Document title

BONDMATCH EXCHANGE DATA PUBLISHER (XDP) CLIENT SPECIFICATION

Document type or subject

CLIENT SPECIFICATION

Revision number

Revision Number: 2.0.0

Date

July 2016

This document is for information purposes only and is not a recommendation to engage in investment activities. The information and materials contained in this document are provided 'as is' and Euronext does not warrant the accuracy, adequacy or completeness of the information and materials and expressly disclaims liability for any errors or omissions. This document is not intended to be, and shall not constitute in any way a binding or legal agreement, or impose any legal obligation on Euronext. This document and any contents thereof, as well as any prior or subsequent information exchanged with Euronext in relation to the subject matter of this document, are confidential and are for the sole attention of the intended recipient. All proprietary rights and interest in or connected with this publication shall vest in Euronext. No part of it may be redistributed or reproduced without the prior written permission of Euronext.

Euronext refers to Euronext N.V. and its affiliates. Information regarding trademarks and intellectual property rights of Euronext is located at https://www.euronext.com/terms-use.

© 2016, Euronext N.V. – All rights reserved.

© 2016, Euronext.

PREFACE

DOCUMENT HISTORY

The following table provides a description of all changes to this document.

For historical document history details please see Appendix B. The Current Version change history is provided in the following section.

VERSION NO.	DATE	CHANGE DESCRIPTION
2.0.0	July 2016	- Modification of reference Data – 553 Message to add the following new field: * **GuaranteeIndicator**

CONTACT INFORMATION

For technical support please contact the Customer Technical Support Group at CTSG@euronext.com / +33 1 8514 8588.

FURTHER INFORMATION

For additional product information, visit: https://www.euronext.com/market-data

For details of IP addresses, visit our IP address pages at: https://www.euronext.com/en/it-documentation/market-data (Technical Documentation tab).

CONTENTS

© 2016, Euronext.

1. EURONEXT CASH MARKETS PROCESSING INFORMATION

1.1 EXCHANGE DATA PUBLISHER OVERVIEW

The Exchange Data Publisher (XDP) provides high-speed, real-time market data for BondMatch.

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 wide range of European market data sets

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

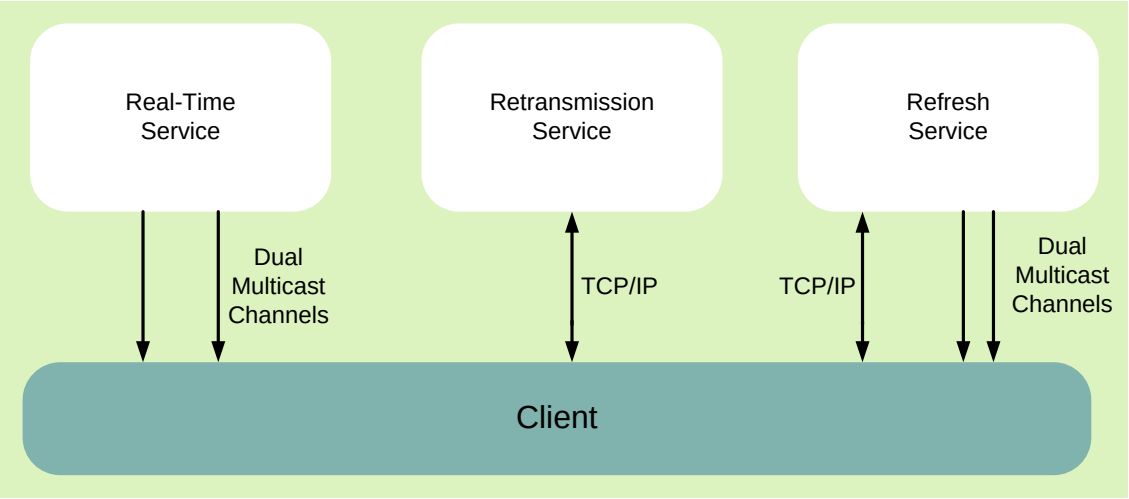
The other chapters of this document provide details that are specific to each of the Euronext market data sets, including formats for each message type.

1.2 ACCESS TO MARKET DATA

Clients access market data as follows:

- **Real time service** Clients connect to multicast channels to receive market data messages
- **Retransmission service** Clients connects to a TCP/IP server for packet retransmissions
- **Refresh service** Clients connect to a TCP/IP server for the initial request–response transaction and the data is then sent over multicast channels

Figure 1 Access to Market Data



1.2.1 Real Time Market Data

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

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

For capacity reasons, market data will be split across a number of multicast groups organized into predefined data sets. Each multicast group will deliver a set of data for a certain market segment. **Please note that BondMatch will initially be delivered over a single multicast line, and may be split horizontally over time.**

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

See Production Feed Configuration and External User Acceptance Feed Configuration for detailed content of Production and External User Acceptance multicast groups.

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 will terminate delivery of data to that host's local network. If a client application terminates without issuing an 'unjoin' message, the network will eventually issue a 'timeout' for the multicast group subscription that will automatically terminate 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.

1.2.2 Retransmission Functionality

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

Note: The Packet Sequence Reset packet (packet number one – see Packet Sequence Number Reset) is *not* available on the Retransmission Server.

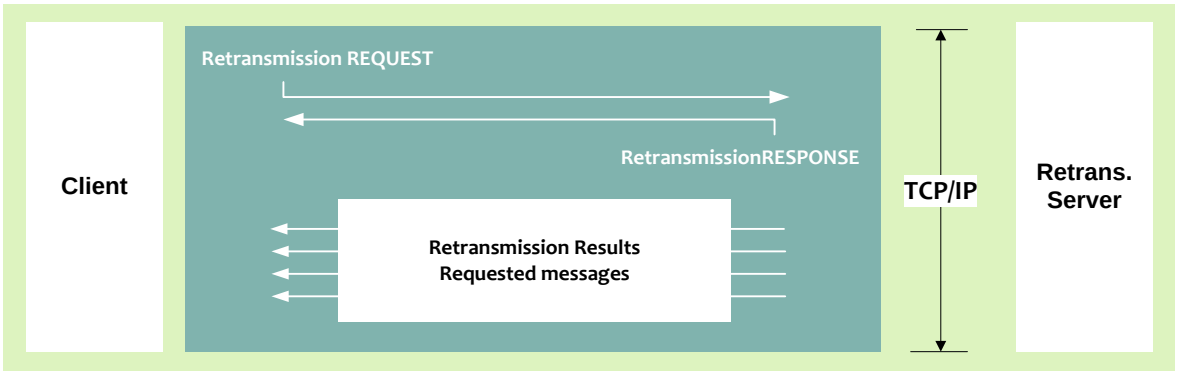
It is not intended that clients use the retransmission functionality to recover data after long outages or on late start up. Accordingly, the number of packets that the user can request is limited. The number of retransmission requests permitted per user is also limited per day.

The client makes a TCP/IP connection with the retransmission server, and receives the requested messages also via the TCP/IP channel, as shown in

Figure 2.

Figure 2 Retransmission Request shows the sequence of messages and the transport protocols employed when making a retransmission request.

Figure 2 Retransmission Request



The retransmission request will include a Source ID (username) which will be validated by the retransmission server. It is important to note that only one Source ID can be used per application session, per service ID.

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

- Invalid Source ID
- Invalid packet sequence number
- PSN no longer in cache, or is not yet available due to an incorrect request
- 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
- Retransmission request message incorrectly formatted

In the case of such a failure, the user will receive a retransmission response with a rejection code to advise of the reason for failure.

If an application connects with a Source ID that is currently in use, the application will be disconnected and no retransmission response will be sent.

Applications which send 200 invalid requests (for any of the reasons above) will no longer be serviced by the retransmission server for the given trading day. The offending applications will not be disconnected and will continue to receive heartbeats, however any requests, valid or otherwise, will be ignored and not responded to by the retransmission server. In this situation clients may use a separate Source ID once one is blocked, or recover packet loss via the refresh functionality.

1.2.3 Refresh Functionality

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

The refresh messages are compressed using the Zlib compression format, which delivers a significant reduction in network latency.

The refresh server is designed to allow the user to update 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 will respond to a request with a Refresh Response message to indicate whether the request was accepted or rejected.

The refresh messages will then be sent on the multicast channels. This will be 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 wait for the next refresh cycle.

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

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

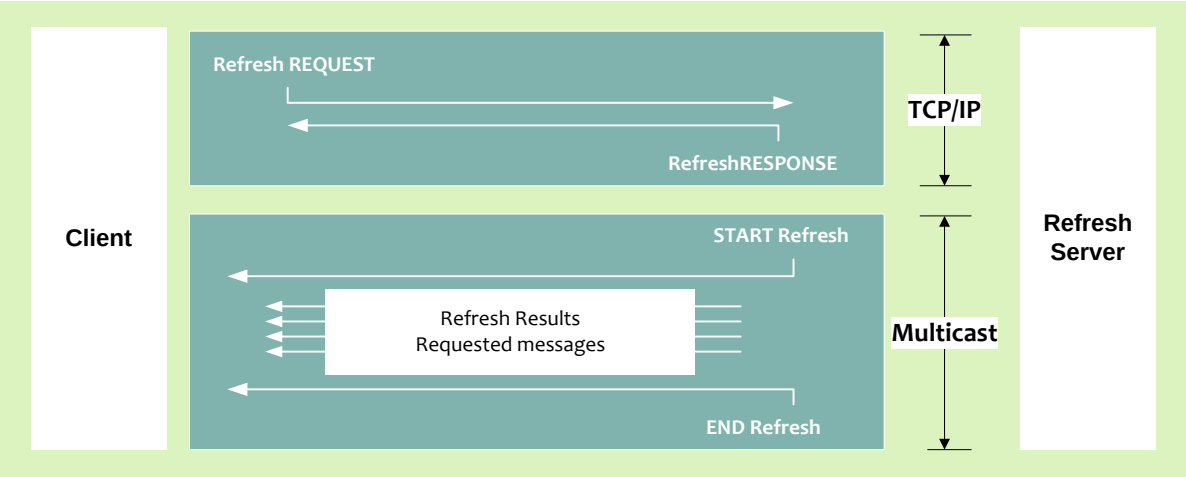
The refresh request may be rejected for any of the following reasons:

- Invalid Source ID
- Invalid Service ID
- Incorrectly formatted message sent
- Incorrect packet type sent
- 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 will receive a rejection message to advise of the reason for failure.

Figure 3 shows the sequence of messages and the transport protocols employed when making a refresh request.

Figure 3 Refresh Request



1.3 PROCESSING GUIDELINES

1.3.1 General Processing Notes

The following processing notes apply to all messages:

- All fields will be sent for every message
- Only field values will appear in the published messages (for example, no names or 'tags' will 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 will not be supported. This means that a message will never straddle a packet boundary.

1.3.2 Packet Structure

In the feed, messages are sent contained within packets. All packets of data sent on the XDP feed will have a common packet header followed by one or more messages (with the exception of some technical messages that do not contain any messages). Messages within a packet are laid out sequentially, one after another without any spaces between them. Here is a graphic that illustrates message sequencing within a packet:

PACKET HEADER	MESSAGE 1	MESSAGE 2	...	MESSAGE N
16 bytes	n1 bytes	n2 bytes	...	nN bytes

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 will start with message size (MsgSize) and message type (MsgType).

The maximum length of a packet is 1400 bytes.

A packet will only ever contain complete messages. A single message will never straddle multiple packets.

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

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 is described in detail in Packet Header Format.

The format of each message within a packet will vary according to message type.

However, regardless of the message type (with the exception of technical messages), each message will start with a two-byte message size (MsgSize) followed by a two-byte message type (MsgType). These are described in **Table 1**.

Table 1 MsgSize and MsgType Fields

FIELD	OFFSET (BYTES)	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	-	2	Binary Int.	Size of the message in bytes (excluding these two bytes).
MsgType	-	2	Binary Int.	Type of message. See Real-Time Message Specifications for details.

1.3.3 Packet Sequence Numbers

All messages conform to the line level sequencing. Each channel has its own Packet Sequence Number (PSN). Clients can use PSNs to determine the following:

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

1.3.4 Detecting and Recovering Missed Data

UDP is an 'unreliable' protocol and therefore may drop packets from line A and line B.

The 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 described in **Table 2**.

