BINARY ENTRYPOINT

Messaging Guidelines 8.4.2.3

Last Modified: 28/09/2026

Contacts

- Contract Services Department manages all requests for connectivity setup and general exchange-supported services.
    - contratacao@b3.com.br
    - +55 11 2565-5081
- Certification and Testing Center: performs certification of all software solutions applying for *EntryPoint* connectivity.
    - tradingcertification@b3.com.br
    - +55 11 2565-5029
- Trading Support Department (GSN): provides real-time connectivity monitoring and troubleshooting.
    - tradingsupport@b3.com.br
    - +55 11 2565-5021

1 PREFACE 12

1.1 BENEFITS 12
1.2 CLARIFICATION NOTES 12

2 MARKET SEGMENTS 13

2.1 ORDER EXECUTION RULES AT B3 13
2.2 TRADING PLATFORM SCHEDULE 14

3 SBE PROTOCOL DETAILS 14

3.1 SBE DESIGN PRINCIPLES 14
3.2 SBE SPECIFICATION 14
3.3 SBE FIELD ORDER AND SPEED OF ACCESS 14
3.4 ALIGNMENT AND PADDING 15
3.5 SBE, OPTIONAL FIELDS AND DEFAULT VALUES, EMPTY FIELDS 15
3.6 DECIMAL 'NULL' VALUE 17
3.7 SCHEMA EXTENSION MECHANISM 17
3.7.1 OBJECTIVE 17
3.7.2 MESSAGE SCHEMA FEATURES FOR EXTENSION 18
3.7.3 WIRE FORMAT FEATURES FOR EXTENSION 19
3.8 COMPATIBILITY STRATEGY TO HANDLE SCHEMA EVOLUTION 19
3.8.1 INCREMENT SCHEMA VERSION AND CHANGES ARE MADE FOR SOME TEMPLATES 19
3.8.2 SCHEMA VERSION IS INCREMENTED, NEW TEMPLATE IS CREATED FOR EXISTING MESSAGE 20
3.9 LATEST SBE TEMPLATES 21
3.10 SCHEMA VERSION SUPPORT POLICY 22

4 CONNECTIVITY 23

4.1 NETWORK LAYER 23
4.1.1 RCB 23
4.2 SESSION CONNECTION 23
4.3 AUTHENTICATION 23
4.4 PACKET COMPOSITION 23
4.5 SESSION-LEVEL MESSAGES 25

4.5.1 DAILY RESET 25
4.5.2 NEGOTIATE 25
4.5.3 ESTABLISH 27
4.5.4 SEQUENCE 28
4.5.5 NOTAPPLIED 29
4.5.6 RETRANSMIT 30
4.5.7 TERMINATE 30
4.5.8 EXAMPLE – ESTABLISH MESSAGE WITH CREDENTIALS 32
4.6 APPLICATION-LEVEL MESSAGES 32
4.6.1 MESSAGES AND DIRECTIONS 32
4.6.2 MESSAGE SEQUENCE NUMBERS 33
4.6.3 BUSINESS HEADER 34
4.6.4 RECEIVED TIMESTAMPS 38
4.6.5 EXAMPLE – SIMPLENEWORDER MESSAGE 38
4.7 CANCEL ON DISCONNECT (*COD*) 39
4.7.1 *COD* TYPE 39
4.7.2 *COD* TIMEOUT WINDOW 40
4.7.3 TRIGGERING *COD* MECHANISM ON GATEWAY-FORCED DISCONNECTIONS 40
4.7.4 *COD* LIMITATIONS 41
4.8 MASS CANCEL 41
4.8.1 MASS CANCEL ON BEHALF 42
4.8.2 MASS CANCEL FLOWS 43
4.9 THROTTLE 44
4.9.1 EXAMPLE SCENARIOS 45
4.10 BASIC VALIDATIONS FROM THE GATEWAY APPLIED FOR INBOUND MESSAGES 46

5 MESSAGE-FLOW EXAMPLES 47

5.1 NORMAL CONNECTIVITY 47
5.2 INVALID CREDENTIALS IN NEGOTIATION PROCESS 47
5.3 LOSS OF CONNECTIVITY DURING THE SESSION 48
5.4 CLIENT-SIDE FAILURE 49
5.5 GATEWAY FAILURES 50

