UMDF – Unified Market Data Feed

FIX/FAST Market Data Messaging Specification

Version: 2.0.14
Last modified: 2015-03-04

Contacts

- Services Development Department (GDS)**: handles all enquiries for connectivity setup and general exchange supported services.**

  ✉ **bvmfsolution@bvmf.com.br**

- Certification and Testing Center (CTC)**: performs certification of all software solutions applying for EntryPoint connectivity.**

  ✉ **ctc@bvmf.com.br**

- Trading Support Channel**: provides real time connectivity monitoring and troubleshooting.**

  ✉ **tradingsupport@bvmf.com.br**

  ☎ +55 11 2565-5000, option 2

Index

Revision History

Date	Version	Description	Author
Mar, 4th, 2015	2.0.14	- Removed section 1.1 (What´s new on UMDF), and every mention of UMDF GTS and UMDF MEGA. Also removed mentions of GLOBEX.	JLRM
Dec, 19th, 2014	2.0.13	- Added note on section 9.1 warning to that the only safe way to handle book entry references is by using tag 290-MDEntryPositionNo. - Updated sections 5.2.6 and 6.1 clarifying step 6 for Market Initial Synchronization and Recovery regarding usage of tag 369.	JLRM
Nov, 13th, 2014	2.0.12	- Updated section 7 changing the definition of VWAP that now includes cross trades. - Updated section 10.9 removing statement regarding Trading Reference Price: this tag is only applied to Economic Indicator instruments.	RNKH
Jun, 10th, 2014	2.0.11	- Clarifications regarding Fixed Income on section 13.2.	JLRM
May, 16th, 2014	2.0.10	- Typo in section 14.6, where there was a misspelling on the name of tag 326. - Updated section 16.3 with the correct URL for the Channel Definitions file. - Updated section 17 with the correct location for the Message Reference specification.	JLRM
Apr, 15th, 2014	2.0.9	- Edited chapter 1.1 removing reference of the release date for Derivatives on UMDF 2.0.	JLRM
Mar, 27th, 2014	2.0.8	- Updated section 13.2 reflecting the changes to Corporate Fixed Income products being merged into PUMA Trading Platform, thus 35=n messages are DEPRECATED. - Added a note on section 10.6 clarifying that tags 272 and 273 refer to message generation time, not settlement date/time. - Added a note to section 8.1 clarifying that when an instrument is created during session, its phase and state are to be considered invalid until the first SecurityStatus message is received.	JLRM
Jan, 2nd, 2014	2.0.7	- Clarified section 5.1.5, indicating there is a maximum limit of messages per request on TCP Historical Replay. - More information added about Final Closing Call phase and state on sections 11.1 and 11.2. - Updated section 13.4 with more information regarding Trade deletes.	JLRM
Dec, 26th, 2013	2.0.6	- Sections 6.2 and 5.1.5 were altered giving more details about TCP historical replay. - Clarified the snapshot recovery scenarios in sections 5.2.6 (item 8) and 6.1 (item 4). - Changed table on section 13.6 updating the use of tag 1150 for Equities.	JLRM
Nov, 19th, 2013	2.0.5	- Updated section 10.6 with the correct behavior of settlement prices.	JLRM
Oct, 11st, 2013	2.0.4	- Marked block 269=D (Index Composite Price) as future use only, since it won´t be currently used on UMDF 2.0 index channel. - Updated sections 13.3 and 13.4 reflecting changes for index instruments. - Updated section 12.1 explaining that Trade Volume information is generated each 30 seconds, not each 1 minute. - Updated section 14.10 reflecting the correct behavior of Trade Volume for equities on PUMA UMDF. - Added a note on section 14.10 explaining that trade volume is always generated per instrument, never per segment. - Added section 9.6 explaining the Overlay book update type. - Added section 13.9 about TCP Historical Replay on Equities. - Reviewed section 5.1.5, on par with the launch of the new TCP Historical Replay feed for equities.	JLRM
Jul, 30th , 2013	2.0.3	- Added a note at the end of section 5.2.5, indicating that customers should shut down their connections during weekends unless when participating in scheduled mock tests.	JLRM
Jun, 11st , 2013	2.0.2	- Corrected “Default” operator example on section 3.1.10 - On section 13.4, added notes about different equities behavior on tags 37-OrderID, 272-MDEntryDate and 273-MDEntryTime - Added a note on chapter 11 to indicate that the group phase may be inferred from the instrument state whenever it rejoins the group.	JLRM

Feb, 22nd , 2013	2.0.1	- Added warning about Settlement price usage on section 10.6. - Emphasized on section 3.1.16 that customer applications must follow the FIX specification for field types, not the FAST template.	JLRM
Jan, 8th, 2013	2.0	- Reviewed FAST examples - Renamed Market Recovery to Snapshot Recovery - Reviewed channel diagram - Explained the Global TCP feeds - New architecture diagrams including deprecation information - New template information - Reviewed the steps at section 5.3.5 - Removed sections 9.1.1 and 9.2.2 (moved to Messaging Spec) - Moved previous section 5.2. Market Data Contingency Feed into FIX/FAST Engagement Rules - Created section 1.1 What´s New on UMDF - Renamed chapter 12 to indicate the changes apply to PUMA Derivatives - Overhaul of chapter 13 to include specifics on PUMA Equities - **Moved Chapter 12 (Certification Process) to the end of the document** - Reviewed table on section 9.1 for book updates - Reviewed all tables on chapter 10 to reflect PUMA messages	JLRM
Oct, 24th, 2012	1.6.4.1	- Upgraded version to match Message Reference update.	JLRM
Oct, 5th, 2012	1.6.4	- Removed Volatilities from Option Strike Price esceptions. - Added note about the need for a single CompID to use TCP Recovery on PUMA. - Added a note to explain that Co-location customers will not have access to feed B for incremental messages. - Corrected the scale of some architecture figures.	JLRM
Jul, 11th, 2012	1.6.3	- Changed CCB to TSG (Trading Support Group) through the text.	JLRM
Jul, 10th, 2012	1.6.3	- Added a special remark on section 6.2 regarding the time a message takes to become available on TCP Recovery feed.	JLRM
Jun. 27th, 2012	1.6.2	- Changed contact information to the Trading Support Group	JLRM
May 18th, 2012	1.6.1	- Added a diagram describing the full TCP Recovery scenarios on section 6.2. - Small corrections on diagram in section 6.2	JLRM
Apr. 5th, 2012	1.6	- **Replaced chapter 15 explaining the conversion from RLC to FIX for PUMA specific impacts** - Added 279=2 for Closing Price on Section 10.5 - Replaced SecurityExchange default value from XBSP to BVMF all over the document - Mention on chapter 5.3.3, indicating the MTU for PUMA is 1420 instead of 1400 for UMDF Legacy - Renamed TCP Replay to TCP Recovery - On Section 6.2.2, explain that the customer must wait 10-20ms before requesting the message on TCP Recovery, to confirm that the message is indeed lost - Added section 11.5 directing to Phases and States handling on PUMA - Added a note on chapter 10 regarding special handling of tag 1500 - On section 11.4, marked tags 625 and 326 on Snapshot (35=W) as non-required. - Added chapter 13.4 to describe the behavior of Option Strike Price for derivatives products. - Removed references to the GTS system (that has been discontinued.	JLRM
Sep. 2nd, 2011	1.5.1	- On section 9.1.1, added the tag 37016-MDInserDate for equities market. - On sections 15.13 and 15.14, added tag 37016-MDInserDate whilst translating messages S3 and S4 from RLC.	JLRM
Aug. 30th, 2011	1.5.1	- On section 3.1.3, corrected the math formula so it displays well in PDF. - On section 3.1.3, added a clarification about the encoding of Boolean values.	JLRM
Jul. 27th, 2011	1.5.1	- On section 10.7, added Imbalance to the list of statistics that customers must delete when an auction is over. - Fixed diagram on section 5.3.8, changing tag 55 to 48. - On Section 15.9, added tag 286=1 (session entry). - Changed the description below the table on section 10.9.	JLRM
Jul. 1st, 2011	1.5.1	- On section 10.8, corrected tag number for MDEntrySize from 270 to 271.	JLRM

May 25th, 2011	1.5.0	- Added TradingReferencePrice handling to section 15.3 - Added a note to the table at the end of section 5.3.4, explaining the times it contains are in local market time.	JLRM
May 5th, 2011	1.5.0	- Clarified SequenceReset (35=4) usage on all streams (5.3.7) - Corrected red box on section 10.2, explaining Trade Volume block (269=B) is available for derivatives now. - Updated section 10.1, explaining the new domain '3' for tag 277-TradeCondition. - Updated section 5.3.8 explaining that the tag 83-RptSeq won´t be sent on 269=J (Empty book) blocks	JLRM
Apr. 20th, 2011	1.5.0	- Removed conflicting template HTTP link - In section 15.11, added strategy leg tags from message 63 - Added section 14.3 to explain about ISIN in underlying instruments.	JLRM
Apr. 16th, 2011	1.5.0	- Section 9.2.2, changed MDUpdateAction. - Section 10.5, changed domain of tag 286-OpenCloseSettlFlag. - In section 10.9, added tag 6939-PriceBandType again. - In section 15.9, added 286-OpenCloseSettlFlag. - In section 15.11, added tags 1194-ExerciseStyle, 201-PutOrCall and 37012-PriceDivisor. - In section 15.12, removed mention to ignore S0 in certain cases.	JLRM
Feb. 9th, 2011	1.4.2.1	- NewSeqNo for stream reset messages should be 1 not 0.	JLRM
Jan. 19th, 2011	1.4.2.0	- Typo for pages 72;73, should be tag 269, not 279	JLRM
Aug. 5th, 2010	1.4.1.0	- Price banding block adjusted.	RNKH
Jul. 27th, 2010	1.4.0.0	- Symbol removed from the spec except from SecurityList. - Default *SecurityExchange* field is changed to "BVMF". - Domain of phases and states are changed to be compatible with new Matching Engine. - Imbalance and Trade Volume block included. - Book Reset mechanism changed. - Derivatives post-trading information included. - Adjusted section describing the behavior for each marketplace.	RNKH
May 5th, 2010	1.3.0.1	- Security Status to RLC 07 mapping added.	EEW/RNKH/JML
Apr. 20th, 2010	1.3.0.0	- DayCumQty removed: TradeVolume replace it. - Include NoMDEntryTypes in the SecurityStatus message do indicate the entry types to be reset by client systems. - Closing Price to RLC 5J mapping added.	RNKH
Jan. 14th, 2010	1.2.2.5	- Included SettlPriceType to incremental and snapshot messages - Included (9 and U) as new values to TradeCondition field	RNKH/TAT
Jan. 13th, 2010	1.2.2.4	- RLC to FIX mapping revised - Included Trading Statistics Reset Flag - Included NewsSource to News message - Included DayCumQty to incremental and snapshot messages	RNKH/TAT
Jan. 8th, 2010	1.2.2.3	- Including best description for index related messages	RNKH/TAT
Dec. 30th, 2009	1.2.2.2	- Texts and links revised - Bovespa RLC revised	JML/RNKH/TAT
Dec. 17th, 2009	1.2.2.1	- TCP Replay revised - Added information on the cash equities index channel	DRSF/JML/RNKH
Oct. 23th, 2009	1.2.1	- Texts and links revised	RNKH/JML
Sep. 29th, 2009	1.2.0	- Added price banding information - Tag 207 is now required in the instrument identification block - Added section 11.3 – clarification on instrument state in the snapshot message - Added state mappings from RLC	RNKH/JML
Apr. 6th, 2009	1.0.0	- First version	RNKH/JML

1. Preface

This document outlines the BVMF Unified Market Data Feed (UMDF) specification contemplating the use of FIX 5.0/FAST protocol over UDP multicast transport and the integration of Equities, Derivatives and FX, using the consolidated trading platform, called **PUMA Trading System** for both market segments.

During the transition from the legacy platforms to PUMA Trading System, the legacy feeds will remain available, but sometime after the migration is concluded, these feeds will be discontinued. Please pay heed to the Circular Letters, available at BVMF´s website at:

http://www.bmfbovespa.com.br/oficiosComunicados/oficiosComunicados.aspx?idioma=en-us

BVMF provides this market data feed based on the Financial Information eXchange ("FIX") Protocol. FIX is a technical specification for electronic communication of trade-related messages. It is an open standard managed by members of FIX Protocol Limited (http://www.fixprotocol.org/). It is assumed that the reader of this document has basic knowledge of the FIX protocol.

1.1 Abbreviations

Abbreviation	Description
BVMF	Bolsa de Valores, Mercadorias & Futuros, or BM&FBOVESPA.
CBOT	Chicago Board of Trade
TSG	BVMF Trading Support Group.
CFI Code	Classification of Financial Instruments Code.
CME	Chicago Mercantile Exchange
CMEG	CME Group – the holding that encompasses the CME, CBOT, NYMEX and other exchanges.
FAST	FIX Adapted for STreaming – a specification for data compression to reduce bandwidth usage, especially for market data feeds.
FIX	Financial Information Exchange Protocol
IP	Internet Protocol
SSL	Secure Socket Layer
TCP	Transport Control Protocol
UDP	User Datagram protocol
EQT	The Equities segment, previously available as BOVESPA signal.
DER	Derivatives and FX segment, previously available as BMF segment.

1.2 Glossary

Term	Definition
BM&FBOVESPA	Securities, Commodities and Futures Exchange, based in São Paulo, Brazil. For more information, visit http://www.bmfbovespa.com.br.
Broker	A broker is an individual or firm who acts as an intermediary between a buyer and seller, usually charging a commission.
Brokerage	Used interchangeably with broker when referring to a firm rather than an individual. Also called brokerage house or brokerage firm.
Counterparty	Party to a trade.
DMA	Direct Market Access – functionality that allows end-customers, such as hedge funds or investment banks, to directly access the exchange electronically without the need to go over physical broker firm infrastructure.

Term	Definition
FIX Gateway	Service that provides connectivity to third-party clients and brokerages using the FIX protocol.
Instrument	Financial capital in a readily tradable form.
Market Data	A collective term for quotes, last sales, volume statistics and other information used by the market to evaluate trading opportunities.
Matching	The process by which two counterparties that have engaged in a trade compare the settlement details of the offers provided by both. Matching is done to verify all aspects of a trade and ensure that all parties agree on the terms of the transaction.
IP Multicast	Method of forwarding IP datagrams to a group of interested receivers.
Security	A stock, bond or contract that has been authorized for trading on, and by, a registered exchange. Each exchange has different criteria to determine a security's eligibility for listing.
Vendor	Institution that sells services to its clients. In the context of this document, a vendor is an institution that sells access to market data feeds and order management interfaces to an Exchange.
PUMA	BVMF´s PUMA Trading System to unify the trading for all exchange products.

2. Trading Hours

2.1 Trading Session Hours

For a list of FX, derivatives and equities trading hours and sessions, please visit:

http://www.bmfbovespa.com.br/en-us/intros/intro-trading-hours.aspx

2.2 Exchange Holidays

For a list of exchange holidays for the FX, derivatives and equities segments, please visit:

http://www.bmfbovespa.com.br/en-us/rules/market-calendar/market-calendar.aspx

3. FAST Introduction

FIX Adapted for STreaming (FAST) encoding has been developed by the FIX Market Data Optimization Working Group. FAST is designed to optimize electronic exchange of financial data, particularly for high volume, low latency data dissemination. This document describes implementation of FAST in receiving and processing BVMF's FIX/FAST-encoded electronic market data feed.

The implementation of BVMF's market data feed is based on the FAST 1.1 specification, available at:

http://www.fixprotocol.org/fastspec

FAST is a data compression algorithm that significantly reduces bandwidth requirements and latency between sender and receiver. FAST works especially well at improving performance during periods of peak message rates. FAST extends the base FIX specification and assumes the use of FIX message formats and data structures.

It compresses data by removing redundant data and doing binary encoding. It does not use general-purpose, data compressing methods like Lempel-Ziv or arithmetic coding; instead, carefully crafted templates are used for describing the structure of the messages. High levels of data compression with low processing overhead and latency can be attained by using FAST.

It is not required that the decoding of a FAST message results in a FIX message; you can streamline your market data feed processing by creating directly data structures suited to your program, if your FAST decoder implementation supports it.

3.1 Implementing FAST Overview

This section provides a brief overview on FAST implementation and basic concepts of FAST and the encoding/decoding process. Customer development teams should refer to the FAST specification for in-depth understanding of such process. **BVMF does not provide support for any FAST decoders, including the reference code.**

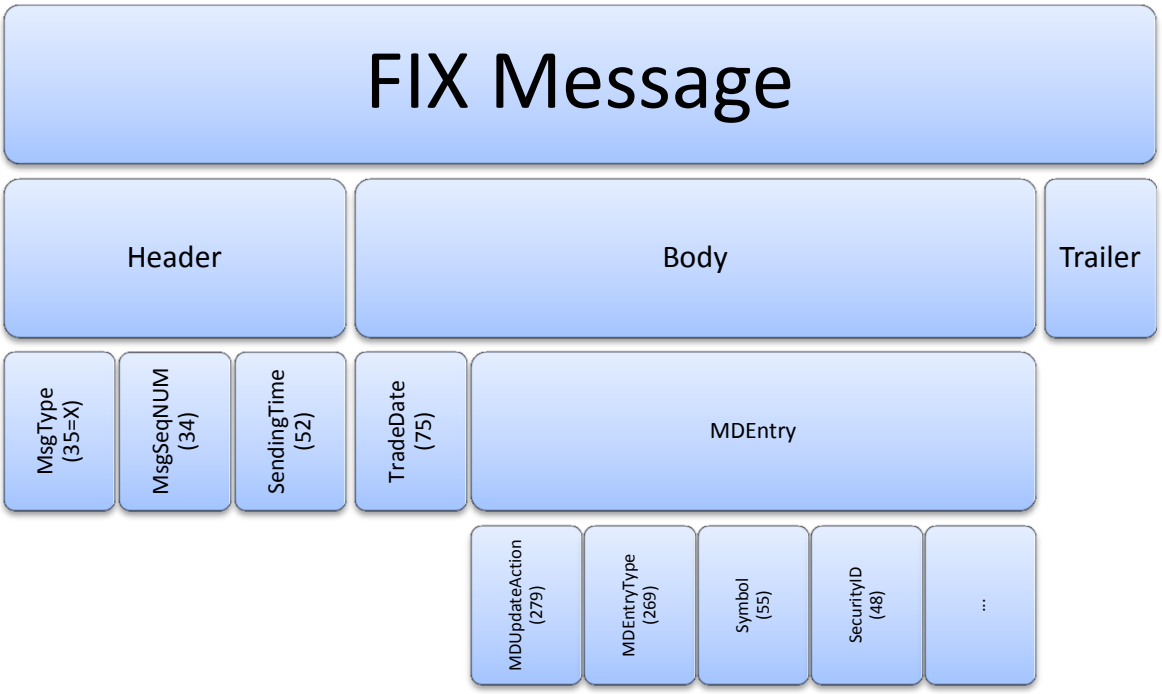
3.1.1 Templates

Every FIX message can be described by one or more FAST templates. Each template describes what fields from the original FIX message are included, and their types and transfer encodings. The templates are kept in a single XML file that obeys the "FAST v1.1 Template Definition Schema", included in the FAST 1.1 specification.

3.1.2 Message Structure

Take as example the FIX message "Market Data Incremental Refresh" (tag 35=X). It is composed by three elements:

- Header;
- Body;
- Trailer (that is not encoded in FAST).



FAST encoding makes no distinction between Header, Body and Trailer. The template for the message "X" simply lists the fields, as follows:

```
<template name="MDIncRefresh_145" id="145" dictionary="145">
	<string name="ApplVerID" id="1128">
		<constant value="9"/>
	</string>
	<string name="MsgType" id="35">
		<constant value="X"/>
	</string>
	<uInt32 name="MsgSeqNum" id="34"/>
	<uInt64 name="SendingTime" id="52"/>
	<uInt32 name="TradeDate" id="75" presence="optional"/>
	<sequence name="MDEntries">
		<length name="NoMDEntries" id="268"/>
		<uInt32 name="MDUpdateAction" id="279">
			<copy value="1"/>
		</uInt32>
	...
	</sequence>
</template>
```

3.1.3 Data Types

The following data types are recognized by FAST:

- String – ASCII (7-bit) strings (no special characters allowed);
- Unicode strings – Internationalized (Unicode) strings, encoded using UTF-8;
- Byte vectors;
- Decimal numbers;
- Signed integers (both 32 and 64 bits);

- Unsigned integers (both 32 and 64 bits).