Table 2 Recovery Mechanisms

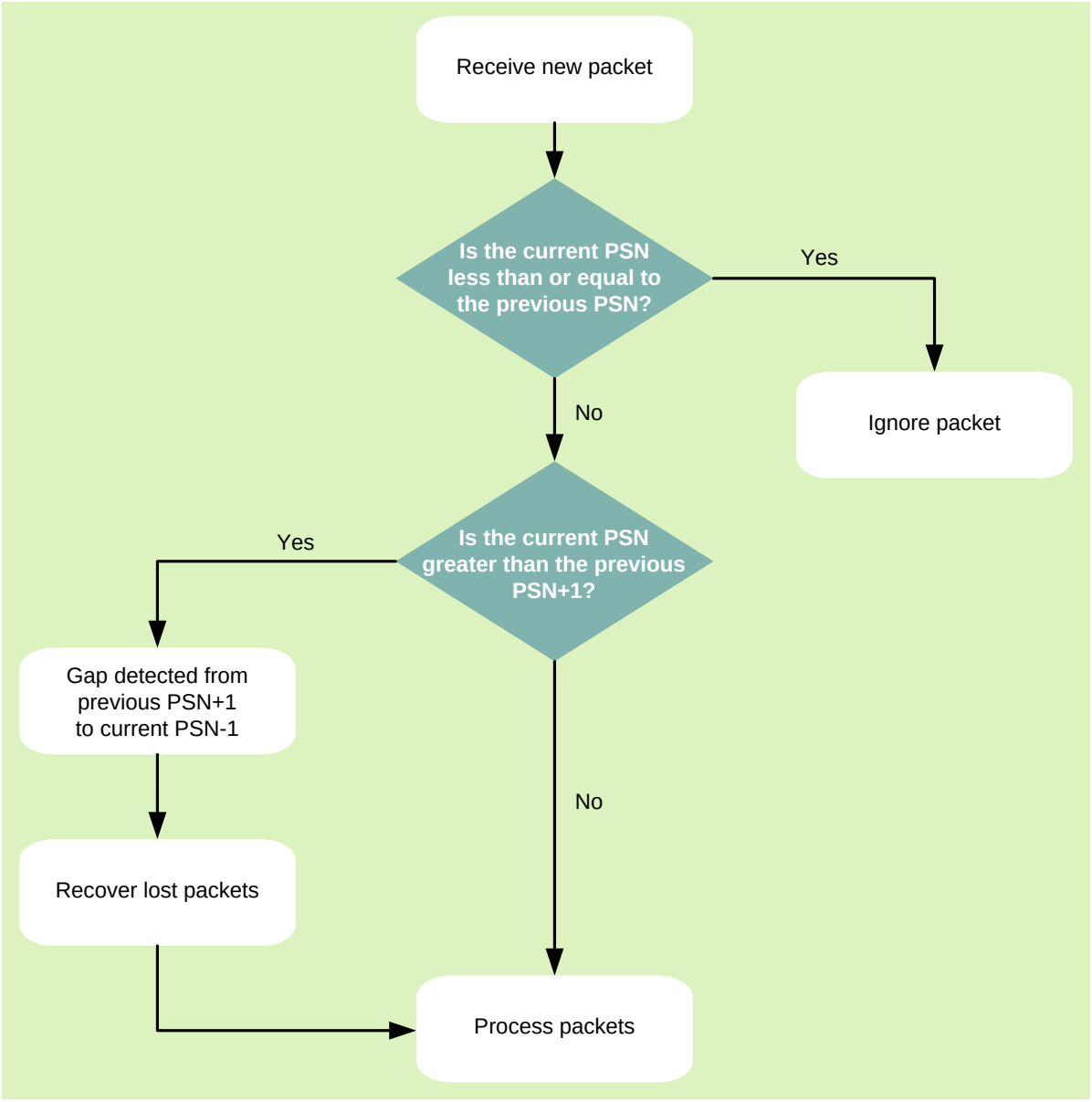
EVENT	ACTION
Packet lost on one of the two lines	Try to recover data from the other line with a configurable timeout.
Dropped packet(s) on both line A and line B	Recover dropped packet(s) from the Retransmission Server.
Late start up or extended intraday outage	Request a refresh of the current market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

1.3.4.1 Gap Detection

Each packet has a PSN that starts at one (1) and increases one-by-one and without gaps with each subsequent packet. Users should use the PSN to detect gaps in the transmission of packets.

Figure 4 Gap Detection using the Packet Sequence Number illustrates how the PSN should be used to detect gaps in the feed.

Figure 4 Gap Detection using the Packet Sequence Number



1.3.4.2 Line Arbitration

Client applications should check the PSN for every packet received.

PSNs are unique and increase monotonically for each service.

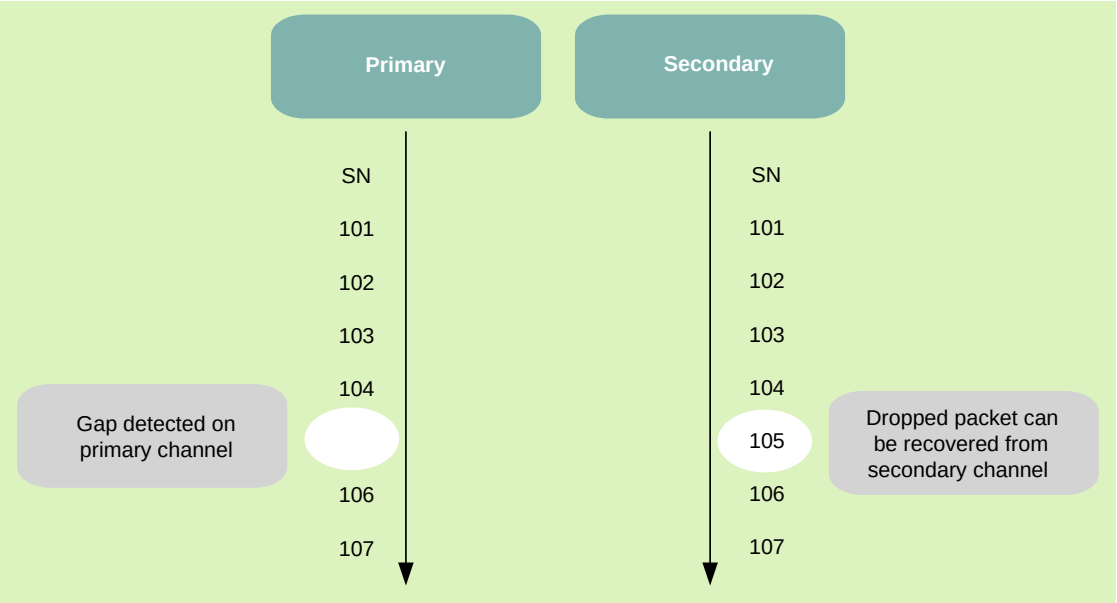
Line A and line B are identical in terms of:

- 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.

See **Figure 5** for a schematic illustrating how missed packets are detected.

Figure 5 Detecting Missed Packets



1.3.4.3 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.

Note: The Packet Sequence Reset packet (packet number one – see _Packet_Sequence_Number Packet Sequence Number Reset) is *not* available on the Retransmission Server.

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

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

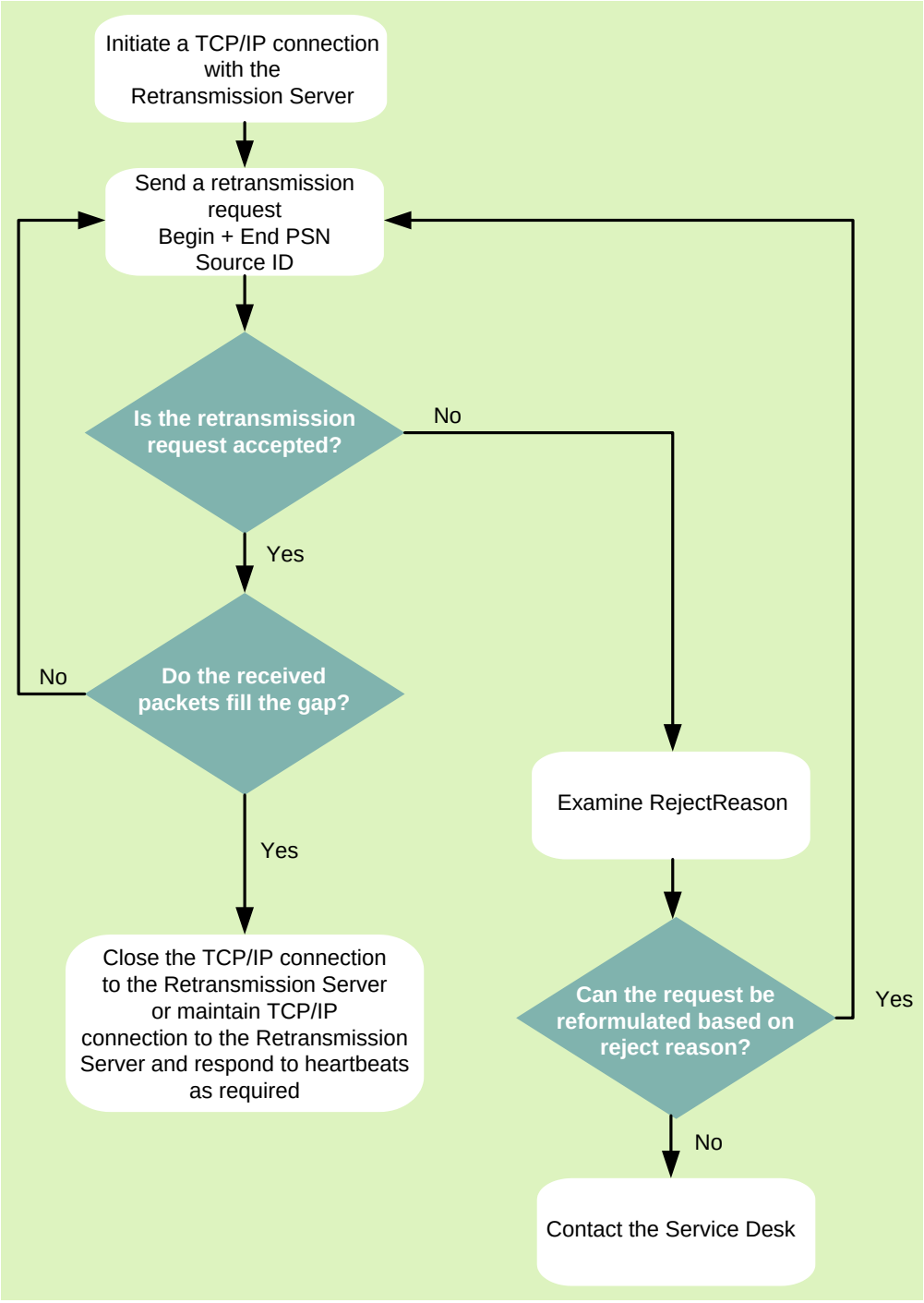
Retransmission requests should contain a Start PSN, an End PSN and a Source ID. The Source ID identifies the client application, and will be supplied by the exchange. The request can be rejected for a number of reasons as defined in Retransmission Response.

The number of retransmissions allowed per client per day is limited and the length of each retransmission is limited to a pre-defined number of packets. See Production Feed Configuration and External User Acceptance Feed Configuration for detailed descriptions of Production and External User Acceptance Retransmission Server limitations.

The length of each retransmission is limited to a predefined number of packets. See Production Feed Configuration and External User Acceptance Feed Configuration for detailed content of Production and External User Acceptance retransmission server limitations).

Figure 6 Requesting Dropped Packets illustrates the process of requesting dropped packets from the Retransmission Server.

Figure 6 Requesting Dropped Packets



1.3.4.4 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 will periodically send heartbeat request messages to the client. Clients must respond to this request with a heartbeat response within a specific timeframe – otherwise, the Refresh Server will close the connection.

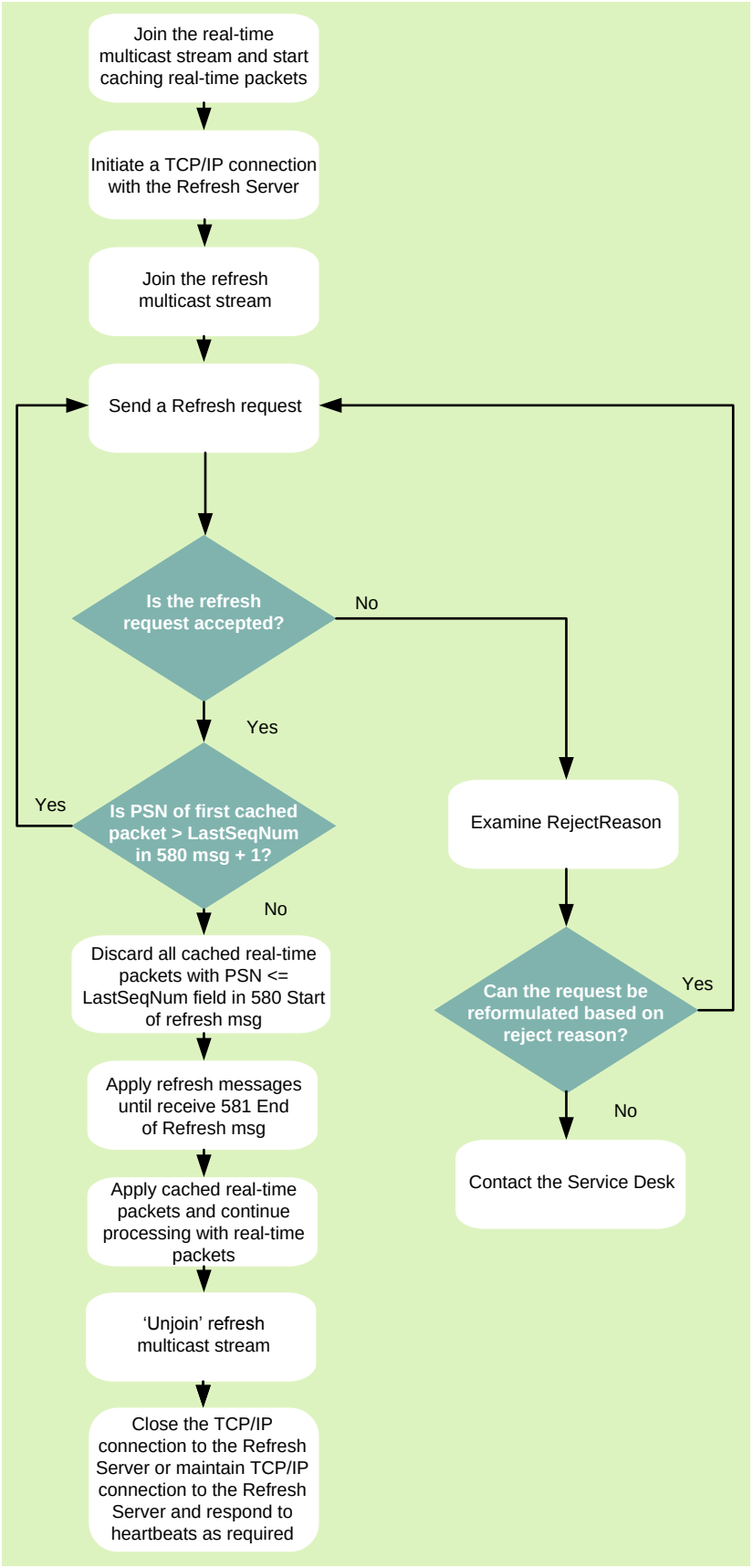
The client makes a TCP/IP connection to the Refresh Server to request a refresh of packets, and at the same time joins the multicast refresh groups. Refresh requests should contain a Service ID and a Source ID. The Service ID will define for which multicast channel a refresh is required. The Source ID identifies the client application, and will be supplied by the exchange. This will be 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 will be 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 number of refreshes that are allowed per client per day is limited (see Refresh Request Limitations).

