127847</instructionId>
      <instrumentId>4001</instrumentId>
      <contentType>MARKET_DEPTH</contentType>
      <startDate>2011-04</startDate>
      <endDate>2011-06</endDate>
</body>
```

Example Response

```
<body>
      <tickHistoryResponse>
            <instructionId>127847</instructionId>
```

```
      <tickHistoryFile>https://testapi.lmaxtrader.com/secure/tickhistory/400
1/2011/4001-2011-04.csv.gz</tickHistoryFile>
      <tickHistoryFile>https://testapi.lmaxtrader.com/secure/tickhistory/400
1/2011/4001-2011-05.csv.gz</tickHistoryFile>
      <tickHistoryFile>https://testapi.lmaxtrader.com/secure/tickhistory/400
1/2011/4001-2011-06.csv.gz</tickHistoryFile>

            </tickHistoryResponse>
      </body>
```

Please note that our Historic Market Data service will be changing soon.

3. ERROR CODES

3.1. AUTHORISATION FAILURE REASONS

Code	Description
UNAUTHORISED	Access unauthorised
UNAUTHENTICATED	Unauthenticated user
SESSION_EXPIRED	Session expired

3.2. LOGIN FAILURE CODES

Code	Description
ACCOUNT_CLOSED	Account is closed
ACCOUNT_LOCKED	Account is locked
ACCOUNT_SUSPENDED	Account is suspended
BAD_CREDENTIALS	Bad credentials used
TEMPORARY_SYSTEM_ERROR	Temporary system error
SERVICE_UNAVAILABLE	Service unavailable
PROTOCOL_VERSION_INVALID	The version of the LMAX Protocol specified is invalid.

3.3. ORDER REJECTION REASONS

Code	Description
INSTRUMENT_DOES_NOT_EXIST	The instrument on which the order was placed doesn't exist.
INSTRUMENT_NOT_OPEN	The instrument on which the order was placed is not currently open for trading.
PRICE_NOT_VALID	The price specified on the order was invalid. Typically outside the range or precision acceptable to the instrument.
DUPLICATE_ORDER	Specified instruction id is a duplicate
QUANTITY_NOT_VALID	The quantity specified on the order was invalid. Typically outside the range or number of decimal places acceptable to the instrument.
EXPOSURE_CHECK_FAILURE	The funds available to the account are not sufficient to place this order.
INTERNAL	An internal system error prevented the order being accepted.
UNKNOWN_ORDER	The instruction relates to an order that does not exist or has been closed.

ORDER_TYPE_INVALID	The combination of Good Until and Allow Unmatched were invalid.
SERVICE_UNAVAILABLE	service is temporarily unavailable
NO_QUANTITY_TO_CANCEL	There is no remaining working quantity to cancel
ALREADY_CANCELLED	The order has been previously cancelled
ACCESS_DENIED	The user is not allowed to submit this instruction
OUTSIDE_ALLOWED_PRICE_RANGE	The order price is invalid; too far away from current market price
STOP_LOSS_OFFSET_INVALID	stop loss price offset was invalid
STOP_PROFIT_OFFSET_INVALID	profit target price offset was invalid
TEMPORARY_SUSPENSION	the market is temporarily suspended

4. Protocol SCHEMAS

4.1. userLogin-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../public/public-datatypes.rng"/>
    <include href="../common/tfx-request.rng">
        <define name="RequestBody">
            <interleave>
                <element name="username">
                    <ref name="UsernameType"/>
                </element>
                <element name="password">
                    <ref name="LoginAttemptPasswordType"/>
                </element>
                <element name="productType">
                    <ref name="ProductType"/>
                </element>
                <element name="protocolVersion">
                     <ref name="NonEmptyStringType" />
                </element>
                </optional>
                <optional>
<!-- Used for password reset when the user must enter the temporary password and the
password they wish to use -->
             <optional>
                    <element name="newPassword">
                        <ref name="PasswordType"/>
                    </element>
                </optional>
            </interleave>
        </define>
    </include>
</grammar>
```

4.2. userLogin-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

<include href="published-datatypes.rng"/>
     <include href="tfx-response.rng">
        <define name="ResponseBody">
            <choice>
                <ref name="ValidLoginResponseType"/>
                <element name="failureType">
                    <ref name="LoginFailureEnum"/>
                    <optional>
                          <element name="minProtocolVersion">
                              <ref name="StringType"/>
                          </element>
                          <element name="maxProtocolVersion">
                              <ref name="StringType"/>
                          </element>
                      </optional>

