Document title

EURONEXT DERIVATIVES MARKETS

Document type or subject

XDP Client Specifications

Version number

2.2.8a

Date

26 Jul 2016

Number of pages

89

Author

Euronext

PREFACE

PURPOSE

XDP (for "eXchange Data Publisher") is a messaging protocol between Euronext and its clients.
It allows them to choose their Market Data Services based on their requirements and trading profile. This also allows them to optimise bandwidth usage on their circuits.

The purpose of this document is to describe each message that Euronext can possibly disseminate via XDP.

TARGET AUDIENCE

This document should be read by Feed Handler developers using the Euronext Derivatives XDP.

WHAT'S NEW?

The following lists only the most recent modification made to this revision/version. For the Document History table, see the Appendix.

VERSION NO.	DATE	CHANGE DESCRIPTION
2.2.8a	26 Jul 2016	Clarification of the use of the 4 update types 1, 2, 3 and 4 in the Market Update – 702 Message.

ASSOCOIATED DOCUMENTS

Please visit https://www.euronext.com/en/it-documentation.

SUPPORT

For any questions about this specification please contact :

- Euronext Market Solutions: MarketSolutions@euronext.com
- Customer Technical Support Group: ctsg@euronext.com

FURTHER INFORMATION

- For additional product information, visit: https://www.euronext.com/market-data
- For updated capacity figures and details of IP addresses, visit: https://www.euronext.com/en/market-data/products-by-type

CONTENTS

1. EURONEXT DERIVATIVES XDP MARKET DATA SOLUTION

1.1 INTRODUCTION

1.1.1 Euronext Derivatives Market Data Overview

The Euronext Derivatives XDP feed provides high-speed, real-time market data for Euronext Derivatives markets. Euronext Derivatives market data is only available via the SFTI® network.

The data feed has the following high-level features:

- Multicast technology
- High Availability
- Ultra-low latency
- Reliable network solution
- High level of scalability
- Access to a wide range of European market data sets

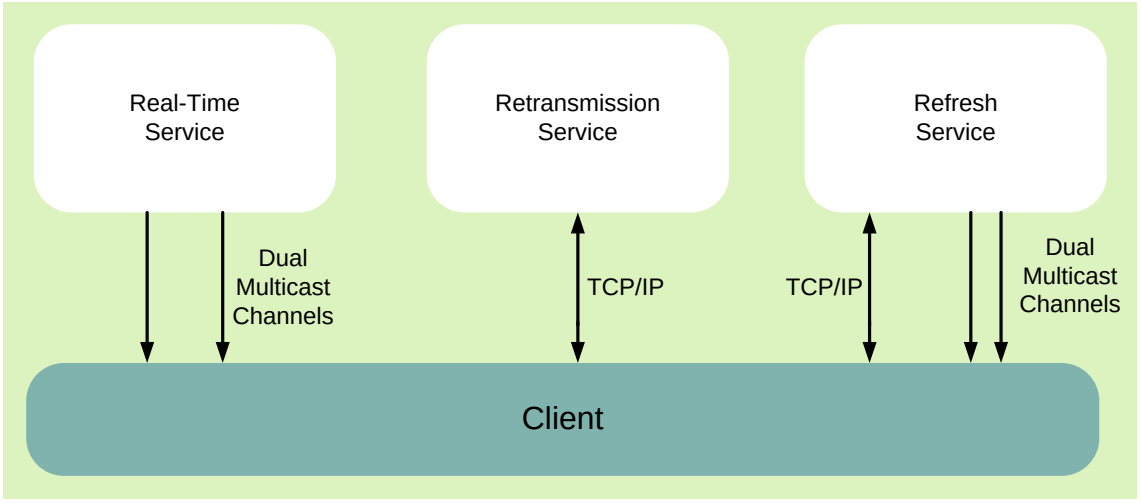
This document provides detailed information about the features of the feed, to support the development of client applications by Members, Independent Software Vendors and Quote Vendors.

The following chapters of this specification provide details that are specific to each of the Euronext Derivatives market data sets, including formats for each message type.

1.1.2 Access to Market Data

Clients connect to multicast addresses for the real-time market data messages and refresh data, and can also connect to a TCP/IP server for packet retransmissions. Requests for retransmission and refresh are performed via TCP/IP.

Figure 1 Access to Market Data



1.1.3 Multicast Streams

Dual multicast streams are made available for the distribution of real-time and refresh data.

Data is provided across multiple multicast streams. Users should refer to the Euronext Derivatives Market Data Service Breakdown for information on what data is carried in each multicast group.

Clients should connect to multicast stream(s) for which they require data.

1.1.4 TCP/IP Channels

TCP/IP channels are made available for retransmission and refresh requests and responses.

The user can choose to disconnect/reconnect in between requests. However if choosing to remain connected, the user needs to respond to heartbeat requests from the exchange.

2. CONFORMANCE TESTING – TEST AND PRODUCTION FEED CONFIGURATION

2.1 CONFORMANCE TESTING

IP Addresses for the test environment for conformance testing can be found here:

https://euronext.com/en/it-documentation/market-data?quicktabs_171=2#quicktabs-171

Once clients have passed conformance testing, production IP addresses can be found in the Production Configuration file ("Euronext_derivatives_market_data_services_xdp_production_environment_v2.1.pdf") through the Euronext https://euronext.com/en/it-documentation/market-data

2.2 REAL-TIME, RETRANSMISSION AND REFRESH - PRODUCTION FEED CONFIGURATION

The UDP/IP configuration details for the Euronext Derivatives XDP feed are provided in a separate file. This file includes primary and secondary data center source IPs, multicast group/port details for the real-time and refresh feeds.

TCP/IP configuration details for the Euronext Derivatives XDP feed will be provided in a separate file. This file will include IP address/port details for the retransmission and refresh services.

2.3 PRODUCTION TIMETABLE

The following table is the overview of the main daily event generating activity and indicative time on the Euronext Derivatives XDP feed.

Table 1 Production Timetable

EVENT	TIME (AMSTERDAM/PARIS-LOCAL)	COMMENT
FIXML standing data files available	03:30	FTP site
Standing data sent on multicast channels	06:00	722, 732 messages
Pre-Open	07:00	AEX and CAC index future (first equities contract to open and last to close)
Open	08:00	
Pre-Close	21:58	
Close	22:00	
System Close	23:30	

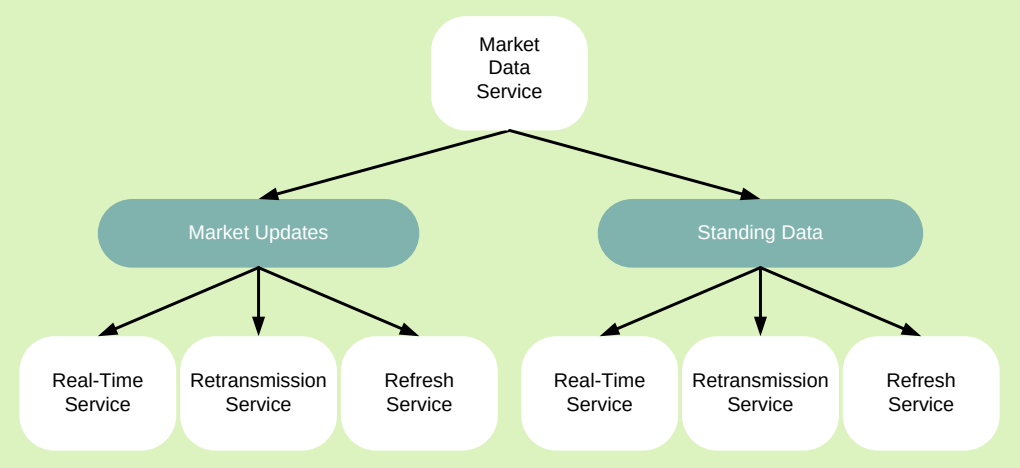
3. EURONEXT DERIVATIVES MARKET DATA SERVICE BREAKDOWN

3.1.1 Euronext Derivatives Market Data Service Breakdown

The diagram below provides an overview of how a market data service is broken down into real-time, retransmission and refresh services for market updates and standing data.

Separate multicast or TCP/IP addresses are provided for each of these interfaces.

Figure 2 Market Data Service Breakdown



3.1.2 Euronext Derivatives Service Types

The Euronext Derivatives market data feed provides a number of different service types. The message types that are disseminated in each service type are described in the following table:

Table 2 Service Types

SERVICE TYPE	DESCRIPTION	MESSAGE TYPES
Level 1	Real-Time market updates without order book	702 Market Update (all update types except 3 and 4) 712 Settlement Prices 741 Market Availability 752 Market Status 761 Exchange Message
Level 1 + 2	Real-Time market updates with order book	702 Market Update 712 Settlement Prices 741 Market Availability 752 Market Status 761 Exchange Message
Standing Data	Start of day standing data plus intraday updates for new option strikes and strategy creations	722 Outright Standing Data 732 Strategy Standing Data
Value Add + Open Interest	Value-Added parameters plus open interest information	772 Value-Added Parameters

SERVICE TYPE	DESCRIPTION	MESSAGE TYPES
		782 Open Interest
AtomX	AtomX trade reports, settlements (For Atomx -traded instruments only)	702 Market Update (update type 39 and 5) 712 Settlement Prices 722 Outright Standing Data

3.1.3 Euronext Derivatives Product Groups

The Euronext Derivatives market data feed is split into a number of product groups. The following table shows the products that are contained within each product group.

Table 3 Product Groups

PRODUCT GROUP	PRODUCTS
Euronext Derivatives Equity & Index Derivatives	CAC 40 Index Futures CAC 40 Dividend Index Futures AEX Index® Futures AEX Dividend Index Futures BEL 20 Index Futures PSI 20 Futures CAC 40 Index Options (PXA) CAC 40 Index Options (PXL) AEX Index® Options AEX Index® Weekly Options BEL 20 Index Options Paris Stock Options Amsterdam Stock Options Brussels Stock Options USFs (Portugal)
Euronext Currency Derivatives	US Dollar / Euro Futures US Dollar / Euro Options Euro / US Dollar Futures Euro / US Dollar Options
Euronext Commodity Derivatives	Corn Futures Corn Options Rapeseed Futures Rapeseed Options Rapeseed Oil Futures Rapeseed Oil Options Milling Wheat Futures Milling Wheat Options Malting Barley Futures Malting Barley Options Skimmed Milk Powder Futures

PRODUCT GROUP	PRODUCTS
	Sweet Whey Food Grade Powder Futures Unsalted Lactic Butter Futures
Amsterdam - Equity Derivatives	Amsterdam Stock Options
Amsterdam - Index Futures	AEX Index® Futures AEX Dividend Index Futures
Amsterdam - Index Options	AEX Index® Options AEX Index® Weekly Options AEX Index® Daily Options
Paris - Equity Derivatives	Paris Stock Options
Paris - Index Futures	CAC 40 Index Futures CAC 40 Dividend Index Futures FTSEurofirst 80 Index Futures (Paris) FTSEurofirst 100 Index Futures (Paris)
Paris - Index Options	CAC 40 Index Options (PXA) CAC 40 Index Options (PXL)
Brussels - Equity & Index Derivatives	BEL 20 Index Futures BEL 20 Index Options Brussels Stock Options
Lisbon - Equity & Index Derivatives	PSI 20 Futures USFs (Portugal)

3.1.4 Euronext Derivatives Service IDs

The following different multicast groups (known as Service IDs) are made available and are defined as follows.

Clients have two options for accessing Equity & Index Level 1+2 data:

- Subscribe to Service ID 702 which contains all of the level 1 and 2 data for equity and index derivatives
- Subscribe to all of Service IDs 721 to 723 and 728 to 732 which contain the same data but split into more granular streams

Clients will need to ensure that the option they select corresponds to the services selected on the SFTI order form.

Table 4 Real-time Service IDs

SERVICE ID	PRODUCT GROUP	SERVICE TYPE
701	Equity & Index Derivatives	Level 1
751	Equity & Index Derivatives	Standing Data (for Level 1)
702	Equity & Index Derivatives	Level 1+2
752	Equity & Index Derivatives	Standing Data (for Level 1+2)

SERVICE ID	PRODUCT GROUP	SERVICE TYPE
703	Currency Derivatives	Level 1
753	Currency Derivatives	Standing Data (for Level 1)
704	Currency Derivatives	Level 1+2
754	Currency Derivatives	Standing Data (for Level 1+2)
705	Commodity Derivatives	Level 1
755	Commodity Derivatives	Standing Data (for Level 1)
706	Commodity Derivatives	Level 1+2
756	Commodity Derivatives	Standing Data (for Level 1+2)
721	Amsterdam - Equity Derivatives	Level 1+2
771	Amsterdam - Equity Derivatives	Standing Data
722	Amsterdam - Index Futures	Level 1+2
772	Amsterdam - Index Futures	Standing Data
723	Amsterdam - Index Options	Level 1+2
773	Amsterdam - Index Options	Standing Data
728	Paris - Equity Derivatives	Level 1+2
778	Paris - Equity Derivatives	Standing Data
729	Paris - Index Futures	Level 1+2
779	Paris - Index Futures	Standing Data
730	Paris - Index Options	Level 1+2
780	Paris - Index Options	Standing Data
731	Brussels - Equity & Index Derivatives	Level 1+2
781	Brussels - Equity & Index Derivatives	Standing Data
732	Lisbon - Equity & Index Derivatives	Level 1+2
782	Lisbon - Equity & Index Derivatives	Standing Data
736	Currency Derivatives	Level 1+2
786	Currency Derivatives	Standing Data
711	Euronext Equity & Index Derivatives	VAP + Open Interest
712	Euronext Currency Derivatives	VAP + Open Interest
713	Euronext Commodity Derivatives	VAP + Open Interest
715	Equity & Index Derivatives	AtomX

Table 5 Refresh Service IDs

SERVICE ID	PRODUCT GROUP	SERVICE TYPE
801	Equity & Index Derivatives	Level 1
851	Equity & Index Derivatives	Standing Data (for Level 1)
802	Equity & Index Derivatives	Level 1+2
852	Equity & Index Derivatives	Standing Data (for Level 1+2)
803	Currency Derivatives	Level 1
853	Currency Derivatives	Standing Data (for Level 1)
804	Currency Derivatives	Level 1+2
854	Currency Derivatives	Standing Data (for Level 1+2)
805	Commodity Derivatives	Level 1
815	Equity & Index Derivatives	AtomX
855	Commodity Derivatives	Standing Data (for Level 1)
806	Commodity Derivatives	Level 1+2
856	Commodity Derivatives	Standing Data (for Level 1+2)
821	Amsterdam - Equity Derivatives	Level 1+2
871	Amsterdam - Equity Derivatives	Standing Data
822	Amsterdam - Index Futures	Level 1+2
872	Amsterdam - Index Futures	Standing Data
823	Amsterdam - Index Options	Level 1+2
873	Amsterdam - Index Options	Standing Data
828	Paris - Equity Derivatives	Level 1+2
878	Paris - Equity Derivatives	Standing Data
829	Paris - Index Futures	Level 1+2
879	Paris - Index Futures	Standing Data
830	Paris - Index Options	Level 1+2
880	Paris - Index Options	Standing Data
831	Brussels - Equity & Index Derivatives	Level 1+2
881	Brussels - Equity & Index Derivatives	Standing Data
832	Lisbon - Equity & Index Derivatives	Level 1+2
882	Lisbon - Equity & Index Derivatives	Standing Data
836	Currency Derivatives	Level 1+2

SERVICE ID	PRODUCT GROUP	SERVICE TYPE
886	Currency Derivatives	Standing Data
811	Euronext Equity & Index Derivatives	VAP + Open Interest
812	Euronext Fixed Income Derivatives	VAP + Open Interest
813	Euronext Commodity Derivatives	VAP + Open Interest

For any question on regard to SFTI network please contact Client Coverage Center team :

- Email : ccc@euronext.com

4. EURONEXT DERIVATIVES MARKET DATA PROCESSING INFORMATION

4.1 REAL-TIME MARKET DATA

Real-time market data is message-based over the UDP IP protocol with FAST-optimized, fixed-length binary and ASCII fields.

It uses the push-based publishing model. This means that data is published based on its availability. Once an update is available, it is published to the appropriate multicast group.

For capacity reasons, market data is split across a number of multicast groups organized into predefined data sets.

Each multicast group delivers a set of data for a certain market segment.

The client application is responsible for issuing multicast subscriptions to one or more of the multicast groups assigned to each product.