Figure 7 Request Process from the Refresh Server illustrates the process of requests from the Refresh Server.

Figure 7 Request Process from the Refresh Server



1.4 OPERATIONAL INFORMATION

Measures are in place to safeguard against unexpected system failures.

1.4.1 Exchange System Failure

1.4.1.1 Dual Multicast Lines

Under normal operating conditions, the exchange system will send real-time messages to two unique multicast addresses. This provides clients with two redundant data feeds. The client application should be designed to handle the loss of one of the two multicast channels without any interruption to service.

1.4.1.2 High Availability

The High Availability (HA) functionality of the market data publisher is set up to ensure that 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 will occur 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 will not change. However, clients should note that there are specific technical details that should be considered.

For details of retransmissions and refresh behaviour that should be included as part of application logic, see the high availability Retransmission and Refresh Behaviour sections in Production Feed Configuration and External User Acceptance Feed Configuration.

1.4.1.3 Disaster Recovery Site

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

Clients should ensure that all configuration surrounding the secondary data centre is included, as described in Production Feed Configuration.

1.4.2 Client System Failure

Real-time market data will be 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.

1.4.3 Gap Detection

The XDP feed provides a unique, sequential packet sequence number for each multicast channel. This will allow 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. See Gap Detection for more details.

1.5 GENERAL PROCESSING NOTES

1.5.1 Environments

This specification provides information on how to develop applications to subscribe to market data. A number of environments are available for clients, all of which have various data products separated into different multicast channels. The environments available, and the different multicast groups available in each (known as Service IDs) are defined in **Table 3** and **Table 4**.

Table 3 Production Environment

SERVICE ID	MULTICAST CONTENT
'115'	BondMatch

Table 4 External User Acceptance Environment

SERVICE ID	MULTICAST CONTENT
'6'	BondMatch

Channel definitions for each of the above can be found in Production Feed Configuration and External User Acceptance Feed Configuration.

1.5.2 Multicast Streams

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

A multicast stream refers to the data available from a given IP address and port number. Users should refer to Production Feed Configuration and External User Acceptance Feed Configuration for information on what data is carried in each multicast group.

Clients should connect to multicast stream(s) for which they require data. In order to complete line arbitrage, clients should connect to both multicast streams with the same data.

1.5.3 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 will need to respond to heartbeat requests from the exchange.

1.5.4 Date and Time Conventions

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

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

The base for timestamps in Message bodies is the number of milliseconds since previous midnight 00:00:00.000 UTC.

For example Wednesday 15:30:00.000 UTC is indicated as 315000000 in a packet header or 55800000 in a message body.

1.5.5 Sequence Numbers

The feed contains two sequence numbers:

- The Packet Sequence Number is part of the packet header, and should be used for retransmission requests. 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.
- The source sequence number is assigned by the source system to this message. Whilst this sequence number increases serially, it does not increase one by one.

© 2016, Euronext.

1.5.6 Price Formats

Prices in the feed are represented by two fields, an integer value and a scale code. The scale code is represented in the PriceScaleCode field.

The value should be calculated using the following formula:

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

For example, a price of 27.56 is represented by an Integer of 2756 and a PriceScaleCode of 2.

1.5.7 Data Types

All “Binary Int.” formatted fields are numeric unsigned binary. All “Binary Int. (signed)” formatted fields are signed binary integer. Binary data is in network byte order (Big-Endian).

All “ASCII Str.” and “ASCII Ch.” fields are alphanumeric, left aligned and null padded.

1.5.8 Instrument Identifiers

An instrument is identified by its SymbolIndex, across all Service IDs that relate to that instrument. The SymbolIndex is arbitrarily assigned by the feed, and will not change for the lifetime of the instrument.

The SymbolIndex can take a different value for the same instrument depending on the environment (Production or Test).

Standard security identifiers (for example ISIN, Euronext Trading Code) can be found in the 553 Reference Data message (see Referential - 553 Message).

2. TECHNICAL MESSAGE SPECIFICATIONS

2.1 INTRODUCTION

There are two types of messages transmitted as part of this protocol: technical and market data. Technical messages do not contain data, they allow conversing parties to exchange session-specific information (e.g. 'reset packet sequence number'). Data messages are product-specific.

2.2 PACKET HEADER FORMAT

All messages will contain a common packet header. **Table 5** describes the header field. The design is intended to minimize the development burden on behalf of clients.

Table 5 Common Packet Header Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
PacketLength	0	2	Binary Int.	Length of the packet including the 16-byte packet header
PacketType	2	2	Binary Int.	Identifier for the type of data contained in the packet: ■ '501' - Market Data Packet ■ '1' - Packet Sequence Number Reset ■ '2' - Heartbeat Packet ■ '10' - Retransmission Response Packet ■ '20' - Retransmission Request Packet ■ '22' - Refresh Request Packet ■ '23' - Refresh Response Packet ■ '24' - Heartbeat Response Packet
PacketSeqNum	4	4	Binary Int.	This field contains the packet sequence number. It is unique for each broadcast stream (multicast group) and is used for gap detection. It increases serially and is reset to 1 at the beginning of each trading day. The PackSeqNum is unique for packets containing market data only. Heartbeats inherit their sequence number from the last market data packet or packet sequence number reset packet.
SendTime	8	4	Binary Int.	Timestamp in milliseconds indicating the packet broadcast time. The number represents the number of milliseconds since midnight of the last Sunday 00:00.000 UTC.

© 2016, Euronext.

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
ServiceID	12	2	Binary Int.	Numeric value identifying the broadcast stream. Possible values are described in Feed Configuration descriptions.
DeliveryFlag	14	1	Binary Int.	Indicates delivery method: ■ '0' - Real Time message (Uncompressed) ■ '2' - Retransmission message (Uncompressed) ■ '17' - Refresh message (ZliB Compressed)
NumberMsgEntries	15	1	Binary Int.	The number of messages that are contained within the packet.

2.3 PACKET SEQUENCE NUMBER RESET

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

2.3.1 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). Firstly, if the value should exceed the maximum value that the SeqNum field may contain, it will be reset to one (1). Secondly, 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.

Table 6 NextSeqNumber Field

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Defined below are the 'body' fields of the Sequence Number Reset				
NextSeqNumber	16	4	Binary Int.	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.

2.4 HEARTBEAT

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

2.4.1 General Heartbeat Processing Notes (TCP and Multicast)

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

Heartbeat messages will only contain the packet header (with PacketType = '2'). The packet will not contain a message body.

Heartbeats will only be sent on the multicast channels when there is no market data. Heartbeat frequency since the last packet, is:

- 2 seconds in the multicast streams
- 30 seconds in the active TCP/IP retransmission and refresh sessions

2.4.2 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.

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 request, or remain connected to the retransmission server.

To determine the health of the user connection on the TCP/IP channel, the retransmission or refresh server will send 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 will be disconnected.

Figure 8 Retransmission Server Heartbeat Message

Figure 9 Refresh Server Heartbeat Response

2.5 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.

The fields in the packet header should be filled as described in

Table 7.

Table 7 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 described in **Table 8**.

Table 8 Heartbeat Response Packet Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Defined below are the 'body' fields of the Heartbeat response				
SourceID	0	20	ASCII Str.	This field represents the Identifier of the source (client) requesting retransmission. Field is null padded, left aligned.

2.6 RETRANSMISSION REQUEST

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

Note: The Packet Sequence Reset packet (packet number one – see Packet Sequence Number Reset) is *not* available on the Retransmission Server.

The fields in the packet header should be filled as described in **Table 9.**

Table 9 Retransmission Request Message Header Format

FIELD	DESCRIPTION
PacketLength	44
PacketType	20
PacketSeqNum	Optional
SendTime	Optional
ServiceID	Service ID of the broadcast stream corresponding to the request, in other

FIELD	DESCRIPTION
	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 described in **Table 10**.

Table 10 Retransmission Request Packet Fields

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Defined below are the 'body' fields of the Retransmission Request				
BeginSeqNum	0	4	Binary Int.	Begin Sequence Number of the requested range of packets to be retransmitted. Note the Sequence Number refers to the PacketSeqNum in the header. Remark: The broadcast stream from which a retransmission is requested has to be stated in the field ServiceID in the Packet header of the RetransmissionRequest packet.
EndSeqNum	4	4	Binary Int.	End Sequence Number of the requested range of packets to be retransmitted. Note the Sequence Number refers to the PacketSeqNum in the header. Remark: The broadcast stream from which a retransmission is requested has to be stated in the field ServiceID in the Packet header of the RetransmissionRequest packet.
SourceID	8	20	ASCII Str.	This field represents the Identifier of the source (client) requesting retransmission. Source-ID is pre-set by the Exchange and subject to validation. Field is null padded, left aligned.

2.7 RETRANSMISSION RESPONSE

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

Table 11 Retransmission Response Fields

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Defined below are the 'body' fields of the Retransmission Response				
SourceSeqNum	0	4	Binary Int.	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 packet.
SourceID	4	20	ASCII Str.	This field represents the Identifier of the source (client) requesting retransmission. Field is null padded, left aligned.
Status	24	1	ASCII Str.	Indicates whether the retransmission request was accepted or rejected. Valid values are: ■ 'A' - Accepted ■ 'R' - Rejected
RejectReason	25	1	Binary Int.	Indicates the reason for the rejection. Valid values are: ■ '0' - Retransmission Request Packet was accepted ■ '1' - Rejected due to permissions (the ServiceID is not granted for the SourceID or an incorrect Source ID has been used) ■ '2' - Rejected due to invalid sequence range ■ '3' - Rejected due to max sequence range reached (> thresholds) ■ '4' - Rejected due to max request reached in a day (> thresholds) ■ '5' - Rejected - Requested packets are no longer available ■ '6' - Rejected - Retransmission request incorrectly formatted
Filler	26	2	ASCII Str.	For future use.

2.8 REFRESH REQUEST

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

The fields in the standard packet header should be filled as described in **Table 12**.

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 described in **Table 13**.

Table 13 Refresh Request Packet Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Defined below are the 'body' fields of the Refresh Request Message				
SourceID	0	20	ASCII Str.	Identifier of the Source ID requesting the refresh.

2.9 REFRESH RESPONSE

This packet will be sent immediately via TCP/IP in response to the client's request for a refresh. Note that the fields in the standard packet header should be filled as described in **Table 14**.

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

© 2016, Euronext.

FIELD	DESCRIPTION
NumberMsgEntries	1

The fields in the message body are described in Table 15.

Table 15 Request Response Packet Format

Field	Offset	Size (Bytes)	Format	Description
Defined below are the 'body' fields of the Refresh Response Message				
SourceSeqNum	0	4	Binary Int	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 packet.
SourceID	4	20	Ascii Str.	Identifier of the source requesting the refresh.
Status	24	1	Ascii Str..	Indicates if the Refresh request has been accepted. Valid values are: ■ 'A' - Accepted ■ 'R' - Rejected
RejectReason	25	1	Binary Int.	Indicates the reason for rejection. Valid values are: ■ '0' - Refresh Request Packet was accepted ■ '1' - Request rejected due to permissions (the ServiceID is not granted for the SourceID or an incorrect Source ID has been used) ■ '2' - Request rejected due to incorrect ServiceID ■ '3' - Refresh request incorrectly formatted ■ '4' - Request rejected due to incorrect packet type sent ■ '5' - Rejected due to max request reached in a day (> thresholds) ■ '6' - Request rejected due to unavailability of refresh data, e.g.: – Requesting reference data refresh before the real time reference data has been fully broadcast. – Requesting order book refresh before the morning market sheet retransmission.