Example fragments of template definitions for fields (not necessarily used in official templates):

- Ascii String

```
<string name="SecurityID" id="48" />
```

- Unicode String

```
<string name="Text" id="58" charset="unicode" presence="optional" />
```

- Byte Vector

```
<byteVector name="Text" id="58" presence="optional">
   <length name="TextLength" id="59" />
</byteVector>
```

- Decimal number

```
<decimal name="MDEntryPx" id="270" presence="optional"/>
```

- Signed Integer (64 bits)

```
<int64 name="MarketDepth" id="264" />
```

- Unsigned Integer (64 bits)

```
<uInt64 name="MarketDepth" id="264" />
```

- Signed Integer (32 bits)

```
<int32 name="MarketDepth" id="264" />
```

- Unsigned Integer (32 bits)

```
<uInt32 name="MarketDepth" id="264" />
```

FIX has more types, but almost all of them can be easily mapped to FAST data types (like Price → Decimal). One exception is the *UTCTimeStamp* type, that's mapped to an unsigned, 64-bit integer in a non-standard[1] way – just remove all separators of the *UTCTimeStamp* value (the value must have the milliseconds part) and convert the resultant decimal string to a number. For instance, if the field SendingTime (52) has the value 20081007-09:12:08.008 (format YYYYMMDD-HH:MM:SS.sss), encode it to the integer "20081007091208008":

```
<uInt64 name="SendingTime" id="52" />
```

Decimal numbers are represented as a pair of integers "mantissa" and "exponent". For instance, the value 23.45 is 2345×10^{-2} and it is represented as "2345" and "-2".

Another exception is the Boolean type that is encoded on FIX within the domain (N, Y). However, BVMF uses the integers "0" and "1" when encoding it to FAST.

3.1.4 Stop Bit Encoding

All fields in FAST are variable-length fields, even the integer ones. Instead of using a length indicator (like ASN.1, DER Encoding) or a separator byte (like FIX), the 8th bit of each byte (for strings and numbers) indicates if this is the last byte of the field.

[1] BVMF uses the same FAST encoding of a FIX *UTCTimeStamp* as the CME Group, and does not follow the tentative FAST 1.2 specification.

Optional fields are encoded slightly differently from mandatory fields, to take into account the special value NULL (missing); the details can be found in the FAST specification document. We will show only the encoding of mandatory fields.

- **Encoding an ASCII (7-bit) string**
  "BM&FBovespa" = ASCII 42 4D 26 46 42 6F 76 65 73 70 61

  The 8th bit of the last byte (61 hex, 0110 0001 binary) must be set to indicate that it's the last byte, so the last byte must be encoded as **1**100 0001 binary = C1 hex. The FAST encoding will be:

  42 4D 26 46 42 6F 76 65 73 70 **C1**

- **Encoding an unsigned integer**
  Integers are encoded using 7 bits per byte; the 8th bit of the last byte must be set.

  123456 = binary 111 1000100 1000000

  The FAST encoding will be:
  00000111 01000100 **1**1000000

  i.e.,
  07 44 C0

- **Encoding a byte vector or an Unicode (UTF-8) string**
  Byte vectors (that represent the FIX DATA type) and Unicode strings are encoded using a length indicator (encoded as an integer in stop-bit encoding) and then the data. For instance, "ação" (stock in Portuguese) is represented in UTF-8 as the following 6-byte array:
  61 C3 A7 C3 A3 6F

  The stop-bit encoding of the integer value 6 is binary 86, so the resulting encoding will be:
  86 61 C3 A7 C3 A3 6F

3.1.5 Data Redundancy Removal

Redundant data in FAST is removed by noting that:

- If you have a template, no metadata information need to be sent (like tags numbers and field separators);
- Optional fields are usually absent;
- Some fields have constant or default values, and could be omitted;
- In repeating groups, some fields can have repeated or similar values.

3.1.6 Templates and Implicit Tagging

The FAST template says exactly what fields must be encoded by FAST, and what the order of the fields is. So the tags are not encoded. If the original FIX message contains fields that are not specified in the template, they are simply ignored when encoding, and will not be decoded as well.

3.1.7 Presence Map (PMAP)

It is a bit vector that helps the decoder to find if data is present or it is implied (omitted). It occurs at the beginning of each FAST message and at the beginning of every sequence/group.

3.1.8 Template IDs

Every FAST message has a *template ID* as the first integer field and it will be used by the decoder to choose what template will be used to decode it. You can have several templates for the same FIX message (*MsgType*=X, for instance), but referring to different versions of the message layout. For instance, if BVMF needs to add a field "Symbol (55)" to the message X, a new template will be generated (with a new template ID) that maps to the new version of the message X including the new field.

Example (taken from the FAST template file):

```
<!-- Template 11 for message MarketDataSnapshotFullRefresh (W) -->
<template name="MarketDataSnapshotFullRefresh_147" id="147">
   <string name="MsgType" id="35">
      <constant value="W" />
   </string>
   …
```

3.1.9 The Dictionary Context

It is a set of values that must be kept in memory for correct operation of the decoder. FAST compares the current value of a field to the prior value of that field, and determines how it will be encoded (according to the "field operator", a directive that is associated to that field).

The BVMF encoding process always resets the dictionary for each message, and uses only the "global dictionary". See FAST Specification Version 1.1 for more details:

http://www.fixprotocol.org/fastspec

3.1.10 Field Operators

A field within a FAST template will usually have one of the following Field Operators:

- **(None)** – The field will be encoded directly *as is*.

  ```
  <uInt64 name="MsgSeqNum" id="34" />
  ```

- **Constant** – The field will always contain a predetermined value. For instance, to encode a *MarketDataIncrementalRefresh* message (tag 35=X), the value of tag 35 is constant and always X, so it can be omitted. (The messages are distinguished by their template IDs.)

  ```
  <string name="MsgType" id="35">
     <constant value="X" />
  </string>
  ```

- **Default** – The field is omitted from the message if it is equal to the default value. For instance, the field *SecurityExchange* (tag 207) is usually "BVMF" and can be omitted if the value is exactly "BVMF".

```
<string name="SecurityExchange" id="207">
   <default value="BVMF">
      <!--  Possible values:  -->
      <!-- BVMF : BMF&BOVESPA -->
      <!-- XCME : CME -->
      <!-- XCBT : CBOT -->
   </default>
</string>
```

- **Copy** – Omit the field if it was already used with that exact value (usually in a previous repeating group). For instance, if the field Currency (tag 15) occurs several times in the same FIX message with the value "BRL", the first occurrence of that field is sent and the other occurrences are copies, so they don't need to be encoded.

```
<decimal name="MDEntryPx" id="270" presence="optional">
   <copy />
</decimal>
```

- **Delta** (for numbers) – Encode the difference between the previous value and the current value. It can save some bytes because smaller numbers are encoded with lesser bytes.

```
<decimal name="MDEntryPx" id="270" presence="optional">
   <delta />
</decimal>
```

- **Delta** (for strings) – Encode the "string difference" between the previous value and the current value. For instance, to encode two fields *Symbol*, one with the value "BMFBR123456" and the other with the value "BMFBR789012" (both start with "**BMFBR**"), encode the binary value "-5" and the string value "789012".

```
<string name="Symbol" id="55" presence="optional">
   <delta />
</string>
```

- **Increment** –If the difference between the current value and the previous value is exactly 1 (one), the field can be omitted.

```
<int64 name="NumberOfOrders" id="346" presence="optional">
   <increment />
</int64>
```

- **Tail** – Encode just the "tail" difference. It's like "delta" but the strings must have exactly the same length. For instance, to encode the *Symbol* fields given above ("BMFBR123456" and "BMFBR789012"), encode just "789012".

```
<string name="Symbol" id="55" presence="optional">
   <tail />
</string>
```

3.1.11 Sequence Numbers and Groups

In FAST, a “group” is an unordered set of fields (a FAST “group” is roughly equivalent to the FIX “component” type). For instance, you can define a “group” that groups the fields of a single instrument together.

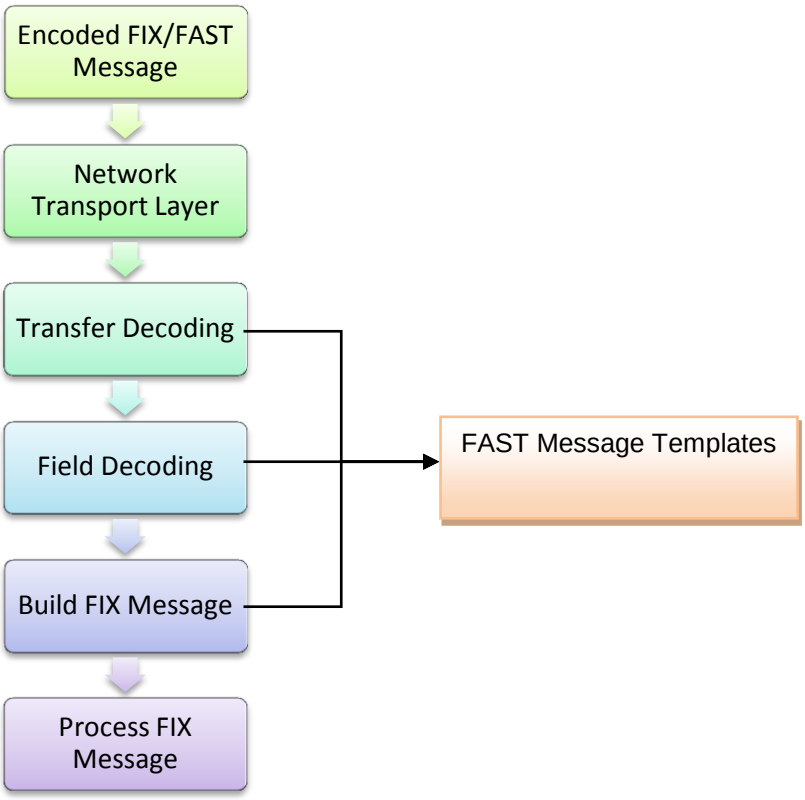
A “sequence” is a length and an ordered set of FAST groups (a FAST “sequence” is a FIX “repeating group”). For instance, you can define a “sequence” that lists *MDEntries* (market data incremental refresh blocks). You can specify directly the fields, dispensing the FAST “group”.

The terminology is somewhat confusing; just remember, “FAST Sequence” = “FIX Repeating Group”.

The repeating group “*NoMDEntries*” from FIX (message “W”, *MarketDataSnapshotFullRefresh*) is represented in FAST as:

```
      <sequence name="MDEntries">
         <length name="NoMDEntries" id="268" />
         <string name="MDEntryType" id="269">
            <default value="0" />
         </string>
         <decimal name="MDEntryPx" id="270" presence="optional">
            <copy />
         </decimal>
...
      </sequence>
```

3.1.12 The FAST Decoding Process



1) The FAST Encoder translates the original FIX message into a FAST message.
2) Such message is transmitted (via UDP within a datagram, for instance), and received by the client system. If the message needed to be split in pieces, the client must join them to get a complete message.
3) Transfer decoding:
    a. Identify template (get the template ID and find the matching template)
    b. Extract binary encoded bits
    c. Map bits to fields per template field
4) Field decoding: apply operators (like <copy> or <delta>) to determine values per template field.
5) Build/Process FIX Message (optional)

3.1.13 Transfer Decoding

Transfer decoding is the initial step that converts data from the FAST binary format.

3.1.14 Field Decoding

Field decoding reconstruct data values according to the template definitions.

3.1.15 Decoder State Reset for Every Message

BVMF resets the encoder state for each message sent, so the client decoder must reset the decoder state as well. The reason is because the client can join the market data dissemination at any time and it cannot be dependent on data in a previous message.

3.1.16 Template Implementation Considerations

The following items must be taken into account whenever implementing template functionality:

- Client systems should use the defined sizes and types for each tag in the FIX Message Specification as a guide for storing data, not just only the FAST template.
- If the structure of the underlying FIX message is changed, a new template will be generated, with a new ID and BVMF will release a new version of the template XML file.

The field types on the FAST template may be different from the types described on the FIX Message Specification, as a transport optimization. Always follow the FIX message Specification when implementing the protocol.

Template changes should be handled by client systems without any changes to their decoder.

3.2 Reference Source Code for FAST Decoding

BVMF makes available reference source code for client system's developers who wish to decode BVMF's market data stream.

The source code comes with absolutely no warranties and is not intended for production use. The decoder is implemented in C++ and can be compiled by MSVC++ (Windows platform) and gcc/g++ (Unix/GNU platform).

They can be found at the URL:

ftp://ftp.bmf.com.br/FIXFAST/reference/FASTDecoder.zip

4. Legacy Electronic Market Data Feeds

Currently there are 3 separate electronic market data feeds available from BVMF. All 3 feeds will remain available after FIX/FAST is rolled out for facilitating the migration process. However, support for these feeds will cease at a later date, to be established by the exchange.

4.1 BELL (FIX 4.4 over TCP)

This feed carries the derivatives and FX segments' market data. It is based on the FIX 4.4 protocol and is transmitted via TCP unicast using a subscription mechanism.

The feed specification is available at:

http://www.bmf.com.br/bmfbovespa/pages/gts2/arquivos/BELL-BMF-Electronic-Link-Specification-v3.0.11.zip

This feed also provides the market data for CME Group's CME Futures and CBOT Futures.

4.2 RLC/MMTP over TCP

This feed carries the equities segment market data. It is based on the RLC protocol using the TCP unicast mechanism as transport, in a push data model (no subscription). The RLC messages are compressed using the ZLIB algorithm before being sent to the market, reducing the necessary bandwidth.

The feed specification is available at:

http://www.bmfbovespa.com.br/pt-br/download/Manual_Production_NewVersion.pdf

4.3 SDM over TCP

This feed carries the derivatives clearing house market data. It is a proprietary protocol using the TCP unicast mechanism as transport, in a push data model (no subscription).

5. System Architecture

The market data systems at BVMF publishes an unified FIX FAST feed. These components are specific to each market segment (Equities and Derivatives), although their output will be the same from the client system standpoint.

There are two focal points on this market data architecture: the concept of a “market data channel” – which defines how the feed is logically distributed according to a set of instruments and level of information of the book; and the “FIX/FAST engagement rules” – which define the transport of the information and how the client system should synchronize the data that is provided in the market data channels.

5.1 Market Data Channel

A channel is a logical group of multicast IP addresses, UDP ports. Every channel provides market data of a list of instruments that have common characteristics, as determined by the exchange.

A channel is broken up into 3 streams, as listed below:

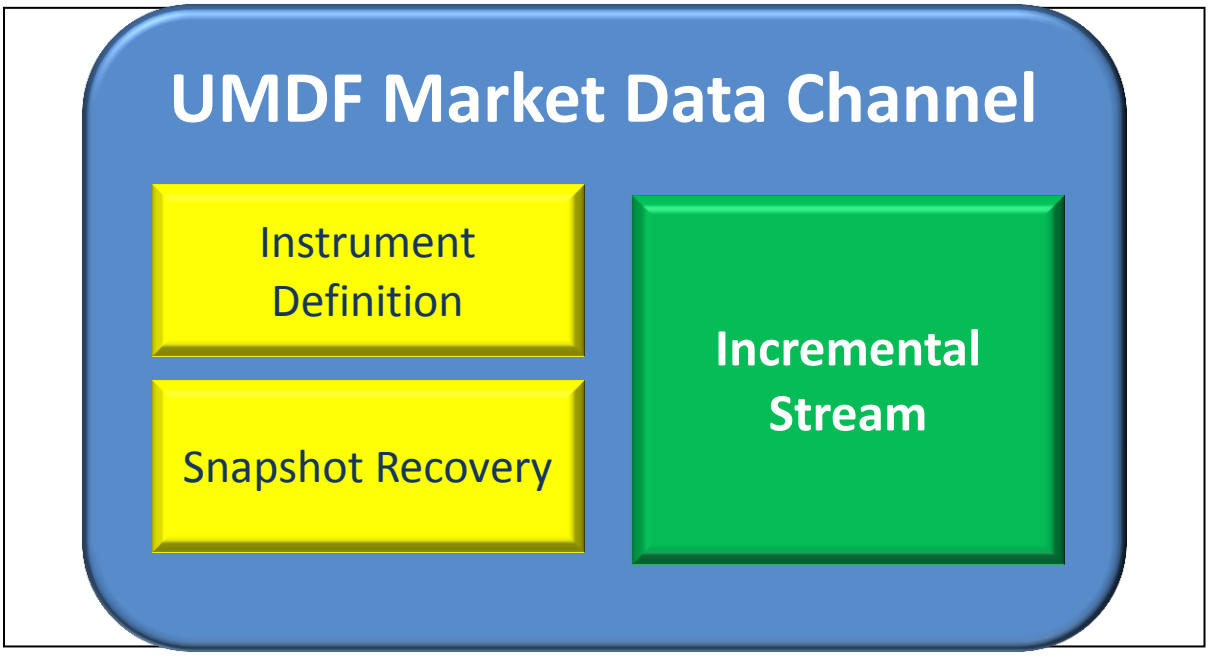


Figure 5.1: Channel overview.

For contingency purposes, BVMF provides a backup feed that is generated at its contingency site.

The backup feed contains the exact same data that is sent over the primary feed, however with different connectivity information (different UDP multicast addresses and ports).

Besides the 3 feeds present in the channel, there is a TCP recovery feed, global to all channels, allowing the recovery of lost messages.

BVMF strongly suggest that customers sign up to receive both feeds.

5.1.1 Incremental Stream

Used to disseminate BVMF incremental market data and other real time data such as news, instrument updates, instrument status using FAST encrypted FIX messages.

If no data is sent through the incremental stream for more than 10 seconds, BVMF will issue a heartbeat message for maintaining connectivity. If client systems do not receive this message within 30 seconds, the incremental stream should be considered not functional and the book state should be considered inconsistent.

A single FIX/FAST message can contain multiple updates for multiple instruments.

5.1.2 Snapshot Recovery Stream

Snapshot recovery is used to disseminate BVMF market data snapshot message for instruments belonging to that channel. The snapshot for a book is transmitted in only one message in one or more chunks of data. This is very common with order depth books that may not fit into the maximum UDP packet size. The market data snapshot messages are replayed at a specific rate and should be used as the primary source for initial book synchronization.

Once the books are synchronized and the client starts using only the incremental stream, the client should unjoin the stream as it would take up unnecessary bandwidth.

The number of snapshots sent in the snapshot recovery stream in one loop could be less than the number of instruments assigned to the related channel. Client systems must handle instruments with no snapshots as have empty books and statistical data before applying incremental data.

5.1.3 Instrument Definition Stream

The instrument definition stream is used to relay the list of all instruments belonging to that channel. The list is replayed at a specific rate and starts over once the last instrument definition message is received.

The exchange sends more than one instrument in each SecurityList message.

5.1.4 TCP Recovery

The global TCP Recovery Connection (previously known as TCP Replay) functionality allows a client to request messages that were already sent through the incremental stream. These messages will be returned to the client over a TCP connection (a FIX 4.4 session). The same connection is used for both the request and the retransmission.

The request specifies a range of messages to be retransmitted. The client system must use an *ApplicationMessageRequest* message (tag 35=BW) to request the lost messages in the incremental stream (UDP channel). For each request, BVMF should send an Application Message Request

Acknowledgment (tag 35=BX) to report whether the request was accepted or not. After sending a positive acknowledgment, BVMF should start resending the available requested messages wrapped in one or more Application Raw Data Reporting messages (*tag 35*=URDR). To indicate the end of the retransmission, for each *ApplID* (channel id) in the request, BVMF sends an *ApplicationMessageReport* (tag 35=BY) message.

This method of recovery should only be used if few messages were lost. For late joiners to the market, or if the lost exceeded 10000 messages, the snapshot recovery stream should be used.