6 CERTIFICATION 51

7 ORDER CHARACTERISTICS 51

7.1 ORDER TYPES 51
7.1.1 MARKET ORDERS WITH PROTECTION (ORDTYPE = 1) 52
7.1.2 LIMIT ORDERS (ORDTYPE = 2) 53
7.1.3 STOP ORDERS WITH PROTECTION (ORDTYPE = 3) 53
7.1.4 STOP LIMIT ORDERS (ORDTYPE = 4) 54
7.1.5 MARKET WITH LEFTOVER AS LIMIT (ORDERTYPE = K) 54
7.1.6 RLP – RETAIL LIQUIDITY PROVIDER (ORDERTYPE = W) 55
7.1.7 ORDER VALIDITY TYPES (TIME IN FORCE) 62
7.1.8 DAY (TIMEINFORCE = 0) 62
7.1.9 GOOD TILL CANCEL (GTC) (TIMEINFORCE = 1) 62
7.1.10 IMMEDIATE OR CANCEL (IOC) (TIMEINFORCE = 3) 62
7.1.11 FILL OR KILL (FOK) (TIMEINFORCE = 4) 63
7.1.12 GOOD TILL DATE (GTD) (TIMEINFORCE = 6) 63
7.1.13 AT THE CLOSE (MOC) (TIMEINFORCE = 7) 64
7.1.14 GOOD FOR AUCTION (MOA) (TIMEINFORCE = A) 64
7.1.15 ORDER QUANTITIES 64
7.1.16 DISCLOSED QUANTITIES (ICEBERG ORDERS) 64
7.1.17 MINIMUM QUANTITY 65
7.1.18 TRADE-RELATED QUANTITIES 65
7.1.19 CANCELED QUANTITY 65
7.1.20 IN-FLIGHT MODIFICATION 66
7.2 CANCELLATION BEHAVIOR 68
7.3 SIMPLE ORDER MESSAGES 69
7.3.1 SIMPLE NEW ORDER 69
7.3.2 SIMPLE MODIFY ORDER 69
7.4 ORDER CHARACTERISTICS MODIFICATION/REMOVAL/CANCEL 69
7.5 IMPACT OF THE CHANGES ON THE ORDER'S PRIORITY. 71
7.6 ORDER IDENTIFICATION 71
7.6.1 PARTICIPANT-ISSUED IDENTIFIERS 71
7.6.2 EXCHANGE-ISSUE IDENTIFIERS 74
7.6.3 ORDER IDENTIFIER RULES 76
7.7 STRATEGY IDENTIFICATION 77

8 EXECUTION REPORT 78

8.1 AGGRESSOR INDICATOR 78
8.2 REJECTION CODES 78
8.3 DAILY GTC/GTD RESTATEMENT 78
8.3.1 HOW IT WORKS 78

9 PARTICIPANT IDENTIFICATION 80

10 SECURITY IDENTIFICATION 81

11 ACCESS CATEGORIES 81

12 MEMO 82

13 CLIENT IDENTIFICATION 83

13.1 ACCOUNT NUMBER 83
13.2 ACCOUNT ANNOTATION 83
13.3 TRADING SUB-ACCOUNT NUMBER 83

14 MARKET-SEGMENT SPECIFIC RULES 84

14.1 BOVESPA SEGMENT (EQUITIES) 84
14.1.1 TRADING HOURS 84
14.1.2 ORDERS TRIGGERING INSTRUMENT FREEZE (FROZEN ORDERS) 84
14.2 BM&F SEGMENT (DERIVATIVES) 85
14.2.1 TRADING HOURS 85
14.2.2 TRADE GIVE-UPS 85

15 ADVANCED FUNCTIONALITIES 86

15.1 USER-DEFINED SPREADS (UDS) 86
15.1.1 CREATION RULES 86
15.1.2 EXPIRATION DATE 87
15.1.3 SECURITY STRATEGY TYPES 87
15.2 EXERCISE & BLOCKING 87
15.3 FORWARD DECLARATION/ACCEPTANCE (“TERMO”) 87

15.3.1 Forward Types 87
15.3.2 Forward + Cash ("Termo Vista") 88
15.3.3 Forward + Registered Cash ("Termo Vista Registered") 88
15.3.4 Security Code 88
15.3.5 Instrument States 88
15.3.6 Quote Lifecycle 89
15.3.7 Contract Details 90
15.4 Self-Trading Prevention 91
15.4.1 Self-Trading Prevention Instruction 91
15.4.2 Investor ID 91
15.5 Market Protections 94
15.5.1 Protection Types 94
15.5.2 Protection Counters 104
15.5.3 Orders cancellation on protection activation 105
15.5.4 Automatic Reset 105
15.5.5 FIXP Tags Usage 106
15.6 Implied Order 107
15.6.1 Implied In orders 108
15.6.2 Implied Out orders 109
15.6.3 Matching involving an implied order 109