© 2016, Euronext.

Field	Offset	Size (Bytes)	Format	Description
				■ '7' - Refresh request rejected as sent to incorrect server (secondary instead of primary)
Filler	26	2	Ascii Str.	For future use

2.10 REFRESH MESSAGES

2.10.1 Refresh Compression

The refresh messages disseminated over the refresh multicast channels are compressed using the ZliB compression format. As mentioned before this method of compression considerably reduces network latency.

The complete specifications for the ZliB compression format can be found here:

http://www.zlib.net/zlib_docs.html

Clients should use the DeliveryFlag field in the packet header to determine if a packet is ZliB compressed.

All messages delivered over the refresh multicast channels will be compressed. The messages delivered over the real time multicast channels will not be compressed.

2.10.2 Refresh Packet Type

In order to allow refresh data to consist of different message types, defined by different generic packet types, a new packet type has been created (999). For example:

The 553 reference data message will be sent within a generic market information packet type 995. A trade full information 252 message will be sent within a generic trade message packet 998. In order to combine these two message types within a single multicast stream for refresh, the new packet type below has been defined.

Table 16 Refresh Packet Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
PacketLength	0	2	Binary Int.	Length of the packet including the 16-byte packet header.
PacketType	2	2	Binary Int.	Identifier for the type of data contained in the packet. '501' – Market Data Packet
PacketSeqNum	4	4	Binary Int.	This field contains the packet sequence number. It is unique for each broadcast stream (multicast group) and is used for gap detection. It increases serially and monotonically and is reset to 1 at the beginning of each trading day. The PackSeqNum is unique for packets containing market data only. Heartbeats inherit their sequence number from the last market data packet or packet sequence number reset packet.
SendTime	8	4	Binary Int.	Timestamp in millisecond indicating the packet broadcast time. The number represents the number of milliseconds since midnight of the last Sunday 00:00 UTC
ServiceID	12	2	Binary Int.	Numeric value identifying the broadcast stream. Possible values are described in Feed Configuration descriptions
DeliveryFlag	14	1	Binary Int.	Indicates delivery method: ■ '0' - Real Time message (Uncompressed) ■ '2' - Retransmission message (Uncompressed) ■ '17' - Refresh message (ZliB Compressed)
NumberMsgEntries	15	1	Binary Int.	The number of messages that are contained within the packet.

2.10.3 Start and End Refresh

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

Table 17 Start Refresh Message

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Defined below are the 'body' fields of the Start Refresh Message				
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	'580' – Start Refresh Message
LastSeqNum	4	4	Binary Int.	Contains the last cached PacketSeqNum that the Refresh is valid to.

Table 18 End Refresh Message

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Defined below are the 'body' fields of the End Refresh Message				
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	'581' – End Refresh Message
LastSeqNum	4	4	Binary Int.	Contains the last cached PacketSeqNum that the Refresh is valid to.

© 2016, Euronext.

3. REAL TIME MESSAGE SPECIFICATIONS

3.1 MARKET INFORMATION

3.1.1 Overview

The BondMatch Market Information Feed contains the following message types:

- 505 – Stock State Change
- 516 – Class State Change
- 537 – Collars
- 539 – Session Timetable
- 550 – Start referential
- 551 – End referential
- 553 – Referential

3.1.2 Packet Header Format

All messages are preceded by a common packet header format. **Table 19** describes the header fields of a Market Information packet.

Table 19 Market Information Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
PacketLength	0	2	Binary Int.	Length of the packet including the 16-byte packet header.
PacketType	2	2	Binary Int.	Identifier for the type of data contained in the packet. '501' – Market Data Packet
PacketSeqNum	4	4	Binary Int.	This field contains the packet sequence number. It is unique for each broadcast stream (multicast group) and is used for gap detection. It increases serially and monotonically and is reset to 1 at the beginning of each trading day. The PackSeqNum is unique for packets containing market data only. Heartbeats inherit their sequence number from the last market data packet or packet sequence number reset packet.
SendTime	8	4	Binary Int.	Timestamp in millisecond indicating the packet broadcast time. The number represents the number of milliseconds since midnight of the last Sunday 00:00 UTC

© 2016, Euronext.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
ServiceID	12	2	Binary Int.	Numeric value identifying the broadcast stream. Possible values are described in Feed Configuration descriptions
DeliveryFlag	14	1	Binary Int.	Indicates delivery method: ■ '0' - Real Time message (Uncompressed) ■ '2' - Retransmission message (Uncompressed) ■ '17' - Refresh message (ZliB Compressed)
NumberMsgEntries	15	1	Binary Int.	The number of messages that are contained within the packet.

3.1.3 Stock State Change - 505 Message

3.1.3.1 Message Overview

A 'Stock State Change' message is generated by the trading engines to announce a change in the state of an instrument.

There is a relationship between ClassState (516 message) and InstrumentState (505 message). In case the state of the Class to which the instrument belongs differs from the state of that individual instrument, the most restrictive value supersedes the other. If the Instrument State shows "tradable" but its class is declared "not tradable" (e.g. Interrupted, Forbidden), effectively trading is not allowed in this instrument.

3.1.3.2 Message Sending Rules

This is sent for tradable instruments:

- When an instrument is halted
- When an instrument is unhalted
- Prior to the trading session to carry over instruments halted the day before

Behaviour when an instrument is halted

If an instrument is halted for functional reasons (for collars for example), it will reopen automatically once the group enters an auction phase. The behaviour is as follows:

- The instrument is halted and a 505 message is sent indicating that it is halted.
- A 516 message (Class State Change) is sent, indicating that the group that the instrument belongs to is entering an auction phase and as such, the instruments of this group will open.
- The Trades occur (if any)
- A 505 message is sent to confirm the status of the instrument

© 2016, Euronext.

3.1.3.3 Message Structure

Table 20 describes the body fields of an BondMatch Market Information message, MsgType = '505' Stock State Change.

Table 20 505 Message Structure

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	'505' – Stock State Change
SymbolIndex	4	4	Binary Int.	XDP proprietary identification of the instrument
SourceSeqNum	8	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically.
SourceTime	12	4	Binary Int.	This field specifies the message generation time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain 47576170
SystemID	16	4	Binary Int.	The ID of the originating Exchange/System of the message. See System ID for possible values.
SourceTimeMicroSecs	20	2	Binary Int.	Number of micro seconds to be combined with SourceTime
Filler	22	2	Binary Int.	For future use.
StartDateHalting	24	8	ASCII Str.	Date that trading was halted for an instrument. Valid values are: ■ YYYYMMDD ■ 00000000 if not provided
StartTimeHalting	32	6	ASCII Str.	Date that trading was halted for an instrument. Valid values are: ■ HHMMSS

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				■ 000000 if not provided
ProgOpeningTime	38	6	ASCII Str.	Time that trading was unhalted for an instrument. Valid values are: ■ HHMMSS ■ 000000 if not provided
OrderEntryRejection	44	1	ASCII CH.	Indicates whether order entry is allowed or forbidden. Valid values are: ■ 'N' - Order entry allowed ■ 'Y' - Order entry forbidden ■ Null - Not provided
InstrumentState	45	1	ASCII Ch.	Indicates the state of the instrument. Valid values are: ■ 'H' - Halted ■ Null (or space) - Inherited (following the state of the class the instrument belongs to)
InstrumentTrading Status	46	1	ASCII Ch.	Indicates whether trading on an instrument is suspended, halted or resumed: Valid values are: ■ Null - Not provided ■ 'S' - Suspended
HaltReason	47	1	ASCII Ch.	Indicates the origin of halting for an instrument Valid values are: ■ 'C' – Opening or trade price outside dynamic collars ■ 'M' - Manuel halting by Market Operations ■ Null or space - Instrument not halted or information not available
ActionAffectingState	48	1	ASCII Ch.	Code indicating the event that caused a change in the instrument state. Valid values are:

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				■ Null - Not provided ■ 'C' - Trading on the instrument at the opening (sent before O) ■ 'D' - Cancelled programmed opening ■ 'M' - Instrument manually halted by Market Operations ■ 'N' - Instrument is being initialized (beginning of the trading day) ■ 'O' - Instrument opened ■ 'P' - Deferred programmed opening
InstrumentStateTCS	49	1	ASCII Ch.	This field is not applicable for BondMatch.
PeriodSide	50	1	ASCII Ch.	This field is not applicable for BondMatch.
Filler	51	1	Binary Int.	For future use.

3.1.4 Class State Change - 516 Message

3.1.4.1 Message Overview

The Class State Change message indicates a change in the state of a class.

There is a relationship between ClassState (516 message) and InstrumentState (505 message). In case the state of the Class to which the instrument belongs differs from the state of that individual instrument, the most restrictive value supersedes the other. If the Instrument State shows "tradable" but its class is declared "not tradable" (e.g. Interrupted, Forbidden), effectively trading is not allowed in this instrument.

3.1.4.2 Message Sending Rules

This message is sent each time a class changes state during the trading day.

3.1.4.3 Message Structure

Table 21 describes the body fields of an BondMatch Market Information message, MsgType = '516' Class State Change.

Table 21 516 Message Structure

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	'516' – Class State Change
SourceSeqNum	4	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically.
SourceTime	8	4	Binary Int.	This field specifies the message generation time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain 47576170
SystemID	12	4	Binary Int.	The ID of the originating Exchange/System of the message
SourceTimeMicroSecs	16	2	Binary Int.	Number of micro seconds. To be combined with SourceTime.
ClassCode	18	3	ASCII Str.	Class Identification
InstrumentGroupState	21	1	ASCII Ch.	This field is not applicable for BondMatch.
SessionType	22	1	ASCII Ch.	Market Session. Valid values are: ■ 'E' - Early ■ 'C' - Core Session ■ 'L' - Late
Filler	23	1	Binary Int.	For future use.
ClassState	24	4	ASCII CH.	Class status representing the current trading market phase for instruments belonging to that class and whose status is inherited. Valid values are: ■ 'EAMO' – Early Monitoring ■ 'LAMO' – Late Monitoring ■ 'COCA' – Core Call ■ 'CLCA' – Closing Call

© 2016, Euronext.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				■ 'COAU' – Core Auction ■ 'CLAU' – Closing Auction ■ 'COCO' – Core Continuous ■ 'HALT' – Halted ■ 'CLSD' – Closed

3.1.5 Collars – 537 Message

3.1.5.1 Message Overview

A Collars message informs clients of modifications in authorized price intervals for an instrument. This message is subjected to selective transmission according to the class of instruments to which the instrument belongs.

3.1.5.2 Message Sending Rules

This message is sent:

- At the start-up of the XDP trading system in the morning
- Each time a collar is changed

3.1.5.3 Message Structure

Table 22 describes the body fields of an Corporate Bond Market Information message, MsgType = '537' Collars.

Table 22 537 Message Format

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	'537' - Collars
SymbolIndex	4	4	Binary Int.	XDP proprietary identification of the instrument
SourceSeqNum	8	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically.
SourceTime	12	4	Binary Int.	This field specifies the message generation time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				47576170
HighCollar	16	4	Binary Int.	Higher bound of the collar price range (to be calculated with the HighScaleCode).
LowCollar	20	4	Binary Int.	Lower bound of the collar price range (to be calculated with the LowScaleCode).
SystemID	24	4	Binary Int.	The ID of the originating Exchange/System of the message. See System ID for possible values.
SourceTimeMicroSecs	28	2	Binary Int.	Number of micro seconds. To be combined with SourceTime.
HighScaleCode	30	1	Binary Int.	To be combined with HighCollar
LowScaleCode	31	1	Binary Int.	To be combined with LowCollar

3.1.6 Session Timetable - 539 Message

3.1.6.1 Message Overview

The session timetable message indicates the timetable for changes in the state of a Class for the current trading day.

3.1.6.2 Message Sending Rules

This message is sent:

- Automatically for each Class code, before the market opening to indicate the times at which the market session will change from one phase to another.
- On an exceptional basis, it may be sent during the trading day in cases where normal hours have changed.

3.1.6.3 Message Structure

Table 23 describes the body fields of an BondMatch Market Information message, MsgType = ‘539’ Session timetable.

Table 23 539 Message Structure

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	‘539’- Session timetable
SourceSeqNum	4	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				monotonically.
SourceTime	8	4	Binary Int.	This field specifies the message generation time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain 47576170
SystemID	12	4	Binary Int.	The ID of the originating Exchange/System of the message. See System ID for possible values.
SourceTimeMicroSecs	16	2	Binary Int.	Number of micro seconds. To be combined with SourceTime.
ClassCode	18	3	ASCII Str	Indicates for which Class code the different times are provided.
SessionType	21	1	ASCII Ch	Market Session. Valid values are: ■ 'E' - Early session ■ 'C' - Core session ■ 'L' - Late session
TimePreOpening1	22	6	ASCII Str.	Indicates the pre-opening time of the Class/Instrument Group of the session.
TimeOpening1	28	6	ASCII Str.	Indicates the opening time of the Class for the session
TimeClosing1	34	6	ASCII Str.	Indicates the closing time of the Class for the session
TimePreOpening2	40	6	ASCII Str.	Indicates, if applicable, the pre-opening time of second auction.
TimeOpening2	46	6	ASCII Str.	Indicates, if applicable, the opening time of second auction.
TimeClosing2	52	6	ASCII Str.	Indicates, if applicable, the closing time of second auction.
TimePreOpening3	58	6	ASCII Str.	Indicates, if applicable, the pre-opening time of third auction.
TimeOpening3	64	6	ASCII Str.	Indicates, if applicable, the opening time of third auction.
TimeClosing3	70	6	ASCII Str.	Indicates, if applicable, the closing time of

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				third auction.
EodTime	76	6	ASCII Str.	Time of End of day inquiry.
Filler	82	2	ASCII Str.	For future use

3.1.7 Start Referential – 550 Message

3.1.7.1 Message Overview

Start referential message is sent to signal the start of transmission of the Instrument Reference Data Flow. It is sent in the morning and in the evening.