The process of subscribing to a multicast group ID is also known as 'joining' a multicast group. Upon session termination, the client's host system should issue an 'unjoin' message. This terminates delivery of data to that host's local network. If a client application terminates without issuing an 'unjoin' message, the network eventually issues a 'timeout' for the multicast group subscription that automatically terminates delivery of the multicast packets to the host's local network.

The 'join' and 'unjoin' processes are standard functions. No specific instructions are provided here, as they are specific to the user's operating system and programming language.

4.1.1 Packet Structure

There are two types of packets transmitted as part of this protocol:

- **Control packets** These do not contain data, they allow conversing parties to exchange session-specific information (for example, 'reset sequence number').
- **Data packets** These are product-specific.

All packets sent on the Euronext Derivatives XDP feed have a common packet header followed by one or more messages (with the exception of some technical packets that do not contain any messages).

The packet header format is the same for all packets, and contains packet length, number of messages within the packet, packet sequence number, and so forth.

The format of each message in the packet depends on message type, however each message starts with message size and message type.

The maximum length of a packet is 1400 bytes.

A packet only ever contains complete messages. A single message never straddles multiple packets.

The message size never exceeds the maximum packet length (less the packet header size).

PACKET HEADER	MESSAGE 1	MESSAGE 2	...	MESSAGE N

The packet header provides information including the total packet length, a packet sequence number, the number of messages within the packet and a send timestamp. The format of each message within a packet varies according to message type.

4.1.2 Packet Header Format

All messages contain a common packet header. The table below describes the header field. The design is intended to minimize the development burden on behalf of clients. This means that all clients may implement line-level protocol processing once, and then only need to develop parsing algorithms for their choice of message.

Table 6 Packet Header Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
PacketLength	0	2	Binary Integer	Length of the packet including the 16-byte packet header
PacketType	2	2	Binary Integer	Identifier for the type of data contained in the packet: ■ '1' - Sequence Number Reset ■ '2 '- Heartbeat Message ■ '10' - Retransmission Response message ■ '20' - Retransmission Request Message ■ '22' - Refresh Request message ■ '23' - Refresh Response message ■ '24' - Heartbeat Response Message ■ '799' - Generic Derivatives Message
PacketSeqNum	4	4	Binary Integer	This field contains the packet sequence number. It is unique for each broadcast stream (multicast group) and is used for gap detection. It increases monotonically and is reset to 1 at the beginning of each trading day. Note that heartbeats inherit their sequence number from the last market data packet or packet sequence number reset packet.
SendTime	8	4	Binary Integer	Timestamp in milliseconds indicating the packet broadcast time. The number represents the number of milliseconds since midnight of the last Sunday 00:00 UTC. For real-time and refresh packets this represents the time that the packet was broadcast. For retransmission packets this represents the time that the packet was originally broadcast on the real-time service.
ServiceID	12	2	Binary	Numeric value identifying the broadcast

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
			Integer	stream. Possible values are described in Feed Configuration descriptions
DeliveryFlag	14	1	Binary Integer	Indicates the delivery method: ■ '0' - Real Time message (uncompressed) ■ '1' - Refresh message (uncompressed) ■ '2' - Retransmission message (uncompressed) ■ '8' - Real Time message (FAST optimized) ■ '9' - Refresh message (FAST optimized) ■ '10' - Retransmission message (FAST optimized)
NumberMsgEntries	15	1	Binary Integer	The number of messages that are contained within the packet.

4.1.3 Packet Sequence Numbers

All messages conform to the line-level sequencing. Each channel has its own packet sequence number. Clients can use packet sequence numbers to determine the following:

- Missing (gapped) packets
- Unordered packets
- Duplicate packets

4.1.4 Packet Sequence Number Reset Message

This message is sent to 'reset' the Packet Sequence Number at start of day, in response to failures, and so forth. Note that this message contains a valid sequence number. The message contains one field and the format is shown below.

Table 7 Packet Sequence Number Reset Message Field

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
NextSeqNumber	16	4	Binary Integer	Contains the packet sequence number value that the client should expect in the immediately succeeding data packet. Note that this packet will contain its own valid packet sequence number in the header portion of the message.

4.1.5 Packet Sequence Number Reset Processing Notes

Packet Sequence numbers normally begin at one (1) and increase one by one with each subsequent packet. There are two scenarios where the packet sequence number is reset (besides the start of day):

- If the value should exceed the maximum value that the SeqNum field may contain, it will be reset to one (1).
- If the system fails and it recovers, it sends a Packet Sequence Number reset message. The PacketSeqNum field of that packet will be set to one (1) and the NextSeqNumber field will be set to two (2).

Note that the packet sequence number reset is always sent as the first message of the day. The Packet Sequence Number Reset message is never sent as part of a retransmission.

4.2 DETECTING AND RECOVERING MISSED DATA

4.2.1 Introduction

UDP is an 'unreliable' protocol and therefore may drop packets. All multicast data is provided over dual channels (line A and line B).

The Euronext Derivatives XDP feed provides three different mechanisms for recovering missed data:

- Line arbitration – using dual multicast channels
- Retransmission Server – recovery of a limited number of packets
- Refresh Server – snapshot of current market state

These mechanisms should be used as follows:

Table 8 Recovery Mechanisms

EVENT	ACTION
Packet lost on one of the two lines	Try to recover data from the other line with a configurable timeout
Dropped packet(s) on both line A and line B	Recover dropped packet(s) from the Retransmission Server
Late start up or extended intraday outage	Request a refresh of the current market state and then continue with real-time messages

4.2.2 Line Arbitration

Real-time market data is made available on two different multicast groups. This offers clients the possibility to set up more than one receiving system processing the same data. In the event of a client system failure, the backup client system should continue to process the real-time data sent on the second multicast group.

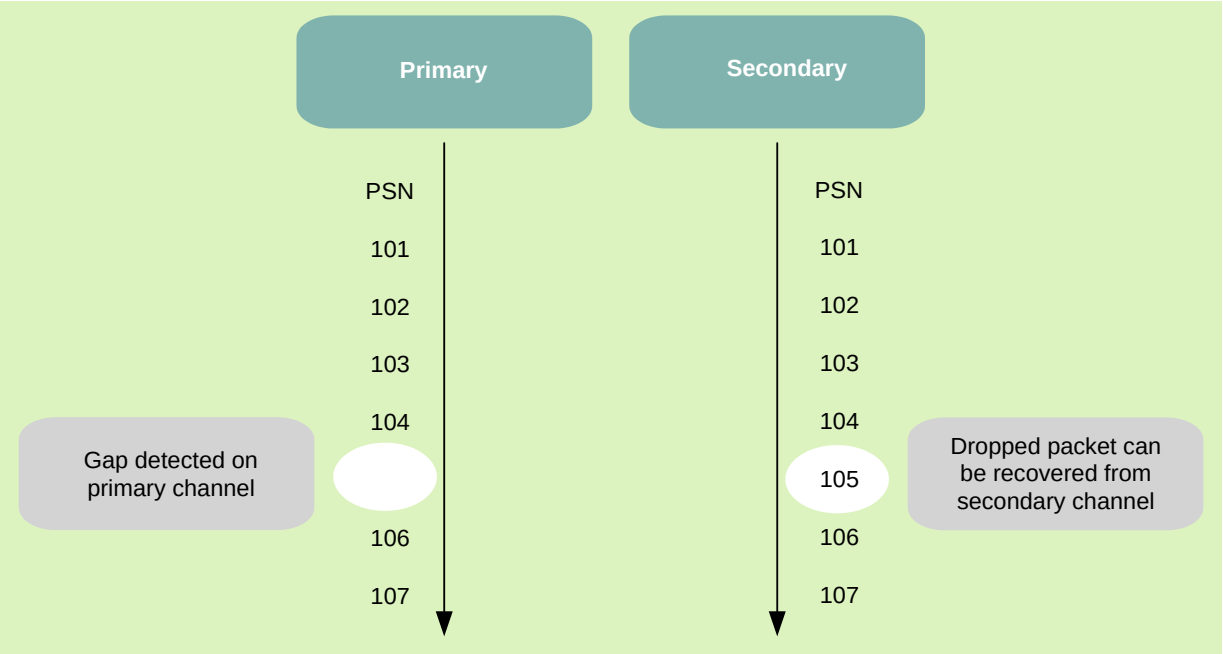
Client applications should check the Packet Sequence Number (PSN) for every packet received. PSNs are unique and increase monotonically for each service.

Line A and line B are identical in terms of:

- Packet contents
- PSNs
- Sequence in which packets are sent

Client applications should listen to both channels in real-time. Clients should look at packets coming from both lines and process the ones that arrive first, regardless of whether they came from line A or line B. It is advisable to apply the 'first come – first served' rule.

Figure 3 Packet Sequence Numbers



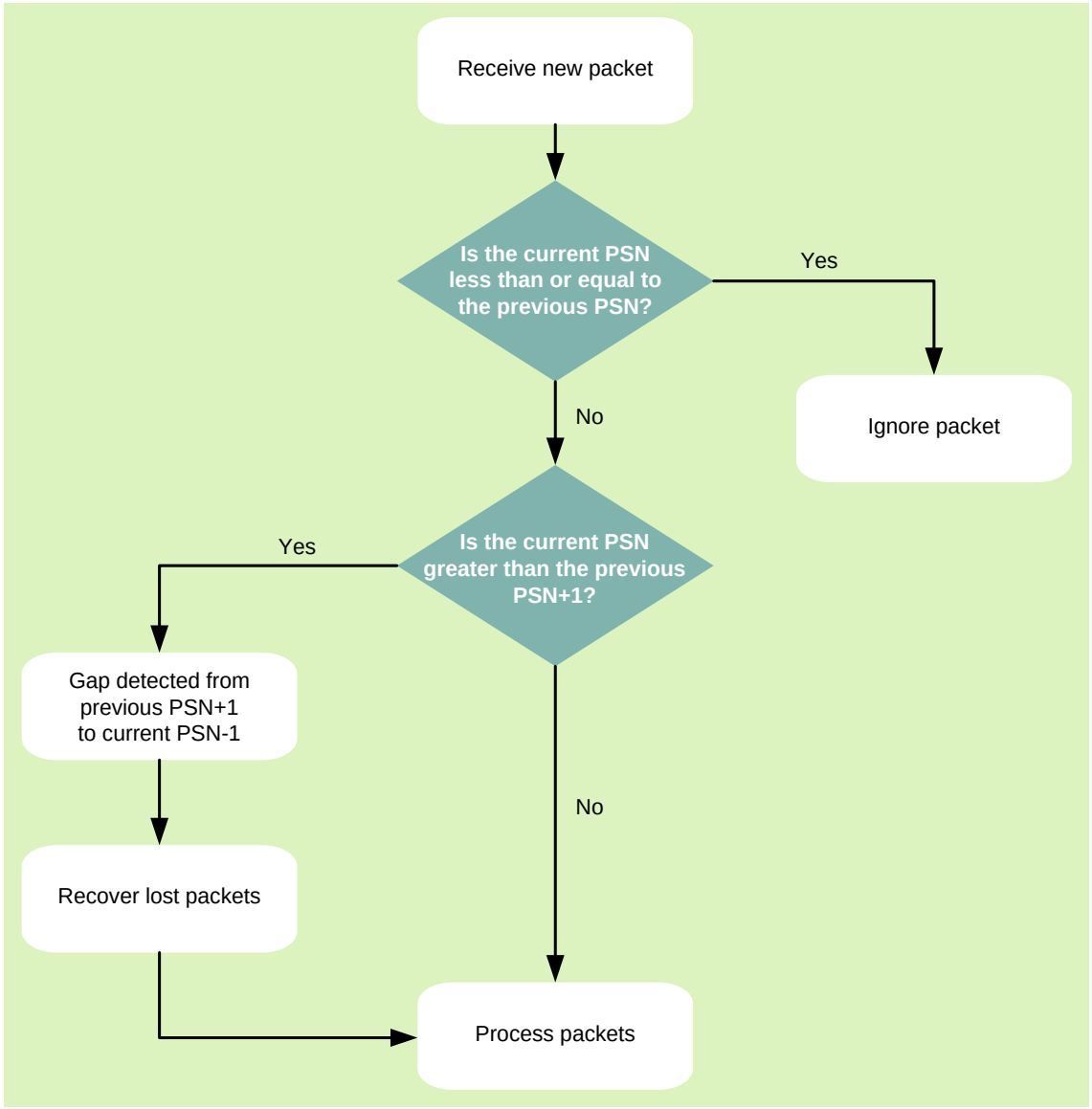
4.2.3 Gap Detection

The Euronext Derivatives XDP feed provides a unique, sequential packet sequence number for each multicast channel. This allows recipients to identify 'gaps' in the message sequence and, if appropriate, reconcile them 'locally' with an alternate channel or request retransmission of the missing/corrupted data packet.

Each packet has a Packet Sequence Number (PSN). PSNs start at one (1) and increase one-by-one and without gaps with each subsequent packet. Users should use the PSN to detect gaps in the transmission of messages.

The following diagram illustrates how the PSN should be used to detect gaps in the feed.

Figure 4 Gap Detection



4.3 RETRANSMISSION

4.3.1 Retransmission Functionality

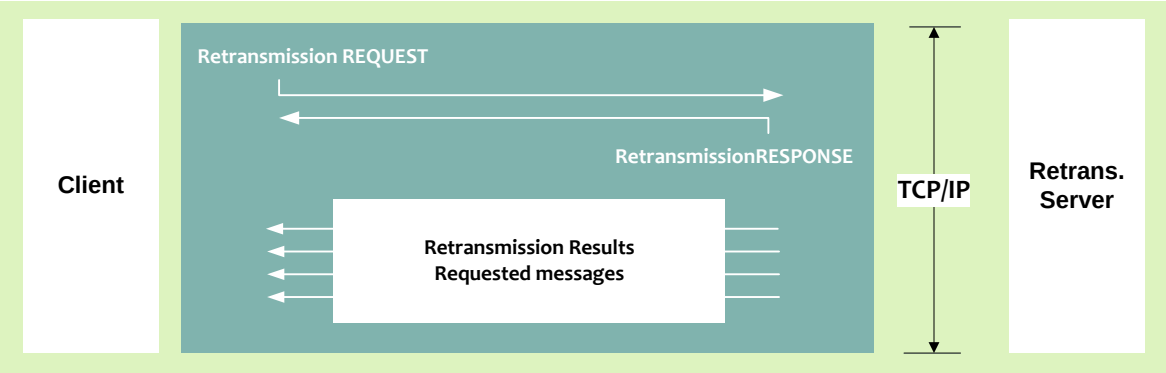
The retransmission functionality is designed to allow the user to recapture a small number of missed packets.

It is not intended that clients use the retransmission functionality to recover data after long outages or on late start up. Accordingly, the number of packets that the user can request is limited. The number of retransmission requests permitted per user is also limited per day.

The client makes a TCP/IP connection with the Retransmission Server, and receives the requested messages also via the TCP/IP channel.

The following diagram shows the sequence of messages and the transport protocols employed when making a retransmission request.

Figure 5 Retransmission Request



The retransmission request includes a Source ID (username) which is validated by the exchange system. It is important to note that only one Source ID can be used per application session.

The retransmission request may be rejected for any of the following reasons:

- Invalid Source ID (username)
- Invalid packet sequence number
- Incorrectly formatted request packet
- Packet no longer in cache
- Total number of packets requested in the current day exceeds the predefined system limit
- Number of retransmission requests in the current day exceeds the predefined system limit

In the case of such a failure, the user receives an error message to advise of the reason for failure.

If the reason for failure is exceeding a predefined system limit, clients are asked to not make any further requests. If further retransmissions are required, the client should contact Euronext.

4.3.2 Retransmission Server

If a packet is lost from both line A and line B, clients then make a TCP/IP request to have the packets resent. Packets are resent from the Retransmission Server.

After a client establishes a TCP/IP connection, the Retransmission Server periodically sends heartbeat request messages to the client. Clients must respond to this request with a heartbeat response within a specific timeframe – otherwise, the Retransmission Server closes the connection.

The client makes a TCP/IP connection to the Retransmission Server for both requesting and receiving retransmitted packets.