16 APPLICATION MESSAGE SCENARIOS 118

16.1 Order management scenarios 118
16.1.1 Order Cancellation by *clOrdID* 118
16.1.2 Order Cancellation Attempt of Filled Order 118
16.1.3 Order Modification 119
16.1.4 Order Mass Action 120
16.1.5 Cross Order 121
16.2 Self-Trading prevention scenarios 122
16.2.1 Self-Trading prevention on aggressing order 122
16.2.2 Self-Trading prevention on Order Modification 123
16.2.3 Self-Trading prevention and Partial Fills 124
16.2.4 Self-Trading prevention on stop orders 126
16.3 Market Protections 126
16.3.1 Protected Mode 126

16.3.2 RESETTING MONITORING MODE 128
16.3.3 ORDER FILLED DURING THE PROTECTED MODE 130
16.3.4 ORDER PARTIALLY FILLED DURING PROTECTED MODE AND REMAINING QUANTITY CANCELED 132
16.3.5 ORDER FILLED AND PROTECTION VALUE EXCEEDED. 134
16.3.6 STOP ORDER TRIGGERED AFTER AUCTION NOT CANCELED AT PROTECTION MODE ACTIVATION 135
16.4 FORWARD 135
16.5 EXERCISE 135
16.6 SWEEP & CROSS 135
16.6.1 MAX SWEEP QUANTITY NOT INFORMED 136
16.6.2 MAX SWEEP QUANTITY INFORMED 139

APPENDIX A: GLOSSARY 142

APPENDIX B: SECURITY STRATEGY TYPES 144

Change Log

Date	Version	Description	Author
Feb. 28th, 2020	1.0	- Initial version.	RC, AYSF
Jan. 17th, 2022	2.0	- Details on standard and recovery message flow.	LMG, AEF
Apr. 17th, 2023	6.4.1	- Version of this document jumps to 6.4.1 to follows the specification. - Guidelines is related to the **version 6.4.1** of Binary Entrypoint specification. - Changes to the order flow (removal of *FinishedSending* and *FinishedReceiving* messages). - More detailed message flows. - Added more details about null types and padding. - Comprehensive revision of terms and scenarios. - Added description of packet composition. - Added *NotApplied* message behavior description.	LMG, RNKH
Apr. 27th, 2023	6.4.1.1	- Guidelines is related to the **version 6.4.2** of Binary Entrypoint specification.	RNKH
Jun. 26th, 2023	7.0.1	- Guidelines is related to the **version 7.0.1** of Binary Entrypoint specification. - Mass Cancel on behalf feature included.	RNKH
Sep. 6th, 2023	7.1.0	- *investorID* field is now a composite field and inserted at the root block of the order-entry messages. See section 15.4.2. - More clarification about how heartbeat's interval is applied. See section 4.5.4.1. - Including scenario of invalid credentials in negotiation process. See section 5.2.	RNKH
Sep. 18th, 2023	8.0.0	- Included a note for future use of investor identification for mass cancel on behalf. See section 15.4.2. - Adjusted two examples of *Binary Entrypoint* messages in hex dump to be compatible with specification 8.0.0. See sections 4.5.8 and 0	RNKH
Nov. 20th, 2023	8.0.0.1	- Clarification in the Session Negotiation process. See section 4.5.2. - A note that informs the need to inform the 'length' of a variable-length string, even if it is an empty string in the section 3.5. - A note that informs more than one message in the TCP packet can occur included in the section 4.4. - Updated diagrams for more clarification in message-flow examples in the section 5.	RNKH
Dec. 6th, 2023	8.0.0.2	- Included conditions that can prevent mass cancel or cancel on disconnection to cancel intended open orders in section **Erro! Fonte de referência não encontrada.** and section 4.8. - New range of prefix and a new type of non-resident investor supported for *InvestorID* composite field in section 15.4.2. - New validations applied for order entry messages from the Gateway in section 4.10. - Added conditions that trigger *Terminate* message from the gateway in section 4.5.7.	RNKH
Jan. 10th, 2024	8.0.1	- Mass cancel on behalf mechanism description added in section 4.8.	RNKH
Apr. 1st, 2024	8.1.0	- Range of prefix adjusted for non-resident investor without CVM code (code 88) for *InvestorID* composite field in section 15.4.2.	RNKH