3.1.7.2 Message Sending Rules

This message is sent:

- Each time referential messages (553) are sent, prior to the first one.
- There is a 550 message per multicast group containing 553 messages.

3.1.7.3 Message Structure

Table 24 describes the body fields of an BondMatch Market Information message, MsgType = '550' Start referential.

Table 24 550 Message Structure

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	'550' - Start referential
Indicator	4	1	ASCII Ch.	This field indicates the start of the Instrument characteristic flow. Always takes the value 'S'
Filler	5	3	ASCII Str.	This is a filler, reserved for a future use

3.1.8 End Referential - 551 Message

3.1.8.1 Message Overview

End referential message is sent to signal the end of transmission of the Instrument Reference Data Flow. It is sent in the morning and in the evening.

3.1.8.2 Message Sending Rules

This message is sent:

- Each time referential messages (553) are sent, after the last one
- There is a 551 message per multicast group containing 553 messages

3.1.8.3 Message Structure

Table 25 describes the body fields of an BondMatch Market Information message, MsgType = '551' End referential.

Table 25 551 Message Structure

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	'551 - End referential
Indicator	4	1	ASCII Ch.	This field indicates the end of the Instrument characteristic flow. Always takes the value 'E'
Filler	5	3	ASCII Str.	This is a filler, reserved for a future use.

3.1.9 Referential - 553 Message

3.1.9.1 Message Overview

The referential message indicates the main characteristics of a listed instrument:

- Characteristics of the instrument itself
- Trading characteristics of the instrument
- Previous trading day price and amount of capital traded

For morning messages, the characteristics are valid for the trading day at the start of which it was transmitted.

For evening messages, the characteristics are valid for the next business date (the trading day D+1 following the Day D of its transmission).

3.1.9.2 Message Sending Rules

This message is sent:

- Every morning. There is a 553 message per instrument, tradable or broadcast the current day
- Every evening. There is a 553 message per instrument, tradable or broadcast the next trading day
- 553 messages will be sent in between the start and end reference data messages (550 and 551)

3.1.9.3 Message Structure

Table 26 describes the body fields of an BondMatch Market Information message, MsgType = '553' Referential.

Table 26 553 Message Structure

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgType	2	2	Binary Int.	'553' - Instrument Characteristic
SymbolIndex	4	4	Binary Int.	XDP proprietary identification of the instrument
SourceSeqNum	8	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically.
SourceTime	12	4	Binary Int.	This field specifies the message generation time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain 47576170
LastAdjPrice	16	4	Binary Int.	Last traded price of the previous trading day after application of the adjustment coefficient (to be calculated with the LastAdjPriceScaleCode) Provided only in the message sent in the morning.
SystemID	20	4	Binary Int.	Possible values: see System ID
PrevVolumeTraded	24	4	Binary Int.	This field is not applicable for BondMatch.
FixPriceTick	28	4	Binary Int.	Indicates the amount of the fixed tick size (according to FixPriceTickScaleCode). Provided only for tradable instruments.
SourceTimeMicroSecs	32	2	Binary Int.	Number of micro seconds. To be combined with SourceTime
Stock Exchange Code	34	2	Binary Int.	Indicates the Market Place. Possible values: see Stock Exchange Code
TypeOfInstrument	36	2	Binary Int.	Type of instrument. Possible values: see Stock Type
EventDate	38	8	ASCII Str.	Date of the last modification of the characteristics of the instrument
InstrumentName	46	18	ASCII Str.	Instrument Name
PeriodIndicator	64	1	ASCII Ch.	Indicates for which day the characteristics of

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				the instrument will be effective. Valid values are: ■ 'M' - Morning, effective for the current day ■ 'E' - Evening, effective for the next trading day
TypeOfMarket Admission	65	1	ASCII Ch.	This field is not applicable for BondMatch.
IssuingCountryCode	66	3	ASCII Str.	Country code of location for the corporate headquarters of the company that issued the instrument. (ISO 3166-3A)
TradingCurrency	69	3	ASCII Str.	Code of the currency in which the instrument is traded. (ISO 4217-3A)
ClassCode	72	3	ASCII Str.	Designates the group of instruments to which the security belongs.
InstrumentCategory	75	1	ASCII Ch.	Indicates to which category, the security belongs. Valid values are: ■ 'O' – Bond
InstrumentCode	76	12	ASCII Str.	ISIN Code, International Securities Identification Number, according to the ISO-6166.
DateOfLastTrade	88	8	ASCII Str.	Date of the trading day during which a trade was executed for the instrument (YYYYMMDD). Provided only in the message sent in the morning.
UnderlyingRepoISIN Code	96	12	ASCII Str.	This field is not applicable for BondMatch
RepoExpiryDate	108	8	ASCII Str.	This field is not applicable for BondMatch.
FirstSettlementDate	116	8	ASCII Str.	This field is not applicable for BondMatch
TypeOfDerivatives	124	1	ASCII Ch.	This field is not applicable for BondMatch.
BICDepositary	125	11	ASCII Str.	This field is not applicable for BondMatch.
ICB	136	4	ASCII Str.	Identifies for a listed instrument, the economic subsector of the issuing company in the ICB (Industry Classification Benchmark) classification.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MIC	140	4	ASCII Str.	Identifies the market to which an instrument y belongs by its MIC (Market Identification code), according to ISO 10383. 'MTCH' - BondMatch (under final validation)
UnderlyingISINCode	144	12	ASCII Str.	Gives the ISIN code for the underlying instrument
Depositary List	156	25	ASCII Str.	This field is not applicable for BondMatch.
MainDepositary	181	5	ASCII Str.	This field is not applicable for BondMatch.
TypeOfCorporate Event	186	2	ASCII Str.	Valid values are: ■ '00' – No specific event
TimeLagEuronextUTC	188	5	ASCII Str.	Effective difference time between CET (Euronext time) and UTC. To be interpreted in conjunction with the time difference between MiFID regulators and UTC. Valid values are: ■ SHHMN format (with S = + / - , HH = Hour, MN=Minutes) ■ Always provided (MiFID instrument) ■ " 0000" means no difference time
TimeLagMiFIDRegUTC	193	5	ASCII Str.	Effective time difference between MiFID regulators and UTC. To be interpreted in conjunction with the time difference between CET (Euronext time) and UTC. Valid values are: ■ SHHMM format (with S = + / - , HH = Hour, MM=Minutes) ■ Always provided (MiFID instrument) ■ " 0000" means no difference time
CFI	198	6	ASCII Str.	Classification code of a financial instrument defined by the ISO-10962 standard. The structure of the CFI code: ■ The CFI reflects characteristics that are defined when a financial instrument is issued, and remain unchanged during its entire lifetime. ■ The CFI consists of six alphabetical characters: First character indicates the highest level of classification (Categories):

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				■ 'E' - Equities ■ 'D' - Debt instruments ■ 'R' - Entitlements (Rights) ■ 'O' - Options ■ 'F' - Futures ■ 'M' - Others/Miscellaneous Second character indicates specific groups within each category: Groups e.g. for equities: ■ Shares ■ Preferred shares ■ Convertible preferred shares ■ Units, i.e. unit trusts/mutual funds etc. ■ Others The four last characters indicate the most important attributes applicable to each group: whereas voting rights, restrictions, payment status and form are useful information in Equities, these features do not exist for Options, which have other attributes (underlying instruments, type of scheme, delivery, standardized/non-standardized). In Equities, Debt instruments and Entitlements, the sixth (last) character indicates the form of the instrument. If the information is not available or applicable at the time of assignment, the code "X" is to be used for the respective element, i.e. X = not applicable, unknown, not available.
QuantityNotation	204	4	ASCII Str.	Specifies the nature of the amount expression used for negotiating the instrument on the market. Valid values are: ■ 'UNT' - In unit (i.e. number of shares), left padded ■ 'FMT' - In facial amount (i.e. bonds expressed in %), left padded ■ Null - Not applicable

© 2016, Euronext.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
IndexSetOfVarPrice Tick	208	2	ASCII Str.	The key for the variable price tick table consists of two data items [the index for a set of variable price ticks, the lowest value in a range of prices]. The index data item refers to a set of lines in that table which make it possible to determine the price tick for an instrument, based on the price range in which a given price for the instrument falls (i.e. a price to be rounded off or a limit to be checked). When a listed security is created, or when the characteristics of an existing listed security are modified, this data item is (re)initialized to the "typical" index value for this listed security (if a value is not provided for the Fixed Price Tick data item). This "typical index" is derived from the following three characteristics: ■ Trading currency (type of unit in which the listed security's price is expressed) ■ Broad instrument category associated with the listed security ■ Trading group
MarketFeedCode	210	2	ASCII Str.	"Market data flow" to which the instrument belongs. Possible values are listed in appendix A
MICList	212	24	ASCII Str.	This field is not applicable for BondMatch.
IndustryCode	236	4	ASCII Str.	This field is not applicable for BondMatch.
PrimaryMIC	240	4	ASCII Str.	Identify the primary regulated exchange where the primary / most liquid stock is listed
SettlementDelay	244	2	ASCII Str.	Period in number of days between the trade date and the settlement date. Valid values are: ■ [0 – 10] – Possible values ■ 'X' – General value assigned for most of securities (Equities, Bonds) and corresponding to 2 days. ■ 'Z' – Dedicated value for Lending/Borrowing and corresponding to

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				3 days.
MaturityDate	246	8	ASCII Str.	The date when a bond or other debt instrument becomes due or payable.
GuaranteeIndicator	254	1	ASCII Ch.	Indicator for Guarantee in the Settlement System. Valid values are: ■ '0' – Any trade executed on this instrument will be cleared and Guaranteed by a Clearing House. ■ ' 1' – Any trade executed on this instrument will be cleared but not Guaranteed by a Clearing House. ■ '2' – Any trade executed on this instrument is not clearable by a Clearing House. ■ '8' – Not applicable as corresponding to non-tradable instruments
DarkEligibility	255	1	ASCII Ch.	This field is not applicable for BondMatch.
DarkLISThreshold	256	8	Binary Int.	This field is not applicable for BondMatch.
DarkMinimumQuantity	264	4	Binary Int.	This field is not applicable for BondMatch.
ParValue	268	4	Binary Int.	Par Value (all called Nominal value) for Instrument. For Bond it represents the par amount to be repaid at maturity (not including interest revenue).
ParValueScaleCode	272	1	Binary Int.	To be combined with ParValue.
Filler	273	67	ASCII Str.	For future use
FinancialMarketCode	340	3	ASCII Str.	See Financial Market Code
USIndicators	343	7	ASCII Str.	This field is not applicable for BondMatch.
Filler	350	2	ASCII Str.	For future use
PrevDayCapitalTraded	352	8	Binary Int.	This field is not applicable for BondMatch.
NomMktPrice	360	8	Binary Int.	Amount of the nominal value of the security (to be calculated with the NomMktPriceScaleCode)
LotSize	368	8	Binary Int.	Amount, expressed in number of shares or in an amount or a volume of the capital, of the lot size. The lot size is a minimum tradable

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				quantity that is set for each instrument by Euronext. The quantity of an order entered by a trading member on the market must be a multiple of the lot size.
NumberInstrument Circ	376	8	Binary Int.	Number of shares issued/bonds outstanding after payments
SharesOut	384	8	Binary Int.	Not used
AuthShares	392	8	Binary Int.	Not used
OfficialQuotationList HeadingNumber SectionNumber	400	2 1	Binary Int. Binary Int.	This field is not applicable for BondMatch.
RepoIndicator	403	1	ASCII Ch.	This field is not applicable for BondMatch.
LastAdjPriceScale Code	404	1	Binary Int.	To be combined with LastAdjPrice.
TypeOfUnitExp	405	1	Binary Int.	Unit in which the security is quoted. Valid values are: ■ '1' – In units ■ '2' – As a % of nominal ■ '5' – As a % of nominal (including accrued interest) ■ '8' – In kilograms ■ '9' – In ounces
MarketIndicator	406	1	Binary Int.	This field is not applicable for BondMatch.
PrevDayCapitalTradedS caleCode	407	1	Binary Int.	This field is not applicable for BondMatch
TaxCode	408	1	Binary Int.	This field is not applicable for BondMatch.
NomMkdtPriceScale Code	409	1	Binary Int.	To be combined with NomMktPrice.
LotSizeScaleCode	410	1	Binary Int.	To be combined with LotSize.
FixPriceTickScale Code	411	1	Binary Int.	To be combined with FixPriceTick.
Mnemo	412	5	ASCII Str.	This field is not applicable for BondMatch.
TradingCode	417	12	ASCII Str.	Trading code of the instrument. This is only instrument identifier that is unique in the feed in addition to the symbol index.