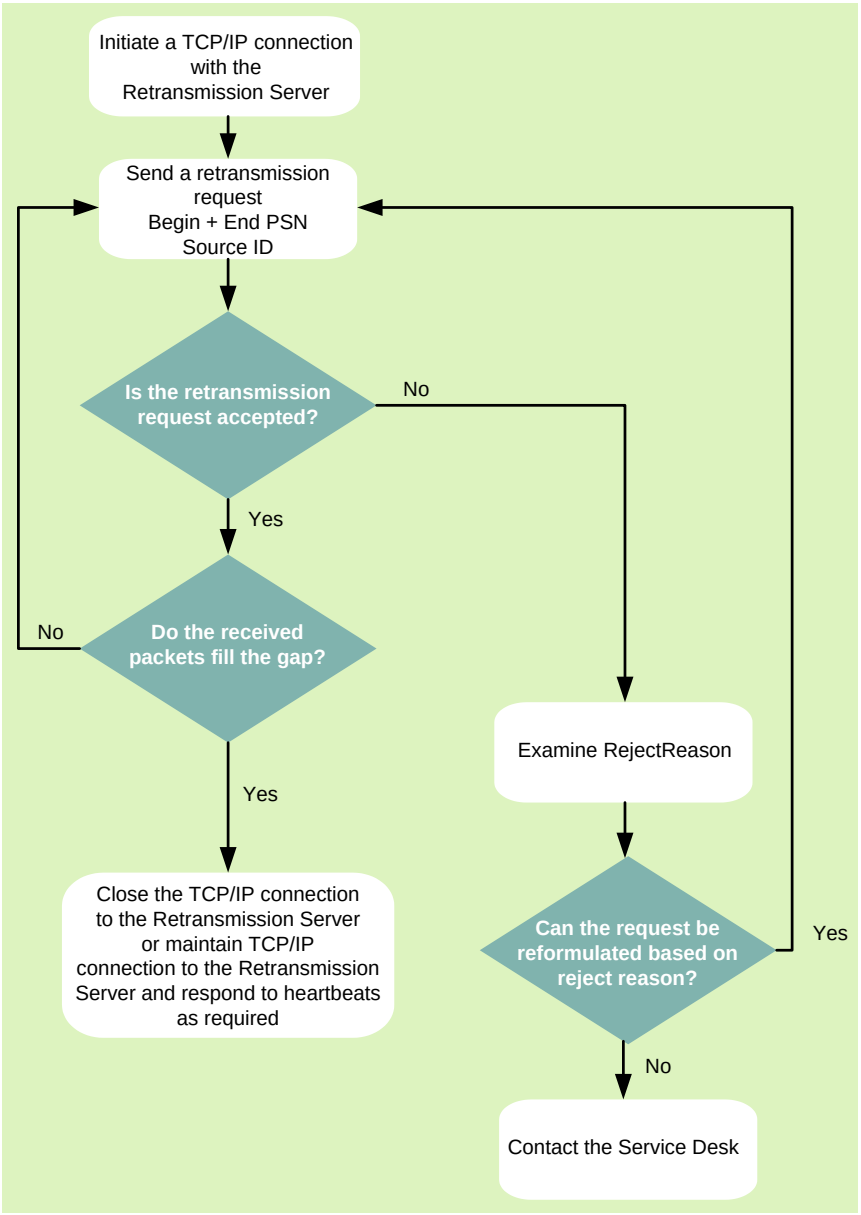
Retransmission requests should contain a Start PSN, an End PSN and a Source ID. The Source ID identifies the client application, and is supplied by the exchange. The request can be rejected for a number of reasons, see Retransmission Response.

The number of retransmissions allowed per client per day is limited, see Retransmission Request Limitations. The length of each retransmission is limited to a pre-defined number of packets.

Note that the Packet Sequence Number Reset message is never sent by the Retransmission Server, therefore if a client request for retransmission is made with Start PSN 1, the first message received in the retransmission response is PSN 2.

The following diagram illustrates the process of requesting dropped packets from the retransmission server.

Figure 6 Requesting Retransmission of Dropped Packets



4.3.3 Retransmission Request

This message is sent by clients requesting missing messages identified by a sequence number gap. Upon receipt of a valid retransmission request message, the requested message(s) will be sent. The requested message(s) have the same message format and content as the original sent by the system.

Note that the fields in the packet header should be filled as follows:

Table 9 Retransmission Request Message Header Format

FIELD	DESCRIPTION
PacketLength	44
PacketType	20
PacketSeqNum	Optional

FIELD	DESCRIPTION
SendTime	Optional
ServiceID	Service ID of the broadcast stream corresponding to the request, in other words the stream for which messages need to be recovered by the client.
DeliveryFlag	0
NumberMsgEntries	1 (only 1 retransmission request should be sent per packet)

The fields in the message body are as follows:

Table 10 Retransmission Request Message Body Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
BeginSeqNum	16	4	Binary Integer	Begin Sequence Number of the requested range of messages to be retransmitted. Note the Sequence Number refers to the PacketSeqNum in the header. The broadcast stream from which a retransmission is requested has to be stated in the field ServiceID in the Packet header of the RetransmissionRequest message.
EndSeqNum	20	4	Binary Integer	End Sequence Number of the requested range of messages to be retransmitted. Note the Sequence Number refers to the PacketSeqNum in the header. The broadcast stream from which a retransmission is requested has to be stated in the field ServiceID in the Packet header of the RetransmissionRequest message.
SourceID	24	20	ASCII String	This field represents the Identifier of the source (client) requesting retransmission. SourceID is pre-set by the Exchange and subject to validation. Field is null padded, left aligned.

4.3.4 Retransmission Response

This message will be sent immediately via TCP/IP in response to the client's request for retransmission messages.

Table 11 Retransmission Response Message Fields

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
SourceSeqNum	16	4	Binary Integer	This field contains the request message sequence number assigned by the client. It is used by the client to couple the request with the response message.
SourceID	20	20	ASCII String	This field represents the Identifier of the source (client) requesting retransmission. Field is null padded, left aligned.
Status	40	1	ASCII String	Indicates whether the retransmission request was accepted or rejected. Valid values: 'A' - Accepted. 'R' - Rejected.
RejectReason	41	1	Binary Integer	Indicates the reason for the rejection. Valid values are: ■ '0' - Message was accepted. ■ '1' - Rejected - invalid username (SourceID). ■ '2' - Rejected - invalid sequence number range. ■ '3' - Rejected - number of packets requested exceeds the predefined system limit. ■ '4' - Rejected - number of retransmission requests in the current day exceeds the predefined system limit. ■ '5' - Rejected - requested packets are not available ■ '6' - Rejected - incorrectly formatted request packet
Filler	42	2	ASCII String	For future use.

4.4 RETRANSMISSION REQUEST LIMITATIONS

The below recommendations apply to production and test.

4.4.1 Source ID

The Source ID allows clients to perform retransmission and refresh requests. Please note that the Source IDs for retransmissions and refresh are identical. Euronext will provide each client with a default of 4 Source IDs for Production if more Source IDs is necessary, please contact CAS team.

Each Source ID may only be logged on to a server once at a given time.

4.4.2 Heartbeat Mechanism

The heartbeat message frequency is set to 30 seconds.

A heartbeat message response has to be sent within 5 seconds to stay connected to the server

4.4.3 Number of Source IDs

Clients are provided with 4 SourceIDs by default. In the event more Source IDs are required, please contact Customer Access Services.

4.4.4 Parallel Sessions

Clients may file several concurrent requests on the server at the same time with the same SourceID; there is no need to wait for the active retransmission to be closed to ask for another one.

But it is important to remind these several requests will be queued and the responses to these requests will be sent in the same order as the initial requests.

4.4.5 Maximum Number of Requests

Maximum number of Retransmission Requests per Source ID per day has been set at 1,000.

4.4.6 Maximum Number of Packets per Request

Maximum number of packets that can be requested in one Retransmission Request has been set at 1,000.

4.4.7 Maximum Number of Packets Stored in the Retransmission Cache

The maximum number of packets that are cached for retransmission request has been set at 500,000 per Service ID. A packet out of cache cannot be retransmitted (A dedicated response “Rejected – requested packets are not available” will be sent in this case).

4.5 REFRESH

4.5.1 Refresh Functionality

The Refresh Server supplies (on demand) a snapshot including reference data, last trade price, high, low and the order book.

The Refresh Server is designed to allow the user to update the market state within their applications before restarting in real-time, following a data outage or late start.

The client makes a TCP/IP connection to the Refresh Server for requesting the refresh, whilst also joining the refresh multicast channels for receiving the refresh messages.

The Refresh Server responds to a request with a refresh response message to indicate whether the request was accepted or rejected.

The refresh messages are then sent on the multicast channels. This is preceded by a Start of Refresh message and followed by an End of Refresh message. No dedicated retransmission service is available for the refresh; if packet loss is detected, clients should submit another refresh request.

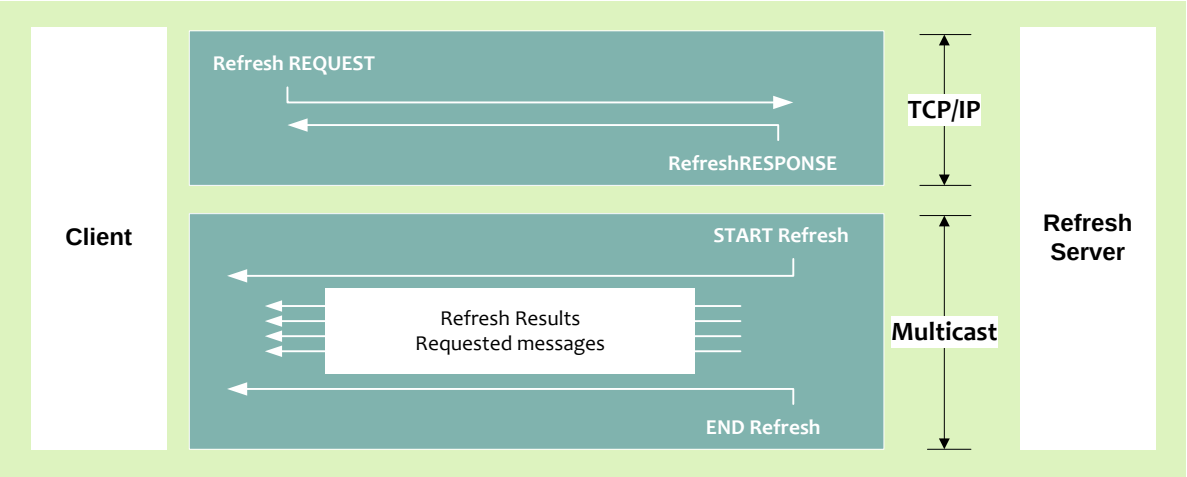
The refresh request includes a Source ID (username) which is validated by the exchange system. It is important to note that:

- **only one Source ID can be used per application session,**
- **it is not possible to make several request at the same time using the same Source ID.**

A pair of refresh multicast channels are provided for each corresponding real-time service. The contents of the refresh and message formats correspond to the contents and message formats contained in the appropriate real-time service.

The following diagram shows the sequence of messages and the transport protocols employed when making a refresh request.

Figure 7 Refresh Request



The refresh request may be rejected for any of the following reasons:

- Invalid Source ID (username)
- Invalid packet sequence number requested
- Incorrectly formatted request packet
- Packet no longer in retransmission cache
- Total number of refreshes requested in the current day exceeds the predefined system limit
- Rejected due to unavailability of refresh data

In the case of such a failure, the user receives an error message to advise of the reason for failure.

If the reason for failure is exceeding a predefined system limit, clients are asked to not make any further requests. If further refreshes are required, the client should contact Euronext.

4.5.2 Refresh Server

If a client starts their application late, or experiences an outage, the Refresh Server should be used to provide the means to get back in synchronization with the real-time market. Clients send and receive the

refresh request and response messages over a TCP/IP connection, and join the refresh multicast groups to receive the content of the refresh.

After a client establishes a TCP/IP connection, the Refresh Server periodically sends heartbeat request messages to the client. Clients must respond to this request with a heartbeat response within a specific timeframe – otherwise, the refresh server closes the connection.

The client makes a TCP/IP connection to the Refresh Server to request a refresh of packets and at the same time joins the multicast refresh groups. Refresh requests should contain a Service ID and a Source ID. The Service ID defines for which multicast channel a refresh is required. The Source ID identifies the client application, and is supplied by the exchange. This is identical to the Source IDs allocated for the Retransmission Server. The request can be rejected for a number of reasons as defined in the refresh response message, see Refresh Response.

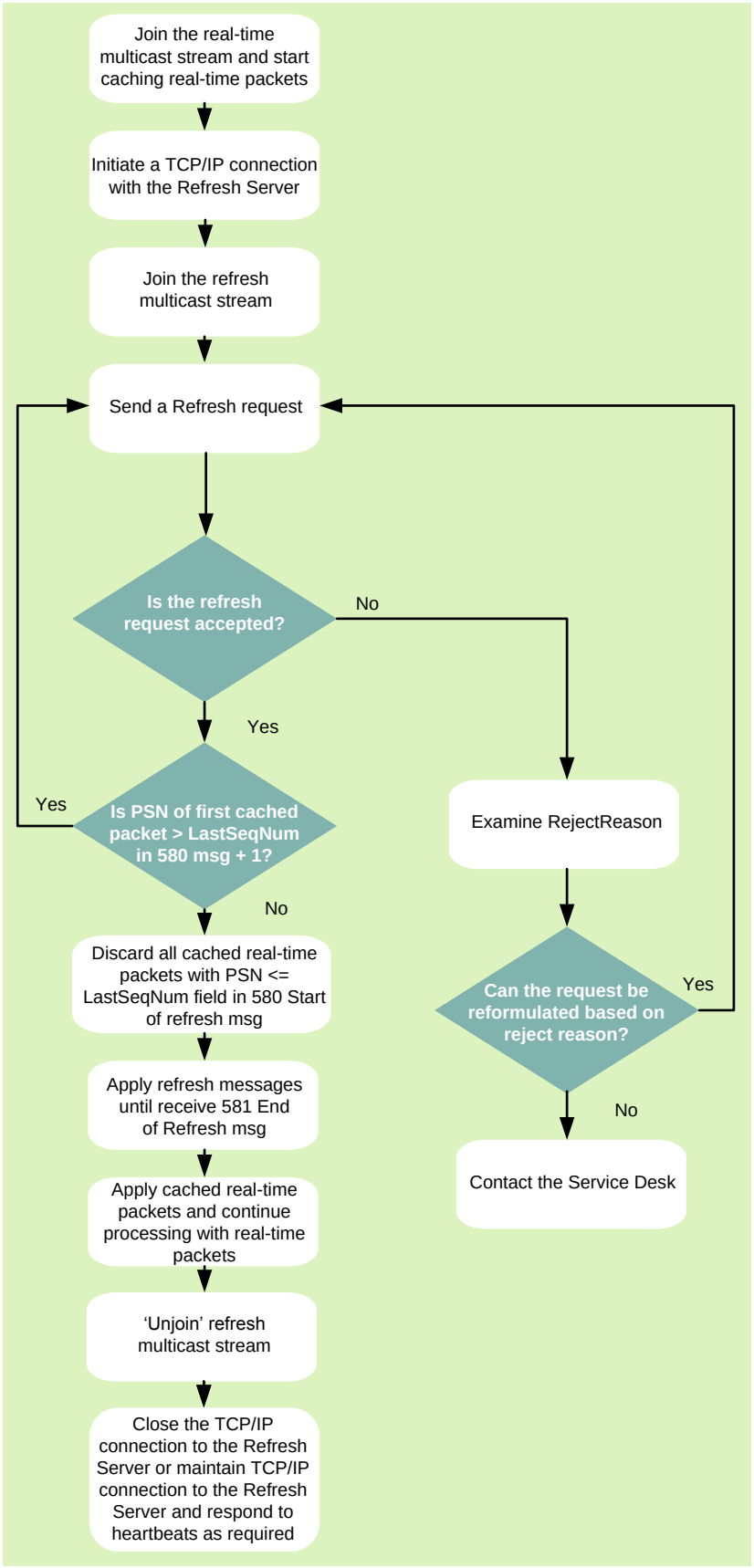
Once a successful request has been received by the server, a refresh response is sent to the client. Clients should then process the refresh content from the refresh multicast groups and synchronize this with the real-time data.

The information contained within a snapshot is not intended to be applied incrementally to your current order book for that product; it is intended to replace the previous data image. Therefore the client should clear the order book image for any instrument associated with that Service ID before applying the contents of the refresh. Note that multiple market data update messages can be delivered during a refresh for an individual instrument.

The number of refreshes allowed per client per day is limited, see Refresh Request Limitations.

The following diagram illustrates the process of requests from the refresh server.

Figure 8 Requesting Refresh Information



4.5.3 Refresh Request

This message is sent by clients requesting a refresh. The system will provide the appropriate message(s) in response.

Note that the fields in the standard packet header should be filled as follows:

Table 12 Refresh Request Message Header Format

FIELD	DESCRIPTION
PacketLength	36
PacketType	22
PacketSeqNum	Optional
SendTime	Optional
ServiceID	Service ID of the broadcast stream corresponding to the request (the stream for which the refresh will be applied)
DeliveryFlag	Ignored in the response
NumberMsgEntries	1

The fields in the message body are as follows:

Table 13 Refresh Request Message Body Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
SourceID	16	20	ASCII String	Identifier of the Source ID requesting the refresh.