                </element>
            </choice>
        </define>
    </include>
</grammar>
```

4.3. getLongPollKey-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../common/tfx-request.rng"/>
</grammar>
```

4.4. getLongPollKey-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../public/public-datatypes.rng"/>
    <include href="../common/tfx-response.rng">
       <define name="ResponseBody">
              <element name="longPollKey">
                 <ref name="LongPollKeyType"/>
              </element>
       </define>
    </include>
</grammar>
```

4.5. heartbeat-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../common/tfx-request.rng">
        <define name="RequestBody">
            <element name="token">
                <text/>
            </element>
        </define>
    </include>
</grammar>
```

4.6. heartbeat-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../common/tfx-response.rng">
       <define name="ResponseBody">
              <element name="token">
                 <text/>
              </element>
       </define>
    </include>
</grammar>
```

4.7. logout-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../common/tfx-request.rng"/>
</grammar>
```

4.8. logout-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../common/tfx-response.rng"/>
</grammar>
```

4.9. placeOrder-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <!-- A request to place a single order. -->
    <include href="../public/public-datatypes.rng"/>
    <include href="../common/tfx-request.rng">
        <define name="RequestBody">
            <element name="order">
                <interleave>
                    <optional>
                        <element name="instructionId">
                            <ref name="SuppliedInstructionIdType"/>
                        </element>
                    </optional>

                    <element name="instrumentId">
                        <ref name="InstrumentIdType"/>
                    </element>
                    <ref name="PlaceOrderPayload"/>
                </interleave>
            </element>
        </define>
    </include>
```

4.10. placeStop-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <!-- A request to place a single order. -->
    <include href="published-datatypes.rng"/>
    <include href="tfx-request.rng">
        <define name="RequestBody">
            <element name="order">
                <interleave>
                  <optional>
                        <element name="instructionId">
                            <ref name="SuppliedInstructionIdType"/>
                        </element>
                    </optional>
                    <ref name="PlaceStopPayload"/>
                </interleave>
            </element>
        </define>
    </include>
</grammar>
```

4.11. placeStop-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <!-- The response when an order is placed. -->
    <include href="published-datatypes.rng"/>
    <include href="tfx-response.rng">
        <define name="ResponseBody">
            <optional>
                <!-- The unique identifier of the instruction for the order placed.
-->
                <element name="instructionId">
```

```
                    <ref name="InstructionIdType"/>
                </element>
            </optional>
        </define>
    </include>
</grammar>
```

4.12. placeOrder-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <!-- The response when an order is placed. -->
    <include href="../public/public-datatypes.rng"/>
    <include href="../common/tfx-response.rng">
        <define name="ResponseBody">
            <optional>
                <!-- The unique identifier of the instruction for the order placed. -->
                <element name="instructionId">
                    <ref name="InstructionIdType"/>
                </element>
            </optional>
        </define>
    </include>
</grammar>
```

4.13. cancelOrder-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <!-- A request to cancel all orders on an instrument. -->
    <include href="../public/public-datatypes.rng"/>
    <include href="../common/tfx-request.rng">
        <define name="RequestBody">
            <interleave>
                <element name="instrumentId">
                    <ref name="InstrumentIdType"/>
                </element>
             <optional>
                    <element name="instructionId">
                        <ref name="SuppliedInstructionIdType"/>
                    </element>
                </optional>
                <element name="originalInstructionId">
                    <ref name="InstructionIdType"/>
                </element>
            </interleave>
        </define>
    </include>
</grammar>
```

4.14. cancelOrder-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../public/public-datatypes.rng"/>
    <include href="../common/tfx-response.rng">
        <define name="ResponseBody">
            <optional>
                <element name="instructionId">
                    <ref name="InstructionIdType"/>
                </element>
            </optional>
        </define>
```

```
    </include>
</grammar>
```

4.15. subscribe-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../public/public-datatypes.rng"/>
    <include href="../common/tfx-request.rng">
        <define name="RequestBody">
            <oneOrMore>
                <ref name="Subscription"/>
            </oneOrMore>
        </define>
    </include>
</grammar>
```

4.16. subscribe-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../common/tfx-response.rng"/>
</grammar>
```

4.17. unsubscribe-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../public/public-datatypes.rng"/>
    <include href="../common/tfx-request.rng">
        <define name="RequestBody">
            <oneOrMore>
                <ref name="Subscription"/>
            </oneOrMore>
        </define>
    </include>
</grammar>
```