Date	Version	Description	Author
		- New section 7.7 included to describe strategy identification. - Limitation for mass cancel on behalf for sessions that belong to the same brokerage firm in section 4.8.1. - *receivedTime* field that is added in *ExecutionReport_New*, *ExecutionReport_Modify*, *ExecutionReport_Cancel*, *ExecutionReport_Reject* messages described in the section 4.6.4.	
May 16th, 2024	8.1.1	- Description of Sweep & Cross functionality included in the section 16.6.	RNKH
May 17th, 2024	8.1.1.1	- Compatibility strategy to handle schema evolution added in section 3.8. - Latest SBE schema location described in section 0.	RNKH
May 22th, 2024	8.1.1.2	- Section 16.6 removed.	RNKH
Jun. 10th, 2024	8.1.1.3	- Section 16.6 restated.	RNKH
Jun. 20th, 2024	8.1.1.4	- Recommendation in case of silence more than *keepAliveInterval* included in section 4.5.4.1.	RNKH
Aug. 1st, 2024	8.1.1.5	- *asset*, *securityID, side* and/or *ordTagID* fields are now optional for mass cancel on behalf requests, *investorID* is the only field required for this purpose in section 4.8.1.	RNKH
Aug. 20th, 2024	8.2.1	- Implied Order functionality added in section 15.6. - *semanticVersion* field is added at the end of *NegotiateResponse* in section 4.5.2.	RNKH
Sep. 13th, 2024	8.2.3	- *possResend* field in the *OutboundBusinessHeader* and *BidirectionalBusinessHeader* types is replaced by a set of bits from eventIndicator field keeping the binary compatibility until new bits will be used. See section 4.6.3.2 and 4.6.3.3. - *marketSegmentID* field added in the *OutboundBusinessHeader* and *BidirectionalBusinessHeader* types in place of an existing padding. See section 4.6.3.2 and 4.6.3.3.	RNKH
October 15th, 2024	8.2.3.1	- Trade busts involving implied orders will also include “Implied Order” item in the orderCategory field and will have a valid value in *impliedEventID* field in *ExecutionReport_Trade* message. See section 15.6.3.	RNKH
December, 7th, 2024	8.2.3.2	- Some small adjustments in section 3.8 for better clarification for handling newer versions of the schema while maintaining compatibility with the last one. - Highlighting a behavior for trade notification of the legs related to a match that involves an implied order: the value of *ordStatus* field is always “FILLED” in section 15.6.3.	RNKH
January 20th, 2025	8.3.1	- Risk account definition is declared at section 13.3. - New *LowPriority* flag in the *eventIndicator* field is present in the header of all outbound messages. The purpose of this flag is described in section 4.6.3.2.	RNKH
January 27th, 2025	8.3.2	- “Risk account” renamed to “Trading sub-account” at section 13.3.	MEF
March 31st, 2025	8.3.3	- Removed the description of updated behavior of implied orders in case of partial execution in section 15.6.	RNKH
December 10th, 2025	8.4.2	- Some changes in the behavior of the *Cancel On Disconnect* mechanism described at section 4.7.3. - New policy related to older schema versions acceptance added in section 3.10. - Include section 8.3 describing how the daily GTC/GTD orders restatement works.	RNKH

Date	Version	Description	Author
March 20th, 2026	8.4.2.1	- Adjusted the possible values of *ordStatus* field in the *ExecutionReport_Modify* message that is published during GTC/GTD restatement process in section 8.3. - Statements that said GTC/GTD are not restated at the start of each trading session are fixed in sections 7.1.9 and 7.1.12.	RNKH
June 18th, 2026	8.4.2.2	- Include new prefix range (900-999) for Investor ID in section 15.4.2.	RNKH
Sep. 28th, 2026	8.4.2.3	- Scenario of in-flight modification that provokes a cancellation of the order is fixed with *ExecutionReport_Cancel* published at the end of the order flow in section 7.1.20.3.	RNKH

1 PREFACE

Binary EntryPoint is based on FIXP protocol (https://www.fixtrading.org/standards/fixp-online/), targeting high-performance trading systems. It is optimized for low latency of encoding/decoding while keeping bandwidth utilization small. For compatibility, it is intended to represent all FIX semantics.

1.1 Benefits

- FPGA-friendly.
- Reduced message size.
- An execution report for each kind of message.