4.5.4 Refresh Response

This message will be sent immediately via TCP/IP in response to the client's request for a refresh.

Table 14 Refresh Response Message Header Format

FIELD	DESCRIPTION
PacketLength	44
PacketType	23
PacketSeqNum	Contains the Packet Sequence Number if sent in the refresh request
SendTime	Ignored in the response
ServiceID	Service ID of the broadcast stream corresponding to the request (the stream for which the refresh will be applied). This will be the same value as in the initial request.
DeliveryFlag	Ignored in the response
NumberMsgEntries	1

The fields in the message body are as follows:

Table 15 Refresh Response Message Body Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
SourceSeqNum	16	4	Binary Integer	This field contains the request packet sequence number assigned by the client. It is used by the client to couple the request with the response message.
SourceID	20	20	ASCII String	Identifier of the source requesting the refresh.
Status	40	1	ASCII String	Indicates if the Refresh request has been accepted. Valid values are: ■ 'A' - Accepted ■ 'R' - Rejected
RejectReason	41	1	Binary Integer	Indicates the reason for the rejection. Valid values are: ■ 0' - Message was accepted. ■ '1' - Rejected – username (SourceID) invalid or already in use. ■ '2' - Rejected – incorrect ServiceID. ■ '3' - Rejected – request incorrectly formatted. ■ '4' - Rejected – incorrect packet type sent. ■ '5' - Rejected – exceeded maximum number of requests per day ■ '6' - Rejected – requested refresh data unavailable
Filler	42	2	ASCII String	For future use

4.5.5 Start Refresh – 580 Message

A refresh cycle begins with a Start Refresh message and ends with an End Refresh message on the multicast channels.

The packet type for a Start Refresh message is 799.

Table 16 Start Refresh Message Fields

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	16	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	18	2	Binary Integer	580 – Start Refresh Message
LastSeqNum	20	4	Binary Integer	Contains the last cached PacketSeqNum that the Refresh is valid to.

4.5.6 End Refresh – 581 Message

A refresh cycle begins with a Start Refresh message and ends with an End Refresh message on the multicast channels.

The packet type for an End Refresh message is 799.

Table 17 End Refresh Message Fields

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	16	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	18	2	Binary Integer	581 – End Refresh Message
LastSeqNum	20	4	Binary Integer	Contains the last cached PacketSeqNum that the Refresh is valid to.

4.5.7 Refresh Contents

The table below describes the Service ID where the refresh functionality will be available in the production environment.

Table 18 Refresh Contents

SERVICE TYPE	REFRESH CONTENTS
Level 1	Best bid/offer Total traded volume Last trade price/volume Implied bid/offer (where applicable) Previous day settlement price

SERVICE TYPE	REFRESH CONTENTS
	Settlement price (if settlement has already occurred) Close price (if after market close) EDSP (if after settlement on expiry day) Product availability and market status
Level 1 + 2	Best bid/offer Full order book Total traded volume Last trade price/volume/type Implied bid/offer (where applicable) Previous day settlement price Settlement price (if settlement has already occurred) Close price (if after market close) EDSP (if after settlement on expiry day) Product availability and market status
Standing Data	Standing data message for each outright and strategy instrument on market
Value Add + Open Interest	Value-added parameters Open interest for current day

4.6 REFRESH REQUEST LIMITATIONS

The below recommendations apply to production and test.

4.6.1 Source ID

The Source ID allows clients to perform retransmission and refresh requests. Please note that the Source IDs for retransmissions and refresh are identical. Euronext will provide each client with a default of 4 Source IDs for Production.

Each Source ID may only be logged on to a server once at a given time.

It is important to note that:

- only one Source ID can be used per application session,
- **it is not possible to make several request at the same time using the same Source ID**.

4.6.2 Heartbeat Mechanism

The heartbeat messages frequency is set to 30 seconds.

A heartbeat message response has to be sent within 5 seconds to stay connected to the server

4.6.3 Number of Source IDs

The Source IDs for the refresh are identical to the retransmission server. They should be reused with the refresh server.

Clients are provided with 4 SourceIDs by default. In the event more Source IDs are required, please contact Customer Access Services.

4.6.4 Maximum Number of Requests

Maximum number of Refresh Requests per Source ID per day has been set at 1,000.

4.7 HEARTBEAT

Heartbeat messages are sent in the multicast streams and the active TCP/IP retransmission and refresh sessions.

4.7.1 General Heartbeat Processing Notes (TCP and Multicast)

The following applies to the TCP channels for retransmissions and refresh, and also the multicast channels for real time and refresh data.

- Heartbeat messages contain only the packet header (with PacketType = '2'). The packet does not contain a message body.
- Heartbeats are sent only on the multicast channels when there is no market data. Heartbeat frequency since the last packet, is:
  - Seconds in the multicast streams
  - 30 seconds in the active TCP/IP retransmission sessions

4.7.2 Retransmission and Refresh Heartbeat Processing Notes (TCP)

- Clients may receive a heartbeat message if they have an active TCP/IP session with the retransmission or refresh server.
- To determine the health of the user connection on the TCP/IP channel, the retransmission or refresh server sends regular heartbeat messages to the user. The heartbeat frequency is 30 seconds. The time out for this heartbeat response message is set at 5 seconds. If no response is received by the server within this timeframe, the TCP/IP session is disconnected.
- Clients that choose to establish and remain connected to the Retransmission or Refresh Server intraday must respond to a heartbeat message with a heartbeat response message. Users can choose to either disconnect following each retransmission or refresh request, or remain connected to the retransmission or refresh server.

Figure 9 Retransmission Server Heartbeats

Figure 10 Refresh Server Heartbeats

4.8 HEARTBEAT RESPONSE

Clients that choose to establish and remain connected to the retransmission server intraday must respond to a heartbeat message with a heartbeat response message.

Note that the fields in the packet header should be filled as follows:

Table 19 Heartbeat Response Message Header Format

FIELD	DESCRIPTION
PacketLength	36
PacketType	24
PacketSeqNum	Optional
SendTime	Optional
ServiceID	Optional
DeliveryFlag	0
NumberMsgEntries	1 (only 1 heartbeat response message should be sent per packet)

The fields in the message body are as follows:

Table 20 Heartbeat Response Message Body Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
SourceID	16	20	ASCII String	This field represents the Identifier of the source (client). Field is null padded, left aligned.

4.9 OPERATIONAL INFORMATION

4.9.1 Packet Sequence Number Reset Processing Notes

Packet Sequence numbers normally begin at one (1) and increase one by one with each subsequent packet. There are two scenarios where the packet sequence number is reset (besides the start of day):

- If the value should exceed the maximum value that the SeqNum field may contain, it will be reset to one (1).
- **If the real-time publisher fails and a failover to the secondary publisher is performed by the Exchange, the secondary publisher sends a Packet Sequence Number reset message. The PacketSeqNum field of that packet will be set to one (1) and the NextSeqNumber field will be set to two (2).**

Note that the packet sequence number reset is always sent as the first message of the day. The Packet Sequence Number Reset message is never sent as part of a retransmission.

4.9.2 System Behaviour on Start and Restart

At the start of the day, the feed sends the following messages:

- Standing Data Messages (722/732) for each future, option and strategy instrument on the market
- Product Availability Message (741) with TradingAvailableFlag = 1 for each available product
- A snapshot of market data for each available product, comprising:
  - Market Update Messages (702) providing best bid/ask, order book and last trade price for each instrument
  - Market Status (752) providing the market status for each product
  - Settlement Prices (712) providing Yesterday Settlement Prices for each instrument

Note that this sequence is also followed on system recovery following a failure. Therefore in exceptional circumstances a user may see this during the trading day.

It is important that each client clears all instrument order books for a particular product on receipt of a Product Availability Message with TradingAvailableFlag equal to 0 or 1 for that product.

4.10 HIGH AVAILABILITY MECHANISM AND DISASTER RECOVERY SITE

4.10.1 Real-Time Market Data High Availability

The High Availability (HA) functionality of the market data publisher is set up to ensure there is no loss of service for clients if there is any kind of outage in the exchange on the primary publisher, for example a hardware failure. The failover to the secondary publisher occurs without any gap in market data packet sequence numbers. The HA failover has been designed to be as transparent as possible for clients, as the connectivity in terms of multicast groups and ports does not change. However, clients should note that there are specific technical details that should be considered.

Clients that use a Firewall must ensure that the Primary, Secondary and Disaster Recovery Source IP Addresses for multicast channels for real-time and refresh that they subscribe to, are allowed through their Firewalls.

Clients also need to take into account that if the real-time publisher fails and a failover to the secondary publisher is performed by the Exchange, the secondary publisher sends a Packet Sequence Number reset packet. The PacketSeqNum field of that packet will be set to one (1) and the NextSeqNumber field will be set to two (2).

4.10.2 High Availability Retransmission Behaviour

In production there are two retransmission servers, the primary and secondary. **Both these retransmission servers are setup behind one virtual IP Address and only the primary server is servicing client requests.** Clients should monitor the TCP/IP connection to the retransmission server to determine if there is an outage. **Clients are required to reconnect to the Virtual IP address in case of a disconnection due to a failure of the primary retransmission server.** The failover from primary to secondary retransmission server is performed by the Exchange. Clients should monitor the availability of the retransmission service by the following means:

- For clients remaining connected to the retransmission server throughout the day, a disconnection from the retransmission server should trigger a reconnect to the same virtual IP address.
- Clients may choose to only connect to the retransmission server if the application requires packets to be serviced.

Note that in case of a failover, the customers is expected to disconnect the session made with the retransmission server to allow the retransmission server to route the connection to the secondary server.

4.10.3 High Availability Refresh Behaviour

In production there are two refresh servers, the primary and secondary**. Both these refresh servers are setup behind one virtual IP Address and only the primary server is servicing client requests.** Clients should monitor the TCP/IP connection to the refresh server to determine if there is an outage. **Clients are required to reconnect to the Virtual IP address in case of a disconnection due to a failure of the primary refresh server.** The failover from primary to secondary refresh server is performed by the Exchange. Clients should monitor the availability of the refresh service by the following means:

- For clients remaining connected to the refresh server throughout the day, a disconnection from the refresh server should trigger a reconnect to the same virtual IP address.
- Clients may choose to only connect to the refresh server if the application requires packets to be serviced.

4.10.4 Disaster Recovery Site

In order to mitigate any serious outage in the primary data center, a secondary data center is online in standby mode, in case of a serious incident.

Clients should ensure that all configuration surrounding the secondary data center is included as described in https://euronext.com/en/it-documentation/market-data

5. REAL-TIME MESSAGE SPECIFICATIONS

5.1 GENERAL PROCESSING NOTES

5.1.1 General Processing Notes

The following processing notes apply to all real-time messages:

- All fields are sent for every message
- Only field values appear in the published messages (for example, no names or 'tags' appear in the message)
- The field names that appear in the message format documents are for reference purposes only
- All the fields are contiguous, with reserved fields for alignment issues
- All field sizes are fixed and constant
- Binary fields are provided in network byte order (Big-endian format)
- ASCII string fields are left aligned and null padded
- Segmentation of messages across packets is not supported. This means that a message never straddles a packet boundary.

5.1.2 FAST Optimization

FAST optimization is used to minimize the bandwidth required for the Euronext Derivatives XDP feed.

Packet headers are not FAST optimized.

Messages within the packet are FAST-optimized (with the exception of technical messages).

Appendix A: FAST Encoding provides details of the FAST template that should be used to decode messages sent on the Euronext Derivatives XDP feed.

5.1.3 Date and Time Conventions

Dates and Times use UTC (Universal Time, Coordinated).

The base for timestamps is the number of milliseconds since the previous Sunday 00:00:00.000 UTC (so in the night from Saturday to Sunday).

For example Wednesday 15:30:00.000 UTC is indicated as 315000000.

5.1.4 Sequence Numbers

The feed contains two sequence number fields:

- Packet Sequence Number

  The Packet Sequence Number is part of the packet header, and should be used for retransmission requests. It should also be used to synchronise between real-time and refresh services. It is unique per service and common across a pair of dual multicast streams. Note that the packet sequence number is only unique for market data packets; heartbeats use the PSN of the last packet.

- Series Sequence Number

  Message type 702 contains a Series Sequence Number.

  For Service Type Level 1+2 the SeriesSequenceNumber increases monotonically and is unique for each instrument (future, option, strategy).

For Service Type Level 1 the SeriesSequenceNumber does not increase monotonically.

The Series Sequence Number should not be used to synchronize between messages in the real-time and refresh services. The refresh may contain separate market update messages for different data points (for example best bid/ask, order book, total traded volume) with differing Series Sequence Number values.

5.1.5 Price Formats

All price fields are sent in integer format.

Unless otherwise specified, prices are sent in absolute ticks. See Prices in Ticks for details on how to convert a price, which is denoted in absolute ticks, to a displayable price, using the price format code.

In other cases, where explicitly specified in the message specifications, some fields appear in an integer/ scale code format. In this case the value is represented by two fields, and should be calculated using the following formula:

$$Value = \frac{Integer}{10^{ScaleCode}}$$

For example, a price of 98.75 is represented by an Integer of 9875 and a ScaleCode of 2.

5.1.6 Prices in Ticks

The tick denominator field is a numeric field indicating how to convert a price, which is denoted in absolute ticks, to a displayable price. It represents the number of absolute ticks in a full point. Four examples are provided below.

Note that Tick Size Denominator and Tick Size Numerator are available in the standing data FIXML file, and are not available on the XDP feed.

5.1.6.1 Example 1

Tick Size Denominator = 1000

Assuming:

- That the price is to be displayed as a decimal, denoted in points.
- That the price field contains "000061525"

Absolute Price = 61525

Points = Absolute Price DIV Tick Size Denominator = 61525 DIV 1000 = 61

Ticks = Absolute Price MOD Tick Size Denominator = 61525 MOD 1000 = 525

Displayable Price = Points + "." + Ticks = "61.525"

5.1.6.2 Example 2

Tick Size Denominator = 200

Assuming:

- That the price is to be displayed as a decimal, denoted in points
- That the price field contains "000020002"

Points = Absolute Price DIV Tick Size Denominator = 20002 DIV 200 = 100

Ticks = Absolute Price MOD Tick Size Denominator = 20002 MOD 200 = 2

Displayable Price = Points + "." + (Ticks/Tick Size Denominator) = 100 + (2/200) = “100.010”

5.1.6.3 Example 3

Tick Size Denominator = 4

- Assuming:
- That the price is to be displayed as a decimal, denoted in points
- That the field contains "000000099"

Points = Absolute Price DIV Tick Size Denominator = 99 DIV 4 = 24

Ticks = Absolute Price MOD Tick Size Denominator = 99 MOD 4 = 3

Displayable Price = Points + "." (Ticks/ Tick Size Denominator) = 24 + (3/4) = “24.75”

5.1.6.4 Example 4

Tick Size Denominator = 64

The price field could be "000005969", i.e. Absolute Price = 5969

Points = Absolute Price DIV Tick Size Denominator = 5969 DIV 64 = 93

Ticks = Absolute Price MOD Tick Size Denominator = 5969 MOD 64 = 17

- Displayable Price = Points + Ticks + "/" + Tick Size Denominator = "93 17/64"
- Displayable Price = Points + "." + (Ticks/Tick Size Denominator) = "93.265625"

5.1.7 Data Types

Binary data is in network byte order (big-endian format).

‘Binary Integer’ fields are unsigned.

‘Binary Signed Integer’ fields are signed, and can take negative values.

All alphanumeric fields are left justified and null padded.

5.1.8 Instrument Identifiers

Instruments on the Euronext Derivatives XDP feed can either be identified by an AMR code or a Symbol Index.

The AMR code is a 15-character ASCII string (including spaces), allocated by the trading engine. It is unique per instrument.

The Symbol Index is an integer value (32 bit unsigned integer). The Symbol Index is unique across all Euronext Derivatives services. It is unique per instrument.

The AMR code and Symbol Index are mutually exclusive. Each instrument is identified by either one or the other. The AMR code and Symbol Index are persisted for the lifetime of the instrument.