4.18. unsubscribe-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="../common/tfx-response.rng"/>
</grammar>
```

4.19. exchangeRate-event

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="public-datatypes.rng"/>
        <start>
            <element name="exchangeRate">
                <interleave>
                    <element name="from">
                        <ref name="CurrencyType"/>
                    </element>
                    <element name="to">
                        <ref name="CurrencyType"/>
                    </element>
                    <element name="buy">
                        <ref name="PriceType"/>
                    </element>
```

```
                    <element name="sell">
                        <ref name="PriceType"/>
                    </element>
                </interleave>
            </element>
        </start>
</grammar>
```

4.20. accountState-event

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
<include href="published-datatypes.rng"/>

    <start>
        <element name="accountState">
            <ref name="AccountStateType"/>
        </element>
    </start>
</grammar>
```

4.21. orderState-event

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

    <include href="public-datatypes.rng"/>
    <start>

       <choice>
                   <element name="orders">
                       <ref name="OrderStatePageType"/>
                   </element>
                   <element name="order">
                       <ref name="OrderStateType"/>
                   </element>
       </choice>

    </start>

    <define name="OrderStatePageType">
        <element name="page">
            <zeroOrMore>
                <element name="order">
                    <ref name="OrderStateType"/>
                </element>
            </zeroOrMore>
        </element>
        <element name="hasMoreResults">
            <ref name="BooleanType"/>
        </element>
        <element name="correlationId">
            <ref name="NonEmptyStringType"/>
        </element>
    </define>

    <define name="OrderStateType">
        <interleave>

       <element name="instructionId">
                <ref name="InstructionIdType"/>
            </element>
            <optional>
                <element name="originalInstructionId">
                    <ref name="InstructionIdType"/>
```

```
    </element>
</optional>
<element name="orderId">
    <ref name="OrderIdType"/>
</element>
<!--The id of the account.-->
<element name="accountId">
    <ref name="AccountIdType"/>
</element>
<!--The id of the instrument.-->
<element name="instrumentId">
    <ref name="InstrumentIdType"/>
</element>
<choice>
    <group>
        <element name="quantity">
            <ref name="QuantityType"/>
        </element>
    </group>
    <group>
        <element name="price">
            <ref name="PriceType"/>
        </element>
        <element name="quantity">
            <ref name="QuantityType"/>
        </element>
    </group>
</choice>
<!--The portion of the order matched.-->
<element name="matchedQuantity">
    <ref name="QuantityType"/>
</element>
<!--The portion of the order cancelled.-->
<element name="cancelledQuantity">
    <ref name="QuantityType"/>
</element>
<optional>
    <interleave>
        <element name="stopReferencePrice">
            <choice>
                <ref name="PriceType"/>
                <empty/>
            </choice>
        </element>
        <element name="stopLossOffset">
            <choice>
                <ref name="PriceType"/>
                <empty/>
            </choice>
        </element>
        <element name="stopProfitOffset">
            <choice>
                <ref name="PriceType"/>
                <empty/>
            </choice>
        </element>
    </interleave>
</optional>
<element name="timestamp">
    <ref name="DateTimeType"/>
</element>
<element name="orderType">
    <ref name="OrderTypeEnum"/>
</element>
<element name="openQuantity">
    <ref name="QuantityType"/>
</element>
```

```
            <optional>
                <element name="openingOrderId">
                    <ref name="StringType"/>
                </element>
            </optional>
            <element name="openCost">
                <ref name="PriceType"/>
            </element>
            <element name="cumulativeCost">
                <ref name="PriceType"/>
            </element>
            <element name="commission">
                <ref name="PriceType"/>
            </element>
            <optional>
                <!--  Represents an execution report -->
                <element name="executions">
                    <!--The identifier of the latest match execution that contributed to the
matched position on this instrument.
         Only present if the account has any matches associated with this position.-->
                    <element name="executionId">
                        <ref name="ExecutionIdType"/>
                    </element>
                    <zeroOrMore>
                        <choice>
                            <!--  An execution at a price point -->
                            <element name="execution">
                                <ref name="ExecutionType"/>
                            </element>
                            <!--  A cancelled quantity at a price point -->
                            <element name="orderCancelled">
                                <ref name="OrderCancellationType"/>
                            </element>
                        </choice>
                    </zeroOrMore>
                </element>
            </optional>
        </interleave>
    </define>

    <define name="ExecutionType">
        <!--The price at which the execution occured.-->
        <element name="price">
            <ref name="PriceType"/>
        </element>
        <!--The quantity that was executed.-->
        <element name="quantity">
            <ref name="QuantityType"/>
        </element>
        <!--The profit that this execution realised for the overall position on the
instrument.-->
        <optional>
            <element name="realisedProfit">
                <ref name="PriceType"/>
            </element>
        </optional>
    </define>

    <define name="OrderCancellationType">
        <!--The quantity that was cancelled.-->
        <element name="quantity">
            <ref name="QuantityType"/>
        </element>
    </define>
</grammar>
```