1.2 Clarification notes

We need to highlight that some **diagrams** in this document are based on Entrypoint FIX version of guidelines. We are going to update them with binary version incrementally in the next revisions. Until then, for clarifications some points can be described as follows:

FIX Entrypoint	Binary Entrypoint correspondent
NewOrderSingle / 35=D	NewOrderSingle / SimpleNewOrder
OrderCancelReplaceRequest / 35=G	OrderCancelReplaceRequest / SimpleModifyOrder
OrderCancelRequest / 35=F	OrderCancelRequest
ExecutionReport / 35=8	Various sub-types of ExecutionReports based on the value of *execType* field (tag 150) or other criteria such as: ➢ ExecutionReport_New (150=0 or 150=D) ➢ ExecutionReport_Modify (150=5) ➢ ExecutionReport_Cancel (150=4) ➢ ExecutionReport_Trade (150=F or 150=H) ➢ ExecutionReport_Reject (150=8 or 35=9) ➢ ExecutionReport_Forward (150=F for forward contracts).
Symbol / 55=XTPO1	securityID (numeric instrument identification)

2 MARKET SEGMENTS

Binary Entrypoint supports two market segments: Derivatives (former BM&F segment) and Equities (former Bovespa segment).

The following table depicts the current product availability via the *EntryPoint* interface:

Market Segments			
	Segment	**Available via EntryPoint**	**Backend System**
Derivatives (Former BM&F Segment)			
→	Futures[1]	✓	PUMA
→	Options[2]	✓	PUMA
→	Forward	✓	PUMA
→	Spot (Gold)	✓	PUMA
Equities (Former Bovespa Segment)			
→	Stocks	✓	PUMA
→	Options on Stocks	✓	PUMA
→	Forward on Stocks	✓	PUMA
→	Exchange-defined Strategies	✓	PUMA
→	User-defined Strategies	✓	PUMA
→	Corporate-issued bonds	✓	PUMA

2.1 Order Execution Rules at B3

B3 matches orders by price/time priority. Lower ask prices take precedence over higher askes prices, and higher bid prices take precedence over lower bid prices. If there is more than one bid or ask at the same price level, earlier bids and asks take precedence over later bids and asks.

Under price/time priority of orders, a bid (ask) fills at best price by the earliest entered ask (bid) at that price. If additional contract units are needed to fill the bid (ask) then the next oldest ask (bid) at that price is matched until all the liquidity at that price has been exhausted. Then matches would commence at the next best price until the order is filled.

[1] Encompass Financial Instruments (e.g., exchange/interest rates), Commodities (e.g., Corn, Soybean), and Indices (e.g., Bovespa Index).

2.2 Trading Platform Schedule

The following table describes the trading schedules for the platform in each market segment:

Segment	Schedule
PUMA Equities	Brought down daily between 00:00 and 1:00 (local time). On weekends between Fri 22:00 and Sun 12:00.
PUMA Derivatives	Brought down daily between 00:00 and 00:00 (local time). On weekends between Fri 22:00 and Sun 12:00.

3 SBE PROTOCOL DETAILS

3.1 SBE Design Principles

As mentioned previously, SBE is optimized for low latency of encoding and decoding, while keeping bandwidth usage small. The main design principles are:

- Usage of native binary data types and simple types derived from native binaries (prices, timestamps).
- Preference for fixed positions and fixed length fields, to streamline direct access to data.
- For compatibility, all FIX semantics are supported.
- Metadata information, like tag numbers and field separators, is not sent; it is available as a message "template" (a collection of message schemas in an XML file).

SBE encoding and decoding is much simpler and faster than FAST encoding and decoding (no presence maps, no "dictionary contexts" containing previous values of tags in messages or repeating groups, no variable-length integers, no special rules for missing values etc.).

3.2 SBE Specification

B3 uses the version 1.0 of SBE, that can be accessed at:
https://www.fixtrading.org/packages/simple-binary-encoding-technical-proposal-final/.

3.3 SBE Field Order and Speed of Access

SBE templates are not expected to lay out the fields in the same order as the original FIX Message; the order and alignment of the fields are important for fast access.

SBE messages that have no repeating groups nor variable length fields behave as fixed-length, fixed-position messages, so they are very friendly to hardware processing (**FPGA**) and low-level languages (**like C**). In fact, it just behaves like a simple struct in C language.

3.4 Alignment and Padding

For speed of access, the most important fields are laid out in the limits of a 64-byte cache line and aligned according to their sizes (8-byte fields start in an offset that is a multiple of 8, 4-byte fields in an offset that's a multiple of 4, and so on).

Alignment is especially important for speeding up FPGA processing. It is guaranteed by judicious placement of the fields, and padding. In SBE the alignment and padding are done by specifying the field offsets explicitly (see more information at https://www.fixtrading.org/standards/sbe-online/#message-body), and by carefully defining the block length of the message. The block length must be greater than or equal to the sum of the sizes of all fields in the message or group: see more detail of padding at the end of a message at https://www.fixtrading.org/standards/sbe-online/#padding-at-end-of-a-message-or-group. Normally we tried to avoid 'dummy fields' for padding because it is harder to reclaim unused space later with updated specification. We highlighted field offsets and block lengths that are different than the sum of size of the fields in the Message Reference document for clarification.