There is a single TCP Recovery feed for all channels.

5.1.5 TCP Historical Replay

The TCP Historical Replay feed allows querying all incremental MD messages for charting purposes, up to the message with sequence number 1. The response time for this feed is considerably higher compared to TCP Recovery, because of that, it´s not recommended to use it for any other purpose than charting.

There is a global TCP Historical Replay feed for all channels, and the customer application can choose to remain connected to either A or B feed through the week (disconnecting during the times the platform is down for maintenance).

The message format is exactly the same as TCP Recovery, without the maximum message limitation, up to the entire trading week (SequenceNumber=1). However, a maximum limit of 2000 messages per request is still enforced.

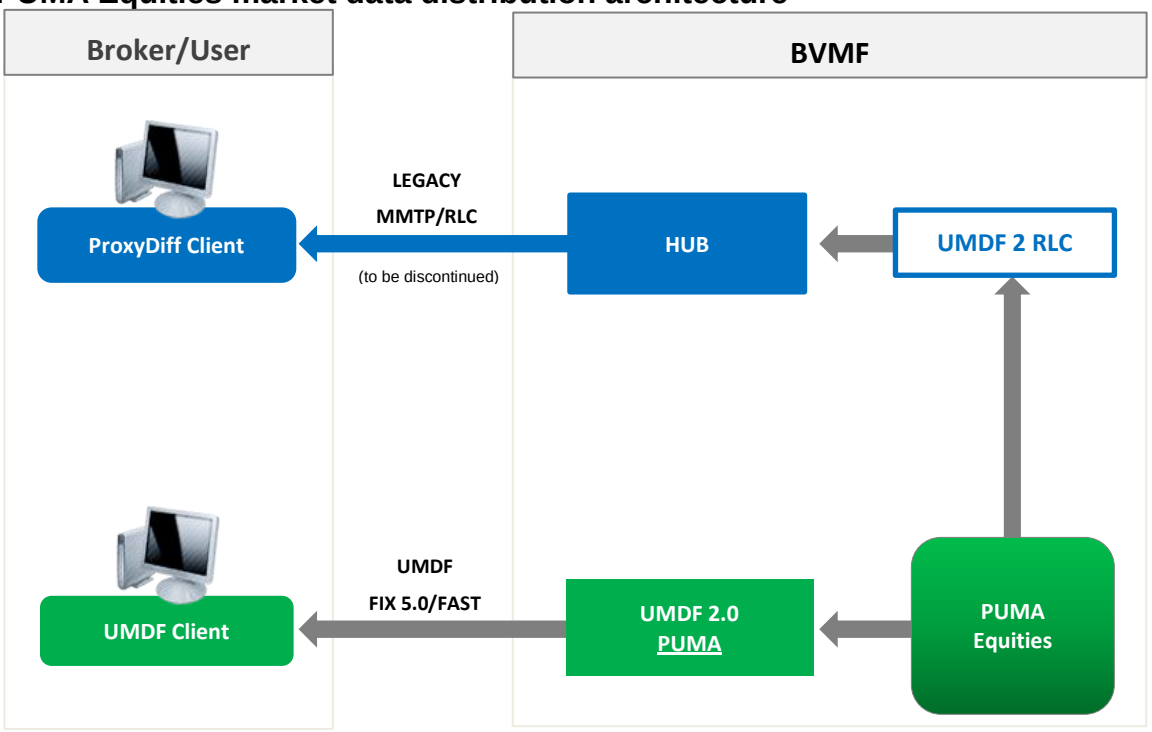
For the complete flow of messages, refer to the chart on section 6.2.

WARNING:
BVMF does not support the usage of this feed for any other purpose than offline charting. This feed is not supposed to be used for recovery or for real time market data consumption.

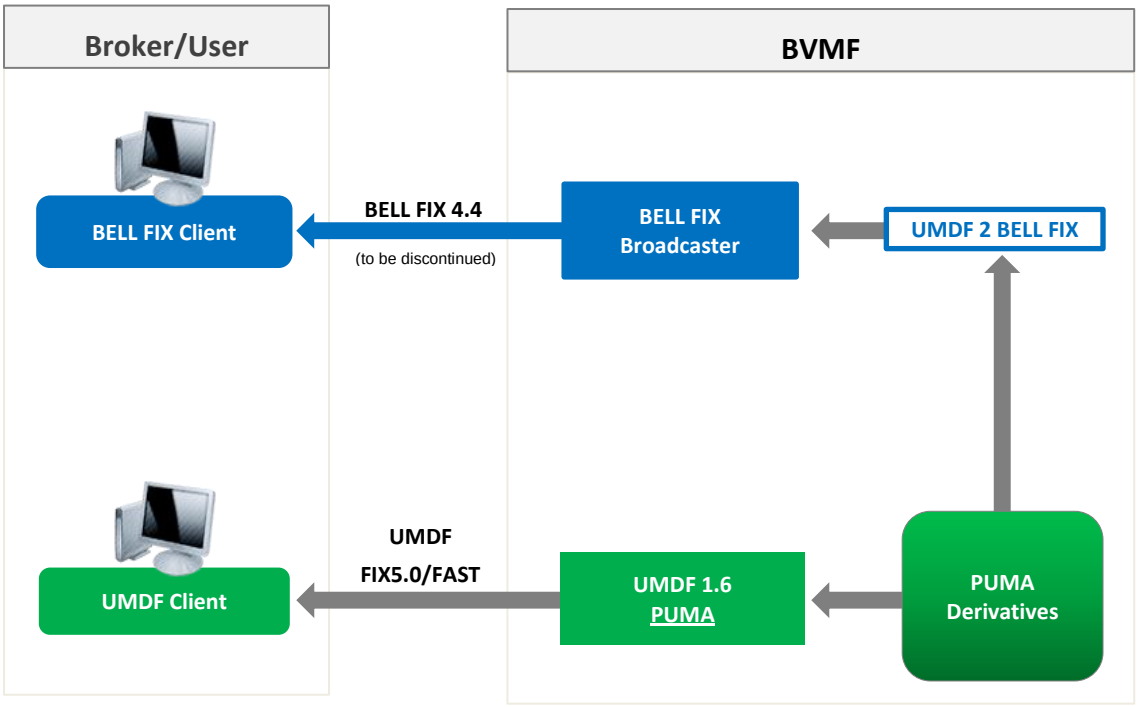
5.1.6 BVMF Market Data Distribution Diagrams

The following diagrams illustrate the market data distribution components involved in the feeds for Equities and Derivatives/FX, respectively.

PUMA Equities market data distribution architecture



PUMA Derivatives/FX market data distribution architecture



5.2 FIX/FAST Engagement Rules

This section contains an overview of engagement architecture for receiving the FIX/FAST market data feed.

5.2.1 FIX/FAST Templates

FIX/FAST templates provide the rules for a FAST decoder to be able to properly decode market data messages. FAST-encoded messages can only be interpreted correctly by using such templates.

The templates are all listed within a single XML file. The templates are subject to change by BVMF as the system evolves and new functionality is added. When a change is done, BVMF will notify market participants in advance for appropriate development and/or testing efforts.

On the template file header comments the customer can obtain the latest template id used for each message available. For example:

```
*Incremental refresh

MDIncrementalRefresh........ = 145 (138)
```

In this case, the latest template id for message MDIncrementalRefresh (35=X) is 145, with 138 being the previous template id.

The customer application must be compatible at least with the **current** and **previous** template ids for each message type specified in the template file.

Please contact the TSG (BVMF Trading Support Group) on how to get the latest template information.

In addition, template files are available at the BVMF public FTP site, at the following address:

For New Release:

ftp://ftp.bmf.com.br/FIXFAST/templates/NewRelease/templates-PUMA.xml (PUMA Equities)

For Certification:

ftp://ftp.bmf.com.br/FIXFAST/templates/Certification/templates-PUMA.xml (PUMA Equities)

ftp://ftp.bmf.com.br/FIXFAST/templates/Certification/templates-UMDF-NTP.xml (PUMA Derivatives/FX)

For Production:

ftp://ftp.bmf.com.br/FIXFAST/templates/Production/templates-PUMA.xml (PUMA Equities)

ftp://ftp.bmf.com.br/FIXFAST/templates/Production/templates-UMDF-NTP.xml (PUMA Derivatives/FX)

FAST SCP (Session Control Protocol) is not currently used by BVMF to exchange template files.

5.2.2 Network Configuration

BVMF will provide clients with the necessary network configuration in order to receive all market data channels.

See the documents: *Market Data Channels Definition*, or contact the TSG for the list of new release, certification and production multicast streams and TCP recovery / historical replay connection information.

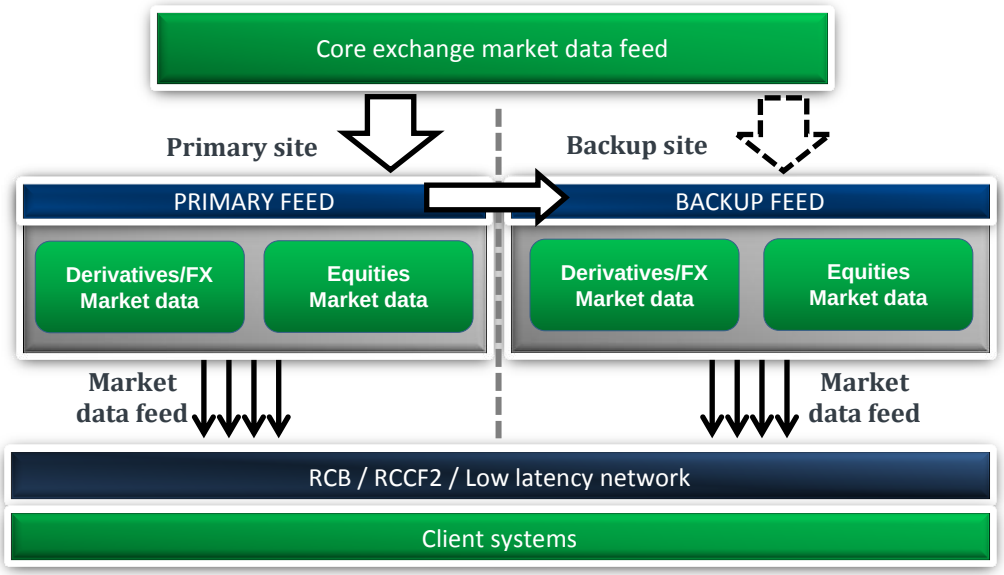
Please note that FIX/FAST multicast data is available through the RCB (Rede de Comunicação BVMF, or BVMF Communications Network).

FIX / FAST Multicast Data is available through the RCB and RCCF2 (low latency network).

5.2.3 Market Data Network Contingency Feed

BVMF provides customers in remote locations the ability to receive a network contingency feed from the backup site, to strengthen stability and provide for disaster recovery. Customers that wish to receive the backup feed should contact the exchange's Market Relations Department at the e-mail bvmfsolution@bvmf.com.br and request the backup feed.

The following diagram illustrates the primary and backup feeds distribution:



BVMF suggests customers to sign up for both feeds, to increase stability. In case of disaster, only the backup feed will be available.

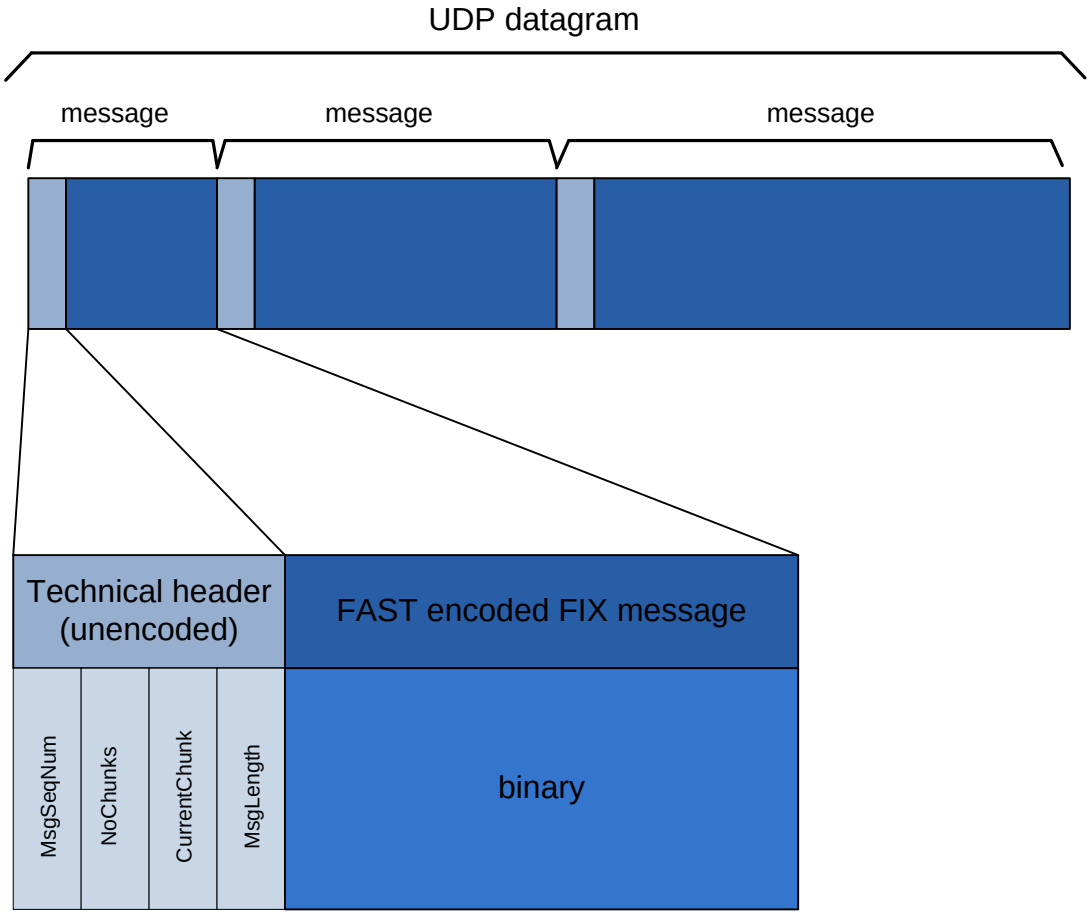
CAUTION!
On PUMA UMDF, both Incremental feeds share the exact same messages, so it´s advised to connect to both feeds simultaneously for better reliability and avoiding packet losses.

IMPORTANT
Co-location customers will not be able to arbitrate between feeds A and B as the secondary feed is not available to them. This should not be a problem since such customers have access to local network interfaces much less prone to packet loss.

5.2.4 Technical Message Header

The FIX/FAST encoded Market Data is transmitted over a network from UDP layer in chunks that is no larger than 1420 bytes including the header.

Each datagram received from client system could contain one or more messages that consist in a set of a not encoded technical header and the payload: a FAST encoded message. It is illustrated below:



The purposes of the UMDF technical message header are:

- Allow the client system to detect sequence number gaps **before** decoding the message;
- Allow for breaking-up of large messages and re-composition (e.g. market data snapshots of order depth-books may be very deep – e.g. over 100 entries for each side, bid and ask).

Before each received FIX/FAST message from UDP feed, there is the following sequence of bytes defining a header (blue rows):

MsgSeqNum	NoChunks	CurrentChunk	MsgLength	FAST message
4 bytes	2 bytes	2 bytes	2 bytes	*MsgLength* bytes

All attributes defined in the header is in "*big-endian*" convention, where bits and bytes are in network byte order, where high order bits precede low order bits, and high order bytes precede low order bytes.

MsgSeqNum – this attribute contains the same value as in the Tag *34-MsgSeqNum*.

NoChunks – total number of chunks that constitutes a single FIX/FAST Message identified by *MsgSeqNum* in the channel at the current trading session.

CurrentChunk – the current position of the chunk of data that constitutes a single FIX/FAST Message identified by *MsgSeqNum* in the channel at the current trading session.

MsgLength – The length of the following sequence of bytes that constitutes a chunk of data.

Client systems need to reassembe all chunks of data with same *MsgSeqNum* in the correct order to have a valid FIX /FAST encoded data before decoding it.

5.2.5 Instrument List Processing

The instrument definition stream replays the list of instruments of a specific channel at an exchange-defined rate. In order to correctly process the entire list of instruments for that channel, client systems must join the instrument definition stream and start decoding messages looking for the *SecurityList* message (*35=y*) which contains tag *34–MsgSeqNum* equal to 1.

From this point on, the instrument database on the client side may be populated. Each *SecurityList* message needs to be processed until the count of instruments of that channel (tag *393-TotNoRelatedSym)* is fully received. The last message in the loop will contain tag *893-LastFragment* set to 'Y'.

Note that a *SecurityList* message may contain more than one instrument definition. Deleted or expired instruments are not sent over the instrument definition stream. For deletion of instruments, the application must process the *SecurityList* message sent over the incremental stream.

The following diagram illustrates correct client system processing of the instrument definition stream:

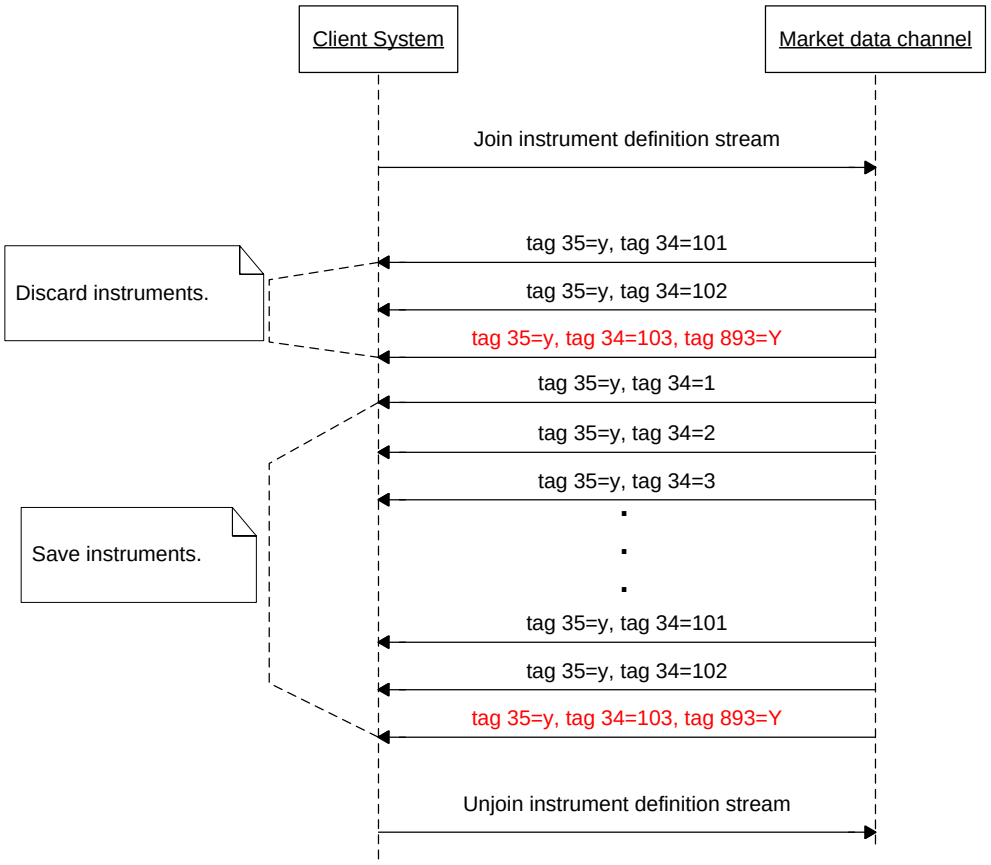


Figure 1.5 - Start of day instrument list processing.

BVMF will start issuing instrument definition messages in the instrument definition stream using the following schedule (all times are local unless stated otherwise):

Trading platform	Segment	Schedule
PUMA	Equities	**Not restarted daily**, brought down between Fri 22:00 and Sun 12:00 (local time)
PUMA	Derivatives/FX	**Not restarted daily**, brought down between Fri 22:00 and Sun 12:00 (local time)

The other feeds (incremental and snapshot recovery) are also activated at this time, but messages are only sent as they become available.