4.22. position-event

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

    <include href="public-datatypes.rng"/>
<start>
        <ref name="PositionEvent"/>
    </start>

<define name="PositionEvent">
        <choice>
            <element name="positions">
                <ref name="PositionPageType"/>
            </element>
            <element name="position">
                <ref name="PositionType" />
            </element>
        </choice>

    </define>

                <

    <define name="PositionPageType">
            <zeroOrMore>
                <element name="position">
                    <ref name="PositionType"/>
                </element>
            </zeroOrMore>
        </element>
        <element name="hasMoreResults">
            <ref name="BooleanType"/>
        </element>
        <element name="correlationId">
            <ref name="NonEmptyStringType"/>
        </element>
    </define>

    <!--A statement of an account's position on an identified instrument
         ...for accountController.getOpenPositions(offset) and for push position events.-->
    <define name="PositionType">
        <interleave>
            <!--The unique identifier of the account the position relates to.-->
            <element name="accountId">
                <ref name="AccountIdType"/>
            </element>
            <!--The unique identifier of the instrument the position relates to.-->
            <element name="instrumentId">
                <ref name="InstrumentIdType"/>
            </element>
            <!-- Cumulative Cost = Long Cost - Short Cost -->
            <element name="cumulativeCost">
                <ref name="PriceType"/>
            </element>
            <!-- Open Cost = Cumulative Cost - Realised Profit -->
            <element name="openCost">
                <ref name="PriceType"/>
            </element>
            <!-- Open Quantity = Long Quantity - Short Quantity -->
            <element name="openQuantity">
                <ref name="QuantityType"/>
            </element>
            <!-- The short unfilled cost -->
            <element name="shortUnfilledCost">
                <ref name="PriceType"/>
            </element>
```

```
            <!-- The long unfilled cost -->
            <element name="longUnfilledCost">
                <ref name="PriceType"/>
            </element>
            <!--The current total valuation on this instrument (matched and unmatched).-->
            <element name="valuation">
                <ref name="PriceType"/>
            </element>
        </interleave>
    </define>
</grammar>
```

4.23. instructionRejected-event

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <include href="public-datatypes.rng"/>
    <include href="errors.rng"/>
        <start>
            <element name="instructionRejected">
                <ref name="InstructionRejectedType"/>
            </element>
        </start>

    <!--An event signalling that an order has been rejected.-->
    <define name="InstructionRejectedType">
        <interleave>
            <element name="instructionId">
                <ref name="InstructionIdType"/>
            </element>
            <element name="accountId">
                <ref name="AccountIdType"/>
            </element>
            <optional>
                <element name="instrumentId">
                    <ref name="InstrumentIdType"/>
                </element>
            </optional>
            <element name="reason">
                <ref name="InstructionRejectionEnum"/>
            </element>
        </interleave>
    </define>
</grammar>
```

4.24. heartbeat-event

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
<include href="published-datatypes.rng"/>
    <start>
        <element name="heartbeat">
            <ref name="HeartBeatType"/>
        </element>
    </start>

<define name="HeartBeatType">
        <element name="accountId">
            <ref name="AccountIdType"/>
        </element>
        <element name="token">
            <text/>
        </element>
    </define>
</grammar>
```

4.25. orderBookStatus-event

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

<include href="published-datatypes.rng"/>

    <start>
        <element name="orderBookStatus">
            <ref name="OrderBookStatusType"/>
        </element>
    </start>

    <define name="OrderBookStatusType">
        <interleave>
            <element name="id">
                <ref name="InstrumentIdType"/>
            </element>
            <element name="status">
                <ref name="InstrumentStatusEnum"/>
            </element>
            <optional>
                <element name="phoneTradingAvailable">
                    <empty/>
                </element>
            </optional>
        </interleave>
    </define>
</grammar>
```