3.5 SBE, Optional Fields and Default Values, Empty Fields

Optional fields in SBE can hold 'null values', that an application can interpret as 'absence of contents (field is not set)'.

There is no 'default values' in SBE. Applications can replace 'null values' with 'default values' if required but it is not in the protocol specification.

Optional fields in SBE do not save space at all. If a field is defined as an optional *int32* (4 bytes), it always occupies 4 bytes, even though the contents of the field are 'null'.

For optional fields, there is a value that represents the 'null' value (the field is not set). It can be specified in the declaration of their types.

If not explicitly defined in the type, the 'null' value is assumed as default for the primitive type: for unsigned fields, it is the largest possible value (something like 0xFFFF… in hexadecimal); for signed fields, it is the most negative value (something like 0x8000… in hexadecimal); for char fields, it is always the binary value 0 ('\0').

For enumeration fields, it depends on the 'encodingType' attribute (for instance, for the *NegotiationRejectCode* enum, whose encoding type is '*uint8'*, the 'null' value is 255; for the *TimeInForce* enum, whose encoding type is 'char', the 'null' value is NUL ('\0').

For some types, the encoding for the 'null value' is not the default, but it is specified in the 'nullValue' attribute for the type. For instance, the type of the *'enteringFirm'* field (tag 37501) is *FirmOptional*, whose 'nullValue' attribute is "0". For such field, the value "0" represents null, not zero.

There are two types of strings in SBE: variable-length strings and fixed-length strings. Null string and empty strings are encoded the same way. For variable-length strings, the field 'length' is 0 for null (empty) strings. For fixed-length strings, if the content is shorter than the specified length, must be delimited by NUL ('\0') character.

NOTE

For all optional variable-length string field such as *memo*, *deskID*, *clientIP*, *clientAppName* and *clientAppVersion*, clients need to correctly inform the length of the field (first byte of the field), even if it is zero (*0x00*) to be correctly decoded in sequence. The length member may not be null but may be set to zero for an empty string.

From the SBE specification, the default values for null value are:

Primitive Type	Value	Decimal	Hexadecimal
int8	-128	-128	0x80
uint8	255	255	0xFF
int16	-32768	-32768	0x8000
uint16	65535	65535	0xFFFF
int32	-2^{31}	-2147483648	0x80000000
uint32	$2^{32}-1$	4294967295	0xFFFFFFFF
int64	-2^{63}	-9223372036854775808	0x8000000000000000
uint64	$2^{64}-1$	18446744073709551615	0xFFFFFFFFFFFFFFFF
char	0 $(ASCII\ NULL)$	0	0x00 ('\0')
decimal (int32 mantissa)	Mantissa: -2^{31}	-2147483648	0x80000000
decimal (int64 mantissa)	Mantissa: -2^{63}	-9223372036854775808	0x8000000000000000

3.6 Decimal 'null' value

For instance, the type 'Price' is defined as a Decimal; the mantissa type is '*int64*' and the exponent is fixed as -4. A price like '+12.34' is encoded as the long value '`123400`' (it is 12.34 times 10^4, or 10000) but the null price (that can be found in market orders, that have no defined price) is encoded as the special value `0x8000000000000000`, or –`9223372036854775808`.

For some decimal fields, it is possible to have null values encoded as NUL ('\0') instead. Typically, they are values that are strictly positive (cannot assume the value 0.0). Check the SBE template searching for *nullValue* attribute in type declaration.

3.7 Schema Extension Mechanism

3.7.1 Objective

It is not always practical to update all message publishers and consumers simultaneously. Within certain constraints, messages and repeating groups can be extended in a controlled way. Consumers using an older version of a schema should be compatible if interpretation of added fields or messages is not required for business processing.

Message templates and repeating groups may be extended with new fields. However, the extension mechanism does not support extension of composite types that back existing fields.

This specification only details compatibility at the presentation layer. It does not relieve application developers of any responsibility for carefully planning a migration strategy and for handling exceptions at the application layer.

3.7.1.1 Constraints

Compatibility is only ensured under these conditions:

- Fields may be added to either the root of a message or to a repeating group, but in each case, they must be appended to end of a block.
- Inserting a new field in a reserved space not used before.
- Existing fields cannot change data type or move within a message.
- Message and repeating group byte alignment may not change.
- A repeating group may be added after existing groups at the root level or nested within another repeating group.