© 2016, Euronext.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
Filler	429	3	ASCII Str	For future use
StrikePrice	432	4	Binary Int.	This field is not applicable for BondMatch.
StrikeCurrency	436	3	ASCII Str	This field is not applicable for BondMatch
StrikeScaleCode	439	1	Binary Int.	This field is not applicable for BondMatch.
CurrencyCoef	440	4	Binary Int.	Change ratio coefficient of currency applied to instrument: Used in conjunction with one of the change rate indicators in order to apply this coefficient to a currency among two available currencies defined for the instrument: ■ Trading currency ■ Currency code of strike price for Derivative Instrument The currency to which this coefficient will be applied depends on just one of the two values set to the related indicators (defined below in the integrity constraints). This coefficient is used when the currency is not compliant with the ISO 4217 (3A) standard as pence (GBP) or cent (USD) expression of an official currency. In this case the formula to apply in order to retrieve the price expressed in an official currency is: Real price in Trading currency = Traded price (i.e. 1565 pence) x Change ratio coefficient value (0.01)
CurrencyCoefScale Code	444	1	Binary Int.	To be combined with CurrencyCoef
TradingCurrency Indicator	445	1	Binary Int.	Change rate indicator for currency of Instrument Traded Price: Indicates if the change rate will be applied to the currency defined for traded prices of the instrument. Permitted values are: ■ '- 0' - Change rate not applied to the traded price. ■ '- 1' - Change rate applied to the traded price.

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				■ Null - Not applicable
StrikeCurrency Indicator	446	1	Binary Int.	This field is not applicable for BondMatch.
UMTFCode	447	6	ASCII Str.	This field defines the new UMTF code, which identify the primary regulated exchange and primary / most liquid stock.
Filler	453	3	Binary Int.	For future use

3.2 TRADES

3.2.1 Overview

The BondMatch Trade service uses the push-based publishing model. This means that data will be published based on its availability. Once a Last Sale is available, it will be published to BondMatch Trades clients.

The BondMatch Trade message reflects the last sale in each traded security.

Below is a list of the functional message types in the BondMatch Trade Feed:

- 221 - Trade Cancel
- 252 - Trade Full Information
- 253 - Trade Price Update

3.2.2 Packet Header Format

All messages are preceded by a common packet header format. **Table 27** describes the header fields of a Trade messages.

Table 27 Trade Message Fields

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
PacketLength	0	2	Binary Int.	Length of the packet including the 16-byte packet header.
PacketType	2	2	Binary Int.	Identifier for the type of data contained in the packet. '501' – Market Data Packet
PacketSeqNum	4	4	Binary Int.	This field contains the packet sequence number. It is unique for each broadcast stream (multicast group) and is used for gap detection. It increases serially and monotonically and is reset to 1 at the beginning of each trading day. The PackSeqNum is unique for packets containing market data only. Heartbeats

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				inherit their sequence number from the last market data packet or packet sequence number reset packet.
SendTime	8	4	Binary Int.	Timestamp in millisecond indicating the packet broadcast time. The number represents the number of milliseconds since midnight of the last Sunday 00:00 UTC.
ServiceID	12	2	Binary Int.	Numeric value identifying the broadcast stream. Possible values are described in Feed Configuration descriptions
DeliveryFlag	14	1	Binary Int.	Indicates delivery method: ■ '0' - Real Time message (Uncompressed) ■ '2' - Retransmission message (Uncompressed) ■ '17' - Refresh message (ZliB Compressed)
NumberMsgEntries	15	1	Binary Int.	The number of messages that are contained within the packet.

3.2.3 Trade Cancel - 221 Message

3.2.3.1 Message Overview

A Trade cancel message informs that a trade which was already executed has been cancelled.

3.2.3.2 Message Sending Rules

This message is sent each time a trade executed on the trading engine is cancelled.

3.2.3.3 Message Structure

Table 28 describes the body fields of an BondMatch Trade message, MsgType = '221' Trade cancel.

Table 28 221 Message Type Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	'221' - Trade Cancel
SymbolIndex	4	4	Binary Int.	XDP proprietary identification of the instrument
SourceTime	8	4	Binary Int.	This field specifies the Trade generation time. The number in this field represents the

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				number of milliseconds since midnight of the same day. Ex: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain 47576170
SourceSeqNum	12	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically
OriginalTradeID Number	16	4	Binary Int.	This field refers to the initial TradeIDNumber of the trade concerned (as reported in a 252 message) in this cancellation message
SystemID	20	4	Binary Int.	The ID of the originating Exchange/System of the message
SourceTimeMicroSecs	24	2	Binary Int.	Number of micro seconds. To be combined with SourceTime field.
Filler	26	2	Binary Int.	For future use.

3.2.4 Trade Full Information - 252 Message

3.2.4.1 Message Overview

Trade full information message is sent each time a trade has occurred and a trade message is sent. This specific message provides the detailed information related to the trade.

3.2.4.2 Message Sending Rules

This message is sent each time a trade is taking place on the trading engine. As for each trade a 252 message is sent, clients should not process both messages as “reporting of a new trade”.

3.2.4.3 Message Structure

Table 29 describes the body fields of an BondMatch Trade message, MsgType = ‘252’ Trade full information.

Table 29 252 Message Type Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	‘252’ - Full Information
SymbolIndex	4	4	Binary Int.	XDP proprietary identification of the instrument

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
SourceTime	8	4	Binary Int.	This field specifies the Trade generation time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain 47576170
TradeIDNumber	12	4	Binary Int.	Unique numeric and increasing Identifier of the trade, set by the trading engine.
QuoteLinkID	16	4	Binary Int.	Not used
SourceSeqNum	20	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically
Price	24	4	Binary Int.	Price of the trade (to be calculated with the PriceScaleCode)
Volume	28	4	Binary Int.	Number of instruments transacted in this trade
CumulativeQuantity	32	4	Binary Int.	Cumulative number of instruments traded on UTP-NBM since the start of the current trading session
HighestPrice	36	4	Binary Int.	Highest Price traded during the day (to be calculated with the PriceScaleCode)
LowestPrice	40	4	Binary Int.	Lowest Price traded during the day (to be calculated with the PriceScaleCode)
VariationLastPrice	44	4	Binary Int. (signed)	Percentage variation of today's price/last reference price (to be calculated with the VariationScaleCode)
SystemID	48	4	Binary Int.	The ID of the originating Exchange/System of the message
SourceTimeMicroSecs	52	2	Binary Int.	Number of micro seconds. To be combined with SourceTime field.
YieldScaleCode	54	1	Binary Int.	Applicable to all Yields in the message *Provided when ActionType = A, M and Y*
SpreadScaleCode	55	1	Binary Int.	Applicable to all Spreads in the message *Provided when ActionType = A, M and Y*

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
TradCond1	56	1	ASCII Ch.	Not used
TradCond2	57	1	ASCII Ch.	Last trade at same price indicator: Valid values are: ■ '0' - Not last of a series of trade at the same price. ■ '1' - Last of a series of trade at the same price
TradCond3	58	1	ASCII Ch.	Cross Trade Indicator: Valid values are: ■ '0' - Trade does not stem from a Cross Order ■ '1' - Trade stems from a Cross Order ■ '4' - Valuation Trade
TradCond4	59	1	ASCII Ch.	Not used
TickDirection	60	1	ASCII Ch.	Symbol of the variation of the price / previous prices, valid values are: ■ '+' - positive ■ '-' - negative ■ '0' - no variation or data field not significant ■ Null - not significant
OpeningTrade Indicator	61	1	ASCII Ch.	If the trade took place during the opening or during the core session. Valid values are: ■ 'O' - Opening ■ 'S' - Core Session
VariationScaleCode	62	1	Binary Int.	To be combined with VariationLastPrice
PriceScaleCode	63	1	Binary Int.	Applicable to all prices in the message
Yield	64	4	Binary Int.	Yield (to be calculated with the YieldScaleCode) *Provided when ActionType = A, M and Y*
Spread	68	4	Binary Int.	Spread (to be calculated with the SpreadScaleCode) *Provided when ActionType = A, M and Y*

3.2.5 Price Update - 253 Message

3.2.5.1 Message Overview

The Price update message contains a bid price or an ask price, or the modification of the last reference price on a given instrument. This message informs of:

- Modifications operated on instrument prices.
- When the message follows the cancellation of a trade, it indicates that an instrument's first, high, low, or most recent price has been modified.
- The updated instrument's last adjusted closing price or the instrument's settlement price when modified.

3.2.5.2 Message Sending Rules

This message is sent:

- When market operations cancels a trade which modifies one of the following price: first, high, low and last (type of price 30, 31, 32, 33,34). There is one msg.253 sent for each modified price.

3.2.5.3 Message Structure

Table 30 describes the body fields of an BondMatch Trade message, MsgType = '253' Price Update.

Table 30 253 Message Type Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int.	'253' - Price Update
SymbolIndex	4	4	Binary Int.	XDP proprietary identification of the instrument
SourceTime	8	4	Binary Int.	This field specifies the Trade generation time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain 47576170
SourceSeqNum	12	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically
Price	16	4	Binary Int.	Last Price of the trading day (to be calculated with the ScaleCode)
HighestPrice	20	4	Binary Int.	Highest Price traded during the day (to be

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				calculated with the PriceScaleCode)
LowestPrice	24	4	Binary Int.	Lowest Price traded during the day (to be calculated with the PriceScaleCode)
VariationLastPrice	28	4	Binary Int. (signed)	Percentage variation of today's price/last reference price (to be calculated with the VariationScaleCode)
SystemID	32	4	Binary Int.	The ID of the originating Exchange/System of the message
SourceTimeMicroSecs	36	2	Binary Int.	Number of micro seconds. To be combined with SourceTime field.
TypeOfPrice	38	2	Binary Int.	European Market-Code identifying the type of price that is updated. Valid values are: ■ '30' – Change of first price ■ '31'– New high price ■ '32'– New low price ■ '33'– New last price ■ '34'– New previous day's closing price ■ '35'– Bond Reference Price
YieldScaleCode	40	1	Binary Int.	Applicable to all Yields in the message *Provided when ActionType = A, M and Y*
SpreadScaleCode	41	1	Binary Int.	Applicable to all Spreads in the message *Provided when ActionType = A, M and Y*
PriceScaleCode	42	1	Binary Int.	Applicable to all prices in the message
VariationScaleCode	43	1	Binary Int.	Available for the price variation in the message
Yield	44	4	Binary Int.	Yield (to be calculated with the YieldScaleCode) *Provided when ActionType = A, M and Y*
Spread	48	4	Binary Int.	Spread (to be calculated with the SpreadScaleCode) *Provided when ActionType = A, M and Y*

3.3 QUOTES AND BBO10

3.3.1 Overview

The BondMatch Quotes service uses the push-based publishing model. This means that data will be published based on its availability. Once a Quote is available, it will be published to clients.

The BondMatch Quote message reflects a configurable number of highest bids and lowest offers (for example, BBO1, BBO10 ...) in each BondMatch traded security.

3.3.2 Packet Header Format

All messages are preceded by a standard header format.

Table 31 describes the header fields of a Quotes message.

Table 31 Quotes Header Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
PacketLength	0	2	Binary Int.	Length of the packet including the 16-byte packet header.
PacketType	2	2	Binary Int.	Identifier for the type of data contained in the packet. '501' – Market Data Packet
PacketSeqNum	4	4	Binary Int..	This field contains the packet sequence number. It is unique for each broadcast stream (multicast group) and is used for gap detection. It increases serially and monotonically and is reset to 1 at the beginning of each trading day. The PackSeqNum is unique for packets containing market data only. Heartbeats inherit their sequence number from the last market data packet or packet sequence number reset packet.
SendTime	8	4	Binary Int.	Timestamp in milliseconds indicating the packet broadcast time. The number represents the number of milliseconds since midnight of the last Sunday 00:00 UTC.
ServiceID	12	2	Binary Int.	Numeric value identifying the broadcast stream. Possible values are described in Feed Configuration descriptions
DeliveryFlag	14	1	Binary Int.	Indicates delivery method: ■ '0' - Real Time message (Uncompressed) ■ '2' - Retransmission message

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				(Uncompressed) ■ '17' - Refresh message (ZliB Compressed)
NumberMsgEntries	15	1	Binary Int.	The number of messages that are contained within the packet.

3.3.3 Quotes - 140 Message

3.3.3.1 Message Overview

A Quotes message indicates the modification of one or more of the ten best limits.

3.3.3.2 Message Sending Rules

This message is sent:

- Each time one or more of the ten best limits for an instrument is modified.
- All Or None orders (AON) will be included in quote messages to ensure quotes reflect the best bid and ask prices available on BondMatch; There is no flag in the Quote message to identify the AON orders.
- Due to the use of AON orders and other order types in the Quote messages it is possible that 'crossed books' occur.

3.3.3.3 Message Structure

Table 32 describes the body fields of an BondMatch Quotes message, MsgType = '140' Quotes.