In general, for PUMA Trading System, customers may connect every day or keep connected through the week. However, BVMF recommends that customers remain disconnected during the weekends, unless when participating in scheduled mock tests.

5.2.6 Initial Market Data Synchronization Procedure

For a startup, follow the process below to ensure that all necessary market data is received:

1. Contact the TSG or visit the BVMF FTP server to get the latest configuration parameters and template files;
2. Connect to the TCP Recovery service. In case of missing packets on the incremental stream, they can be recovered using this service;
3. Join the multicast address/UDP port of the incremental stream and start receiving the market data incremental messages. **Queue** them;
4. Join the multicast address/UDP port of the security definition stream until all instruments have been received (monitor the tag *393–TotNoRelatedSym*);
5. **Unjoin** the security definition stream, to avoid consuming unnecessary bandwidth;
6. Join the multicast address/UDP port of the snapshot recovery stream until all snapshot messages have been received: monitor the tag *34-MsgSeqNum* whose value is cyclical and the tag *911-TotNumReports* = total number of snapshots in the current loop. Client systems could receive and queue snapshots until total number of snapshots received and stored is equal to the value of field *TotNumReports field (tag 911)* of the **last** snapshot message received and the older incremental data queued is **greater than** the next sequence of the lowest value of *LastMsgSeqNumProcessed* field (*tag 369)* of all snapshots stored;
7. **Unjoin** the snapshot recovery stream, to avoid consuming unnecessary bandwidth;
8. Start **by removing from the queue** the incremental stream messages applying over related snapshots until consuming all the queued messages: discard queued 35=X and 35=f messages from the incremental stream until tag *34-MsgSeqNum* in the message has the same value as tag *369-LastMsgSeqNumProcessed* in the snapshot for each instrument in the channel. The discarded messages contain information that was already included in the related snapshot message; **Do not remove from the queue messages of type 35=y (SecurityList) and 35=B (News), as they are not reflected on the received snapshot.**
9. Start normal processing with incremental messages.

NOTE
The number of snapshots sent in the snapshot recovery stream in one loop could be less than the number of instruments assigned to the related channel. Client systems must handle instruments with no snapshots as have empty books and statistical data before applying incremental data.

The following diagram illustrates the graphical representation of the steps listed above.

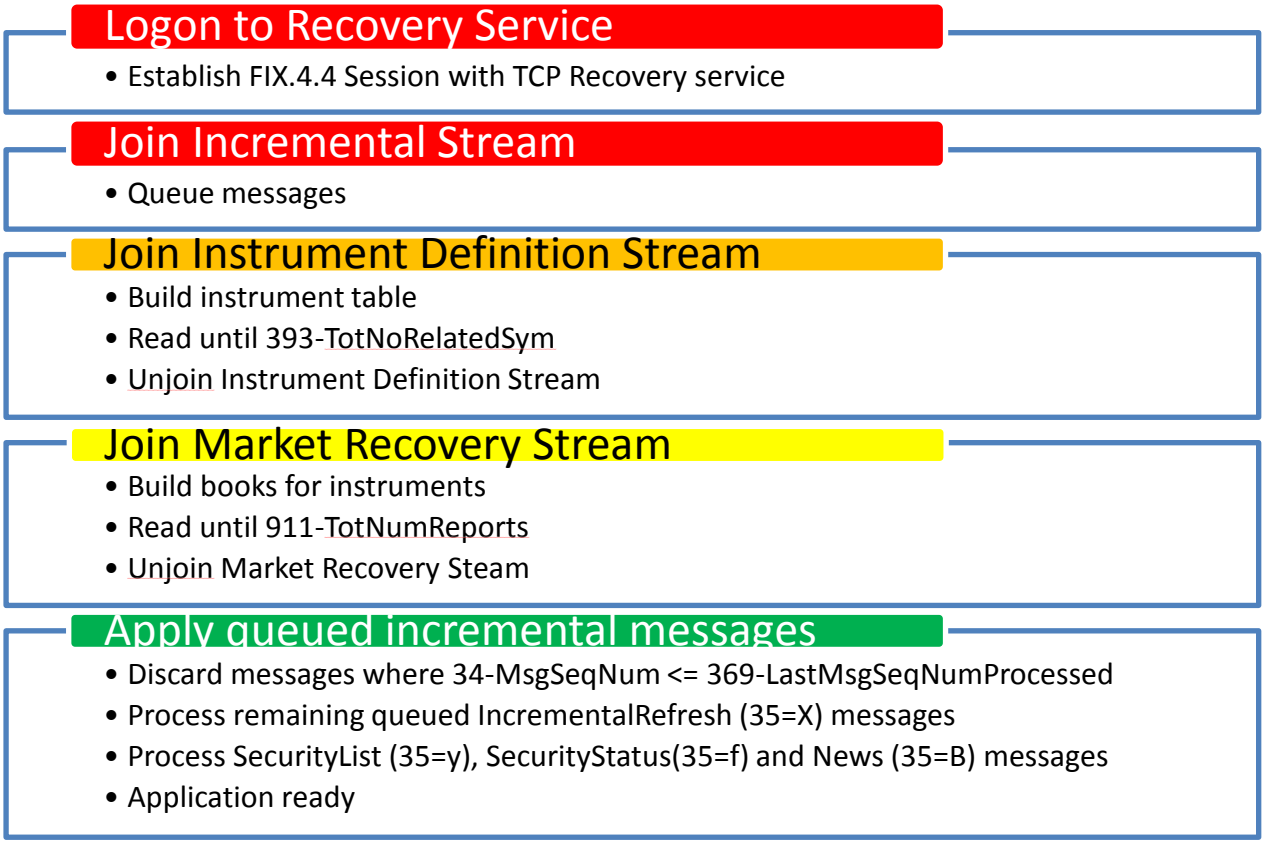


Figure 5.6 - Procedures for initial book synchronization.

5.2.7 Start of Day Heartbeats

In order to provide clients with connectivity testing before the actual streams are activated, BVMF will issue Heartbeat (tag 35=0) messages every 10 seconds (this interval is conficurabl. If client systems do not receive 3 heartbeats in a row it should consider that the multicast is not active. Note that heartbeat message is applicable to all three UDP multicast streams.

5.2.8 Stream Reset Message

Client systems should be able to handle the sequence reset (35=4) message, which is sent by BVMF in the incremental stream of any market data channel.

This message is issued in case of a severe failure in the exchange market data system, or regular start-up. This message will be sent individually for each site, i.e. if the failure occurs in the primary site, only that channel in the primary site is affected, likewise for the backup site.

This message is also sent on security definition and snapshot recovery streams when they are starting or just after a loop is finished to indicate a new loop is commencing. The stream reset is the *Sequence Reset* message (*tag 35=4*) with *NewSeqNo* field (*tag 36*) = 1 (set new sequence number).

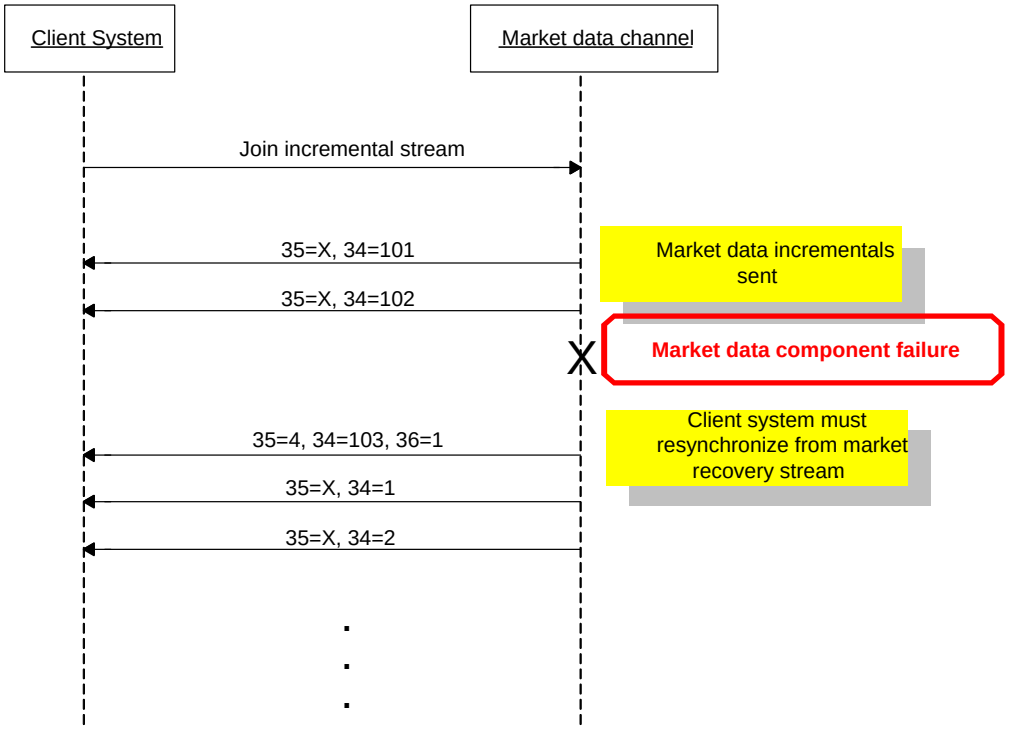
Upon receipt of this message, client systems should:

- Consider that the application sequence number has been reset, and should be started from the value in *NewSeqNo* field;
- Resynchronize their order books according to the snapshot recovery stream, as if it were a start of day synchronization.

IMPORTANT
TCP recovery is not available for messages prior to a Sequence Reset message in that channel.

IMPORTANT
The Sequence Reset message resetting the market data stream is sent at the startup of the market data component, regardless of failure or regular startup.

The following diagram illustrates an example of the Stream Reset procedures:



5.2.9 Channel Reset and Book Reset

Channel Reset will provide a process to synchronizing books (by order or by price) in the unlikely event of component failure, when books on the effected channel may be corrupted. The same mechanism is used for Book Reset, when BVMF detects that a book for specific instrument is corrupted.

In case of component failure, BVMF will issue a market data incremental message with an entry type '**J**' (Empty Book) without any instrument identification to notify client systems of book reset event. This message will also not contain tag 83-*RptSeq*.

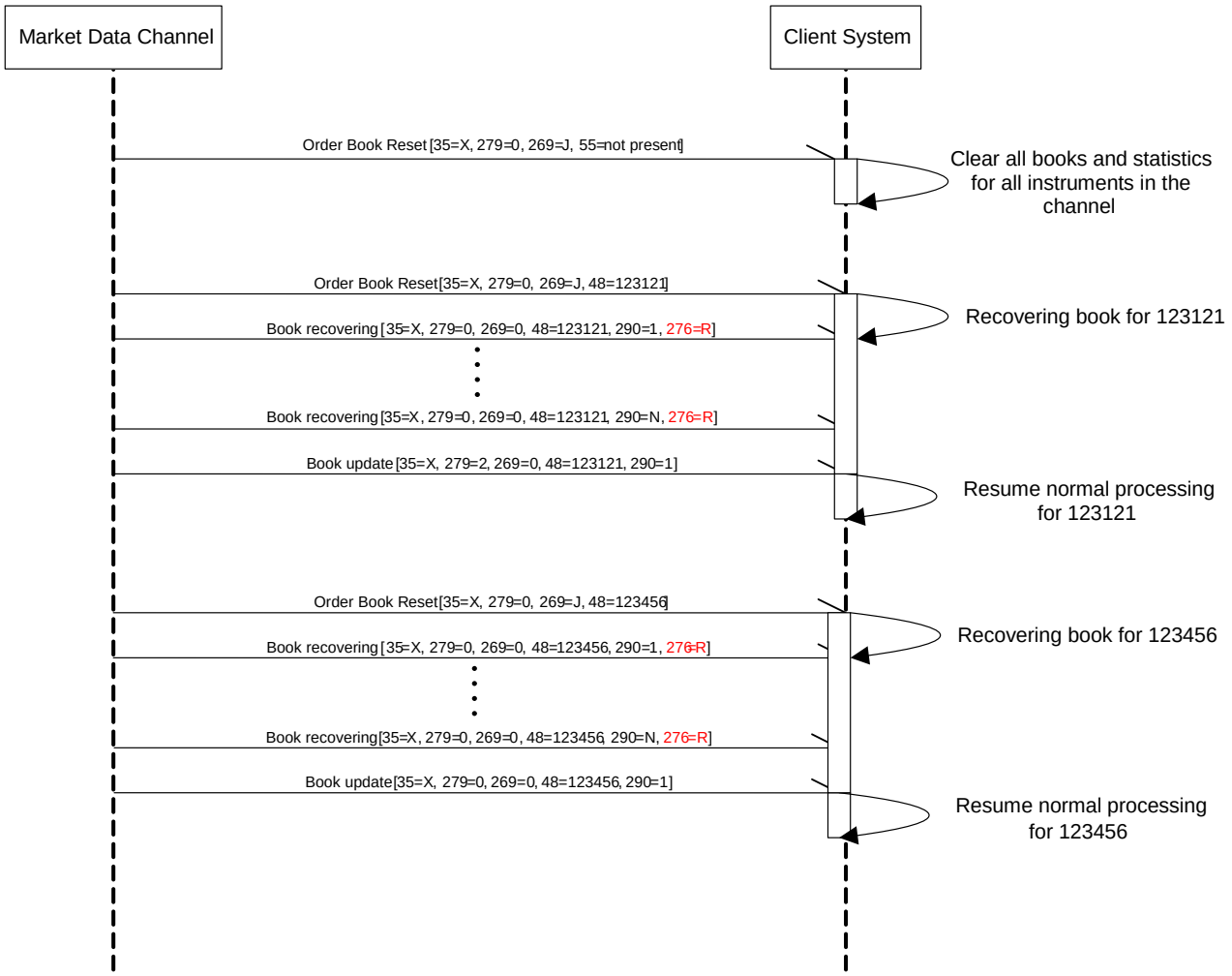
The steps to detect the Channel Reset condition and proceed to recovery process are shown below:

1) The Market Data Incremental Refresh (tag 35-*MsgType* = X) message, Channel Reset data block is sent down the Incremental feed with tag 269-*MDEntryType* = J to indicate that there has been a component failure and books of all instruments on the channel are corrupted;
2) The client system must empty books of all instruments related to the impacted channel;
3) The Market Data Snapshot Full Refresh (tag 35-*MsgType* = W) message on the Snapshot Recovery feed will be deleted for all impacted instruments;
4) Market Data Incremental Refresh (tag 35-*MsgType* = X) messages will be sent at the incremental stream to populate the book of all instruments:
    a) The first incremental Market Data Incremental Refresh message have the first repeating group of type **'J'** containing the instrument identification indicating the starting of recovery process for the given instrument (only sent for instruments that previously had a book);

b) Each of recovered book entry is identified by the presence of tag 276-*QuoteCondition* = 'R'. The tag 83-*RptSeq* fields also reset for the subsequent messages/repeating groups for the related instrument.

5) Once a book for a specific instrument has been recovered, BVMF will disseminate incremental real-time market data for that instrument (book entry will not have tag 276-*QuoteCondition* = 'R'), but other instruments on the channel may still be going through the recovery process.

Below is the sequence diagram to illustrate the full order book reset dynamics:



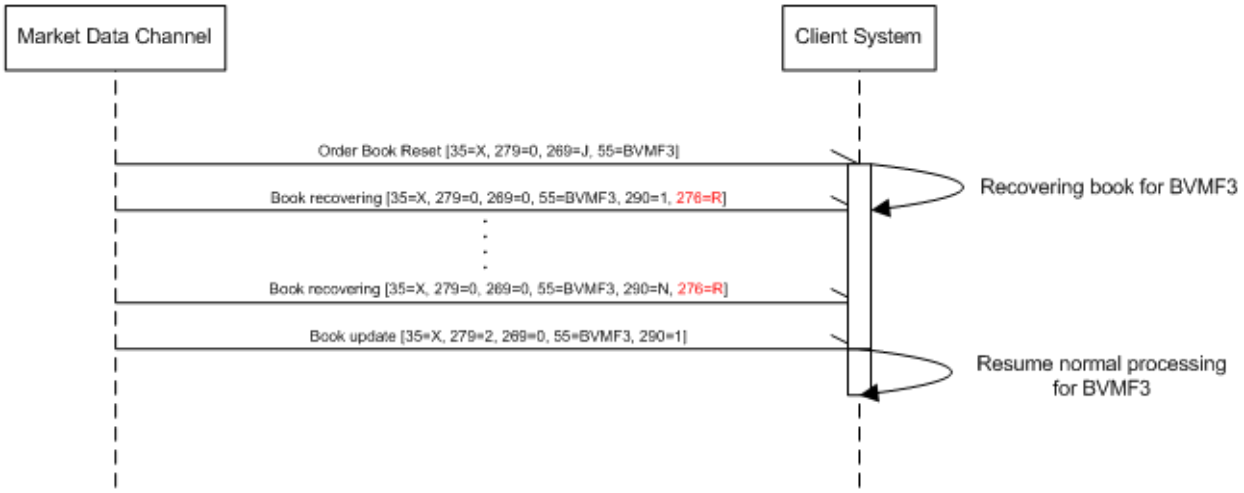
Also BVMF can send a Book Reset notification for specific instrument through issuing a market data incremental message with an entry type '**J**' (Empty Book) with instrument identification.

The steps to detect Book Reset condition and proceed to recovery process are shown below:

1. The Market Data Incremental Refresh (tag 35-*MsgType* = X) message, Book Reset data block is sent down the Incremental feed with tag 269-*MDEntryType* = J and the instrument identification component to indicate the book of this instrument is corrupted;
2. The client system must empty the book of related instrument;

3. The Market Data Snapshot Full Refresh (tag 35-*MsgType* = W) message on the Snapshot Recovery feed will be deleted for the impacted instrument;
4. Market Data Incremental Refresh (tag 35-*MsgType* = X) messages will be sent at the incremental stream to populate the book of the specific instrument, where each of recovered book entry is identified by the presence of tag 276-*QuoteCondition* = 'R'. The tag 83-*RptSeq* fields also reset for the subsequent messages/repeating groups for the related instrument.
5. Once a book for a specific instrument has been recovered, BVMF will disseminate incremental real-time market data for that instrument (book entry will not have tag 276-*QuoteCondition* = 'R').

Below is the sequence diagram to illustrate the order book reset for a specific instrument dynamics:



6. Recovery

BVMF offers some options for recovering missed messages or synchronizing client systems to the latest state: TCP Recovery and Snapshot Recovery.

Message loss is detected using the message sequence number present in the message header and tag *34-MsgSeqNum* in the decoded incremental FIX message. This attribute is an incrementing number. If a gap is detected between messages in tag *34-MsgSeqNum*, this indicates a group of messages have been missed. It should be assumed at this point that all books maintained in the client system may no longer have the correct, latest state maintained by BVMF. Client systems must resynchronize all books to the latest state maintained by BVMF. During this synchronization process, all books are initially assumed to be in an incorrect state and are recovered during the synchronization process.

NOTE

UDP protocol cannot guarantee the order of packets to be maintained, thus the customer application may receive packets out of order, and must be able to handle that gracefully.

IMPORTANT

Before requesting a message that is presumed to be lost on the TCP Recovery / Replay feed, the customer application must wait 10-20ms to be sure that the message is indeed lost (not just out of order).

6.1 Snapshot Recovery Overview

This recovery method should be used for **large-scale** data recovery (i.e. major outage or late joiners) to synchronize client systems to the latest state maintained by BVMF. Client systems can use the Snapshot Recovery stream on each channel to determine the state of each book in affected channels. Each Snapshot Recovery stream constantly loops and sends the Market Data Snapshot Full Refresh (*tag 35=W*) message. The Snapshot Recovery feed is known to be valid as of a sequence number on the Incremental Market Data feed, which is found in tag *369-LastMsgSeqNumProcessed*. This sequence number (tag *369-LastMsgSeqNumProcessed*) is found on each Market Data Snapshot Full Refresh (tag 35=W) message. Client systems will recover the most recent statistics on the Snapshot Recovery stream. Any intermediary statistics (for example trades) will not be recovered.