- A variable-length data field may be added after existing variable-length data at the root level or within a repeating group.
- Message header encoding cannot change.
- Adding a new choice for an enumeration or a set (the encoding type must remain the same).
- Adding a constant field at any position in the message.

In general, metadata changes such as name or description corrections do not break compatibility so long as wire format does not change.

Changes that break those constraints require consumers to update to the current schema used by publishers. A message template that has changed in an incompatible way must be assigned a new template “id” attribute.

3.7.2 Message schema features for extension

3.7.2.1 Schema version

The `<messageSchema>` root element contains a version number attribute. Each time a message schema is changed, the version number is incremented.

Version applies to the schema as a whole, not to individual elements. Version is sent in the message header so the consumer can determine which version of the message schema was used to encode the message.

3.7.2.2 Since version

When a new field, enumeration value, group or message is added to a message schema, the extension may be documented by adding a *sinceVersion* attribute to the element. The *sinceVersion* attribute tells in which schema version the element was added. This attribute remains the same for that element for the lifetime of the schema. This attribute is for documentation purposes only, it is not sent on the wire.

Over time, multiple extensions may be added to a message schema. New fields must be appended following earlier extensions. By documenting when each element was added, it possible to verify that extensions were appended in proper order.

3.7.2.3 Block length

The length of the root level of the message may optionally be documented on a `<message>` element in the schema using the *blockLength* attribute. If not set in the schema, block length of the message root is the sum of its field lengths. Whether it is set in the schema or not, the block length is sent on the wire to consumers.

Likewise, a repeating group has a *blockLength* attribute to tell how much space is reserved for group entries, and the value is sent on the wire. It is encoded in the schema as part of the *numInGroup* field encoding.

3.7.2.4 Deprecated elements

A message schema may document obsolete elements, such as messages, fields, and valid values of enumerations with deprecated attribute. Updated applications should not publish deprecated messages or values, but declarations may remain in the message schema during a staged migration to replace message layouts.

3.7.3 Wire format features for extension

3.7.3.1 Block size

The **length** of the root level of the message is sent on the wire in the SBE message header. Therefore, if new fields were appended in a later version of the schema, the consumer would still know how many octets to consume to find the next message element, such as repeating group or variable-length data field. Without the current **schema version**, the consumer cannot interpret the new fields, but it does not break parsing of earlier fields.

Likewise, block size of a repeating group is conveyed in the *numInGroup* encoding.

3.7.3.2 Number of repeating groups and variable data

Message headers and repeating group dimensions carry a count of the number of repeating groups and a count of variable-length data fields on the wire. This supports a walk by a decoder of all the elements of a message, even when the decoder was built with an older version of a schema. As for added fixed-length fields, new repeating groups cannot be interpreted by the decoder, but it still can process the ones it knows, and it can correctly reach the end of a message.

3.8 Compatibility strategy to handle schema evolution

It is necessary to state that B3 will make any order entry schema updates in tranches fashion following previously released plan. So, it's important to notice that during the roll-out period, not all gateways will be transmitting messages in the same schema version.

Basically, there are two strategies that B3 will adopt, based on the complexity of the changes. But to be make easier for client application be aware of changes, for every change to the message schema, **schema version will be increased**, so the decoder can handle them from reading the SBE header before starting to decode the body of the message.

3.8.1 Increment schema version and changes are made for some templates

This strategy is normally adopted for incremental changes that does not break template backward compatibility. Normally the type of changes that are backward compatible includes:

- Adding new optional fields at the end of **root block** or at the end of **existing repeating group**.
- Adding new optional fields in available unused spaces reserved previously.

- New repeating group block is added at the end of existing repeating group block but there is no variable length data after that.
- New variable length data is added at the end of the message.
- New values are added to existing enumeration types that has already been used by some existing message.
- New choices in unused bits are added to existing sets.

New fields, values and choices need to include "*sinceVersion"* attribute so modified decoders can support extensions smoothly. "Version" attribute in the root node of the "*messageSchema"* needs to be incremented as well so the SBE header of received message will have the updated schema version.

For clarification, we are adding this example: client application is adapted to process version "3" of *ExecutionReport_New* message that includes a new optional field named *receivedTime*.