Table 32 140 Message Type Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 byte MsgSize field.
MsgType	2	2	Binary Int..	'140' - Quotes
SymbolIndex	4	4	Binary Int.	XDP proprietary identification of the instrument
SourceSeqNum	8	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically
SourceTime	12	4	Binary Int.	This field specifies the Quote generation time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				47576170.
QuoteLinkID	16	4	Binary Int.	Not used
AskPrice	20	4	Binary Int.	Ask Price for Quote (to be calculated with the ScaleCode)
AskSize	24	4	Binary Int.	Total number of instruments requested in sell orders at the ask price.
BidPrice	28	4	Binary Int.	Bid Price for Quote (to be calculated with the ScaleCode)
BidSize	32	4	Binary Int.	Total number of instruments requested in buy orders at the bid price.
SystemID	36	4	Binary Int.	The ID of the originating Exchange/System of the message. See System ID for possible values.
NumberAskOrders	40	2	Binary Int.	Number of sell orders at the ask price
NumberBidOrders	42	2	Binary Int.	Number of buy orders at the bid price
SourceTimeMicroSecs	44	2	Binary Int.	Number of micro seconds. To be combined with SourceTime.
TypeOfAskPrice	46	1	Binary Int.	Valid values are: ■ '0' – Limit order ■ '1' – Market order
TypeOfBidPrice	47	1	Binary Int.	Valid values are: ■ '0' – Limit order ■ '1' – Market order
QuoteCondition	48	1	ASCII Ch.	This field is not applicable for BondMatch.
QuoteNumber	49	1	Binary Int.	Indicates the level in the order book for the given aggregate quote, derived from its price value. Level '0' is dedicated to the market summary.
PriceScaleCode	50	1	Binary Int.	Applicable to all prices in the message
Filler	51	1	Ascii Str	For future use.

3.4 ORDER BOOK

3.4.1 Overview

The BondMatch Order Book service uses the push-based publishing model. This means that data will be published based on its availability. Once information is available, it will be published to clients.

List of the messages in the BondMatch Order Book:

- 233 - Order update
- 231 – Order Book retransmission delimiter

3.4.2 Packet Header Format

All messages are preceded by a common header format. **Table 33** describes the header fields of an Order Book message.

Table 33 Order Book Header Format

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
PacketLength	0	2	Binary Int.	Length of the packet including the 16-bytes packet header.
PacketType	2	2	Binary Int.	Identifier for the type of data contained in the packet. '501' – Market Data Packet
PacketSeqNum	4	4	Binary Int.	This field contains the packet sequence number. It is unique for each broadcast stream (multicast group) and is used for gap detection. It increases serially and monotonically and is reset to 1 at the beginning of each trading day. The PackSeqNum is unique for packets containing market data only. Heartbeats inherit their sequence number from the last market data packet or packet sequence number reset packet.
SendTime	8	4	Binary Int.	Timestamp in millisecond indicating the packet broadcast time. The number represents the number of milliseconds since midnight of the last Sunday 00:00 UTC.
ServiceID	12	2	Binary Int.	Numeric value identifying the broadcast stream. Possible values are described in Feed Configuration descriptions
DeliveryFlag	14	1	Binary Int.	Indicates delivery method: ■ '0' - Real Time message (Uncompressed)

FIELD	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
				■ '2' - Retransmission message (Uncompressed) ■ '17' - Refresh message (ZliB Compressed)
NumberMsgEntries	15	1	Binary Int.	The number of messages that are contained within the packet.

3.4.3 Order Update / Market Sheet - 233 Message

3.4.3.1 Message Overview

The Order update message generated by the trading engines, indicates the creation or modification of a line in the market sheet of normal orders for an instrument. It is also used when the Market Sheet of normal orders is rebroadcast. The deletion of an order from the Market Sheet of normal orders is indicated via a Delete N lines from the Market Sheet message:

- This message takes into account any order type, except Stop Loss and Stop Limit (Stop orders).
- Stop orders are not broadcasted to the market participants until they are triggered.

3.4.3.2 Message Sending Rules

This message is sent:

- In the morning, when UTP-NBM is initialized, to retransmit orders remaining in the book from previous days (taking into account expired orders and order book purges). This is known as the 'order book retransmission' or 'market sheet retransmission' – action type - Y.
- During the day, each time an order introduced by a member firm modifies the market sheet (for example, when creating, modifying or changing the priority of an order).

© 2016, Euronext.

3.4.3.3 Notes

- SymbolIndex+ OrderDate+ OrderID uniquely identifies an order
- Market sheet sequencing:
- Orders must be arranged according to:
    1. Order type (OrderType): Priorities are first, Market orders and second, Limit orders
    2. Order price (Price)
    3. Order priority (OrderPriorityDate+OrderPriorityTime+OrderPriorityMicroSecs)

3.4.3.4 Message Structure

Table 34 describes the body fields of an BondMatch Order Book message, MsgType = '233' Order Update

Table 34 233 Message Type Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 bytes MsgSize field.
MsgType	2	2	Binary Int.	'233' - Order Book
SymbolIndex	4	4	Binary Int.	XDP proprietary identification of the instrument
SourceTime	8	4	Binary Int.	This field specifies the time when the Order Update is generated. Note, when an order is added, the SourceTime represents the order entry time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain 47576170
SourceSeqNum	12	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically.
Price	16	4	Binary Int.	Price (to be calculated with the PriceScaleCode) *Provided when ActionType = A, M and Y*
AggregatedVolume	20	4	Binary Int.	Total interest quantity at a price point *Provided when ActionType = A, M, D and Y*
Volume	24	4	Binary Int.	Remaining displayed quantity of the order. *Provided when ActionType = A, M and Y*

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
LinkID	28	4	Binary Int.	Not used
OrderID	32	4	Binary Int.	Identifies the uniqueness of the order, when combined with OrderDate. *Provided when ActionType = A, M, D and Y*
SystemID	36	4	Binary Int.	The ID of the originating Exchange/System of the message. See System ID for possible values.
SourceTimeMicroSecs	40	2	Binary Int.	Number of micro seconds. To be combined with SourceTime.
NumberOrders	42	2	Binary Int.	Number of order at the current price point *Provided when ActionType = A, M, D and Y*
Side	44	1	ASCII Ch.	Indicates the side of the order. Valid values are: ■ 'B' - Buy ■ 'S' - Sell ■ Null - Not provided
OrderType	45	1	ASCII Ch.	Type of Order. Valid values are: ■ '1' - Market order ■ '2' - Limit order
ActionType	46	1	ASCII Ch.	This field identifies why the volume (size) at the price point was modified. Values reserved for future use (US markets): 'O', 'C', 'E', 'X', 'Z' Valid values are: ■ 'A' - New order ■ 'M' - Modification of existing order ■ 'D' - Deletion of order identified by OrderID ■ 'F' - Deletion of all orders for the given instrument (depending on the side. If side is not provided, it means both) ■ 'Y' - Retransmission of all orders for the given instrument

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
PriceScaleCode	47	1	Binary Int.	Applicable to all prices in the message *Provided when ActionType = A, M and Y*
OrderDate	48	4	Binary Int.	Date of order (YYYYMMDD). To be combined with OrderId. *Provided when ActionType = A, M, D and Y*
OrderPriorityDate	52	4	Binary Int.	Date giving the priority of the order (YYYYMMDD) *Provided when ActionType = A, M, D and Y*
OrderPriorityTime	56	4	Binary Int.	Time giving the priority of the order (HHMMSSsss) *Provided when ActionType = A, M, D and Y*
OrderPriorityMicroSecs	60	2	Binary Int.	Number of micro seconds in the current millisecond *Provided when ActionType = A, M, D and Y*
YieldScaleCode	62	1	Binary Int.	Applicable to all Yields in the message *Provided when ActionType = A, M and Y*
SpreadScaleCode	63	1	Binary Int.	Applicable to all Spreads in the message *Provided when ActionType = A, M and Y*
Yield	64	4	Binary Int.	Yield (to be calculated with the YieldScaleCode) *Provided when ActionType = A, M and Y*
Spread	68	4	Binary Int.	Spread (to be calculated with the SpreadScaleCode) *Provided when ActionType = A, M and Y*
ExecInst	72	1	ASCII Ch.	Execution Instructions for order handling. Valid values are: ■ 'Null' – Not provided ■ 'G' – All Or None
Filler	73	3	ASCII Ch.	For future use.

3.4.4 Order Book Retransmission Delimiter - 231 Message

3.4.4.1 Message Overview

Order book retransmission delimiter message is sent at the beginning and at the end of the order book retransmission.

3.4.4.2 Message Sending Rules

This message is sent to notify market participants in the following situations:

- At the beginning and at the end of order book retransmission at the start up of the trading engine
- During the trading day in case of a situation where order books need to be retransmitted

3.4.4.3 Message Structure

Table 35 describes the body fields of an BondMatch Order Book message, MsgType = '231' Order Book retransmission delimiter.

Table 35 231 Message Fields

FIELD NAME	OFFSET	SIZE (BYTES)	FORMAT	DESCRIPTION
MsgSize	0	2	Binary Int.	Length of the message body, excluding the 2 bytes MsgSize field.
MsgType	2	2	Binary Int.	'231' - Order Book retransmission delimiter Message
SourceTime	4	4	Binary Int.	This field specifies the time when the Order Update is generated. Note, when an order is added, the SourceTime represents the order entry time. The number in this field represents the number of milliseconds since midnight of the same day. Example: If SourceTime = 13:12:56 secs, 170ms and 30 microsecs, this field will contain 47576170
SourceSeqNum	8	4	Binary Int.	This field specifies the sequence number assigned by the source system to this message. Please note that while the sequence number increases serially, it does not increase monotonically.
TradingEngineID	12	2	ASCII Str.	Code identifying the trading engine. Valid values are: ■ 'C0' – UTP NBM
InstanceID	14	1	Binary Int.	Indicates the number of instances for a given trading engine rebroadcasting order books.
RetransmissionIndicator	15	1	ASCII Ch.	Indicates the status of the retransmission for the instance of the trading engine: ■ 'B' - Beginning of the retransmission ■ 'E' - End of the retransmission

4. PRODUCTION FEED CONFIGURATION

4.1 INTRODUCTION

4.1.1 Data Content

The information supplied in this chapter applies to the BondMatch XDP services.

4.1.2 Data Delivery

BondMatch XDP is available via SFTI®.

4.1.3 Configuration

For details, see https://www.euronext.com/en/it-documentation/market-data.

4.2 MESSAGE TYPE AND BANDWIDTH PER SERVICE ID

Table 36 describes which messages are disseminated in the multicast channel for BondMatch:

Table 36 Message Type and Bandwidth Per Service ID

MESSAGE TYPE	BONDMATCH 115
140 - Quotes	•
221 - Trade Cancel	•
233 - Order Update	•
231 - Order Retransmission	•
252 - Trade - Full Information	•
253 - Trade - Price Update	•
505 - Stock State Change	•
516 - Group State Change	•
537 - Collars	•
539 - Session Timetable	•
550 - Start Referential	•
551 - End Referential	•
553 - Referential	•
Bandwidth Size (Kb)	512

4.3 REFRESH MESSAGE TYPE AND BANDWIDTH PER SERVICE ID

Table 37 describes which messages are disseminated in the refresh multicast channel for the BondMatch.

Table 37 Refresh Message Type and Bandwidth Per Service ID

MESSAGE TYPE	BONDMATCH 215
140 - Quotes	•
221 - Trade Cancel	•
233 - Order Update	•
231 - Order Retransmission	
252 - Trade - Full Information	•
253 - Trade - Price Update	•
505 - Stock State Change	•
516 - Group State Change	•
537 - Collars	•
539 - Session Timetable	•
550 - Start Referential	•
551 - End Referential	•
553 - Referential	•
Bandwidth Size (Kb)	64

4.4 PRODUCTION TIMETABLE

Table 38 provides an overview of the main daily event generating activity and indicative time on the XDP Market Data Feed.

Table 38 Production Timetable

EVENT	TIME (CET)	COMMENT
Application start-up	~ 06:10	
Referential data sent	~ 06:11	
Order Book Retransmission	~ 06:20	Retransmission of outstanding orders and associated messages from previous day
Open	~ 09:00	
Close	17:30	
Referential data sent	Between 20:00 and 21:00	Manual action
Application close down	23:00	**It should be noted that the close down of the application depends**

EVENT	TIME (CET)	COMMENT
		on the dissemination of the end of day Reference Data. The end of day Reference Data is mandatory, therefore If it is delayed for any reason, the application close down will be delayed until it is disseminated to customers. Customers can use the 551 End Reference Data message to know there will be no more market data disseminated for a trading day.

4.5 RETRANSMISSION AND REFRESH CONFIGURATION

4.5.1 High Availability Retransmission Behaviour

As per the above tables, in production there are two redundant retransmission servers in the exchange. Clients should monitor the connection to the retransmission server to determine if there is an outage, and have the ability to switch to connect to the 'secondary' retransmission server in the event of a failover. The secondary retransmission server will maintain the same cache of packets as the primary therefore providing redundancy. Clients should monitor the availability of the primary / secondary retransmissions server by the following means:

- For clients remaining connected to the retransmission server throughout the day, a disconnection from the retransmission server should trigger a failover to the secondary server.
- Clients may choose to only connect to the retransmissions server if the application requires packets to be serviced. In this instance, clients should fail over to the secondary retransmission server if they cannot establish a connection with the primary.