Some considerations:

1. Client systems should queue real-time data until all snapshot data is retrieved from a given channel. After this, the queued data should then be applied as necessary, where all queued incremental message with tag *34-MsgSeqNum* less or equal than the value of tag *369-LastMsgSeqNumProcessed* of processed snapshot should be discarded.

2. BVMF strongly recommends that the Snapshot Recovery streams be used for recovery purposes only. Once client systems have retrieved recovery data, client systems **should stop** listening to the Snapshot Recovery streams.

Recommended procedure for recovering:

1. Identify channel(s) in which the client system is out of sync;
2. Listen to and queue all the messages from incremental stream;

3. Join the multicast address/UDP port of the snapshot recovery stream until all snapshot messages have been received: monitor the tag *34-MsgSeqNum* whose value is cyclical and the tag *911-TotNumReports* = total number of snapshots in the current loop. Client systems could receive and queue snapshots until total number of snapshots received and stored is equal to the value of field *TotNumReports (tag 911)* of the last snapshot message received and the older incremental data queued is **greater than** the next sequence of the lowest value of *LastMsgSeqNumProcessed* (*tag 369)* of all snapshots stored;
4. Start **by removing from the queue** the incremental stream messages applying over related snapshots until consuming all the queued messages: discard queued 35=X and 35=f messages from the incremental stream until tag *34-MsgSeqNum* in the message has the same value as tag *369-LastMsgSeqNumProcessed* in the snapshot for each instrument in the channel. The discarded messages contain information that was already included in the related snapshot message; **Do not remove from the queue messages of type 35=y (SecurityList) and 35=B (News), as they are not reflected on the received snapshot.**
5. Unjoin the snapshot recovery stream, to avoid consuming unnecessary bandwidth;
6. Start normal processing with incremental messages.

6.2 TCP Recovery and TCP Historical Replay Overview

TCP Recovery service (previously known as TCP Replay) allows client to request a replay of a set of messages already published on the Incremental stream, using a regular FIX 4.4 session over TCP connection. The request specifies messages to be replayed based on the *MsgSeqNum* range and uses a FIX message of type Application Message Request (tag 35=BW) adapted to the FIX 4.4 specification in order to be used with the standard FIX 4.4 session layer. All messages requested are returned as FIX5.0SP2/FAST binary encoded and are embedded in one or more Application Raw Data Reporting messages (tag 35=URDR), inside the *96-RawData* field. See the message specification for more details.

The same protocol is used for TCP Historical Replay, with small differeces on the scope of what can be recovered, see section 5.1.5 for more details.

The following message types are expected from this connection:

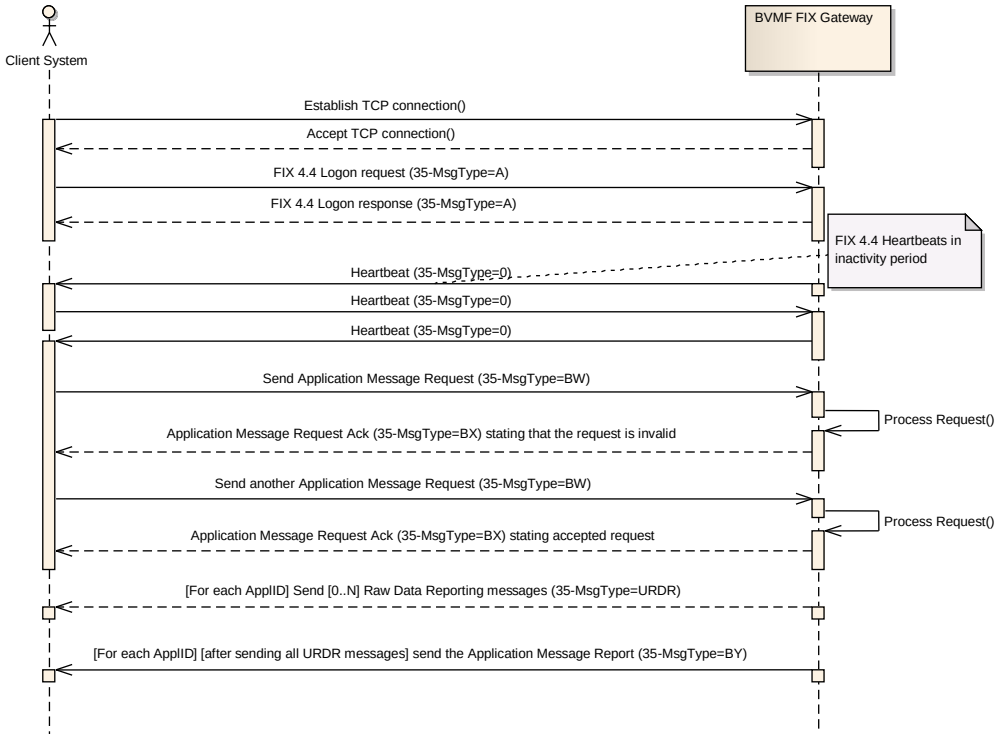
- The *Application Message Request Acknowledgment* (*tag 35=BX*) message is sent to confirm the receiving of the Application Message Request (tag 35=BW) message. The *ApplRespType* field (*tag 1348*) contains the type of acknowledgment being sent. The requested messages are resent only when the value of this tag is “0” (Request accepted) or “1” (Request partially accepted), for the later not all of the messages are resent, in this case the client application must iterate through all the *NoApplIDs* (*tag 1351*) instances to check the presence and value of the *ApplRespError* field, which has the reason for the error related to a specific *RefApplID* (*tag 1355*). The other values (greater than 1) for *ApplRespType* indicate Negative acknowledgment and the client application should verify and treat the error (see the message specification for more details).
- The *Application Raw Data Reporting* (*tag 35=URDR*) message is a BVMF user defined message created to encapsulate and make feasible the transportation of the FAST encoded messages (binary data) over a regular FIX 4.4 session using a TCP connection. The *RawData* field (*tag 96*) contains one or more FAST encoded messages. The *NoApplSeqNums* field (*tag 10054*) is the repeating group that contains the list of the message sequence numbers and related offset/length to retrieve each message in the *RawData* field (*tag 96*). The *ApplLastSeqNum* field (*tag 1350*) can be used to detect gaps (i.e., a sequence reset during the trading session). See the message specification for more details.
- The *Application Message Report* (tag 35=BY) message is used to indicate that the application resend process is complete or was interrupted because of an error. The *ApplReportType* field (*tag 1426*) reports whether the resending was successfully completed (value=3) or there was an error (value=4). A separate *Application Message Report* message is issued for each channel ID

in the request. Thus, in all messages of this type, *NoApplIDs* field (*tag 1351*) is always equal to 1. The field *RefApplID* (*tag 1355*) identifies the channel ID (incremental stream) being reported. This message might be sent immediately after the *Application Message Request Acknowledgment* (*tag 35=BX*) message (if an error occurs and messages cannot be resent), or just after the resending of the last *Raw Data Reporting* message for the related channel ID. Client application must always check the presence of the *ApplRespError* field to detect error occurrence and the field's value to know the error reason.

Some considerations:

1. The replayed messages from the current trading session are available until the next weekly trading session starts. After that, *MsgSeqNum* is reset and the old messages become unavailable.
2. The maximum number of messages that can be requested is 10000 messages.
3. BVMF strongly recommends that the client application should keep connected to TCP recovery during the whole trading session (establishing a connection for each request is not efficient and is not recommended).
4. This type of connection conforms to FIX Session layer standard defined by FPL, but *Application Message Request* (tag 35=BW), *Application Message Report* (tag 35=BY), *Application Message Request Acknowledgment* (*tag 35=BX*) and *Application Raw Data Reporting* (*tag 35=URDR*).
5. Concerning the URDR (BVMF Raw Data Reporting) message, The FAST encoded messages appended in the *RawData* (*tag 95*) field do not contain the header that is sent in the incremental stream for fragmentation/reassembly purposes. After correctly extracting a message from the *RawData* (*tag 95*) field using *RawDataOffset* (*tag 10055*) and *RawDataLength* (*tag 96*), the client application can immediately submit it to the application FIX/FAST decoder routines to obtain the final FIX.5.0SP2 *MarketDataIncrementalRefresh* (*tag 35=X*), *SecurityList* (*tag 35=y*), *SecurityStatus* (*tag 35=f*), *News* (*tag 35=B*) and *Heartbeat* (*tag 35=0*) messages.
6. BVMF expects that the adopted FIX engine at the client application side to take care of all FIX 4.4 session layer routines (i.e., the sending of heartbeat messages during the periods of inactivity)
7. BVMF recommends to reset the sequence numbers on every logon (client application should send the logon message with tag *141-ResetSeqNumFlag=Y*).
8. Retransmission from the session level is not implemented at BVMF's side; all Resend Request messages (*35=2*) are responded with a Sequence Reset (*35=4* with Gap Fill). Thus, the client application should not rely on retransmissions at the session level because this feature isn´t available through the TCP recovery connection.

The following sequence diagram describes an example of a successful scenario for the TCP recovery process:



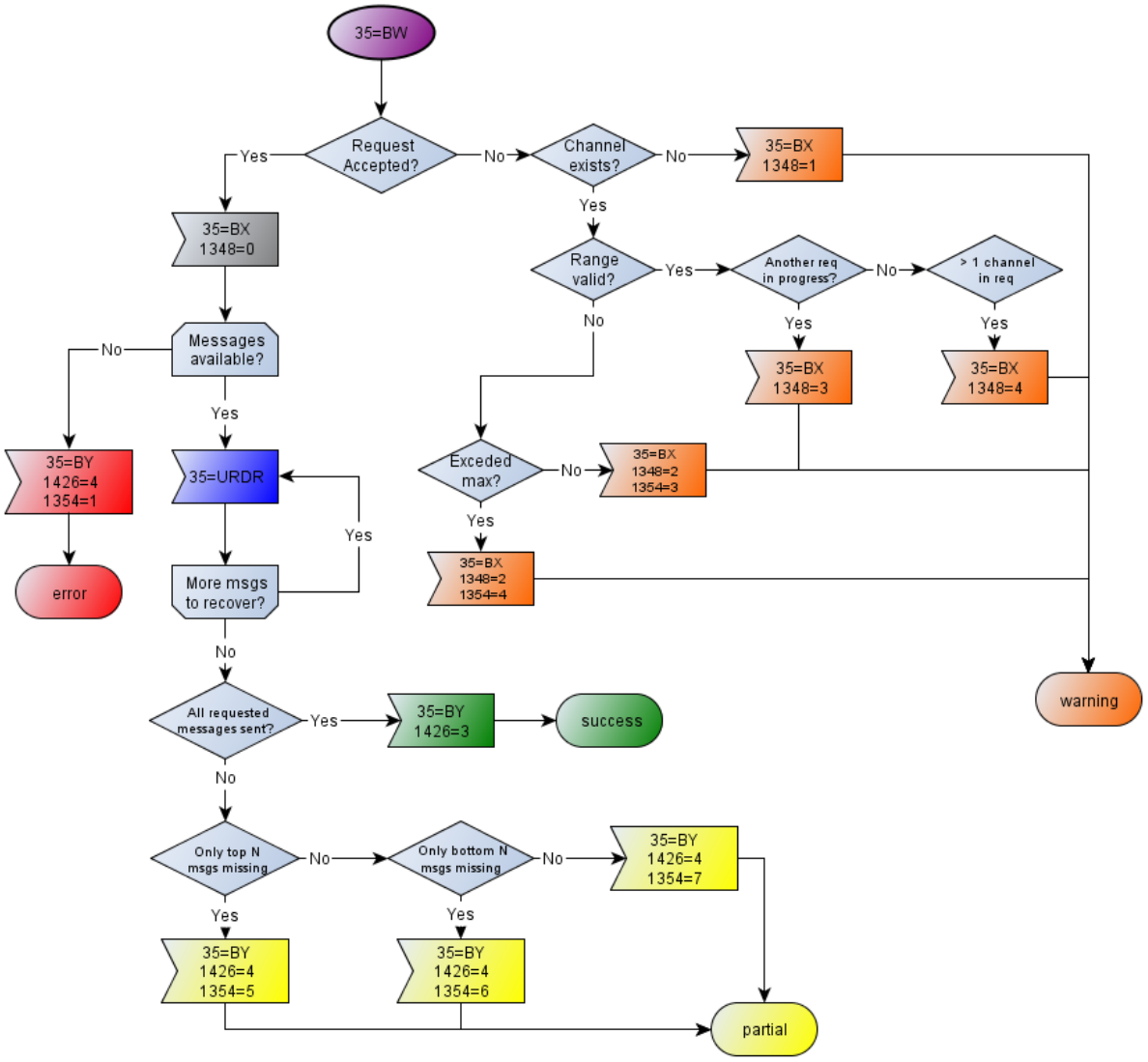
WARNING

After the message is sent on the incremental feed, it usually takes around 10-20ms for it to become available on the TCP Recovery feed. *In extreme conditions, messages may take more than 140ms to become available.*

NOTE

When using TCP Recovery / Historical Replay, since both feeds are identical, it´s preferable to use feed A always, unless this feed is not available. Only then should the application connect to feed B.

The complete map of possible scenarios and message flow when using TCP Recovery is described in the following diagram:



There are two strategies that client systems can apply to determine whether this is the moment to use TCP Recovery:

6.2.1 Message Level Sequencing

Each message received from Incremental stream has the following important information for recovery process in the header: *MsgSeqNum*, *NoChunks* and *CurrentChunk*. Before sending a sequence of bytes to the FAST decoder, the client system needs to assemble a message in correct order using the *CurrentChunk* as index with same *MsgSeqNum*. If any chunk of data is missing, the client system must request, via TCP Recovery mechanism, the entire message identified by *MsgSeqNum*, receiving the as a whole message, not broken down into chunks.

Please note that on the PUMA Trading System, as incremental feeds A and B both have messages with synchronized MsgSeqNums, the customer is advised to check feed B before proceeding to request the missing message from TCP Recovery. Even after the message is published on the incremental feeds, it may take 10-20ms for it to become available on TCP Recovery.

6.2.2 Instrument Level Sequencing

Market Data Incremental Refresh messages (tag *35=X*) contain instrument level sequence numbers (tag *83-RptSeq*), in addition to message sequence numbers (tag *34-MsgSeqNum*). Every repeating group instance of a market data entry contains an incrementing sequence number (tag *83-RptSeq*) that is associated with the instrument for which data is present in the block. Instrument level sequencing can be used to identify which instruments you have missed messages for, and apply using the TCP Recovery mechanism.

Client systems can keep track of the instrument sequence number (tag *83-RptSeq*) for every instrument by inspecting incoming data and determining whether there is a gap in the instrument sequence number (tag *83-RptSeq*).

- If there is a gap in the instrument sequence number (tag *83-RptSeq*), it indicates that data was missed for the instrument when message loss occurred.
- If there is no gap, the data can be used immediately, and it also indicates that the book for this instrument still has a correct current state.

It is likely that if only a small number of messages have been missed, there will be data in subsequent messages which are not affected by the missing data. If there are 100 instruments in a channel, for example, and the missed messages contain data for 4 of these instruments, any subsequent messages containing data about the other instruments (not affected by message loss) are still valid.

7. Market Data Entry Types

This section lists the market data entry types supported in the BVMF feed. Each entry type contains relevant trading information such as order book, trade and statistical data. Note that availability of each of these types is subject to the trading platform functionality:

MDEntryType	Description	Comment
0	Bid	The book on the buy side for the security. The book could be **order-depth** based or **price-depth** based depending on channel parameters definition. In order-depth based book, each individual order will appear as a separate book entry, even if it contains the same price as other orders, while in price-depth based book, each book entry corresponds to a price, and may contain more than one order.
1	Offer	The book on the sell side for the security. The book could be **order-depth** based or **price-depth** based depending on channel parameters definition. In order-depth based book, each individual order will appear as a separate book entry, even if it contains the same price as other orders, while in price-depth based book, each book entry corresponds to a price, and may contain more than one order.
2	Trade	The completed trades for the security.
3	Index Value	Data related to indexes and ETFs (Exchange Traded Funds).
4	Opening price	The opening price of the security (first trade).
5	Closing price	The closing price of the security (previous day's last trade).
6	Settlement price	The settlement price of the security.
7	Trading session high price	The highest price traded for the security in the trading session.
8	Trading session low price	The lowest price traded for the security in the trading session.
9	Trading session VWAP price	Volume-Weighted Average Price, the ratio of the value traded to total volume traded over the trading session. Calculated using the formula: $P_{VWAP} = \frac{\sum_j Q_j P_j}{\sum_j Q_j}$ Where: P_{VWAP} = Volume Weighted Average Price P_j = price of trade *j* Q_j = quantity of trade *j* *j* = each individual trade that takes place over the defined period of time (including cross trades).
A	Imbalance	Information related to imbalance of auctions such as side and quantity.
B	Trade volume	The total volume traded for that security in the trading session.
C	Open Interest	Total number of contracts in a commodity or options market that are still open; that is, they have not been exercised, close out, or allowed to expire. The term also applies to a particular commodity or, in case of options, to the number of contracts outstanding on a particular underlying security. The level of open interest is reported daily.
J	Empty Book	Indicates that the order book for the related instrument (or for all instruments of the channel) is no longer valid.

MDEntryType	Description	Comment
c	Security trading state	The trading status and/or phase of the security.
g	Price band	Contains price banding information.
h	Quantity band	Contain quantity band information.
D	Composite underlying price	Contains price composition for BDR index instruments. *For future use*.

8. Intraday Instrument Definition Updates

The definition of instruments may change intraday, i.e. instruments may be added, deleted, or have their characteristics changed at the exchange's market operations discretion. Hence, client systems must be able to handle these events in order to correctly list the tradable instruments to its customers.

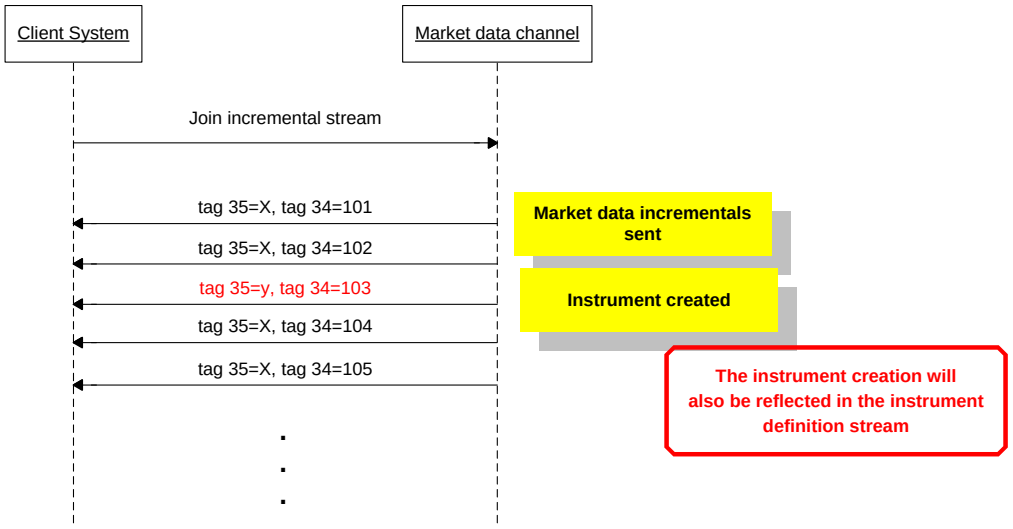
After the start of day procedure to retrieve the list of instruments, any new updates will be sent over the incremental stream of the market data channel. These updates will be available in the TCP recovery functionality as well.

Updates to the instrument definitions will also be reflected in the instrument definition stream for late joiners, however client systems that have already constructed their instrument database as per the start of day procedure should rely on the incremental stream updates instead.

The following sections illustrate the three possible types of instrument updates that will be sent over the incremental channel.