4.26. orderBook2-event

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
<include href="published-datatypes.rng"/>
   <element name="ob2">
     <ref name="StringType"/>
   </element>
</grammar>
```

4.27. published-datatypes

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

<include href="primitives.rng"/>
<include href="public-enumerations.rng"/>

    <define name="PlaceOrderPayload">
        <interleave>
            <element name="quantity">
                <ref name="QuantityType"/>
            </element>
            <optional>
                <element name="stopLossOffset">
                    <ref name="StopOffsetType"/>
                </element>
            </optional>
            <optional>
                <element name="stopProfitOffset">
                    <ref name="StopOffsetType"/>
```

```
                </element>
            </optional>
            <optional>
                <element name="allowUnmatched">
                    <ref name="BooleanType"/>
                </element>
            </optional>
            <optional>
                <element name="price">
                    <ref name="PriceType"/>
                </element>
            </optional>
            <optional>
                <element name="goodUntil">
                    <ref name="GoodUntilEnum"/>
                </element>
            </optional>
        </interleave>
    </define>

<define name="PlaceStopPayload">
        <interleave>
            <element name="instrumentId">
                <ref name="InstrumentIdType"/>
            </element>
            <element name="stopCondition">
                <element name="price">
                    <ref name="PriceType" />
                </element>
            </element>
            <element name="quantity">
                <ref name="QuantityType"/>
            </element>
            <element name="goodUntil">
                <ref name="GoodUntilEnum"/>
            </element>
            <element name="allowUnmatched">
                <ref name="BooleanType"/>
            </element>
        </interleave>
    </define>

    <define name="ValidLoginResponseType">
        <!-- logged in user's username. -->
        <element name="username">
            <ref name="UsernameType"/>
        </element>
        <!-- user's currency -->
        <element name="currency">
            <ref name="CurrencyType"/>
        </element>
        <!-- logged in user's account id. -->
        <element name="accountId">
            <ref name="AccountIdType"/>
        </element>
        <element name="accountType">
            <ref name="AccountTypeType"/>
        </element>
        <element name="productType">
            <ref name="ProductType"/>
        </element>
        <element name="fundingDisallowed">
            <ref name="BooleanType"/>
        </element>
    </define>

    <define name="Subscription">
```

```
        <element name="subscription">
            <choice>
                <!-- The unique identifier of the instrument whose events are required. -->
                <element name="instrument">
                    <ref name="InstrumentIdType"/>
                </element>
                <!-- The unique identifier of the instrument whose price and status events
are required. -->
                <element name="orderBook">
                    <ref name="InstrumentIdType"/>
                </element>
                <!-- The unique identifier of the instrument whose status events are
required. -->
                <element name="orderBookStatus">
                    <ref name="InstrumentIdType"/>
                </element>
                <!-- The from and to currencies. -->
                <element name="exchangeRate">
                    <ref name="ExchangeRateReferenceType"/>
                </element>
                <!-- A type for a generic subscription (not id specific) -->
                <element name="type">
                    <ref name=" "/>
                </element>
            </choice>
        </element>
        <optional>
            <element name="longPollKey">
                <ref name="LongPollKeyType"/>
            </element>
        </optional>
    </define>

    <define name="LongPollKeyType">
        <ref name="StringType1to20"/>
    </define>

<!--Definition of a password. A password can be any valid string.-->
    <define name="PasswordType">
        <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
            <param name="pattern">[-0-9a-zA-
Z!@#$%\^&#x00A3;&amp;*()_+|~=\\`{}\[\]:";'&lt;&gt;?,./]{8,20}</param>
        </data>
    </define>

<define name="LoginAttemptPasswordType">
        <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
    </define>

<define name="UsernameType">
        <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
            <param name="pattern">[0-9a-zA-Z\-_]{1,20}</param>
        </data>
    </define>

<define name="CurrencyType">
        <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
            <param name="pattern">[A-Z]{3}</param>
        </data>
    </define>

<define name="QuantityType">
        <data type="decimal" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
            <param name="totalDigits">19</param>
            <param name="fractionDigits">2</param>
        </data>
    </define>
```

```
<define name="StopOffsetType">
        <data type="decimal" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
                 <param name="minExclusive">0</param>
            <param name="fractionDigits">5</param>
        </data>
    </define>

<define name="ExchangeRateReferenceType">
        <interleave>
            <element name="from">
                <ref name="CurrencyType"/>
            </element>
            <element name="to">
                <ref name="CurrencyType"/>
            </element>
        </interleave>
    </define>

<define name="AccountIdType">
        <ref name="PositiveNonZeroLongType"/>
</define>

    <define name="PriceType">
        <ref name="DecimalType"/>
    </define>

<!--Definition of an instrument identifier.-->
    <define name="InstrumentIdType">
        <ref name="PositiveNonZeroLongType"/>
    </define>

    <!--A representation of a price and quantity.-->
    <define name="PricePointType">
        <!--The price portion.-->
        <element name="price">
            <ref name="PriceType"/>
        </element>
        <!--The quantity portion.-->
        <element name="quantity">
            <ref name="QuantityType"/>
        </element>
    </define>

    <define name="ExecutionIdType">
        <ref name="PositiveNonZeroLongType"/>
    </define>

    <define name="OrderIdType">
        <ref name="StringType1to16"/>
    </define>

    <define name="AccountStateType">
        <interleave>
            <!--The unique identifier of the account.-->
            <element name="accountId">
                <ref name="AccountIdType"/>
            </element>
            <!--The amount of money credited to the account.-->
            <element name="balance">
                <ref name="PriceType"/>
            </element>
            <!--The amount of money available for trading.-->
            <element name="availableFunds">
                <ref name="PriceType"/>
            </element>
            <!--The amount of money available for withdraw.-->
```

```
            <element name="availableToWithdraw">
                <ref name="PriceType"/>
            </element>
             <element name="unrealisedProfitAndLoss">
               <ref name="PriceType"/>
             </element>
             <element name="margin">
                <ref name="PriceType"/>
             </element>
            <optional>
                <element name="onMarginCallExpiryTimestamp">
                    <ref name="DateTimeType"/>
                </element>
            </optional>
            <element name="wallets">
                <zeroOrMore>
                    <element name="wallet">
                        <element name="currency">
                            <ref name="CurrencyType"/>
                        </element>
                        <element name="balance">
                            <ref name="PriceType"/>
                        </element>
                    </element>
                </zeroOrMore>
            </element>
            <!-- The position of this state in the life span of the account. -->
            <element name="active">
                <ref name="BooleanType"/>
            </element>
        </interleave>
    </define>

<define name="SuppliedInstructionIdType">
        <ref name="PositiveNonZeroLongType"/>
    </define>


<define name="InstructionIdType">

       <ref name="LongType"/>
    </define>

    <define name="MarketClosePriceType">
        <element name="price">
            <choice>
                <ref name="PriceType"/>
                <empty/>
            </choice>
        </element>
        <element name="timestamp">
            <choice>
                <ref name="DateTimeType"/>
                <empty/>
            </choice>
        </element>
    </define>

</grammar>
```