4.5.2 High Availability Refresh Behaviour

As per the above tables, in production there are two redundant refresh servers in the exchange. Clients should monitor the TCP/IP connection to the refresh server to determine if there is an outage, and have the ability to switch to connect to the 'secondary' refresh server in the event of a failover. The secondary refresh server will maintain the same cache of packets as the primary therefore providing redundancy. Clients should monitor the availability of the primary / secondary refresh server by the following means:

- For clients remaining connected to the refresh server throughout the day, a disconnection from the refresh server should trigger a failover to the secondary server.
  - It should be noted that if a disconnection is caused for any other reason than by an outage with the refresh server itself, a client sending a request to the secondary will be rejected with rejection code 7. For example, this can be due to a client side network disconnection. In this instance, clients should reconnect to the primary.
- Clients may choose to only connect to the refresh server if the application requires packets to be serviced. In this instance, clients should fail over to the secondary refresh server if they cannot establish a connection with the primary.

© 2016, Euronext.

Clients should note that unlike the secondary retransmission server, the secondary refresh server will not service requests while it is in secondary mode. Clients sending a request to the secondary refresh server when there is no outage and the primary server is available, will receive a response with rejection code 7 - Refresh request rejected as sent to incorrect server (secondary instead of primary).

4.5.3 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 a default of 4 Source IDs for Production. If more Source IDs are required:

- Email: CTSG@euronext.com
- Tel: +33 1 8514 8588.

Clients should clearly state for which environment (EUA or Production) a Source ID is requested.

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

4.6 RETRANSMISSION REQUEST LIMITATIONS

The below recommendations apply to Production and EUA.

4.6.1 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

4.6.2 Number of Source IDs

Clients are provided with 4 Source IDs by default. In the event more Source IDs are required, please contact the XDP Helpdesk.

4.6.3 Parallel Sessions

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

Responses to these requests are sent in the same order as the initial requests.

4.6.4 Maximum Number of Requests

Maximum number of Retransmission Requests per Source ID per day has been set at 1,000. A defensive feature of the retransmission server is that it will not respond to clients requests after 200 invalid requests are sent per Source ID, per day. An invalid request is defined as any of the descriptions 1-6 in the RejectReason field of the retransmission response message. Client applications will remain connected and continue to receive heartbeats, however requests for retransmission will receive no response (accepted or rejected). This defensive feature is designed to stop badly-behaved client applications from impacting the performance of the service for other participants.

4.6.5 Maximum Number of Packets per Request

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

4.6.6 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).

4.7 REFRESH REQUEST LIMITATIONS

The below recommendations apply to Production and EUA.

4.7.1 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

4.7.2 Number of Source IDs

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

Clients are provided with 4 Source IDs by default. In the event more Source IDs are required, please contact the XDP Support Desk.

4.7.3 Maximum Number of Requests

Maximum number of Retransmission Requests per Source ID per day has been set at 100. A defensive feature of the refresh server is that it will not respond to clients requests after 20 invalid requests are sent per Source ID, per day. An invalid request is defined as any of the descriptions 1-7 in the RejectReason field of the retransmission response message. Client applications will remain connected and continue to receive heartbeats, however requests for refresh will receive no response (accepted or rejected). This defensive feature is designed to stop badly-behaved client applications from impacting the performance of the service for other participants.

© 2016, Euronext.

5. EXTERNAL USER ACCEPTANCE FEED CONFIGURATION

5.1 INTRODUCTION

5.1.1 Data Content

The information supplied in this chapter applies to the XDP Euronext Cash markets EUA service. BondMatch EUA is part of the Euronext Cash Markets EUA service in Paris. The remainder of this chapter will refer to BondMatch configuration details only. For complete details of the Euronext Cash EUA service please refer to the XDP Euronext Cash specifications, which also include BondMatch.

5.1.2 Data Delivery

BondMatch XDP EUA is available via SFTI®.

5.1.3 Configuration

For details, see https://www.euronext.com/en/it-documentation/market-data.

5.2 REFRESH CONTENTS

Table 39 describes the Service ID where the refresh functionality will be available in the EUA environment

Table 39 Refresh Contents

EUA SERVICE ID ELIGIBLE FOR REFRESH FUNCTIONALITY	FEED CONTENTS
6 (Refresh 16)	BondMatch Referential Data, Quotes, Trades and Orders

5.3 MESSAGE TYPE AND BANDWIDTH PER SERVICE ID

Table 40 below describes which messages are disseminated in the multicast channel for BondMatch:

Table 40 Message Type and Bandwidth Per Service ID

MESSAGE TYPE	BONDMATCH 6
140 - Quotes	•
221 - Trade Cancel	•
233 - Order Update	•
231 - Order Retransmission	•
252 - Trade - Full information	•
253 - Trade - Price update	•
505 - Stock State Change	•
516 - Group State Change	•
537 - Collars	•
539 - Session Timetable	•
550 - Start Referential	•

© 2016, Euronext.

MESSAGE TYPE	BONDMATCH 6
551 - End Referential	•
553 - Referential	•
Bandwidth Size (Kb)	128

5.4 REFRESH MESSAGE TYPE AND BANDWIDTH PER SERVICE ID

Table 41 describes which messages are disseminated in the refresh multicast channel for the BondMatch.

Table 41 Messages Disseminated in Refresh Multicast Channel

MESSAGE TYPE	BONDMATCH 16
140 - Quotes	•
221 - Trade Cancel	•
233 - Order Update	•
231 - Order Retransmission	•
252 - Trade - Full information	•
253 - Trade - Price update	•
505 - Stock State Change	•
516 - Group State Change	•
537 - Collars	•
539 - Session Timetable	•
550 - Start Referential	•
551 - End Referential	•
553 - Referential	•
Bandwidth Size (Kb)	64

© 2016, Euronext.

5.5 EUA TIMETABLE

The starting time for the XDP EUA feed is ~ 10 minutes delayed in comparison to the production feed.

The closing time for the XDP EUA feed varies daily depending on which testing is scheduled (afternoon upgrades, after hours testing etc).

5.6 RETRANSMISSION AND REFRESH CONFIGURATION

5.6.1 High Availability Retransmission Behaviour

As per the above table, in EUA there are two redundant retransmission servers in the exchange. Clients should monitor the connection to the retransmission server to determine if there is an outage, and have the ability to switch to connect to the 'secondary' retransmission server in the event of a failover. The secondary retransmission server will maintain the same cache of packets as the primary therefore providing redundancy. Clients should monitor the availability of the primary / secondary retransmissions server by the following means:

- For clients remaining connected to the retransmission server throughout the day, a disconnection from the retransmission server should trigger a failover to the secondary retransmissions server.
- Clients may choose to only connect to the retransmissions server if the application requires packets to be serviced. In this instance, clients should fail over to the secondary retransmission server if they cannot establish a connection with the primary.

5.6.2 High Availability Refresh Behaviour

As per the above tables, in EUA there are two redundant refresh servers in the exchange. Clients should monitor the TCP/IP connection to the refresh server to determine if there is an outage, and have the ability to switch to connect to the 'secondary' refresh server in the event of a failover. The secondary refresh server will maintain the same cache of packets as the primary therefore providing redundancy. Clients should monitor the availability of the primary / secondary refresh server by the following means:

- For clients remaining connected to the refresh server throughout the day, a disconnection from the refresh server should trigger a failover to the secondary server.
  - It should be noted that if a disconnection is caused for any other reason than by an outage with the refresh server itself, a client sending a request to the secondary will be rejected with rejection code 7. For example, this can be due to a client side network disconnection. In this instance, clients should reconnect to the primary.
- Clients may choose to only connect to the refresh server if the application requires packets to be serviced. In this instance, clients should fail over to the secondary refresh server if they cannot establish a connection with the primary.

Clients should note that unlike the secondary retransmission server, the secondary refresh server will not service requests while it is in secondary mode. Clients sending a request to the secondary refresh server when there is no outage and the primary server is available, will receive a response with rejection code 7 - Refresh request rejected as sent to incorrect server (secondary instead of primary).

5.6.3 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 a default of four Source IDs for EUA. If more Source IDs are required:

- Email: CTSG@euronext.com
- Tel: +33 1 8514 8588.
- Clients should clearly state for which environment (EUA or Production) a Source ID is requested.

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

APPENDIX A: CODES AND TYPES

A.1 MARKET FEED CODE

MARKET	CODE	CONTENT
	00	None Applicable
	CC	Technical message
European instruments	25	BondMatch

A.2 FINANCIAL MARKET CODE

CODE	CONTENT
303	BondMatch

A.3 STOCK EXCHANGE CODE

CODE	CONTENT
600	BondMatch

A.4 STOCK TYPE

CODE	STOCK TYPE
010	Participating share
041	Ordinary share
042	Bonus share
043	Preferred share
044	Saving share
050	Preferred stock
058	Preference share
059	Preference
061	Unit of international investment trust
062	Unit of unit trust
069	Unit
072	Share warrant
075	Miscellaneous
080	Founder's stock
081	Partnership interest
082	Part de réserve (Belgium)

© 2016, Euronext.

CODE	STOCK TYPE
084	Deferred share
085	Regional development company share
086	Venture capital company
087	Real estate company share
141	Convertible ordinary share
144	Convertible saving share
150	Convertible preference share
162	MBO share
264	Venture Cap. mutual fund share
265	Mutual fund for innovation share
268	Accumulating share
269	Distribution share
272	'Beneficial interest' share

A.5 SYSTEM ID

ID	APPLICATION
1	Not provided
21 to 29	UTP for Equities – NBM

APPENDIX B: CHANGE HISTORY PREVIOUS VERSIONS

VERSION NO.	AUTHOR	CHANGE DESCRIPTION	IMPACT / LIVE DATE DETAILS
1.0	BA Team – Euronext Technologies	Initial distribution of Universal Trading Platform Market Data - BondMatch™ client specification. Note: live production date to be confirmed.	19/10/2010
1.1	BA Team – Euronext Technologies	Added message type, Collars – 537 Message	01/11/2010
1.2	BA Team – Euronext Technologies	■ Added 'Reference Material' section to Preface, providing link to BondMatch™ Market Feed configuration document. ■ Also some minor edits made throughout document.	17/01/2011
1.3	BA Team – Euronext Technologies	Amended message: ■ 253, to show TypeOfPrice '34' ■ 140, to include valid value '1' – Market Order for TypeOfAskPrice and TypeOfBidPrice ■ 233, to include ExecInst and Filler. Also removed the 'P' and 'A' options from the OrderType field.	04/02/2011
1.4	BA Team – Euronext Technologies	Added more valid values for the ClassState field in Table 18.	14/02/2011
1.5	BA Team – Euronext Technologies	■ Amended message 553 (Referential) to include a valid value of '00' (No specific event) for the 'Type Of Corporate Event' field ■ Amended the 'OrderType' field in message 233 to include a valid value of '1' (Market order); also removed the 'K' value ■ Removed references to Peg Orders ■ In Chapters 4 and 5, references to Message Type 537 changed from 'Threshold' to 'Collars' ■ Amended the descriptions of the opening and closing fields for the second and third auctions in Table 20 (Session Timetable 539 Message) ■ Re-coloured diagrams	03/03/2011
1.6	BA Team – Euronext Technologies	■ Corrected Section 3.3.1 to refer to BondMatch ■ Replaced references to "UTP-MD" with "XDP" ■ Updated description of the 'HaltReason' value 'C' field	15/08/2012

VERSION NO.	AUTHOR	CHANGE DESCRIPTION	IMPACT / LIVE DATE DETAILS
		in the Stock State Change message (505) ■ Added description of "Behaviour when an instrument is halted" to Stock State Change message (505) section ■ Added clarifications that the Packet Sequence Reset packet is not available on the Retransmission Server. ■ Replaced the value of '3' with the value of '5' in the description of the 'TypeOfUnitExp' field in the 553 Reference Data message. ■ Document rebranded with new template	
1.7	BA Team – Euronext Technologies	Removed Trade Creation message 220	12/09/2012
1.8	BA Team – Euronext Technologies	Removed NYSE from NYSE BondMatch Added text below to Chapter 3.3.3.2 Message Sending Rules for the 140 Quote Messages: ■ All Or None orders (AON) will be included in quote messages to ensure quotes reflect the best bid and ask prices available on BondMatch; There is no flag in the Quote message to identify the AON orders. ■ Due to the use of AON orders and other order types in the Quote messages it is possible that crossed books' occur.	07/2014 EUA: 30 July 2014 Prod: Q3 2014
1.9.0	BA Team – Euronext Technologies	- Global offset position updated on every messages - Modification of reference Data – 553 Message to add the following new fields: * **GuaranteeIndicator** * **ParValue** * **ParValueScaleCode** * **MaturityDate** * **SettlementDelay** * **DarkEligibility** * **DarkMinimumQuantity** * **DarkLISThreshold** - Contact information updated	Q3 2016
2.0.0	BA Team – Euronext Technologies	- Modification of reference Data – 553 Message to add the following new fields: * **GuaranteeIndicator**	Q3 2016