8.1 Intraday Instrument Creation

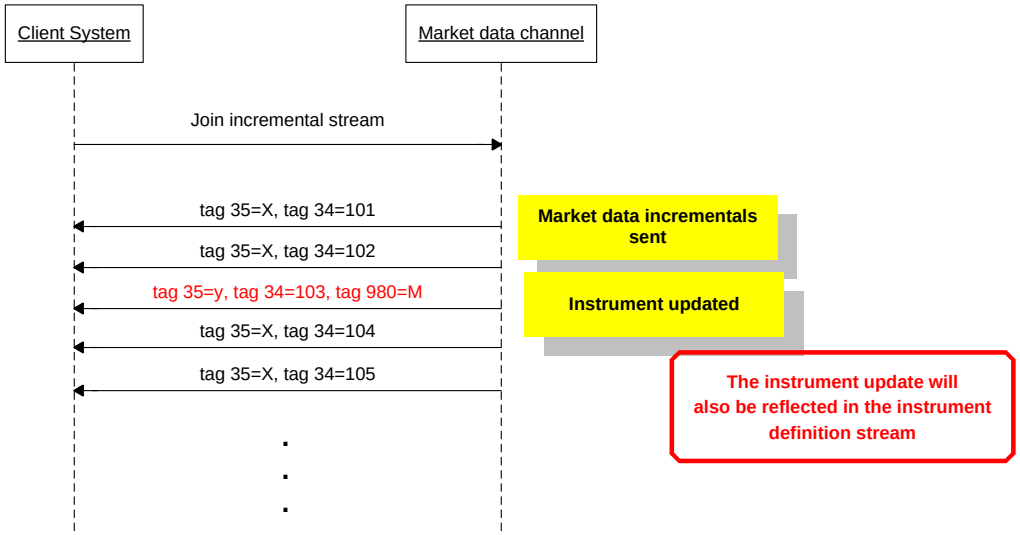
In this case, an instrument is created during the trading session.



NOTE
After an instrument is created during a session, before receiving the first 35=f (SecurityStatus) message referring that instrument, its phase and state are to be considered invalid and unknown.

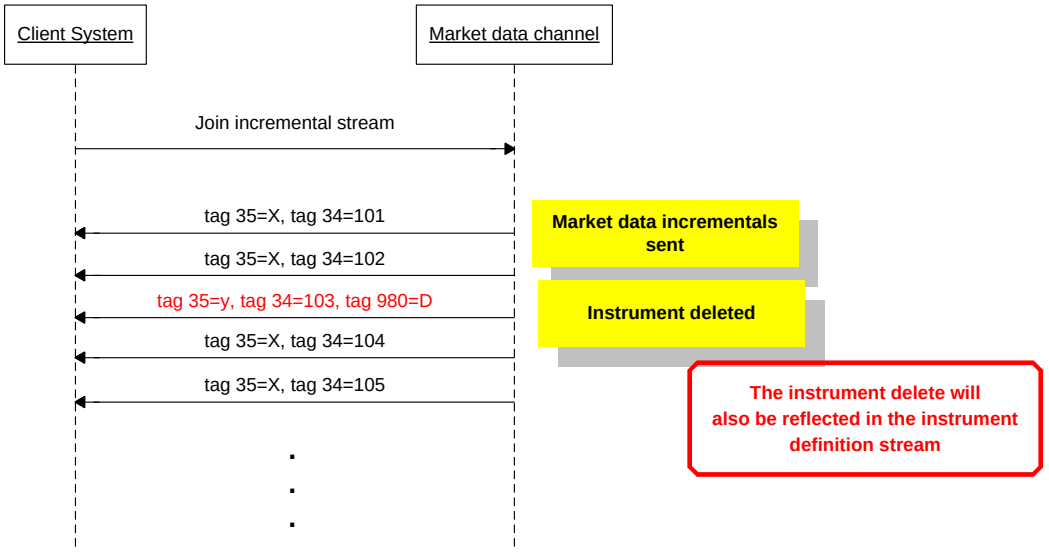
8.2 Intraday Instrument Update

In this case, one of the characteristics of the instrument was changed. BVMF will send all the instrument's characteristics, being the client systems responsibility to update its instrument database.



8.3 Intraday Instrument Deletion

In this case, the instrument was deleted, due to the fact that it is not tradable anymore. Client systems should remove the instrument from its instrument database and invalidate the order book associated to it.



9. Incremental Book Management

Books received via the FIX 5.0/FAST feed are incremental, i.e. changes to the book are relayed on individual messages providing "deltas" of the previous state of the book.

The actions to be executed by the client system receiving the incremental message are determined by tag *279-MDUpdateAction*, whose value carries an instruction that can be either add, delete, change, delete from, delete thru or overlay. The items in the order book that are affected by the action stated in tag 279 are stated in tag *290-MDEntryPositionNo*, which contains a position in the order book.

For bid or offer book entries (order and price depth book), the deletion is based on the entry's position (tag *290-MDEntryPositionNo*). For example, assume ten bids for a security. Adding a bid with *290-MDEntryPositionNo* = 4 requires the receiver to shift down other Market Data Entries, i.e. the Market Data Entry in the 4th display position will shift to the 5th, the 5th shifts to the 6th, etc. until the 10th shifts to the 11th. BVMF will not send a modification of all entries in the 4th through 10th positions just to update the *290-MDEntryPositionNo* field; the receiver of the market data must infer the change.

Similarly, deleting a Market Data Entry in the 7th position causes the 8th Market Data Entry to move into the 7th position, the 9th to shift into the 8th position, etc. BVMF will not issue a change action to modify the position of an entry in the book. Change updates are only sent when a value applicable to a specific tag *290-MDEntryPositionNo* – such as total quantity or number of orders – changes.

BVMF publishes two types of book depth: order depth and price depth using the same *MDEntryType*: 0 (Bid) and 1 (Offer). To determine which type of book is currently defined in a given channel, see "FIX/FAST Channel Definitions" documents on the website or from tag *264-MarketDepth* in the Market Data Snapshot Full Refresh (tag 35=W) message for each instrument: if it is absent, the book is order-depth based, if present, it is price-depth based and the level is determined by the value of the tag where the value 1 (one) indicates top of book.

IMPORTANT
The book could be order-depth based or price-depth based depending on channel parameters definition. Please see "FIX/FAST Channel Definitions" documents to determine which type of book each channel supports.

9.1 Order depth book

Order depth book contains order by order information, where each entry represents an individual order. For example, this is how an order-depth book looks like:

Bid			Offer		
PosNo	Size	Px	Px	Size	PosNo
1	5000	10.58	11.03	7000	1
2	4000	10.58	11.03	2000	2
3	3000	10.57	11.05	1000	3
4	4000	10.54			4

One entry per order: same price on more than one entry.

BVMF provides the full depth of the book for order-depth book, i.e. the client will always receive updates for all the orders that are in the order book, even if it is the last one (worst price).

In general, if a trade occurs, BVMF will send a delete or change data block to update the book. The trade data block itself is not used to update the order book.

Below are the data blocks sent for order depth book update:

Tag	Name	Values	Comments
279	MDUpdateAction	"0","1","2","3","4"	
269	MDEntryType	"0","1"	
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
1500	MDStreamID	C	"L" - for BTC book
270	MDEntryPx	C	Not sent for MOA and MOC
271	MDEntrySize	X	
432	ExpireDate	C	Used for BTC contracts only
37019	EarlyTermination	C	Used for BTC contracts only
272	MDEntryDate	X	
273	MDEntryTime	X	
37016	MDInsertDate	X	
37017	MDInsertTime	X	
276	QuoteCondition	"C", "R"	"R" – On book retransmission

288	MDEntryBuyer	C	Sent on bids, but not on MBP/TOB or FX
289	MDEntrySeller	C	Sent on offers, but not on MBP/TOB or FX
290	MDEntryPositionNo	X	
37	OrderID	X	

For more details, please check the UMDF Message Reference document.

IMPORTANT
The only safe way to treat references to an entry in the book is by using tag 290-MDEntryPositionNo. Using other tags such as 37-OrderID is highly disregarded and can lead to book inconsistencies.

9.2 Price depth book

Price-depth book contains price by price information, where each entry represents the aggregation of all order quantities at that price. The following table illustrates the price-depth book of the same book described above:

Bid				Offer			
PosNo	**NoOrders**	**Size**	**Px**	**Px**	**Size**	**NoOrders**	**PosNo**
1	2	9000	10.58	11.03	9000	2	1
2	1	3000	10.57	11.05	1000	1	2
3	1	4000	10.54				3

One entry per price: more than one order per entry.

In addition to the quantity and the price, the price-depth book also makes the number of orders that compose a specific price available. BVMF presets the depth of the book that will be made available per instruments, usually defaulting to 5. Client systems must determine the book-depth for an instrument from tag *264-MarketDepth* in the Market Data Snapshot Full Refresh (tag *35=W*) message.

BVMF sends an add data block if there is a new price level. Client systems should then shift price levels down, and delete any price levels past the defined depth of the book as indicated in tag *264-Market-Depth* in the Market Data Snapshot Full Refresh (tag *35 =W*) message.

The change data block is sent to update characteristics of a price level without changing the price itself, or impacting any other prices on the book (update to the order count or quantity at that price).

9.2.1 Price-depth Bottom Row Handling

For price-depth book only, the recipient of the market data must know how many price levels are being supplied by BVMF. The recipient must delete the bottom price row when the number of price rows is exceeded – BVMF will not send a delete of the bottom row when the number is exceeded. BVMF will send the bottom row again when a higher level row is deleted.

The following example illustrates this behavior:

Bid			
PosNo	**NoOrders**	**Size**	**Px**
1	2	9000	10.58
2	1	3000	10.57
3	1	4000	10.54
4	4	10000	10.53
5	3	8000	10.50

Top row of the book (best bid).

Bottom row of the book.

New buy order is received (BUY 1000 @ 10.60), updating the top of the book (bid):

Market Data Incremental Refresh	
MDEntryPositionNo	1
MDUpdateAction	New
MDEntrySize	1000
MDEntryPx	10.60
NumberOfOrders	1

Bid			
PosNo	**NoOrders**	**Size**	**Px**
1	1	1000	10.60
2	2	9000	10.58
3	1	3000	10.57
4	1	4000	10.54
5	4	10000	10.53
6	3	8000	10.50

New bottom row of the book.

Implicit deletion of the previous bottom row.

The order with price 10.57 is deleted (CANCEL BUY 3000 @ 10.57):

Market Data Incremental Refresh	
MDEntryPositionNo	3
MDUpdateAction	Delete
MDEntryPositionNo	5
MDUpdateAction	New
MDEntrySize	8000
MDEntryPx	10.50
NumberOfOrders	3

So, the book will miss the last row until the insert at the last position:

Bid			
PosNo	**NoOrders**	**Size**	**Px**
1	1	1000	10.60
2	2	9000	10.58
3	1	4000	10.54
4	4	10000	10.53

New bottom row will be sent by BVMF:

Market Data Incremental Refresh	
MDEntryPositionNo	5
MDUpdateAction	New
MDEntrySize	8000
MDEntryPx	10.50
NumberOfOrders	3

The book after the event will be:

Bid			
PosNo	NoOrders	Size	Px
1	1	1000	10.60
2	2	9000	10.58
3	1	4000	10.54
4	4	10000	10.53
5	3	8000	10.50

New bottom row will be sent by BM&F.

For information on data blocks sent for price depth book update, please check the UMDF Message Reference document.

9.3 Top of the Book (best bid and best offer)

The best bid and best offer prices are used to indicate aggregation of all order quantities at the best bid price of the current book and aggregation of all order quantities at best offer price of the current book respectively. In addition to the quantity and the price, the price-depth book also makes the number of orders that compose a specific price available.

This information is represented by a price depth book with market depth = 1 (as described at the previous sub-topic: Price depth book), and is largely used at some client systems for comprehensive overview of market data behavior of several instruments at same time.

This book management mode makes use of the Overlay method (see below) when updating the sole price level for each book.

9.4 Delete From

FIX 5.0/FAST allows for more efficient book management by providing an extension to tag *279-MDUpdateAction* allowing delete from a position.

When an order is entered that causes several executions and sweeps the order book, causing several price levels to be deleted, instead of sending deletions for several price levels, the *MDUpdateAction* "Delete From" (*tag 279* = 4) is used. It indicates that all positions from the position stated in tag *MDEntryPositionNo* up until position 1 must be deleted. This will cause the market data entry that was in position *MDEntryPositionNo* + 1 to be the first position now.

The following example of an order-depth book illustrates this behavior:

Bid			Offer		
PosNo	Size	Px	Px	Size	PosNo
1	5000	10.58	11.03	7000	1
2	4000	10.58	11.03	2000	2
3	3000	10.57	11.05	1000	3
4	4000	10.54			4

A sell order is sent with quantity 12000 and price 10.57, which executes against the 3 existing buy orders in the book. BVMF will send an incremental market data message with the following characteristics:

Market Data Incremental Refresh	
NoMDEntries repeating group instance	
MDUpdateAction	Delete From (4)
MDEntryType	Bid (0)
MDEntryPositionNo	3

The resulting book as displayed by the client system should be:

Bid			Offer		
PosNo	Size	Px	Px	Size	PosNo
1	4000	10.54	11.03	7000	1
2			11.03	2000	2
3			11.05	1000	3

9.5 Delete Thru

FIX 5.0/FAST allows for more efficient book management by providing an extension to tag *279-MDUpdateAction* allowing delete thru a position. This functionality is supported in the equity market data feed only and in this case, the value of *MDEntryPositionNo* field (tag 290) is always 1 (one). Therefore, all entries of related side of the book (Bid or Offer) are deleted.

The following example of an order-depth book illustrates this behavior:

Bid			Offer		
PosNo	Size	Px	Px	Size	PosNo
1	5000	10.58	11.03	7000	1
2	4000	10.58	11.03	2000	2
3	3000	10.57	11.05	1000	3
4	4000	10.54			4

The market supervisor decided to cancel all bid entries, so BVMF will send an incremental market data message with the following characteristics:

Market Data Incremental Refresh	
NoMDEntries repeating group instance	
MDUpdateAction	Delete Thru (3)
MDEntryType	Bid (0)
MDEntryPositionNo	1

The resulting book as displayed by the client system should be:

Bid			Offer		
PosNo	Size	Px	Px	Size	PosNo
1			11.03	7000	1
2			11.03	2000	2
3			11.05	1000	3

9.6 Overlay

Another possibility for book update is the overlay method (tag *279-MDUpdateAction=5*). This mode indicates that the price level should be replaced or inserted not implying an inconsistency when that price level is not defined. For example:

Market Data Incremental Refresh	
NoMDEntries repeating group instance	
MDUpdateAction	Overlay (5)
MDEntryType	Bid (0)
MDEntryPositionNo	1

Please note that an Overlay update can be sent without containing tag 270-MDEntryPx, meaning that price level should be removed.

10. Trade and real-time statistical data

There is a number of statistics (market data events) which are related to changes in a book but are not used to update the book. The following type of information fit this category: last best price, trade, high/low trade price, and pre-opening statistics. These events describe the behavior of the market and allow a user to know when the market is moving in a certain direction and provide historical information on how the market has performed.

For more details on each of the blocks and the message structure, please refer to UMDF Message Reference document.

IMPORTANT
Whenever handling trades and other real-time statistics, if the tag 1500-MDStreamID is present, this data must be stored separately, as they may contain information from different venues.

10.1 Trade

The trade data block is sent when a trade occurs to provide volume and trade statistics.

When a cross order is accepted in the trading system, the related market data contains a trade, tag 269-*MDEntryType* = 2 (trade) with tag 277-*TradeCondition* containing character '**X**'.

If a repeating group with tag 269-*MDEntryType* = 2 (trade) contains tag 277-*TradeCondition* containing character '**R**', it informs that this is one of trade that forms the opening trade event that indicates when an instrument is traded for the first time in the trading session in progress.

If a trade contains tag 277-*TradeCondition* containing character '**L**', it indicates that the related trade is the last trade of match or opening event.

For termo vista trades, the tag 277-*TradeCondition* will contain the character '**3**', indicating this trade was matched on that origin.

For trades happening on Strategy products, a trade on each leg is also generated. This trade is marked with 277-*TradeCondition* containing value '**1**'

Here are the FIX tags sent for a trade repeating group (X= required, C=conditional):

Tag	Name	Values	Comments
279	MDUpdateAction	"0","2"	New or Delete
269	MDEntryType	"2"	Trade
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
1500	MDStreamID	"E","T","O"	
270	MDEntryPx	C	Not sent for Block Trades (BTF)
271	MDEntrySize	X	

37014	MDEntryInterestRate	C	Only sent for Termo trades
272	MDEntryDate	X	
273	MDEntryTime	X	
37016	MDInsertDate	X	
37017	MDInsertTime	X	
274	TickDirection	C	Not sent on first trades and trades of equal price
277	TradeCondition	C	Sent on specific conditions (see above)
336	TradingSessionID	C	Sent on NRS trades
288	MDEntryBuyer	C	Not sent for FX trades
289	MDEntrySeller	C	Not sent for FX trades
451	NetChgPrevDay	C	Always sent on New Trades
287	SellerDays	C	Only sent for Termo trades
1020	TradeVolume	X	Total traded volume for the session
1003	TradeID	X	

NOTE 1

The last received repeating group MDEntryType (tag 269) = 2 (Trade) does not mean it is the last traded price. Please per attention to the following fields to determine the last traded price: MDEntryTime (tag 273) and TradeID (tag 1003).

NOTE 2

When receiving a message with the repeating group whose MDEntryType (tag 269) = 2 (Trade) and TradeCondition (tag 277) contains U (Exchange Last), it means that it is not a "real" trade, but only information of the last valid trade. Therefore, the related repeating group only informs the price and quantity, and not contains other information like buying and selling parties, trade identification, etc.

10.2 Trade Volume

This repeating group contains information about trade volume or options exercise summary from the registration system such as financial traded volume (in local and foreign currencies), number of trading events etc. The full description for the group is below (X= required, C=conditional):

Tag	Name	Values	Comments
279	MDUpdateAction	"0"	No Delete, always New (replace)
269	MDEntryType	"B"	
83	RptSeq	X	

48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
1500	MDStreamID	“E”,”O”,“L”,”N”	
270	MDEntryPx	C	Indicates the financial volume (R$1 mi traded)
271	MDEntrySize	C	Indicates the number of trades of the session
15	Currency	X	
272	MDEntryDate	X	
273	MDEntryTime	X	
1020	TradeVolume	C	Total traded volume for the session, TRS only sends it for E&B.

10.3 Trading Session High/Low/VWAP Price

The high trade price data block is sent for a trade event that has produced the highest trade price for the current session. Likewise, the low trade price data block indicates that a trade event has produced the lowest trade price for a given session. High, low and Volume-Weighted Average Price (VWAP) trade prices are helpful in tracking market trends. They also provide historical information for the current session regarding market behavior.

The FIX message syntax for Session High/Low/VWAP Trade Price repeating groups lie below (X= required, C=conditional):

Tag	Name	Values	Comments
279	MDUpdateAction	“0”,”2”	
269	MDEntryType	“7”,”8”,”9”	High Price, Low Price or VWAP Price
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
1500	MDStreamID	“E”,”O”,”T”	
270	MDEntryPx	X	
272	MDEntryDate	X	
273	MDEntryTime	X	

10.4 Opening price

This repeating group carries the summary information about opening trading session events per market data stream.

The theoretical opening price is also sent on this block (indicated by the presence of the tag *286-OpenCloseSettlFlag*) and is calculated and updated based on the orders presented in the book during every auction including the pre-opening / pre-closing auction.

The repeating group is described below (X= required, C=conditional):

Tag	Name	Values	Comments
279	MDUpdateAction	"0","2"	
269	MDEntryType	"4"	Opening Price
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
1500	MDStreamID	"E","O","N"	Not sent for theoretical open
270	MDEntryPx	C	For theoretical open, represents TOP
271	MDEntrySize	C	Only sent for theoretical open, as TOQ
37014	MDEntryInterestRate	C	Only sent for Termo trades
272	MDEntryDate	X	
273	MDEntryTime	X	
286	OpenCloseSettlFlag	"0","5"	
451	NetChgPrevDay	C	Not sent for theoretical open

10.5 Closing Price

Summary information about closing trading sessions per market data stream is described below (X= required, C=conditional):

Tag	Name	Values	Comments
279	MDUpdateAction	"0"	No Delete, always New (replace)
269	MDEntryType	"5"	Closing Price
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
1500	MDStreamID	"E","O","T" ,"N"	
270	MDEntryPx	X	
9325	LastTradeDate	C	Always sent for Adjusted close (286=4)
37013	PriceAdjustmentMethod	C	Can be sent for Adjusted close (286=4)

272	MDEntryDate	X	
273	MDEntryTime	X	
286	OpenCloseSettlFlag	“0”,”3”,”4”	3 – used for index preliminary close

10.6 Settlement Price

The price data block is sent to update opening (current trading session), previous day adjustment and settlement price. This data block is useful for obtaining the settlement price and the previous day’s adjusted closing price and is sent after the close of the trading session and the opening price (price of first trades in the current session).