4.28 errors

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <define name="WarningEnum">
        <choice>
            <value>ACCOUNT_TYPE_UNAVAILABLE</value>
```

```
            <value>AUTHORIZATION_FAILED</value>
            <value>INVALID_PASSWORD</value>
            <value>NOT_LOGGED_IN</value>
            <value>TEMPORARY_SYSTEM_ERROR</value>
            <value>UNAUTHENTICATED</value>
            <value>UNAUTHORISED</value>
            <value>UNKNOWN_ACCOUNT_ID</value>
            <value>UNSUPPORTED_PROTOCOL_VERSION</value>
            <value>VALIDATION_ERRORS</value>
              <value>INVALID_FIELD</value>
        </choice>
    </define>

    <define name="InstructionRejectionEnum">
        <choice>
            <value>NONE</value>
            <value>UNKNOWN</value>
            <value>INTERNAL</value>
            <value>ACCOUNT_DOES_NOT_EXIST</value>
            <value>ACCOUNT_TYPE_NOT_SET</value>
            <value>INSTRUMENT_DOES_NOT_EXIST</value>
            <value>INSTRUMENT_NOT_OPEN</value>
            <value>PRICE_NOT_VALID</value>
            <value>QUANTITY_NOT_VALID</value>
            <value>NO_QUANTITY_TO_CANCEL</value>
            <value>EXPOSURE_CHECK_FAILURE</value>
            <value>DUPLICATE_ORDER</value>
            <value>UNKNOWN_ORDER</value>
            <value>INSUFFICIENT_LIQUIDITY</value>
            <value>ORDER_CAPACITY_NOT_SET</value>
            <value>ORDER_TYPE_INVALID</value>
            <value>SERVICE_UNAVAILABLE</value>
            <value>STOP_PROFIT_OFFSET_INVALID</value>
            <value>STOP_LOSS_OFFSET_INVALID</value>
            <value>STOP_WOULD_BE_TRIGGERED_IMMEDIATELY</value>
            <value>STOP_CANCEL_NOT_ALLOWED</value>
            <value>ORDER_INVALID</value>
            <value>ORDER_ID_INVALID</value>
            <value>NO_QUANTITY_TO_CLOSE</value>
            <value>STOP_LOSS_OFFSET_REQUIRED</value>
            <value>ALREADY_CANCELLED</value>
            <value>WOULD_GUARANTEE_LOSS</value>
            <value>INVALID_MARKET_DEPTH</value>
            <value>INVALID_UPDATE_TYPE</value>
            <value>INVALID_ORDER_INSTRUCTION</value>
            <value>UNAUTHORISED</value>
            <value>CROSSED_PRICES</value>
            <value>INSTRUMENT_SUSPENDED</value>
            <value>TEMPORARY_SUSPENSION</value>
            <value>OUTSIDE_VOLATILITY_BAND</value>
            <value>OUTSIDE_ALLOWED_PRICE_RANGE</value>
            <value>STOP_PROFIT_WORSE_THAN_MARKET_PRICE</value>
            <value>MAXIMUM_POSITION_EXCEEDED</value>
            <value>UNSUPPORTED_MARKET_DATA_TYPE</value>
            <value>ACCESS_DENIED</value>
            <value>DUPLICATE_PRICE</value>
        </choice>
    </define>