5.1.9 Standing Data

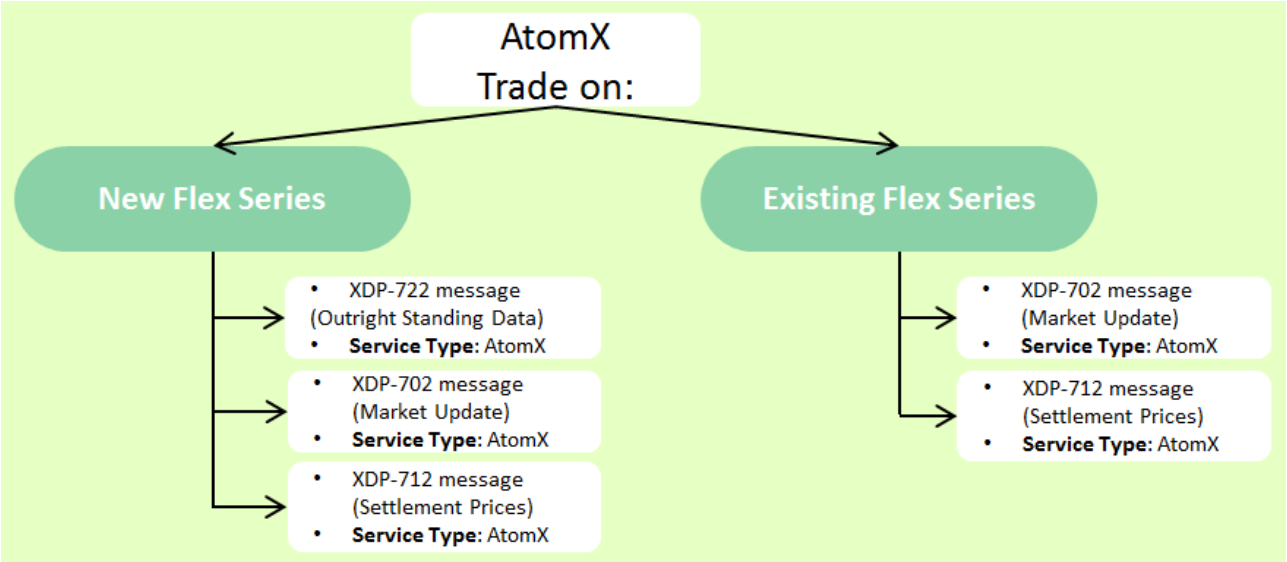
Standing data messages are disseminated via the multicast channels. Standing data messages for all futures, options and strategies are sent at the start of the trading day, and subsequently for any intraday changes (new option strikes, creation of strategy instruments). Standing data may also be sent following recovery from a system failure.

Note that the standing data messages provide basic characteristics of each future, option and strategy.

Additional referential data (for example last trading date, trading currency, underlying instrument characteristics) can be obtained from a separate FTP service. Refer to the Euronext Derivatives FTP Client Specification for further details on this service.

5.1.10 AtomX Trades

The following diagram explains the global publication process into the “AtomX” Service Type in case of a trade on a Flex contract.



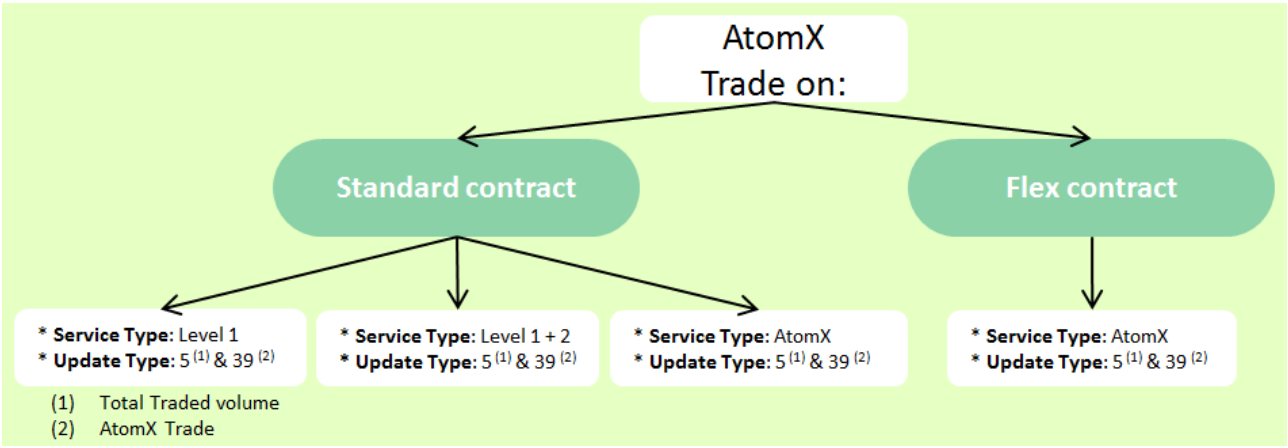
In case of a trade on a new Flex series, an Outright Standing Data message (XDP-722) will be broadcasted via the AtomX channel.

Following this, a Market Update message (XDP-702) will be broadcasted (see the bellow for more details about Market Update publication).

Settlement Prices message (XDP-712) will be published at any time after validation from the Exchange.

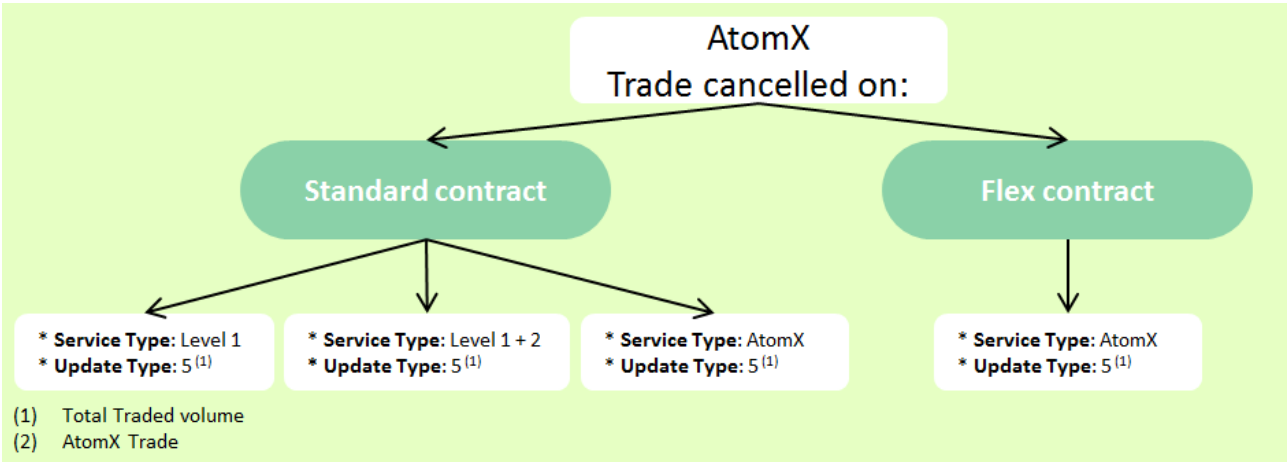
The following diagrams describe the impact of trade update publication on the AtomX service through the XDP Service Types:

New trade through AtomX (message XDP-702):



In case of an AtomX Trade being reported on a standard contract, the total traded volume is calculated by aggregating both Central Order Book (COB) and AtomX traded volume. This AtomX trade and the cumulative volume traded on a standard contract is sent out via one of the Equity and Index Derivatives channels and also via the AtomX channel. In the AtomX channel we will only publish the cumulative volume for a given standard contract when a trade on that contract is performed on AtomX and this is the cumulative volume for both Central order Book and AtomX volumes at the time that the AtomX Trade took place. In case new trades occur on the COB after the AtomX Trade took place, the AtomX channel will not show the new cumulative volume for Standard Contracts. This will only be available in the Equity and Index Derivatives channels

Trade Cancelled through AtomX (message XDP-702):



5.1.11 Cancellations and Corrections

The Euronext Derivatives XDP feed does not support explicit trade cancellations and/or corrections.

In the event of a trade cancellation/correction the total traded volume (Update Type 5 in 702 message) is adjusted.

5.2 MARKET INFORMATION

5.2.1 Overview

The Euronext Derivatives XDP feed uses the push-based publishing model. This means that data is published based on its availability. Once information is available, it is published to clients.

The following message types are in the Euronext Derivatives XDP feed:

- 702 – Market Update
- 712 – Settlement Prices
- 722 – Outright Standing Data
- 732 – Strategy Standing Data
- 741 – Product Availability
- 752 – Market Status
- 761 – Exchange Message
- 772 – Value-Added Parameters
- 782 – Open Interest

5.2.2 Packet Header Format

All messages are preceded by a common packet header format. The following table describes the header fields of a Euronext Derivatives XDP real-time message.

Table 21 Packet Header Format

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
PacketLength	0	2	Binary Integer	Length of the packet including the 16-byte packet header.
PacketType	2	2	Binary Integer	Identifier for the type of data contained in the packet. Possible values: '799' - Generic Derivatives Message
PacketSeqNum	4	4	Binary Integer	This field contains the packet sequence number. It is unique for each broadcast stream (multicast group) and is used for gap detection. It increases monotonically and is reset to 1 at the beginning of each trading day. Note that heartbeats inherit their sequence number from the last market data packet or packet sequence number reset packet.
SendTime	8	4	Binary Integer	Timestamp in millisecond indicating the packet broadcast time. The number represents the number of milliseconds since midnight of the last Sunday 00:00 UTC

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
ServiceID	12	2	Binary Integer	Numeric value identifying the broadcast stream. Possible values are described in Feed Configuration descriptions
DeliveryFlag	14	1	Binary Integer	Indicates the delivery method: ■ '8' - Real Time message (FAST optimized) ■ '9' - Refresh message (FAST optimized) ■ '10' - Retransmission message (FAST optimized)
NumberMsgEntries	15	1	Binary Integer	The number of messages that are contained within the packet.

5.2.3 Market Update – 702 Message

5.2.3.1 Message Overview

A Market Update message provides information about price updates for any outright or strategy instrument.

5.2.3.2 Message Sending Rules

A Market Update 702 message is sent as a result of one of the following events:

- New top of book price and/or volume (explicitly entered or implied)
- New depth of book price and/or volume
- New Indicative Opening price calculated
- New Last Trade
- Request for quote (RFQ)

5.2.3.3 Message Structure

The table below describes the body fields of a Market Update message, MsgType = '702'. Note that this is the message content prior to the application of FAST. Some fields may not appear once FAST has been applied and only included for reference, for example MsgSize. See Appendix A for more details about the application of FAST.

Table 22 Market Update Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Integer	Numeric message type identifier: '702' - Market Update
SymbolIndex	4	4	Binary	Index of the symbol

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
			Integer	Set to '0' if the instrument uses a SecurityID
SecurityIDSource	8	1	Binary Integer	Type of the security code '8' - AMR Set to '0' if the instrument uses a SymbolIndex
SecurityID	9	15	ASCII String	Security code (source of code indicated by SecurityIDSource field) If SecurityIDSource = 8 then this field will contain the AMR code. Set to all spaces if instrument uses a SymbolIndex
SourceTime	24	4	Binary Integer	Milliseconds since the previous Sunday 00:00 (UTC)
SeriesSequenceNumber	28	4	Binary Integer	Sequence number for the message, monotonically increasing and unique for each instrument (future, option, strategy). 1 – 4,294,967,294 ■ For Service Type Level 1+2 the SeriesSequenceNumber increases monotonically and is unique for each instrument (future, option, strategy). ■ **For Service Type Level 1 the SeriesSequenceNumber does not increase monotonically.** ■ The Series Sequence Number should not be used to synchronise between messages in the real-time and refresh services. The refresh may contain separate market update messages for different data points (for example best bid/ask, order book, total traded volume) with differing Series Sequence Number values.
SnapshotFlag	32	1	Binary Integer	Indicates whether the price updates in the message are snapshots. ■ '0' - Real-time event/update ■ '1' - Snapshot
Filler	33	1	-	Reserved for future use
UpdateCount	34	2	Binary	Number of updates. Indicates number of

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
			Integer	times the following group of three fields (Update Type, Price and Volume) will be repeated in the message.
> UpdateType	-	2	Binary Integer	Type of update: ■ '1' - Best Bid ■ '2' - Best Offer ■ '3' - Bid ■ '4' - Offer ■ '5' - Total Traded Volume ■ '6' - Conventional Trade ■ '7' - Large in Scale (LiS) Trade ■ '8' - Basis Trade ■ '9' - Large in Scale (LiS) PackageTrade ■ '10' - Guaranteed Cross Trade ■ '11' - Against Actual Trade ■ '12' - Asset Allocation Trade ■ '13' - External Match Trade ■ '14' - Exchange For Swap Trade ■ '15' - Exchange For Physical Trade ■ '17' - Implied Bid ■ '18' - Implied Offer ■ '19' - Indicative Open Price ■ '29' - Strategy Leg Conventional Trade ■ '30' - Strategy Leg Large in Scale (LiS) Trade ■ '31' - Strategy Leg Basis Trade ■ '33' - Strategy Leg Guaranteed Cross Trade ■ '34' - Strategy Leg Against Actual Trade ■ '35' - Strategy Leg Asset Allocation Trade ■ '36' - Strategy Leg External Match Trade ■ '37' - Strategy Leg Exchange For Swap Trade ■ '38' - Strategy Leg Exchange For Physical Trade ■ '39' *– AtomX Trade* ■ '40' - Request for Quote ■ '41' - Request for Cross Activated ■ '42' - Request for Cross Queued ■ '43' - Request for Cross Trade ■ '44' - Strategy Leg Request for Cross

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				Trade
> StrategyCode	-	1	ASCII Char	Exchange-recognized market code For example 'E' - Calendar Spreads Present for strategy leg trades only.
> Filler	-	1	-	Reserved for future use.
> Price	-	4	Binary Signed Integer	Price. See notes below for full explanation of contents.
> Volume	-	4	Binary Integer	Volume. See notes below for full explanation of contents.

The UpdateCount is used to specify the number of price/volume updates for the instrument, included within the message. For example, a trade execution could result in updates to the last trade price, best bid and offer and total trade volume.

The SnapshotFlag indicates whether the price/volume updates correspond to a real-time event or a snapshot. The information contained within a snapshot is not intended to be applied incrementally to your current order book for that product, it is intended to replace the previous data image. Therefore the client should clear the order book image for the instrument before applying the contents of the snapshot. Note that multiple market data update messages can be delivered during a snapshot for an individual instrument. A snapshot will be sent at a pre-defined time at the start of day to provide details of any orders such as GTC orders (Good 'Til Cancelled), and exceptionally following an intraday service restart.

The UpdateType indicates the type of price/volume update as follows:

- The Best Bid/Offer is the best explicit buy or sell price and aggregated volume at the best price.
- The Bid/Offer update is the explicit buy or sell price and aggregated volume at any price level. In the case where the Bid/Offer is the best price, both the Bid/Offer and the Best Bid/Offer will be sent.
- Best Bid/Offer update types (1 and 2) and Bid/Offer update types (3 and 4) cannot be mixed and must be used separately:
  - 1 and 2 to keep track of the BBO only
  - 3 and 4 to keep track of the full price limit depth (including the BBO)
- In case a message contains both update types 1 and 2, both updates have to be processed as one to update the BBO, or else a crossed book could occur after processing each update individually.
- The Total Traded Volume indicates the volume traded in this instrument since the start of the trading session.
- The Trade update types (6-15, 29-38, 43, 44) provide the last traded price and volume. The update type indicates the type of trade involved. For example a Conventional trade (UpdateType 6) is a central order book trade. A Large in Scale (LiS) Trade (UpdateType 7) is a wholesale trade type. The strategy leg trade update types (29-38,44) provide the traded price and volume in an outright leg that has been traded as part of a strategy trade.

- Implied bid/offer prices are sent for a given outright series when either a) an implied out buy/sell price can be calculated, and it is better than or equal to the best explicitly quoted price, or b) a previously transmitted implied buy/sell price or volume changes, or can no longer be implied.
- The Indicative Open Price indicates that the information is a change to an indicative opening price and or volume, transmitted during Pre-Opening only.
- A Request for Quote (RFQ) notifies market participants that a price is required for the market (futures/options expiry month, series or strategy). An RFQ has an associated volume but no price.
- A Request for Cross Activated notifies market participants that a Request for Cross has been activated.
- A Request for Cross Queued notifies market participants that a Request for Cross has been submitted and is waiting for previous Request For Crosses to terminate before being activated.

The StrategyCode field is only provided for strategy leg updates, and indicates the type of strategy that was traded. It will be present for update types 29, 30, 31, 33, 34, 35, 36, 37, 38,44.

Note that trade prices will not be provided for Against Actual trades, as per market convention.