Here are the FIX tags normally sent for this type of repeating group (X= required, C=conditional):

Tag	Name	Values	Comments
279	MDUpdateAction	“0”	No Delete, always New (replace)
269	MDEntryType	“6”	Settlement Price
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
1500	MDStreamID	“A”,”N”	
270	MDEntryPx	X	
423	PriceType	C	Never sent for index settlement
272	MDEntryDate	X	Message creation date (not settlement date)
273	MDEntryTime	X	Message creation time (not settlement time)
286	OpenCloseSettlFlag	“1”, “4”	For today (1) or previous day (4)
731	SettlPriceType	“1”, ”2”, “3”	Final(1), Preview(2), Updated(3)

Usually, settlement prices are sent in a regular order during the trading day. However, Market Surveillance can issue corrections or updates to the settlement price for the previous or current session. Hence, it´s advised that the customer application is able to process any combination for the tags 286-OpenCloseSettlFlag and 731-SettlPriceType.

The following table illustrates the regular schedule:

Event time	[286]	[731]	Description
Before session	4	1	Previous day final settle
During session	4	3	Previous day updated settle
During session	1	2	Current day preview settle
After session	1	1	Current day Final settlement

10.7 Auction Imbalance

This repeating group carries auction imbalance information, indicating the remaining quantity and to which side (buyer or seller) the auction is pending towards.

Here are the FIX tags normally sent for this type of repeating group (X= required, C=conditional):

Tag	Name	Values	Comments
279	MDUpdateAction	"0","2"	
269	MDEntryType	"A"	Imbalance
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
271	MDEntrySize	X	Indicates the remaining auction quantity
272	MDEntryDate	X	
273	MDEntryTime	X	
277	TradeCondition	"P","Q"	More buyers (P) or More Sellers(Q)

NOTE

Customer applications are responsible for deleting the existing theoretical opening price and imbalance information from memory after trading phase changes from Pre-Open (tag 625-TradingSessionSubID = 21) for each instrument in the group whose trading status is different from Pre-Open status (tag 326-SecurityTradingStatus = 21).

10.8 Open Interest

Open interest (also known as open contracts or open commitments) denotes the total number of contracts in a commodity or options market that are still open; that is, they have not been exercised, close out, or allowed to expire. The term also applies to a particular commodity or, in case of options, to the number of contracts outstanding on a particular underlying security.

Below is the basic template of both market data entry type (***C***) (X= required, C=conditional):

Tag	Name	Values	Comments
279	MDUpdateAction	"0"	No Delete, always New (replace)
269	MDEntryType	"C"	
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	

1500	MDStreamID	"E","L"	"L" – for BTC statistics
270	MDEntryPx	C	Sent only for BTC (security lending) contracts
271	MDEntrySize	X	Indicates volume of contracts currently open
272	MDEntryDate	X	
273	MDEntryTime	X	

10.9 Price and Quantity Band Information

Most of the information regarding price and quantity tunnels and bands is relayed on the Market Data channel for each specific instrument. For a comprehensible list of all tunnels and bands, see section 13.6.

Here are the FIX tags normally sent for this type of repeating group (X= required, C=conditional):

Tag	Name	Values	Comments
279	MDUpdateAction	"0"	No Delete, always New (replace)
269	MDEntryType	"g", "h"	Price band (g), Quantity band (h)
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
6939	PriceBandType	"1","2","3","4"	Not sent for reference price and quantity limits
1306	PriceLimitType	"0","2"	Not sent for reference price and quantity limits
1148	LowLimitPrice	X	
1149	HighLimitPrice	X	
1150	TradingReferencePrice	C	Only sent for Economic Indicator instruments
37008	PriceBandMidpointPriceType	"0","1","2"	Only sent for Rejection and Auction Bands
37003	AvgDailyTradedQty	C	Only sent for Quantity Limits
1140	MaxTradeVol	C	Only sent for Quantity Limits
272	MDEntryDate	X	
273	MDEntryTime	X	

The tunnels don´t change intraday. It is also known as "oscillation tunnel" establishing the price limits (lower and higher) of an instrument. Any order submitted with a price below the low limit or above the high limit will be rejected.

10.10 Index Statistical Data

This group of information is only applicable to indexes channel (indexes and ETFs) for equity markets. For this specific channel, only the following entry types are valid: 3 (Index Value), 4 (Opening Price), 5 (Closing Price), 6 (Settlement Price), 7 (Trading Session High Price), 8 (Trading Session Low Price), 9 (Trading Session Average Price) and D (Index Composite Underlying Price).

Market Data entry type Index Value (3) is used to inform the current value of given index and described as follows:

Tag	Name	Values	Comments
279	MDUpdateAction	"0","1","2"	
269	MDEntryType	"3"	
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
1500	MDStreamID	"N"	
270	MDEntryPx	X	The current index point value
37100	IndexSeq	X	
272	MDEntryDate	X	
273	MDEntryTime	X	
274	TickDirection	C	

The Composite Underlying Price (269=D) block is used to inform the price composition of BDR indexes (*for future use*):

Tag	Name	Values	Comments
279	MDUpdateAction	0	
269	MDEntryType	"D"	Composite Underlying Price
83	RptSeq	X	
48	SecurityID	X	
22	SecurityIDSource	X	
207	SecurityExchange	X	
1500	MDStreamID	"N"	
37100	IndexSeq	X	
711	NoUnderlyings	X	
309	UnderlyingSecurityID	X	
305	UnderlyingSecurityIDSource	X	
308	UnderlyingSecurityIDExchange	X	

810	UnderlyingPx	X	
37018	UnderlyingPxType	C	Sent for TOB average
272	MDEntryDate	X	
273	MDEntryTime	X	

11. Group phase/Instrument State Information

BVMF will relay the state of an individual instrument or a group of instrument using two messages:

- ***MarketDataSnapshotFullRefresh (35=W)*** message in the snapshot recovery stream: used for initial setup of the instrument or instrument group upon client system startup;
- ***SecurityStatus (35=f)*** message: used to relay instrument state changes intraday.

When the client system starts up, it should consider that all snapshots contain the current state of the individual instrument. Intraday updates may be done on the instrument group level.

NOTE
Group codes may repeat amongst different exchanges, hence it is advisable that client systems use the key group code (tag 1151 – SecurityGroup) + exchange (tag 207 – SecurityExchange).

When processing the *SecurityStatus* message (*tag 35=f*), client systems must first look for tag *1151-SecurityGroup*. This tag contains the group identification of a set of instruments. That being the case, all individual instruments of that set will have their status changed to the value of tag *326–SecurityTradingStatus*. The following message example illustrates the change of trading phase of the group "XX" to "Pause":

MsgType = f (*SecurityStatus*)		
Tag	**Tag Name**	**Value**
1151	SecurityGroup	XX
207	SecurityExchange	BVMF
625	TradingSessionSubID	02

If tag *1151-SecurityGroup* is not present in the message, then the message refers to an instrument, referred by tag *48-SecurityID*. The following message show a case where the instrument changes to Pause, separating from the group Phase (tag 1174-SecurityTradingEvent=101).

MsgType = f (*SecurityStatus*)		
Tag	**Tag Name**	**Value**
48	SecurityID	99999999
22	SecurityIDSource	8
207	SecurityExchange	BVMF
326	SecurityTradingStatus	02
1174	SecurityTradingEvent	101

Please see the complete *SecurityStatus* message format at UMDF Message Reference document.

NOTE
Either tag 625 or 326 are sent at once per 35=f message. Whenever an instrument state rejoins the group phase (1174=102), it´s safe to infer the group phase (tag 625) from the current instrument state (tag 326).

11.1 Possible Instrument States

The list of possible instrument states available on the tag ***326***-*SecurityTradingStatus*, are indicated by the following table:

Name	Tag Value	Description
Trading halt (*Pause*)	02	This instrument state is used by surveillance to prevent order entry and matching by market operations or schedule. Orders in the book are not eliminated when instrument entering this state.
No-Open (*Close*)	04	This instrument state is used by market surveillance in order to perform a limited number of functions, including in particular, consultations. Users have no order entry, modification or cancel capability during this phase.
Ready to trade (*Open*)	17	This instrument state is used by subscribers and surveillance to enter, modify and cancel orders, subject to cancellation and modification rules. The orders entered during this period result in immediate trading if counterparty is matched and the specific instrument status also equals to “Open”. Otherwise, the more restrictive status rules.
Not available for trading (*Forbidden*)	18	This instrument state is used by surveillance to prevent order entry and matching by market operations command or schedule. Users have no order entry, modification or cancel capability during this phase.
Pre-Open (*Reserved*)	21	Reserved state is the auction functionality that can be triggered (auction band and self-trading for illiquid instrument, for example) or started by Market Operations by command. Time of state can be pre-defined. The reserve state can have a defined time with the fixed closed (opening) time or random closed (opening) time. This state is used by subscribers and surveillance to enter, modify and cancel (subject to cancellation and modification rules) orders. The orders entered during this period do not result in immediate trading but are used to determine a Theoretical Opening Price. State ended when trades are created based on auction algorithm prior to transition to another state.
Final Closing Call	101	This state indicates when the instrument is on the final closing call for the trading day (Equities only). It behaves similarly to Reserved state.

11.2 Trading Phases

A trading phase identifies the "state" of a whole group of instruments in terms of trading session. By default all instruments follow the trading phase of the group they belong to.

For example, group "XX" may be in trading phase "*Open"*, but instrument ABCD that belongs to group "XX" is in the "*Pause"* status – due to market surveillance command. This information is especially useful when client systems want to determine the state of the group altogether, and outlining the individual state of the instrument.

Trading phase information is relayed to client systems using tag ***625***–*TradingSessionSubID*.

The following table presents the domain of possible trading phases:

Name	Tag Value	Description
Pause	02	This trading phase is used by surveillance to prevent order entry and matching by market operations or schedule.
Close	04	This trading phase is used by market surveillance in order to perform a limited number of functions, including in particular, consultations. As a general rule, during this phase, surveillance checks the consistency of data and post-market state process results before the start of the trading day. Users have no order entry, modification or cancel capability during this phase.
Open	17	This trading phase is used by subscribers and surveillance to enter, modify and cancel orders, subject to cancellation and modification rules. The orders entered during this period result in immediate trading if counterparty is matched and the specific instrument status also equals to "Open". Otherwise, the more restrictive status rules.
Pre-Close	18	This trading phase is used to indicate an intervention by surveillance. If the specific instrument is in "Reserved" state, the auction continues, otherwise users have no order entry, modification or cancel capability during this phase.
Pre-Open	21	This trading phase is used to indicate that all instruments belongs to the group is in "Reserved" state except the status of the instrument indicates "Forbidden" state.
Final Closing Call	101	This phase indicates when the instruments on this group are on the final closing call for the trading day (Equities only). It behaves similarly to Pre-Close phase, however when entering this phase, MOC orders are triggered.

11.3 Trading Statistics Reset

This is a type of flag in the *SecurityStatus* (*tag 35=f*) message that advice customer systems for an event of an end of day trading statistics reset. Represented by a presence of tag 1174 – *SecurityTradingEvent* with a value = 4 (Change of Trading Session).

On receipt of this message and flag, client systems are expected to reset the following entry types:

MDEntryType	Description
2	Last Trade
3	Index Value
4	Opening Price
4	Theoretical Price (tag 286=5)
7	Trading Session High Price
8	Trading Session Low Price
9	Trading Session VWAP Price
B	Trade Volume

The same statistics will no longer be available at the snapshot recovery stream upon the receipt of message *SecurityStatus* (tag *35=f*) with *SecurityTradingEvent* field (*tag 1174*) = 4 (Change of Trading Session).

11.4 Group Phase and Instrument State in the Snapshot Messages

In the full market data recovery process, the current phase and/or state of given instrument is published in the *MarketDataSnapshotFullRefresh (35=W)* message at the following format:

Tag number	Tag name	Required for this MDEntry	Description
269	MDEntryType	Y	**c** (SECURITY_TRADING_STATE_PHASE).
83	RptSeq	Y	Sequence number per instrument update.
48	SecurityID	Y	Composes the Instrument Identification Block.
22	SecurityIDSource	Y	Composes the Instrument Identification Block.
207	SecurityExchange	Y	Composes the Instrument Identification Block.
272	MDEntryDate	Y	Date of the event.
273	MDEntryTime	Y	Time of the event.
625	TradingSessionSubID	N	Group phase.
326	SecurityTradingStatus	N	Instrument state.
336	TradingSessionID	N	Sent during after-hours sessions.
342	TradSesOpenTime	N	Sent when the auction has a defined open time.
1174	SecurityTradingEvent	N	Can only be 101 or 102 (4 is never stored).

The “snapshot” reflects the last state of the instrument and the last correlated phase that affected the group where the instrument belongs to.

11.5 Derivatives/FX Phases and States handling

Please refer to section 14.6 below for specific handling instructions for the Derivatives/FX segment.

11.6 Equities Phases and States handling

For specific information on how to handle phases and states on the Equities segment refer to section 13.5.

12. BM&F Market Data Functionality

This section refers to the BM&F segment (derivatives) functionality only, and important technical information for BVMF customers on how to process this data.

12.1 Trade Volume

BVMF sends trade volume information for derivatives instruments as published by the Derivatives Clearing House, *in notional*. Client systems should expect volume information to be sent independently from trades, since the data is updated every 30 seconds and not on a tick by tick basis.

12.2 Open Interest

The Open Interest in the Derivatives Clearing House is published every minute on a per instrument basis, in case there was any change from the previous open interest value.

12.3 News Messages

News messages related to the derivatives markets will be sent in all incremental streams of all derivatives market data channels. These messages will be available for retransmission in the TCP recovery functionality.

12.4 Option Strike Price

Some instruments disseminated on the options channels are not options per se, but actually spreads on options, this happens with Rollovers and Strategies. The decision was to keep them on the same channel as their underlying options, even though they are not proper options.

Hence, as such instruments are not options, **the strike price (tag 202-StrikePrice) field will not be sent, or sent as zero**.

The proper way to identify these instruments is checking their security subtype (tag 762-SecuritySubType). The following subtypes are used identify them:

Product Description	Tag 762 value
Strategies	90
Financial Rollover	140
Agricultural Rollover	141

13. Equity Specific Market Data Functionality

This section refers to the PUMA Equities segment functionality only, and important technical information for BVMF customers on how to process this data.

13.1 Security Lending contracts (BTC)

The following entry types are supported by Security Lending contracts:

MDEntry	Description
269=0,1,2	BTC Book Entries
269=J	BTC Book Reset
269=B	BTC Trade Volume
269=C	BTC Open Interest

Special remarks:

- The BTC book order does not imply matching priority;
- All market data for BTC comes with the tag 1500-MDStreamID=L to differentiate the market data entries from other venues;
- Tag ***432-ExpireDate*** carries information on when the lending expires;
- Tag ***37019-EarlyTermination*** indicates if the lending can be terminated earlier

13.2 Fixed Income products (BovespaFIX)

As of the date mentioned in the Circular Letter 003/2014-DI, corporate fixed income market data will be published in the same fashion as other equity products, via messages 35=X, y, W, f and so on. **This message (35=n) is then DEPRECATED and will no longer be generated (see NOTE below).**

Supported by the new message 35=n (NonFixData), which has the following layout (header included):

Tag	Tag name	Req	Data type	Comment
35	MsgType	Y	String	Defines message type. Fixed value: **n** (NonFixData)
1128	AppVerID	Y	String	Specifies the service pack release being applied at message level. Fixed value: **9** (for FIX50SP2)
34	MsgSeqNum	Y	Int	Integer message sequence number.
52	SendingTime	Y	UTCTimestamp	UTC Date/ Time at message transmission.
212	DataLen	Y	Int	Integer field representing the length of the field Data (213) in bytes. Value must be positive.
213	Data	Y	Data	Actual message data stream (encoded Z5).
347	MessageEncoding	Y	String	Type of the encoded message inside the current message. Fixed value: **RLC-Z5**

This message (35=n) is used to carry a payload that is the unmodified RLC-Z5 message, from ProxyDiff, that can be extracted and processed as usual by customers and vendors already capable of processing this message.

NOTE
The Fixed Income products are currently disseminated as regular PUMA instruments, identifiable by tag *460-Product*, tag *167-SecurityType* and tag *762-SecuritySubType*. Please refer to the Message Reference document (see chapter 17) for tag usage.

13.3 Instrument definition changes

Most of the changes in this section bring more instrument details, in special for equity contracts, so there are changes to the FAST template. However, in terms of functionality, the instrument definition message remains the same.

A summary of what has been changed is listed below:

- New tag 37021-IndexTheoreticalQty only available for index instruments;
- Tag 37012-PriceDivisor also used as Double when sending index instruments, representing the index reducer;
- All values for tag 48-SecurityID (and derivated) are now Integer values (no longer Strings for Equities);
- New tag 37014-MDInterestRate for reporting the interest rate for Termo (Forward Market);
- Indicator for Non-tradable products and eligibility for GTD/GTC orders (using 870-NoInstrAttribs block);
- Support for multiple Lot Types (using 1234-NoLotTypeRules block);
- Indicator for User-defined spreads (UDS) versus Exchange-defined spreads (EDS) using tag 1377-MultiLegModel;
- Indicator for when strategy products legs contain individual prices (tag 1378-MultiLegPriceMethod);
- Corporate Action Event identification (tag 37010-CorporateActionEventID);
- Market segment indicator (tag 1300-MarketSegmentID);
- Governance level indicator (tag 37011-GovernanceIndicator);
- Special Auctions indicator (tag 37015-SecurityMatchType);
- Reviewed and enhanced security classification (tags 460-Product, 167-SecurityType and 762-SecuritySubType);
- Option Exercise and Blocking (change tag 167 domain from OPTEXC to OPTEXER, a more compatible value with FIX5.0SP2);

For specific changes for each message, please refer to the UMDF Message Reference document.

13.4 Book and statistics changes

The changes for statistics and books in this release are minor, most of them are only changes to the FAST template alone.

A summary of what has been changed is listed below:

- Adjusted closing price (269=5,286=4) also available for indexes;
- The price oscillation (tag 451-NetChgPrevDay) is also available for index instruments;

- Index sequence (tag 37100) and Composite Underlying Prices (block 269=D and group 711-NoUnderlyings in incremental messages) ***this is for future use only****;
- Date when the instrument last had a trade (tag 9325-LastTradeDate) on 269=5 (Adjusted closing price) block;
- Indicator of previous day´s closing price adjustment (tag 37013-PriceAdjustmentMethod);
- All tags of the type UTCTimeOnly and UTCTimestamp now includes milliseconds;
- Trades marked with 277-TradeCondition=U (Exchange Last) will only be generated at the beginning of the weekly session or after intraweek platform restart;
- On trades with 277-TradeCondition=U (Exchange Last), besides sending the last trade price(tag 270-MDEntryPx), the last trade quantity is also sent on tag 271-MDEntrySize;
- When processing leg trades (with tag 277=1), the trade price should not be used to infer the Last Trade Price;
- Changed the behavior for tag 37-OrderID. Whenever an order loses priority in the book, the order gets deleted and added again (279=2 followed by 279=0), with the OrderID tag being different. Also in the case of Iceberg orders, when the order is refilled, the OrderID tag will be different, making them impossible to track and differentiate from regular orders.
- Tags 272-MDEntryDate and 273-MDEntryTime behave differently in UMDF 2.0. On the improved version these tags always carry the date and time when the marked data message was generated at engine level rather than informing the date and time where the order entered the book (as was the behavior in Legacy UMDF). For book insertion date and time, refer to tags 37016-MDInsertDate and 37017-MDInsertTime.
- **When a trade deletion is sent (269=2, 279=2), the only prices that might be resent are High, Low and VWAP (in case they are changed). The other prices are only changed automatically if all trades are cancelled. In this case some of the prices can be deleted as well.**

For specific changes for each message, please refer to the UMDF Message Reference document.