</grammar>
```

4.29. tfx-request

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

    <start>
        <!-- A request -->
```

```
        <element name="req">
            <!-- request id -->
            <text/>
            <!-- request body -->
            <element name="body">
                <ref name="RequestBody"/>
            </element>
        </element>
    </start>

    <!--Base definition of a request message body.-->
    <define name="RequestBody">
        <empty/>
    </define>

</grammar>
```

4.30. tfx-response

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
<include href="errors.rng"/>

    <start>
        <!-- A response -->
        <element name="res">
            <choice>
                <group>
                    <element name="header">
                        <element name="status">
                            <!--The request was successful.-->
                            <value>OK</value>
                        </element>
                    </element>
                    <element name="body">
                        <ref name="ResponseBody"/>
                    </element>
                </group>
                <group>
                    <element name="header">
                        <element name="status">
                            <choice>
                                <!--The request was unsuccessful.-->
                                <value>WARN</value>
                                <!--There was a system problem that caused an error.-->
                                <value>ERROR</value>
                            </choice>
                        </element>
          <!--Optional list of warnings encountered during processing of the request.-->
                        <optional>
                            <element name="warnings">
                                <ref name="WarningsType"/>
                            </element>
                        </optional>
                    </element>
                    <element name="body">
                        <optional>
                            <ref name="ResponseBody"/>
                        </optional>
                    </element>
                    <optional>
                        <!-- Authorization status -->
                        <element name="auth">
                            <ref name="AuthorisationEnum"/>
                        </element>
                    </optional>
```

```
                </group>
            </choice>
        </element>
    </start>

    <!--Base definition of a request message body.-->
    <define name="ResponseBody">
        <empty/>
    </define>

<!--Definition of a warning messages that can be provided in a response that has the status
WARN.-->
    <define name="WarningsType">
        <zeroOrMore>
            <!--There may be many individual warnings associated with a response.-->
            <element name="warning">
                <!--Identifies the field name in the request that this warning is associated
with if applicable.-->
                <element name="fieldName">
                    <choice>
                        <text/>
                        <empty/>
                    </choice>
                </element>
                <!--The warning message.-->
                <element name="message">
                    <choice>
                        <!--Either a plain text English warning message...-->
                        <text/>
                        <!--or a 'warning code' from one of the following enumerations: -->
                        <ref name="WarningEnum"/>
                        <ref name="InstructionRejectionEnum"/>
                    </choice>
                </element>
            </element>
        </zeroOrMore>
    </define>

    <!--Enumeration of authorisation failure reasons.-->
   <define name="AuthorisationEnum">
       <choice>
           <!--The user was unauthorised to perform that operation.-->
           <value>UNAUTHORISED</value>
           <!--The user was not authenticated. This maybe because they have not successfully
logged in or because their session has expired.-->
           <value>UNAUTHENTICATED</value>
           <!--The user's session has expired, or they have been logged off for security
reasons.-->
           <value>SESSION_EXPIRED</value>
       </choice>
   </define>
</grammar>
```

4.31. enumerations

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

<!--Definition of the domain of instrument statuses.-->
    <define name="InstrumentStatusEnum">
        <choice>
            <!--The instrument has been newly created.-->
            <value>New</value>
            <!--The is closed and waiting to be opened.-->
            <value>Closed</value>
            <!--The instrument is temporarily not accepting orders.-->
```

```
            <value>Suspended</value>
            <!--The instrument is no longer trading.-->
            <value>Withdrawn</value>
            <!--The instrument and all orders upon it have been made void.-->
            <value>Settled</value>
            <!--The instrument is accepting orders.-->
            <value>Opened</value>
        </choice>
    </define>

</grammar>
```