The content of the price/volume fields depends on the value of the field UpdateType as follows:

Table 23 Price and Volume Contents

UPDATE TYPE	PRICE	VOLUME
Best Bid / Offer	Best explicit buy/sell price	Aggregated volume of explicit buy/sell orders at best
Bid / Offer	Explicit buy/sell price in depth	Aggregated volume of orders at this price level
Total Traded Volume	N/A	Accumulated volume for instrument for trading date.
Conventional Trade	Trade price	Traded volume
Large in Scale (LiS) Trade	Trade price	Traded volume
Basis Trade	Trade price	Traded volume
Large in Scale (LiS) Package Trade	Trade price	Traded volume
Guaranteed Cross Trade	Trade price	Traded volume
Against Actual Trade	N/A	Traded volume
Asset Allocation Trade	Trade price	Traded volume
External Match Trade	Trade price	Traded volume
Exchange for Swap Trade	Trade price	Traded volume
Exchange for Physical Trade	Trade price	Traded volume
Implied Bid / Offer	Implied buy/sell price in depth	Aggregated volume of implied orders at this price
Indicative Open Price	Indicative Opening price	Indicative Opening volume
Strategy Leg Conventional Trade	Trade price	Traded volume
Strategy Leg Large in Scale (LiS) Trade	Trade price	Traded volume
Strategy Leg Basis Trade	Trade price	Traded volume
Strategy Leg Guaranteed Cross Trade	Trade price	Traded volume
Strategy Leg Against Actual Trade	N/A	Traded volume
Strategy Leg Asset Allocation Trade	Trade price	Traded volume
Strategy Leg External Match Trade	Trade price	Traded volume
Strategy Leg Exchange For Swap Trade	Trade price	Traded volume
Strategy Leg Exchange For Physical Trade	Trade price	Traded volume
Request for Quote	N/A	RFQ volume

UPDATE TYPE	PRICE	VOLUME
Request for Cross Activated	N/A	N/A
Request for Cross Queued	N/A	N/A
Request for Cross Trade	Request For Cross Trade price	Request For Cross Traded volume
Strategy Leg Request for Cross Trade	Request For Cross Trade price	Request For Cross Traded volume

5.2.4 Settlement Prices – 712 Message

5.2.4.1 Message Overview

A Settlement Prices message provides settlement prices for outright instruments.

5.2.4.2 Message Sending Rules

This message can be transmitted at any time during Market Trading or Session End. It gives official (final) settlement and closing prices for outright products.

5.2.4.3 Message Structure

The table below describes the body fields of a Settlement Prices message, MsgType = '712'. Note that this is the message content prior to the application of FAST. Some fields may not appear once FAST has been applied and only included for reference, for example MsgSize. See Appendix A for more details about the application of FAST.

Table 24 Settlement Prices Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Integer	Numeric message type identifier: '712' - Settlement Prices
SourceTime	4	4	Binary Integer	Milliseconds since the previous Sunday 00:00 (UTC)
InfoBlockType	8	1	Binary Integer	Type of Info Block '1' - Connect Commodity
InfoBlock	9	7	ASCII String	Identifies the entity (format indicated by InfoBlockType field). If InfoBlockType = 1 then the format will be 1 char Exchange Code, 1 char Generic Contract Type, 3 chars Product Code.
UpdateType	16	2	Binary Integer	Type of update: ■ '1' - Provisional Daily ■ '2' - Official Daily ■ '3' - Provisional Market Close

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				■ '4' - Official Market Close ■ '5' - Provisional Expiry ■ '6' - Official Expiry ■ '7' - Provisional IntraDay ■ '8' - Official IntraDay ■ '10' - Official YDSP
UpdateCount	18	2	Binary Integer	Number of updates. Indicates number of times the following group of four fields (SymbolIndex, SecurityIDSource, SecurityID and Price) will be repeated in the message.
> SymbolIndex	-	4	Binary Integer	Index of the symbol Set to '0' if the instrument uses a SecurityID
> SecurityIDSource	-	1	Binary Integer	Type of the security code '8' - AMR Set to '0' if the instrument uses a SymbolIndex
> SecurityID	-	15	ASCII String	Security code (source of code indicated by SecurityIDSource field) If SecurityIDSource = 8 then this field will contain the AMR code. Set to all spaces if instrument uses a SymbolIndex
> Price	-	4	Binary Signed Integer	Settlement price.

The UpdateCount is used to specify multiple updates for a given settlement type to be disseminated. For example settlement prices for all AEX futures can be updated in a single message.

InfoBlockType and InfoBlock are used to define standard (product) information for the instruments within the message.

The following types of settlement prices can be transmitted (indicated by the UpdateType field):

- Daily - Used for daily margining and settlement calculations, and may be published before the market closes.
- Market Close - Published at the end of each day's trading in the contract. No further trades will be sent after the Market Close has been received.
- Expiry - Exchange Delivery Settlement Price (EDSP). Final official settlement prices for an expiring contract, and are published as soon as a contract finishes trading on its expiry day.
- Intraday - Disseminated in the same way as settlement prices. The intraday settlements are differentiated from daily settlements.

- YDSP – Yesterday Daily Settlement Price. Previous trading day's daily settlement price.

Where relevant, provisional settlement prices can be sent prior to the publication of official prices. These are sent with UpdateType equal to 1, 3, 5 or 7.

5.2.5 Outright Standing Data – 722 Message

5.2.5.1 Message Overview

An 'Outright Standing Data' message provides standing (or referential) data for each outright instrument.

5.2.5.2 Message Sending Rules

This message is transmitted by the system during the referential data download at the start of the day, or when there is an intra-day change to the list of valid outright markets available for a particular product. The feed may also send standing data following a system failure/recovery.

A single outright standing data update message will be sent for every outright (future or option) instrument that will update on the feed.

5.2.5.3 Message Structure

The table below describes the body fields of an Outright Standing Data message, MsgType = '722'. Note that this is the message content prior to the application of FAST. Some fields may not appear once FAST has been applied and only included for reference, for example MsgSize. See Appendix A for more details about the application of FAST.

Table 25 Outright Standing Data Message

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Integer	Numeric message type identifier: '722' - Outright standing data
SourceTime	4	4	Binary Integer	Milliseconds since the previous Sunday 00:00 (UTC)
SymbolIndex	8	4	Binary Integer	Index of the symbol Set to '0' if the instrument uses a SecurityID
ExchangeCode	12	1	ASCII Char	Single character code that indicates the type of products traded on the exchange.
ProductCode	13	3	ASCII String	Physical product code.
ExpiryDate	16	4	Binary Integer	Date of expiry. Format is YYYYMMDD. For contracts with one expiry per month the day component may be "00". For Flex contracts (AtomX) value is the exact expiry day
ExercisePrice	20	4	Binary Integer	Exercise Price.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				Will be zero for futures.
LotSize	24	4	Binary Integer	Number of items of underlying in one lot of the contract.
ContractType	28	1	ASCII Char	Generic contract type: ■ 'F' – Future ■ 'O' – Option ■ 'S' – Stock ■ 'I' – Index ■ 'X' – Currency
OptionType	29	1	ASCII Char	For options only: ■ 'C' - Call ■ 'P' - Put
Filler	30	1	-	Reserved for future use
NoSecurityIDs	31	1	Binary Integer	Number of alternate security IDs given. Indicates number of times the group of two fields (SecurityIDSource and SecurityID) will be repeated in the message. Set to '0' if the instrument uses a SymbolIndex
> SecurityIDSource	-	1	Binary Integer	Type of the security code: ■ '8' – AMR ■ '14' – Instrument Code 2
> SecurityID	-	15	ASCII String	Security code (source of code indicated by SecurityIDSource field) If SecurityIDSource = 8 then this field will contain the AMR code. If SecurityIDSource = 14 then this field will contain Instrument Code 2, an alternative code used by the Clearing System

The NoSecurityIDs field indicates the different codes used to represent the instrument. This allows multiple references to be connected to the instrument.

Note that the AMR code will always be the first SecurityID in the repeating group.

An additional Instrument Code 2 identifier will be provided for some options instruments to allow reconciliation with the Clearing System. This will always be in addition to the AMR code which remains the principal instrument identifier.

5.2.6 Strategy Standing Data – 732 Message

5.2.6.1 Message Overview

A 'Strategy Standing Data' message provides standing (or referential) data for each strategy instrument.

5.2.6.2 Message Sending Rules

This message is transmitted by the system during the referential data download at the start of the day, or when there is an intra-day change to the list of valid strategy markets available for a particular product. The feed may also send standing data following a system failure/recovery.

A single strategy standing data update message will be sent for every strategy instrument that will update on the feed.

5.2.6.3 Message Structure

The table below describes the body fields of a Strategy Standing Data message, MsgType = '732'. Note that this is the message content prior to the application of FAST. Some fields may not appear once FAST has been applied and only included for reference, for example MsgSize. See Appendix A for more details about the application of FAST.

Table 26 Strategy Standing Data Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Integer	Numeric message type identifier: '732' - Strategy standing data
SourceTime	4	4	Binary Integer	Milliseconds since the previous Sunday 00:00 (UTC)
SymbolIndex	8	4	Binary Integer	Index of the symbol Set to '0' if the instrument uses a SecurityID
ExchangeCode	12	1	ASCII Char	Single character code that indicates the type of products traded on the exchange.
ProductCode	13	3	ASCII String	Physical product code.
ExpiryDate	16	4	Binary Integer	Date of expiry. Format is YYYYMMDD. For contracts with one expiry per month the day component may be "00".
ContractType	20	1	ASCII Char	Generic contract type: 'F' - Future 'O' - Option
StrategyCode	21	1	ASCII String	Exchange-recognized market code For example 'E' = Calendar Spreads
Filler	22	1	-	Reserved for future use

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
NoSecurityIDs	23	1	Binary Integer	Number of alternate security IDs given. Indicates number of times the group of two fields (SecurityIDSource and SecurityID) will be repeated in the message. Set to '0' if the instrument uses a SymbolIndex
> SecurityIDSource	-	1	Binary Integer	Type of the security code '8' - AMR
> SecurityID	-	15	ASCII String	Security code (source of code indicated by SecurityIDSource field) If SecurityIDSource = 8 then this field will contain the AMR code.
NumLegs	-	1	Binary Integer	Number of legs in the strategy. Maximum 32.
Filler	-	3	-	Reserved for future use.
> LegSymbolIndex	-	4	Binary Integer	Index of the symbol for the leg. Set to '0' if the leg instrument uses a LegSecurityID
> LegPrice		4	Binary Integer	Price of underlying leg for a delta neutral strategy.
> LegRatio	-	4	Binary Integer	Ratio of lots for the leg. For contingent trades, the delta. Should be used in conjunction with LegRatioScaleCode.
> LegRatioScaleCode	-	1	Binary Integer	Scale code for LegRatio field.
> LegBuySell	-	1	ASCII Char	'B' - Buy 'S' - Sell
> Filler	-	1	-	Reserved for future use.
> NoLegSecurityIDs	-	1	Binary Integer	Number of alternate security IDs given for the strategy leg. Indicates number of times the following group of two fields (LegSecurityIDSource and LegSecurityID) will be repeated in the message. Set to '0' if the leg instrument uses a SymbolIndex
> > LegSecurityIDSource	-	1	Binary	Type of the security code

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
			Integer	'8' - AMR
> > LegSecurityID	-	15	ASCII String	Security code (source of code indicated by LegSecurityIDSource field) If LegSecurityIDSource = 8 then this field will contain the AMR code.

The NoSecurityIDs field indicates the different codes used to represent the instrument. This allows multiple references to be connected to the strike.

5.2.7 Product Availability – 741 Message

5.2.7.1 Message Overview

A 'Product Availability' message provides information about the availability of each product.

5.2.7.2 Message Sending Rules

This message indicates whether individual products are available or unavailable for trading. It is transmitted at start of day and whenever the product's availability changes.

5.2.7.3 Message Structure

The table below describes the body fields of a Product Availability message, MsgType = '741'. Note that this is the message content prior to the application of FAST. Some fields may not appear once FAST has been applied and only included for reference, for example MsgSize. See Appendix A for more details about the application of FAST.

Table 27 Product Availability Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Integer	Numeric message type identifier: '741' - Product Availability
SourceTime	4	4	Binary Integer	Milliseconds since the previous Sunday 00:00 (UTC)
InfoBlockType	8	1	Binary Integer	Type of Info Block '1' - Connect Commodity
InfoBlock	9	7	ASCII String	Identifies the entity (format indicated by InfoBlockType field). If InfoBlockType = 1 then the format will be 1 char Exchange Code, 1 char Generic Contract Type, 3 chars Product Code.
TradingDay	16	4	Binary Integer	Trading date of the logical trading day. Format is YYYYMMDD.
TradingAvailableFlag	20	1	Binary Integer	Indicates product availability: ■ '0' - Product is not available for trading ■ '1' - Product is available for trading
Filler	21	3	-	Reserved for future use.
ThrottleSize	24	4	Binary Integer	Flag to identify the number of order messages permitted per second

InfoBlockType and InfoBlock is used to define standard (product) information for the instruments within the message.

Upon receiving a Product Availability message with TradingAvailableFlag equal to 0 or 1, the user should clear all order books for that product. For example if a Product Availability message is received with InfoBlock = KOAEX, the client should clear the order books for all AEX Index Options.

A Product Availability message with TradingAvailableFlag = 1 will always be shortly followed by a snapshot of market data, as set out in System Behaviour on Start and Restart.

5.2.8 Market Status – 752 Message

5.2.8.1 Message Overview

A 'Market Status' message is sent when there is a change to the market mode of a product, expiry or series.

5.2.8.2 Message Sending Rules

Market mode changes may be notified for a product, an expiry or series.

This message is also used to notify the market of session changes.

5.2.8.3 Message Structure

The table below describes the body fields of a Market Status message, MsgType = '752'. Note that this is the message content prior to the application of FAST. Some fields may not appear once FAST has been applied and only included for reference, for example MsgSize. See Appendix A for more details about the application of FAST.

Table 28 Market Status Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Integer	Numeric message type identifier: '752' - Market Status
SourceTime	4	4	Binary Integer	Milliseconds since the previous Sunday 00:00 (UTC)
SymbolIndex	8	4	Binary Integer	Index of the symbol Set to '0' if the instrument uses a SecurityID
SecurityIDSource	12	1	Binary Integer	Type of the security code (according to level at which the market mode applies – market, expiry or product): '8' - AMR '9' - Expiry '10' - Product Set to '0' if market status change is at market (individual future/option) level and the instrument uses a SymbolIndex.
SecurityID	13	15	ASCII String	Security code (source of code indicated by SecurityIDSource field) If SecurityIDSource = 8 then this field will contain the AMR code. If SecurityIDSource = 9 then the format will be 1 char Exchange Code, 1 char Generic Contract Type, 3 chars Product Code, 8 chars Expiry Date (format YYYYMMDD). If SecurityIDSource = 10 then the format will be

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				1 char Exchange Code, 1 char Generic Contract Type, 3 chars Product Code. Set to all spaces if market status change is at market (individual future/option) level and instrument uses a SymbolIndex.
NoUpdates	28	2	Binary Integer	Number of market mode updates for the specified SecurityID. Indicates number of times the following field (Market Mode) will be repeated in the message.
> MarketMode	-	2	Binary Integer	Market mode setting: '1' - Closed '4' - ExPit Extend Open '6' - Halted '7' - Open '8' - Pre Closed '9' - Pre Open '10' - Price Limits Enabled '11' - Price Limits Disabled '12' - Restricted Open '13' - Session 1 '14' - Session 2 '15' - Session 3 '23' - Quote Width Exemption 1 '24' - Quote Width Exemption 2 '25' - Quote Width Exemption 3 '28' - Dark Series '29' - Light Series '30' - Trading Unhalt '31' - Terminate '32' - Un-Terminate '39' - Expire '40' - Pre-Expiry '41' - Hold '42' - Unhold '51' - ExPit Large in Scale (LiS) trades extend open '52' - ExPit Basis trades extend open '53' - ExPit Against Actuals trades extend open '55' - ExPit Large in Scale (LiS) Package trades extend open '57' - ExPit Exchange For Swaps trades extend

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				open

Market mode changes may be notified for a product, an expiry or a series (future/option).

If a market mode change is transmitted for a product, this indicates that all individual outright and strategy markets in that product have changed. In this case the SecurityID field will contain the Exchange Code, Generic Contract Type and Product Code. For example, for CAC 40 futures this field will take the value 'FCE '.

If a market mode change is transmitted for an expiry, this indicates that all individual outright markets in that product expiry month, and in some cases strategy markets with at least one leg in that product expiry month, have changed. In this case the SecurityID field will contain the Exchange Code, Generic Contract Type, Product Code and Expiry Date.