13.5 Phases and states changes

The remarkable changes for phase and state handling are:

- New tag 336-TradingSessionID informing when entering non-regular trading sessions;
- New “Final Closing Call” phase and state to indicate when the group/instrument is on the final closing call for the trading day;
- New tag 1174-SecurityTradingEvent to indicate when an instrument state is separating or rejoining its group phase, facilitating the handling of instruments that behave differently from the group they are in;
- These changes are also reflected on the Snapshot message (35=W);

For specific changes for each message, please refer to the UMDF Message Reference document.

13.6 Bands and limits changes

The Equities version of PUMA Trading System has numerous enhancements regarding the reporting of bands and limits on the market data feed. The following types of bands and limits are now supported:

- Hard limits
- Rejection band

- Auction band
- Static limits
- Quantity limit

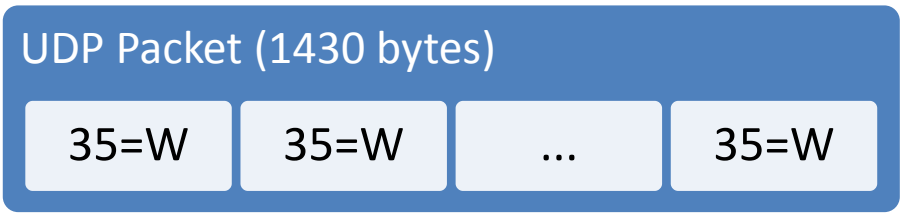
The bands are reported on the incremental refresh message (35=X) using the following tags:

Tag	Name	Reference Price	Hard Limits	Rejection Band	Auction Band	Static Limits	Quantity Limits
279	MDUpdateAction	0	0	0	0	0	0
269	MDEntryType	g	g	g	g	g	h
48	SecurityID	X	X	X	X	X	X
22	SecurityIDSource	X	X	X	X	X	X
207	SecurityExchange	X	X	X	X	X	X
83	RptSeq	X	X	X	X	X	X
272	MDEntryDate	X	X	X	X	X	X
273	MDEntryTime	X	X	X	X	X	X
6939	PriceBandType	-	1 (Hard Limit)	3 (Rejection Band)	2 (Auction Limit)	4 (Static Limit)	-
1306	PriceLimitType	-	0 (Price)	2 (in %)	2 (in %)	0 (Price)	-
1148	LowLimitPrice	-	X	X	X	X	-
1149	HighLimitPrice	-	X	X	X	X	-
1150	TradingReferencePrice	C*	-	-	-	-	-
37008	PriceBandMidpointPriceType	-	-	0,1,2	0, 1, 2	-	-
37003	AvgDailyTradedQty	-	-	-	-	-	X
1140	MaxTradeVol	-	-	-	-	-	X

(tags marked with an “X” are required, those marked with “-“ are not sent, otherwise they have the specified values)
* Currently not in use for Equities, as the reference price is always the Adjusted Close Price (see section 10.5).

13.7 Snapshot Recovery

The Snapshot Recovery feed now supports multiple Snapshot messages (35=W) per packet, greatly improving the recovery speed for channels that contain small Snapshot messages that can be bundled into a single UDP datagram (1430 bytes each).



If the customer application is designed to process a single message per UDP packet, this impact can lead to considerable functional changes.

13.8 TCP Recovery

The TCP Recovery feed can now be used to recover up to 10,000 messages missed in both Incremental feeds A and B. The response time in these circumstances should be below the millisecond range. However, it is **mandatory** to note that customers should only request messages on the TCP Recovery 20ms later, after the message being considered got lost in the incremental feeds.

Another important remark for TCP Recovery is that there is a single IP/port centralizing the recovery for all channels. The customer application should be able to connect a single time using a single session for all channels.

NOTE
Customer applications must be capable of arbitrating between both incremental feeds A and B, to be able to recover missing packages more efficiently and avoid using the TCP Recovery feed.

IMPORTANT
Co-location customers will not be able to arbitrate between feeds A and B as the secondary feed is not available to them. This should not be a problem since such customers have access to local network interfaces much less prone to lose packets.

13.9 TCP Historical Replay

Another feature available in this release of PUMA Trading System is the **TCP Historical Replay** feed that allows querying messages for the whole current trading week, starting from sequence number 1, for charting purposes. The response time for this feed is considerably slower than TCP Recovery, as it should not be used for any other purpose than charting.

For more details, refer to section 5.1.5.

13.10 Book and Channel Reset

Just after a channel reset (35=X, 269=J), the PUMA trading engine will now also resend Security Status messages (35=f) informing Phases and States for all instruments in the channel. These messages should arrive right alter the book resend (using tag 276=R) is finished.

Another important remark is that the tag 272-MDEntryDate is not sent for Channel Reset messages.

13.11 News message changes

There are some changes to the News messages (35=B) to augment the following:

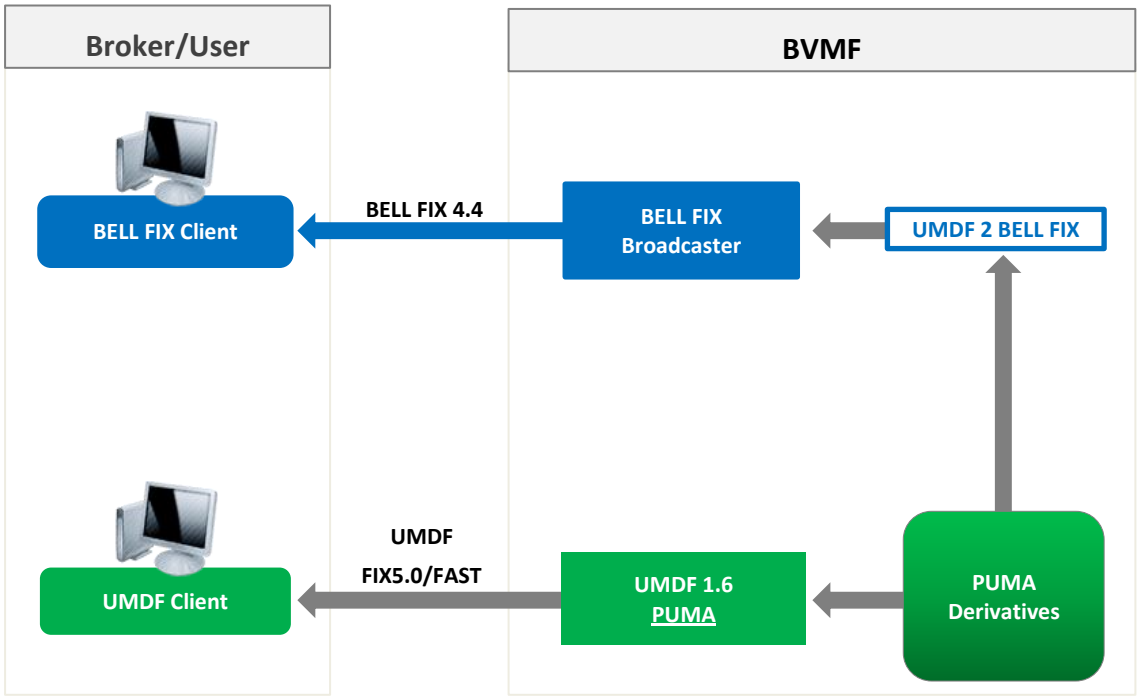
- A new Unified News Channel reserved for global news broadcast that is able to send encoded headlines and text with special characters (accented letters for instance);
- Revised news sources (tag 6940-NewsSource);

- Cross-news referencing (using tag 1472-NewsID);
- Deprecated News routing tags;

For specific changes for the News message (35=B), please refer to the UMDF Message Reference document.

14. Derivatives/FX Specific Market Data Functionality

Currently, all derivatives and FX products are traded on PUMA, but they are also available on the legacy BELL platform, according to the following diagram:



This section describes the market data improvements from PUMA Trading System.

14.1 Market on Auction (MOA) and Market on Close (MOC) Orders

Incremental messages sent for MOA and MOC orders do not contain a price and rest at the top of book while a theoretical price is calculated to initiate matching.

Legacy BELL format does not support MOA and MOC. For the conversion, incremental messages for MOA and MOC are filled up with prices based on the following algorithm:

- The best price plus one tick;
- If the book has just MOA / MOC orders, the Theoretical Opening Price (TOP) will be sent;
- If the book has just MOA / MOC orders and TOP is undefined, the last traded price will be sent;
- In the absence of a last traded price, the settlement price will be sent.

The following table illustrates an example of market by order book considering a tick increment of 0.05:

PUMA UMDF - MBO				LEGACY BELL - MBO			
Bids		Offers		Bids		Offers	
Qty	Price	Price	Qty	Qty	Price	Price	Qty
8400	MOC	MOC	1500	8400	50.05	22.45	1500
3900	MOA	MOA	1000	3900	50.05	22.45	1000
11000	MOA	MOA	100	11000	50.05	22.45	100
300	MOA	22.50	3000	300	50.05	22.50	3000
300	50.00	22.50	2000	300	50.00	22.50	2000

1100	23.20	22.78	100	1100	23.20	22.78	100
1100	23.20	22.78	17700	1100	23.20	22.78	17700
1200	23.20	22.89	200	1200	23.20	22.89	200
		22.89	500			22.89	500

The following table illustrates an example of market by price book considering a tick increment of 0.05:

PUMA UMDF - MBP						LEGACY BELL - MBP					
Bids			Offers			Bids			Offers		
NoOrd	Qty	Price	Price	Qty	NoOrd	NoOrd	Qty	Price	Price	Qty	NoOrd
4	23600	N/A	N/A	2600	3	4	23600	50.05	22.45	2600	3
1	300	50.00	22.50	5000	2	1	300	50.00	22.50	5000	2
3	3400	23.20	22.78	17800	2	3	3400	23.20	22.78	17800	2
			22.89	700	2				22.89	700	2

14.2 Snapshot Feed

To indicate that the snapshot message loop is empty, Legacy UMDF feed sent two sequence resets and heartbeat messages can also be sent. For PUMA UMDF, no sequence reset message is sent at start. However it keeps sending heartbeat messages to indicate that the snapshot message loop is empty. When there are messages, after each looping is complete a sequence reset is sent to indicate the sequence number is going to be reset to 1.

For Legacy UMDF, field RptSeq (tag 83) was sent within the MDEntries repeating group, but for PUMA UMDF this tag is outside the group, in the message body, and it is related to all repeating groups in the message.

14.3 Incremental Feed

On PUMA, the incremental feeds A and B synchronously sends the **same messages** having the same sequence numbers, hence customers are advised to **connect to both feeds simultaneously** to diminish the need for snapshot recovery using TCP. As a special remark, the CERT environment consists only of feed A.

Delete From is not supported by PUMA UMDF.

Delete Thru is not supported by BELL feed. PUMA UMDF and Legacy UMDF support this functionality and the value of MDEntryPositionNo field (tag 290) is always 1 (one). For PUMA UMDF delete thru can be sent even if the book is empty.

14.4 TCP Recovery Feed

PUMA UMDF will keep using the same kind of FIX 4.4 session used on Legacy UMDF TCP Recovery (previously known as TCP Replay), however there is only a single IP address and port for all the channels. To distinguish to which channel one wants to recover messages from, the channel id must be specified on the 1180-ApplID and 1355-RefApplID tags, using the channel ids as strings, the same way they are specified on the Market Data Channels Definition document (see section 5.1).

The TCP Recovery feed for PUMA allows recovery of up to 10000 messages lost on the incremental feed, 2000 per request.

For customers needing to recover more than 10000 messages, it´s advised to do a full snapshot recovery, as explained on section 6.1.

IMPORTANT
On PUMA, as there is a unified TCP Recovery channel, the customer only needs to have a single SenderCompID and a single connection to recover message from all channels. If, for some reason the application requires additional CompIDs, they need to be requested separately.

14.5 Trading Phases

Trading phase information is relayed to client systems using the field TradingSessionSubID (tag 625). For Legacy UMDF and PUMA UMDF the data type of this field is string and the following table presents the domain relation between BELL and UMDF for possible trading phases:

Phase Name	BELL Tag Value	LEGACY / PUMA UMDF Tag Value
Pause	N	02
Close	M	04
Open	S	17
Pre-close	F	18
Pre-open	E	21

14.6 Phase and State behavior

The trading state of an instrument is usually controlled at the group level. Security groups contain one or more instruments that are similar in terms of market rules and behavior. Managing instrument at the Security group level simplifies operations for BVMF and member firms. Instruments normally operate within a security **group phase** contained in field TradingSessionSubID (tag 625). Changes to security group phase are communicated using the SecurityStatus (35=f) message. Security group level messages will contain only the SecurityGroup (tag 1151) and the TradingSessionSubID (tag 625) fields. The Symbol (tag 55), SecurityID (tag 49), and SecurityIDSource (tag 22) will not be populated in SecurityStatus (35=f) messages communicating group status information.

An individual instrument may not be able to operate in the same group phase based upon market conditions and operational status. For example, an equity futures contract may not be able to trade because of a halt in trading of the underlying. In such as case BVMF market operations will change the state of the instrument.

When an instrument state is changed and is no longer operating within the security group phase, a separate SecurityStatus (35=f) message will be transmitted, which serves as indication that the instrument has separated from its security group phase. The SecurityStatus(35=f) message for an individual instrument will contain the Symbol (tag 55), SecurityID (tag 49), SecurityIDSource (tag 22), and the SecurityTradingStatus (tag 326) fields reports the new **instrument state**.

Users should note that PUMA UMDF fields SecurityTradingStatus (tag 326) and TradingSessionSubID (tag 625) fields use the same domain of values, where the first carries information of the instrument level state and the later provides group level phase updates.

Instruments that separate from their security group can later rejoin the group. The current version of UMDF does not provide an explicit message to alert the user when the instrument has rejoined its security group. Applications must store the value of the TradingSessionSubID (tag 625) within your application for each security group and use it to compare to the value contained in SecurityTradingStatus (tag 326) provided in Security Status (35=f) messages sent at the instrument level to determine if the instrument has rejoined the group. When an instrument separates from the group it becomes important to save the value contained in SecurityTradingStatus (tag 326) in your application program as well. Your

application should also store whether an instrument is separated from its security group. And use the following tables to determine the security group membership status of an instrument.

The following table is used to determine how to process an instrument level Security Status (35=f) message (Symbol (tag 55), SecurityID (tag 48), SecurityIDSource (tag 22), and SecurityTradingStatus (tag 326) are populated).

Event (35=f)	**contains 326 equals to stored 625**	**contains 326** different **from stored 625**	**contains 326 equals to stored 625**	**contains 326** different **from stored 625**
Group following state	**Separated**	**Separated**	**Following**	**Following**
Action	Remove 326 for that Instrument	Replace 326 for that Instrument	Store 326 for that Instrument	Store 326 for that Instrument
Following behavior	Instrument rejoins and follows group	Instrument separates from group	Instrument separates from group	Instrument separates from group

When a security group level Security Status (35=f) message is received (SecurityGroup (tag 1151) and Symbol (tag 55), SecurityID (tag 48), SecurityIDSource (tag 22) are not provided) the following table determines the state of the instrument in terms of security group membership. This table shows that the receipt of a security group level Security Status (35=f) message will never cause a change in terms of leaving or rejoining the security group, but it is important to store the current security group state for that instrument.

Event (35=f)	**contains 625 equals to stored 326**	**contains 625 different from stored 326**	**contains 625**
Group following state	**Separated**	**Separated**	**Following**
Action	Store 625	Store 625	Store 625
Following behavior	Instrument remains separated from group	Instrument remains separated from group	Instrument continues to follow group

14.7 Sequence Number Resets

Client systems must expect that the incremental market data feed generated by PUMA only resets the message sequence numbers (tag 34 – MsgSeqNum) on a weekly basis. Current implementation of Legacy UMDF resets sequence numbers on a daily basis.

Instrument replay and snapshot message loops in Legacy UMDF are reset by Sequence Reset message (35=4). This behavior is also present on PUMA UMDF.

14.8 Theoretical Price Deletion Behavior

Legacy UMDF did not remove the theoretical price after the end of an auction. PUMA UMDF will send a delete action after all auction types (pre-opening, triggered and pre-closing).

14.9 Theoretical Opening Price

Imbalance information is sent at repeating group whose MDEntryType is “A” (Imbalance), but Legacy UMDF does not send remaining unfilled quantity (tag 271) and side of remaining unfilled quantity (tag 277). In PUMA UMDF these fields will be sent.

14.10 Trade Volume

On PUMA UMDF, the total traded volume on the electronic platform (1500-MDStreamID=E) is reported on each Trade block (tag 269=2), the volumes generated in other venues is reported on the Trade Volume block (269=B).

Also, the Trade Volume block (269=B) also carries more detailed information regarding financial volume (on tag 270-MDEntryPx) and number of trades (tag 271-MDEntrySize) and also informs the total traded volume for non-electronic venues (for instance option exercise and ex-pit, 1500=O and 1500=X respectively).

For PUMA UMDF, the trade volume is sent every 30 seconds.

NOTE
All trade volume information is sent per instrument not per segment. All summarization must be processed by the client.

14.11 Time of Market Data Entry

For PUMA UMDF, field MDEntryTime (tag 273**) contains milliseconds**. For example, for time of market data entry “16:21:09.364” (format HH:MM:SS.sss), field MDEntryTime will contain “162109364”.

There are also two new fields (tag 37016-MDInsertDate and tag 37017-MDInsertTime) that inform the date and time the order has entered the book, or, for trades, the original trade date and time.

15. Certification Process for FIX/FAST

Client system must certify against the FIX 5.0/FAST feed before being deployed in production. The certification process consists of:

- Establishing connectivity to the certification environment;
- Developing and testing against the certification environment feed;
- Once the client system is deemed ready for certification, the customer must schedule the certification process with BVMF's certification support team (e-mail ctc@bmfbovespa.com.br);
- Customer and BVMF will execute the certification script;

If the client system correctly executes the certification script, it may proceed into production.

15.1 Connectivity to the Certification Environment

Client systems have different options of establishing connectivity to the certification environment. They are:

- Internet VPN: in this case, the customer must be able to configure a GRE tunnel with the exchange, to allow for multicast traffic to be sent.

For a detailed description of network connectivity options, please contact to the BVMF Customer Services Department:

bvmfsolution@bvmf.com.br

16. FIX/FAST Channel Definitions

BVMF make available two documents for developers and network engineering teams of client systems to connect to the unified market data feed.

16.1 Certification Environment

The certification environment multicast and TCP recovery / Historical replay channel definitions is available at the following website:

http://www.bmfbovespa.com.br/pt-br/servicos/download/MarketDataChannels_PUMA_CERT.pdf

Changes to the multicast addresses/ports and TCP recovery / Historical replay information will be notified by the exchange if applicable. Please keep in mind that message rates in the certification environment may be substantially lower than in production.

16.2 New Release Environment

For customers willing to test out new features and new releases, the recommended environment to use is New Release. This environment´s multicast and TCP recovery / Historical replay channel definitions is available at the following website

http://www.bmfbovespa.com.br/pt-br/servicos/download/MarketDataChannels_PUMA_EQT_NR.pdf

Changes to the multicast addresses/ports and TCP recovery / Historical replay information will be notified by the exchange if applicable. Please keep in mind that message rates in the new release environment may be substantially lower than in production.

16.3 Production Environment

The production environment multicast and TCP recovery / Historical replay channel definitions is available at the following website:

http://www.bmfbovespa.com.br/pt-br/servicos/download/MarketDataChannels_PUMA_EQT_PROD.pdf

Changes to the multicast addresses/ports and TCP recovery / Historical replay information will be notified by the exchange if applicable.

17. FIX/FAST Message Reference

The FIX/FAST message specification allows client systems developers to code for the BVMF market data feed. The specification is maintained in a separate document, available at the BVMF website, in the following URL:

http://www.bmfbovespa.com.br/puma → UMDF → UMDF FIX/FAST Message Reference

Customers must download this document to begin the development process.