4.32. public-enumerations

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

    <!--Definition of the lifespan of an order. The use of GoodUntil has been deprecated in
favour of TimeInForce-->
    <define name="GoodUntilEnum">
        <choice>
            <value>Immediate</value>
            <value>Cancelled</value>
        </choice>
    </define>

<!-- The supported Time In Force Policies -->
<define name="TimeInForceEnum">
       <choice>
             <value>GoodTilCancelled</value>
             <value>GoodForDay</value>
             <value>ImmediateOrCancel</value>
             <value>FillOrKill</value>
       </choice>
</define>

<define name="OrderTypeEnum">
        <choice>
            <value>MARKET</value>
            <value>STOP_MARKET_ORDER</value>
            <value>STOP_MARKET_OPENING_ORDER</value>
            <value>PRICE_LIMIT</value>
            <value>STOP_COMPOUND_PRICE_LIMIT</value>
            <value>STOP_COMPOUND_MARKET</value>
            <value>CLOSE_OUT_ORDER</value>
            <value>STOP_LOSS_ORDER</value>
            <value>STOP_PROFIT_ORDER</value>
        </choice>
    </define>

   <define name="AccountTypeType">
        <choice>
            <value>STANDARD_TRADER</value>
        </choice>
    </define>
    <define name="ProductType">
        <choice>
            <value>CFD_DEMO</value>
            <value>CFD_LIVE</value>
        </choice>
    </define>

    <define name="SubscriptionType">
        <choice>
            <value>account</value>
           <value>order</value>
```

```
            <value>position</value>
        </choice>
    </define>


    <!--Enumeration of login failure codes.-->
    <define name="LoginFailureEnum">
        <choice>
            <!--The account was closed and therefore the user could not log in.-->
            <value>ACCOUNT_CLOSED</value>
    <!--The account is locked or the user has messages waiting for them on the web site.-->
            <value>ACCOUNT_LOCKED</value>
            <!--The account is suspended.-->
            <value>ACCOUNT_SUSPENDED</value>
            <!--The user name or password were not valid.-->
            <value>BAD_CREDENTIALS</value>
  <!--A temporary system error prevented the log in being processed. The log in can be
retrived later.-->
            <value>TEMPORARY_SYSTEM_ERROR</value>
            <!-- The service is not currently available for use. -->
            <value>SERVICE_UNAVAILABLE</value>
            <!-- The temporary password has expired, and can no longer be used -->
            <value>PASSWORD_EXPIRED</value>
<!-- The specified protocol version did not match the current supported version -->
            <value>PROTOCOL_VERSION_INVALID</value>
            <!-- The xxxx -->

        </choice>
    </define>

<!-- from old common/enumerations.rng -->
    <!--Definition of the domain of instrument statuses.-->
    <define name="InstrumentStatusEnum">
        <choice>
            <!--The instrument has been newly created.-->
            <value>New</value>
            <!--The instrument is accepting orders.-->
            <value>Opened</value>
            <!--The instrument is temporarily not accepting orders.-->
            <value>Suspended</value>
            <!--The is closed and waiting to be opened.-->
            <value>Closed</value>
            <!--The trades upon the instrument have been settled.-->
            <value>Settled</value>
            <!--The instrument is no longer trading.-->
            <value>Withdrawn</value>
        </choice>
    </define>

</grammar>
```

4.33. primitives

```
<?xml version="1.0"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

    <define name="NonEmptyStringType">
        <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
            <param name="minLength">1</param>
            <param name="whiteSpace">collapse</param>
        </data>
    </define>

    <define name="StringType">
        <data type="string" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
    </define>
```

```
    <define name="DecimalType">
        <data type="decimal" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
    </define>

    <define name="BooleanType">
        <data type="boolean" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
    </define>

    <define name="DateTimeType">
        <data type="dateTime" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
    </define>

    <define name="PositiveNonZeroLongType">
        <data type="long" datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
            <param name="minInclusive">1</param>
        </data>
    </define>
</grammar>
```

Website: **www.LMAXtrader.com**

Email address: **info@LMAXtrader.com**

Telephone: **0333 700 1000 (+44 203 192 2555)**