If a market mode change is transmitted for a series, this indicates that only the individual outright/strategy instrument has undergone a change of state. In this case either the SecurityID field will contain the either the AMR code or the SymbolIndex field will be filled.

Note that the format of this message allows multiple mode changes for the same entity to be communicated in a single message.

- The Closed mode indicates that the product/expiry/series is in a Closed market state.
- ExPit Extend Open mode indicates that Wholesale trading is currently permitted in the product/expiry/series (which can apply to certain trade types only).
- Halted mode indicates that the product/expiry/series is in a Trading halt market state.
- Open mode indicates that the product/expiry/series is in an Open market state.
- Pre Closed mode indicates that the product/expiry/series is about to move into a Closed market state.
- Pre Open mode indicates that the product/expiry/series is in a Pre Open market state.
- Restricted Open mode indicates that the product/expiry/series is in a Restricted Open state.
- Price Limits Enabled/Disabled modes indicate the dynamic price limits are enabled/disabled for the product/expiry/series.
- Session 1 to 3 modes indicate the current trading session.
- Quote Width Exemption 1 – standard legal widths are in effect.
- Quote Width Exemption 2/3 – extended legal widths are in effect (refer to market maker agreements for details).
- Dark Series – the product/expiry/series is dark, market data will not be published
- Light Series – the product/expiry/series is light (non-dark)
- Trading Unhalt – Trading has resumed for the indicated product/expiry/series.
- Terminate – indicates that the product/expiry/series has been suspended for trading. In this case all orders (including GTC Orders) that remain unmatched in the central order book will be pulled automatically by the Trading Host, and no new orders will be accepted.

- Un-Terminate – indicates that the product/expiry/series has been unsuspended.
- Expired mode indicates that the product/expiry/series has moved into the expired market state.
- Pre expiry mode indicates that product/expiry/series is about to move into an expired market state.
- Hold – the product/expiry/series is held
- Unhold – the product/expiry/series is not held
- ExPit Large in Scale (LiS) trades extend open mode indicates Large in Scale trading is currently permitted in the product/expiry/series.
- ExPit Basis trades extend open mode indicates Basis trading is currently permitted in the product/expiry/series.
- ExPit Against Actuals trades extend open mode indicates Against Actuals trading is currently permitted in the product/expiry/series.
- ExPit Large in Scale (LiS) Package trades extend open mode indicates Large in Scale Package trading is currently permitted in the product/expiry/series.
- ExPit Exchange For Swaps trades extend open mode indicates Exchange For Swaps trading is currently permitted in the product/expiry/series.

5.2.9 Exchange Message – 761 Message

5.2.9.1 Message Overview

An 'Exchange Message' is used to process one of a number of predefined messages and text messages from Exchange officials.

5.2.9.2 Message Sending Rules

The text of the message can be in English or the exchange local language.

5.2.9.3 Message Structure

The table below describes the body fields of an Exchange Message, MsgType = '761'. Note that this is the message content prior to the application of FAST. Some fields may not appear once FAST has been applied and only included for reference, for example MsgSize. See Appendix A for more details about the application of FAST.

Table 29 Exchange Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Integer	Numeric message type identifier: '761' - Exchange Message
SourceTime	4	4	Binary Integer	Milliseconds since the previous Sunday 00:00 (UTC)
InfoBlockType	8	1	Binary Integer	Type of Info Block '1' - Connect Commodity Will be filled if the exchange message is associated with a particular product.
InfoBlock	9	7	ASCII String	Identifies the entity (format indicated by InfoBlockType field). If InfoBlockType = 1 then the format will be 1 char Exchange Code, 1 char Generic Contract Type, 3 chars Product Code. Will be filled if the exchange message is associated with a particular product.
NewsCount	16	4	Binary Integer	Count of text messages. Indicates number of times the following group of fields (MsgType, Importance, TextFormat and Text) will be repeated in the message.
> ExchangeMsgType	-	1	ASCII Char	Generic message classification T = Text message
> Importance	-	1	Binary	Importance of the message, values

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
			Integer	from 1-99: ■ '1' - most important ■ '99' - least important
> TextFormat	-	1	Binary Integer	Format of the Text field '1' - ASCII
> Filler	-	1	-	Reserved for future use.
> Text	-	250	Refer to TextFormat field	Message text.
> Filler	-	2	-	Reserved for future use.

InfoBlockType and InfoBlock is used to define standard (product) information for the instruments within the message.

5.2.10 Value-added Parameters – 772 Message

5.2.10.1 Message Overview

A 'Value-Added Parameters' provides value-added information about outright or strategy instruments.

5.2.10.2 Message Sending Rules

A Value-Added Parameters message is sent as a result of a change in one of the following parameters:

- Daily high/low price
- Yearly high/low price
- Lifetime high/low price
- Open price
- Trade count
- Percentage change
- Last trade price

5.2.10.3 Message Structure

The table below describes the body fields of a Value-Added Parameters message, MsgType = '772'. Note that this is the message content prior to the application of FAST. Some fields may not appear once FAST has been applied and only included for reference, for example MsgSize. See Appendix A for more details about the application of FAST.

Table 30 Value-added Parameters Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Integer	Numeric message type identifier: '772' - Value-Added Parameters
SymbolIndex	4	4	Binary Integer	Index of the symbol Set to '0' if the instrument uses a SecurityID
SecurityIDSource	8	1	Binary Integer	Type of the security code '8' - AMR Set to '0' if the instrument uses a SymbolIndex
SecurityID	9	15	ASCII String	Security code (source of code indicated by SecurityIDSource field) If SecurityIDSource = 8 then this field will contain the AMR code. Set to all spaces if instrument uses a SymbolIndex
SourceTime	24	4	Binary Integer	Milliseconds since the previous Sunday 00:00 (UTC)
UpdateCount	28	2	Binary	Number of updates. Indicates number of

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
			Integer	times the following group of three fields (Update Type, Price and Volume) will be repeated in the message.
Filler	30	2	-	Reserved for future use
> UpdateType	-	2	Binary Integer	Type of update: ■ '1' - Daily High ■ '2' - Daily Low ■ '3' - Yearly High ■ '4' - Yearly Low ■ '5' - Lifetime High ■ '6' - Lifetime Low ■ '10' - Open Price ■ '11' - Trade Count ■ '12' - Percentage Change ■ '13' - Last Trade Price
> Filler	-	2	-	Reserved for future use
> Price	-	4	Binary Signed Integer	Price. See notes below for full explanation of contents.
> Volume	-	4	Binary Integer	Volume. See notes below for full explanation of contents.

The UpdateCount is used to specify multiple price/volume updates for the instrument, included within the message.

For example, a trade execution could result in updates to the last trade price, high price, trade count and percentage change.

Trades are classified as either 'book' or 'ex-pit' as follows (702 message Update Type provided in brackets):

- Book trades – Conventional Trade (6), Guaranteed Cross (10), Strategy Leg Conventional Trade (29), Strategy Leg Guaranteed Cross Trade (33)
- Ex-pit trades – Large in Scale (LiS) Trade (7), Basis Trade (8), Large in Scale (LiS) Package Trade (9), Against Actual Trade (11), Asset Allocation Trade (12), External Match Trade (13), Exchange For Swap Trade (14), Exchange For Physical Trade (15), Strategy Leg Large in Scale (LiS) Trade (30), Strategy Leg Basis Trade (31), Strategy Leg Against Actual Trade (34), Strategy Leg Asset Allocation Trade (35), Strategy Leg External Match Trade (36), Strategy Leg Exchange For Swap Trade (37), Strategy Leg Exchange For Physical Trade (38).

The content of the price/volume fields depends on the value of the field UpdateType as follows:

- Daily High/Low - Highest/Lowest trade price for the current trading day for book trades only.
- Yearly High/Low - Highest/Lowest trade price for the current year for book trades only.
- Lifetime High/Low - Highest/lowest trade price for the lifetime of the instrument for book trades only.
- Open Price - Price of the opening book trade for the instrument.
- Trade Count - Number of book trades matched over the course of the current trading day.
- Percentage Change - The percentage change between the last traded book price the previous day settlement price. Note that the value of the price field will be an integer. The last 2 digits represent decimal points, so for example a field value of 325 will represent a change of 3.25%.
- Last Trade Price – The price of the last book trade.

The content of the price/volume fields depends on the value of the field UpdateType as follows:

Table 31 Price and Volume Contents

UPDATE TYPE	PRICE	VOLUME
Daily High/Low	Highest/Lowest Trade for the Day	N/A
Yearly High/Low	Highest/Lowest Trade for the Year	N/A
Lifetime High/Low	Highest/Lowest Trade for the instrument lifetime	N/A
Open Price	First Trade Price for the Day	N/A
Trade Count	N/A	Number of Trades Matched in the Day
Percentage Change	Percentage Change for the Day (2 decimal places)	N/A
Last Trade Price	Last Trade Price	N/A

5.2.11 Open Interest – 782 Message

5.2.11.1 Message Overview

An 'Open Interest' message provides open interest information for outright instruments.

5.2.11.2 Message Sending Rules

An Open Interest 782 message is sent when the open interest has been calculated. The date to which the open interest value is provided.

5.2.11.3 Message Structure

The table below describes the body fields of an Open Interest message, MsgType = '782'. Note that this is the message content prior to the application of FAST. Some fields may not appear once FAST has been applied and only included for reference, for example MsgSize. See Appendix A for more details about the application of FAST.

Table 32 Open Interest Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Integer	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Integer	Numeric message type identifier: '782' - Open Interest
SourceTime	4	4	Binary Integer	Milliseconds since the previous Sunday 00:00 (UTC)
InfoBlockType	8	1	Binary Integer	Type of Info Block '1' - Connect Commodity
InfoBlock	9	7	ASCII String	Identifies the entity (format indicated by InfoBlockType field). If InfoBlockType = 1 then the format will be 1 char Exchange Code, 1 char Generic Contract Type, 3 chars Product Code.
UpdateCount	16	2	Binary Integer	Number of updates. Indicates number of times the following group of fields (SymbolIndex, SecurityIDSource, SecurityID, OpenInterest and OpenInterestDate) will be repeated in the message.
Filler	18	2	-	Reserved for future use
> SymbolIndex	-	4	Binary Integer	Index of the symbol Set to '0' if the instrument uses a SecurityID
> SecurityIDSource	-	1	Binary Integer	Type of the security code '8' - AMR Set to '0' if the instrument uses a SymbolIndex
> SecurityID	-	15	ASCII	Security code (source of code indicated by

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
			String	SecurityIDSource field) If SecurityIDSource = 8 then this field will contain the AMR code. Set to all spaces if instrument uses a SymbolIndex
> OpenInterest	-	4	Binary Integer	Open interest.
> OpenInterestDate	-	4	Binary Integer	Open interest date Format is YYYYMMDD.

The UpdateCount is used to specify multiple open interest values in a single message (for a single product)..

InfoBlockType and InfoBlock is used to define standard (product) information for the instruments within the message.

APPENDIX A: FAST ENCODING

A.1 OVERVIEW

FAST encoding will be used to minimize the bandwidth required for the EURONEXT DERIVATIVES XDP feed.

FAST version 1.2 will be used. The latest FAST specifications can be found at:
http://www.fixprotocol.org/fastspec

A.2 MESSAGE FORMAT

Each market data packet has the following format (excluding IP/UDP headers):

PACKET HEADER	FAST ENCODED MSG	FAST ENCODED MSG	...	FAST ENCODED MSG

A packet header is followed by one or more FAST encoded messages.

The packet header is never FAST encoded. It does indicate how many FAST encoded messages are available.

Each FAST encoded message has the following format:

PMAP	TEMPLATE ID	FAST ENCODED FIELD	...	FAST ENCODED FIELD

The PMAP field is the Presence Map.

The Template ID indicates which template has been employed to encode this message. One or more FAST encoded fields follow the Template ID.

The FAST dictionary should be reset before processing each new packet. No knowledge of previous values in prior packets is required for decoding a FAST encoded packet.

Copy and Delta operators will apply across all messages within the same packet, and not just between messages of the same type. Therefore these operators will apply between fields of the same name in different FAST templates.

A.3 FAST TEMPLATES

The following FAST templates have been defined for the Euronext Derivatives XDP feed:

- 1 – Market Update
- 11 – Settlement Prices
- 12 – Outright Standing Data
- 13 – Strategy Standing Data
- 14 – Product Availability
- 15 – Market Status
- 16 – Exchange Message
- 17 – Value Added Parameters
- 20 – Open Interest

This appendix will describe the wire format for each FAST encoded message. NOTE: The wire format includes all of the fields that should be expected in the FAST encoded message.

Each message will have its own table describing the wire format. This attempts to detail on a field-by-field basis what can be expected for each encoded message. The table format contains the following columns:

Table 33 Wire Format Table Columns

COLUMN NAME	COMMENTS
Field Name	Assigned Field Name
PMAP Bit	Indicates the Presence map bit this field corresponds to, if applicable. NOTE: The PMAP field itself will never be in the presence map, it is always be present in the message. Other fields marked N/A will always be present in the message.
Presence	Indicates if the field presence is mandatory, optional or not applicable. NOTE: The PMAP field itself does not have a presence attribute.
Field Operator	Indicates the field operator that should be applied to this field. The field operators used consist of: Copy Default Delta Constant NOTE: The PMAP field itself will never have a field operator. It will always be present in the message.
Field Type	Indicates the type of field. The field types consist of: uInt32 (4 byte unsigned integers) int32 (4 byte signed integers) ASCII Strings (fixed length ASCII strings) pmap (Presence map field)
Comments	This field provides general information on the field.

A.3.1 Market Update – 702 Message

The table below describes the wire format of a FAST encoded Market Update (702) message.

Table 34 FAST Format for Market Update Message

FIELD NAME		PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
PMAP		N/A	N/A	N/A	pmap	Presence Bit 1 – TemplateId Presence Bit 2 – SymbolIndex Presence Bit 3 – SecurityIDSource Presence Bit 4 – SecurityID Presence Bit 5 – SnapshotFlag Presence Bit 6 – UpdateCount
TemplateId		1	Mandatory	Copy	uInt32	Value is '1'
MsgType		N/A	Mandatory	Const	uInt32	Const value is '702'
SymbolIndex		2	Mandatory	Default	uInt32	Default value is '0'
SecurityIDSource		3	Mandatory	Default	uInt32	Default value is '0'
SecurityID		4	Mandatory	Default	ASCII String	Maximum of 15 characters Default value is 15 spaces
SourceTime		N/A	Mandatory	Delta	Int32	
SeriesSequenceNumber		N/A	Mandatory	N/A	uInt32	
SnapshotFlag		5	Mandatory	Default	uInt32	Default value is '0'
UpdateCount		6	Mandatory	Default	uInt32	Default value is '1'
Update Sequence	> UpdateType	N/A	Mandatory	N/A	uInt32	
	> Strategy Code	N/A	Optional	N/A	ASCII String	Maximum of 1 character
	> Price	N/A	Optional	Delta	int32	
	> Volume	N/A	Mandatory	N/A	uInt32	

The UpdateCount field indicates the number of Update Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the Update Sequence. The position of the length field will coincide with the UpdateCount field, which is why it is represented as a separate field in the table.

A.3.2 Settlement Prices – 712 Message

The table below describes the wire format of a FAST encoded Settlement Prices (712) message.

Table 35 FAST Format for Settlement Prices Message

FIELD NAME		PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
PMAP		N/A	N/A	N/A	pmap	Presence Bit 1 – TemplateId Presence Bit 2 – InfoBlockType
TemplateId		1	Mandatory	Copy	uInt32	Value is '11'
MsgType		N/A	Mandatory	Const	uInt32	Const value is '712'
SourceTime		N/A	Mandatory	Delta	Int32	
InfoBlockType		2	Mandatory	Default	uInt32	Default value is '1'
InfoBlock		N/A	Mandatory	N/A	ASCII String	Maximum of 7 characters
UpdateType		N/A	Mandatory	N/A	uInt32	
UpdateCount		N/A	Mandatory	N/A	uInt32	
Update Sequence	> PMAP	N/A	N/A	N/A	pmap	Presence Bit 1 – SymbolIndex Presence Bit 2 – SecurityIDSource Presence Bit 3 – SecurityID
	> SymbolIndex	1	Mandatory	Default	uInt32	Default value is '0'
	> SecurityIDSource	2	Mandatory	Default	uInt32	Default value is '0'
	> SecurityID	3	Mandatory	Default	ASCII String	Maximum of 15 characters Default value is 15 spaces
	> Price	N/A	Optional	Delta	int32	

The UpdateCount field indicates the number of Update Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the Update Sequence. The position of the length field will coincide with the UpdateCount field, which is why it is represented as a separate field in the table. Each Update Sequence will have its own PMAP.

A.3.3 Outright Standing Data – 722 Message

The table below describes the wire format of a FAST encoded Outright Standing Data (722) message.

Table 36 FAST Format for Outright Standing Data Message

FIELD NAME	PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
PMAP	N/A	N/A	N/A	pmap	Presence Bit 1 – TemplateId Presence Bit 2 – SymbolIndex Presence Bit 3 – ExchangeCode Presence Bit 4 – ProductCode Presence Bit 5 – ExpiryDate Presence Bit 6 – LotSize Presence Bit 7 – ContractType Presence Bit 8 – OptionType Presence Bit 9 – NoSecurityIDs
TemplateId	1	Mandatory	Copy	uInt32	Value is '12'
MsgType	N/A	Mandatory	Const	uInt32	Const value is '722'
SourceTime	N/A	Mandatory	Delta	Int32	
SymbolIndex	2	Mandatory	Default	uInt32	Default value is '0'
ExchangeCode	3	Mandatory	Copy	ASCII String	Maximum of 1 character
ProductCode	4	Mandatory	Copy	ASCII String	Maximum of 3 characters
ExpiryDate	5	Mandatory	Copy	uInt32	
ExercisePrice	N/A	Optional	Delta	uint32	
LotSize	6	Mandatory	Copy	uInt32	
ContractType	7	Mandatory	Copy	ASCII String	Maximum of 1 character
OptionType	8	Optional	Copy	ASCII	Maximum of 1

FIELD NAME		PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
					String	characters
NoSecurityIDs		9	Mandatory	Default	uInt32	Default value is '0'
SecurityID Sequence	**> SecurityIDSource**	N/A	Mandatory	N/A	uInt32	
	> SecurityID	N/A	Mandatory	N/A	ASCII String	

The NoSecurityIDs field indicates the number of Update Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the SecurityID Sequence. The position of the length field will coincide with the NoSecurityIDs field, which is why it is represented as a separate field in the table.

The SecurityID Sequence has no PMAP.

A.3.4 Strategy Standing Data – 732 Message

The table below describes the wire format of a FAST encoded Strategy Standing Data (732) message.

Table 37 FAST Format for Strategy Standing Data Message

FIELD NAME	PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
PMAP	N/A	N/A	N/A	pmap	Presence Bit 1 – TemplateId Presence Bit 2 – Symbol Index Presence Bit 3 – Exchange Code Presence Bit 4 – Product Code Presence Bit 5 – ExpiryDate Presence Bit 6 – Contract Type Presence Bit 7 – Strategy Code Presence Bit 8 – NoSecurity IDs
TemplateId	1	Mandatory	Copy	uInt32	Value is ‘13’
MsgType	N/A	Mandatory	Const	uInt32	Const value is ‘732’
SourceTime	N/A	Mandatory	Delta	Int32	
SymbolIndex	2	Mandatory	Default	uInt32	Default value is ‘0’
ExchangeCode	3	Mandatory	Copy	ASCII String	Maximum of 1 character
ProductCode	4	Mandatory	Copy	ASCII String	Maximum of 3 characters
ExpiryDate	5	Mandatory	Copy	uInt32	
ContractType	6	Mandatory	Copy	ASCII String	Maximum of 1 character

FIELD NAME			PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
StrategyCode			7	Mandatory	Copy	ASCII String	Maximum of 1 character
NoSecurityIDs			8	Mandatory	Default	uInt32	Default value is '0'
SecurityID Sequence	> SecurityIDSource		N/A	Mandatory	N/A	uInt32	
	> SecurityID		N/A	Mandatory	N/A	ASCII String	
NumLegs			N/A	Mandatory	N/A	uInt32	
Leg Sequence	> PMAP		N/A	N/A	N/A	pmap	Presence Bit 1 – LegSymbol Index Presence Bit 2 – LegPrice Presence Bit 3 – LegRatio Presence Bit 4 – LegRatio ScaleCode Presence Bit 5 – LegBuySell Presence Bit 6 – NoLeg SecurityIDs
	> LegSymbolIndex		1	Mandatory	Default	uInt32	Default value is '0'
	> LegPrice		2	Mandatory	Copy	uInt32	
	> LegRatio		3	Mandatory	Copy	uInt32	
	> LegRatioScaleCode		4	Mandatory	Copy	uInt32	
	> LegBuySell		5	Mandatory	Copy	ASCII String	Maximum number of 1 character.
	> NoLegSecurityIDs		6	Mandatory	Default	uInt32	Default value is '0'
	Leg SecurityID Sequence	> > Leg Security ID Source	N/A	Mandatory	N/A	uInt32	
		> > LegSecurityID	N/A	Mandatory	N/A	ASCII String	

The NoSecurityIDs field indicates the number of SecurityID Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the SecurityID Sequence. The position of the length field will coincide with the NoSecurityIDs field, which is why it is represented as a separate field in the table.

The SecurityID Sequence has no PMAP.

The NumLegs field indicates the number of Leg Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the Leg Sequence. The position of the length field will coincide with the NumLegs field, which is why it is represented as a separate field in the table.

Each Leg Sequence will have its own PMAP.

The NoLegSecurityIDs field indicates the number of Update Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the LegSecurityID Sequence. The position of the length field will coincide with the NoLegSecurityIDs field, which is why it is represented as a separate field in the table.

The Leg SecurityID Sequence has no PMAP.

A.3.5 Product Availability – 741 Message

The table below describes the wire format of a FAST encoded Product Availability (741) message.

Table 38 FAST Format for Product Availability Message

FIELD NAME	PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
PMAP	N/A	N/A	N/A	pmap	Presence Bit 1 – TemplateId Presence Bit 2 – InfoBlockType Presence Bit 3 – TradingDay Presence Bit 4 – TradingAvailableFlag Presence Bit 5 – ThrottleSize
TemplateId	1	Mandatory	Copy	uInt32	Value is ‘14’
MsgType	N/A	Mandatory	Const	uInt32	Const value is ‘741’
SourceTime	N/A	Mandatory	Delta	Int32	
InfoBlockType	2	Mandatory	Default	uInt32	Default value is ‘1’
InfoBlock	N/A	Mandatory	N/A	ASCII String	Maximum of 7 characters
TradingDay	3	Mandatory	Copy	uInt32	
TradingAvailableFlag	4	Mandatory	Copy	uInt32	
ThrottleSize	5	Mandatory	Copy	uInt32	

A.3.6 Market Status – 752 Message

The table below describes the wire format of a FAST encoded Market Status (752) message.

Table 39 FAST Format for Market Status Message

FIELD NAME		PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
PMAP		N/A	N/A	N/A	pmap	Presence Bit 1 – TemplateId Presence Bit 2 – SymbolIndex Presence Bit 3 – SecurityIDSource Presence Bit 4 – SecurityID
TemplateId		1	Mandatory	Copy	uInt32	Value is '15'
MsgType		N/A	Mandatory	Const	uInt32	Const value is '752'
SourceTime		N/A	Mandatory	Delta	Int32	
SymbolIndex		2	Mandatory	Default	uInt32	Default value is '0'
SecurityIDSource		3	Mandatory	Default	uInt32	Default value is '0'
SecurityID		4	Mandatory	Default	ASCII String	Maximum of 15 characters Default value is 15 spaces
NoUpdates		N/A	Mandatory	N/A	uInt32	
Mode Sequence	> MarketMode	N/A	Mandatory	N/A	uInt32	

The NoUpdates field indicates the number of Mode Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the Mode Sequence. The position of the length field will coincide with the NoUpdates field, which is why it is represented as a separate field in the table.

Each Mode Sequence will have no PMAP.

A.3.7 Exchange Message – 761 Message

The table below describes the wire format of a FAST encoded Exchange Message (761) message.

Table 40 FAST Format for Exchange Message

FIELD NAME		PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
PMAP		N/A	N/A	N/A	pmap	Presence Bit 1 – TemplateId Presence Bit 2 – InfoBlockType Presence Bit 3 – NewsCount
TemplateId		1	Mandatory	Copy	uInt32	Value is '16'
MsgType		N/A	Mandatory	Const	uInt32	Const value is '761'
SourceTime		N/A	Mandatory	Delta	Int32	
InfoBlockType		2	Mandatory	Default	uInt32	Default value is '1'
InfoBlock		N/A	Mandatory	N/A	ASCII String	Maximum of 7 characters
NewsCount		3	Mandatory	Default	uInt32	Default value is '1'
News Sequence	> PMAP	N/A	N/A	N/A	pmap	Presence Bit 1 – MsgType Presence Bit 2 – TextFormat
	> ExchangeMsgType	1	Mandatory	Default	ASCII String	Default value is 'T'
	> Importance	N/A	Mandatory	N/A	uInt32	
	> TextFormat	2	Mandatory	Default	uInt32	Default value is '1'
	> Text	N/A	Mandatory	N/A	ASCII String	Maximum of 250 characters

The NewsCount field indicates the number of News Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the News Sequence. The position of the length field will coincide with the NewsCount field, which is why it is represented as a separate field in the table.

Each News Sequence will have its own PMAP.

A.3.8 Value-added Parameters – 772 Message

The table below describes the wire format of a FAST encoded Value-added Parameters (772) message.

Table 41 FAST Format for Value-added Parameters Message

FIELD NAME		PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
PMAP		N/A	N/A	N/A	pmap	Presence Bit 1 – TemplateId Presence Bit 2 – SymbolIndex Presence Bit 3 – SecurityIDSource Presence Bit 4 – SecurityID Presence Bit 5 – UpdateCount
TemplateId		1	Mandatory	Copy	uInt32	Value is '17'
MsgType		N/A	Mandatory	Const	uInt32	Const value is '772'
SymbolIndex		2	Mandatory	Default	uInt32	Default value is '0'
SecurityIDSource		3	Mandatory	Default	uInt32	Default value is '0'
SecurityID		4	Mandatory	Default	ASCII String	Maximum of 15 characters Default value is 15 spaces
SourceTime		N/A	Mandatory	Delta	Int32	
UpdateCount		5	Mandatory	Default	uInt32	Default value is '3'
Update Sequence	> PMAP	N/A	N/A	N/A	pmap	Presence Bit 1 – UpdateType Presence Bit 2 – Volume
	> UpdateType	1	Mandatory	Default	uInt32	Default value is '13'
	> Price	N/A	Optional	Delta	int32	
	> Volume	2	Mandatory	Default	uInt32	Default value is '0'

The UpdateCount field indicates the number of Update Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the Update Sequence. The position of the length field will coincide with the UpdateCount field, which is why it is represented as a separate field in the table.

Each Update Sequence will have its own PMAP.

A.3.9 Open Interest – 782 Message

The table below describes the wire format of a FAST encoded Open Interest (782) message.

Table 42 Fast Format for Open Interest Message

FIELD NAME		PMAP BIT	PRESENCE	FIELD OPERATOR	FIELD TYPE	COMMENTS
PMAP		N/A	N/A	N/A	pmap	Presence Bit 1 – TemplateId Presence Bit 2 – InfoBlockType
TemplateId		1	Mandatory	Copy	uInt32	Value is '20'
MsgType		N/A	Mandatory	Const	uInt32	Const value is '782'
SourceTime		N/A	Mandatory	Delta	Int32	
InfoBlockType		2	Mandatory	Default	uInt32	Default value is '1'
InfoBlock		N/A	Mandatory	N/A	ASCII String	Maximum of 7 characters
UpdateCount		N/A	Mandatory	N/A	uInt32	
Update Sequence	> PMAP	N/A	N/A	N/A	pmap	Presence Bit 1 – SymbolIndex Presence Bit 2 – SecurityIDSource Presence Bit 3 – SecurityID Presence Bit 4 – OpenInterestDate
	> SymbolIndex	1	Mandatory	Default	uInt32	Default value is '0'
	> Security IDSource	2	Mandatory	Default	uInt32	Default value is '0'
	> SecurityID	3	Mandatory	Default	ASCII String	Maximum of 15 characters Default value is 15 spaces
	> OpenInterest	N/A	Mandatory	N/A	uInt32	
	> Open InterestDate	4	Mandatory	Copy	uInt32	

The UpdateCount field indicates the number of Update Sequences contained in the message. This field is not an explicit field in the template – it will be the length element in the Update Sequence. The position of the length field will coincide with the UpdateCount field, which is why it is represented as a separate field in the table.

Each Update Sequence will have its own PMAP.

APPENDIX A: REVIEW LOG, DOCUMENT HISTORY, SIGN-OFF

REVIEW LOG

DOCUMENT TITLE	Euronext Derivatives
DOCUMENT TYPE/SUBJECT	XDP Client Specifications
REVISION VERSION	2.2.

DOCUMENT HISTORY

REVISION NO	DATE	AUTHOR	CHANGE DESCRIPTION
2.0	Dec 2014		**The BBO Cash underlying equities data that is part of Service IDs 1, 2 and 24 will not be migrated to the upgraded XDP infrastructure.** Removed reference to BBO from chapters: • Euronext Derivatives Service Types • Euronext Derivatives Service ID's • Refresh Contents Option Valuation and Bclear will also not migrate to the upgraded XDP infrastructure. Removed reference to Option Valuation from chapters: • Euronext Derivatives Service Types • Euronext Derivatives Service ID's • Option Valuation – 792 Message • Refresh Contents • FAST encoded Option Valuation – 792 message Removed reference to Bclear from chapters: • Euronext Derivatives Service Types • Euronext Derivatives Service ID's • Instrument Identifiers • Market Update – 702 Message • Outright Standing Data – 722 Message • Refresh Contents Changed chapter Euronext Derivatives Product Groups, removed all non-Euronext Derivatives from the list. Changed chapter Euronext Derivatives Service ID's, removed all Service ID's that will not migrate to the upgraded XDP infrastructure. Changed chapter Sequence Numbers and chapter Market Update – 702 Message, field SeriesSequenceNumber. **Customers subscribing to Service Type Level 1 need to take into account that the SeriesSequenceNumber does not increase monotonically on the upgraded XDP infrastructure.** Changed chapters High Availability Retransmission Behaviour and High

REVISION NO	DATE	AUTHOR	CHANGE DESCRIPTION
			Availability Refresh Behaviour, removed the text referring to the secondary server. On the upgraded XDP infrastructure we will use Virtual IP Addresses for the Retransmission and Refresh Servers. Customers are no longer required to use separate IP Addresses for the Primary and Secondary RTS and RFS, only one IP Address will be needed. Changed chapter High Availability Refresh Behaviour, removed reference to Error Code 7 – Refresh request rejected as sent to incorrect server (secondary instead of primary) – which will no longer be used on the upgraded XDP infrastructure.
2.1	June 2014		Rebranded to Euronext Template
2.2	Jul 2014		Chapter 3.1.4 updated with new Service ID's.
2.2.1	Sep 2014		Correct of Service ID for Paris - Index Futures - Standing Data page 13.
2.2.2	Dec 2014		Change to the Production Time Table on page 9.
2.2.3	Feb 2015		Change to Market Update message 702, removed Update Types '15' Exchange for Physical Trade and '38' Strategy Leg Exchange For Physical on page 49. Addition of Dairy Complex products to the Product Group Euronext Commodity Derivatives.
2.2.4	Apr 2015		Addition of AtomX to paragraph 3.1.2 Service Types and 3.1.4 Euronext Derivatives Service IDs, including new Service IDs 715 and 815 for AtomX. Addition of UpdateType 39 "AtomX Trade" to the 702 Market Update message.
2.2.5	May 2015		Update of paragraph 3.1.2 (Service Type): • Addition of update type "5" to the service type "AtomX". • "Level 1" section - update type 39 is now available • "Level 1 + 2" section – update type 39 is now available
2.2.6	Jul 2015		New paragraph (5.1.10) to explain the publication process between Standard and Flex contracts concerning message XDP-702. Addition of a general overview regarding publication process between 702,712 and 722 messages
2.2.7	30 Oct 2015	VPO, BA Team – Euronext IT	Change in the possible values of the 'MarketMode' field within the 752 message.
2.2.8	04 Feb 2016	PCH, BA Team – Euronext IT	Added the following values for Update Type in Market Update Message (702) ■ '41' - Request for Cross Activated ■ '42' - Request for Cross Queued ■ '43' - Request for Cross Trade ■ '44' - Strategy Leg Request for Cross Trade
2.2.8a	26 Jul 2016	TCH, BA Team – Euronext IT	Clarification of the use of the 4 update types 1, 2, 3 and 4 in the Market Update – 702 Message.

REQUIRED APPROVER SIGNOFF

DOCUMENT APPROVER NAME	PASS/ FAIL P / F	APPROVAL DATE	COMMENTS Must be entered if a stakeholder does not approve of the document.