1. If the received version is equal to the decoder's version, then all fields known to the decoder may be parsed, and no further analysis is required. *receivedTime* field can be parsed, and if the business logic is correctly implemented, it can know the time of receipt of related inbound message in the gateway.
2. If the received version is less than the decoder's version (that is, the producer's encoder is older than the consumer's decoder from the gateway that has not updated yet), then only the fields of the old version may be parsed. This information is available through metadata as "*sinceVersion*" attribute of an added field. If "*sinceVersion*" (in this case: 3) is greater than received schema version (in this case: 2), then the field is not available. How a decoder signals an application that a field is unavailable is **an implementation detail**. One strategy is for an application to provide a default value for unavailable fields (or *null* value) if it makes sense. For this case (*receivedTime* field), consider the new field as not provided makes more sense.
3. If the received version is greater than the decoder's version (that is, the producer's encoder is newer than the consumer's decoder), then all fields known to the decoder may be parsed but it will be unable to parse newly added fields. In most cases, the state of the client application probably will be in consistent state but without additional information provided by the updated version of the schema.

3.8.2 Schema version is incremented, new template is created for existing message

This strategy is normally adopted for disruption changes that breaks template backward compatibility. B3 will mark the old template as deprecated using "*deprecated*" attribute.

Because B3 will make any order entry schema updates in tranches of gateways, B3 strongly recommended that client applications support decoding/processing both templates: the old and the new one and process them accordingly.

To illustrate this strategy, we are giving this hypothetical example: client application is adapted to process both templates of *ExecutionReport_Trade*: the new one (**223**) and also the old one (**203**). The new template "id" = 223 of the *ExecutionReport_Trade* messages changes the type of *price* field to support 8 decimals. The older template "id" = 203 of *ExecutionReport_Trade* messages still has the *price* field supporting 4 decimals.

If the received version is the new one and is equal to the decoder's version, then all fields known to the decoder may be parsed, and no further analysis is required. The value of *price* field has 8 decimals and the client application needs be adapted to process the value accordingly.

If the received message has old template "id" (203), that is, the producer's encoder is publishing older template "id" than the consumer's decoder expected – the gateway that has not updated yet, then the value of *price* field has 4 decimals, and the client application needs to process the value as previously has been done.

If the received version is greater than the decoder's version (that is, the producer's encoder is newer than the consumer's decoder) for migrated gateway, client application will not receive any messages with the old template "id" (in this example, template "id" = 203) and will receive a message with the newer template "id" (223) that does not know how to process them and possibly will be in **inconsistent state or miss the information**.

Client systems must process SBE Header for each message.

It is extremely important that client applications that *process/send messages* in SBE format must process *schemaVersion* and *blockLength* fields in the **SBE Header** instead of assuming them from the template definition because additional compatible versions of the same message (same template "id") will likely have different schema version and greater block length to accommodate new fields. If the message has repeating groups or variable length data, these structures will be shifted to the right according to the block length of new optional fields added to the end of the root-level block.

3.9 Latest SBE Templates

SBE messages are not self-describing, and all metadata (message schemas) must be previously fetched before decoding.

Client systems must be prepared for template changes.

If schema evolution is required, B3 will release an updated version of the schema file, with new or updated templates.

Schema files are available at the B3 web site, at the following addresses:

In Portuguese: https://www.b3.com.br/pt_br/solucoes/plataformas/puma-trading-system/para-desenvolvedores-e-vendors/entrypoint/

In English: https://www.b3.com.br/en_us/solutions/platforms/puma-trading-system/for-developers-and-vendors/entrypoint/

3.10 Schema Version support policy

Removing support for older versions of the message schema is crucial for several reasons, primary centered on improving efficiency, performance, innovation and also reducing costs related to maintenance and support from the exchange side.

To ensure a smooth transition for clients and users when new schema versions are introduced, while maintaining support for legacy systems, the most recent previous version will be maintained.

During the negotiation process, the gateway will keep track of the schema version of the received messages (informed at the SBE header). The rule is: schema versions older than the most recent previous version are deprecated and it will be subject to rejection by the gateway.

As this policy is being introduced, we will inform customers when it takes effect.

Client systems must be prepared to update schema version regularly.

That said, it is important that customers update the application with newer schemas regularly to avoid staying on deprecated versions for too long and with possibility of having connections rejected by this reason.

4 CONNECTIVITY

The following sections describe all connectivity options to the binary order entry gateways.

4.1 Network Layer

Market participants can choose from the following connectivity options.

4.1.1 RCB

RCB (“Rede de Comunicação B3” or B3 Communications Network) is a newer communication option available to B3 customers.

Based on Ethernet over SONET (EoS/EoSDH), it allows participants to choose from a vast array of link speeds and service levels, which contrasts with RCCF, as the latter offers packaged, predefined solutions.

4.2 Session Connection

All FIXP session schedules for *EntryPoint* are activated daily, being terminated by the order entry gateway at the end of each session.

The session version identification (*sessionVerID* field) must be incremented during each Negotiate message sent across the week.

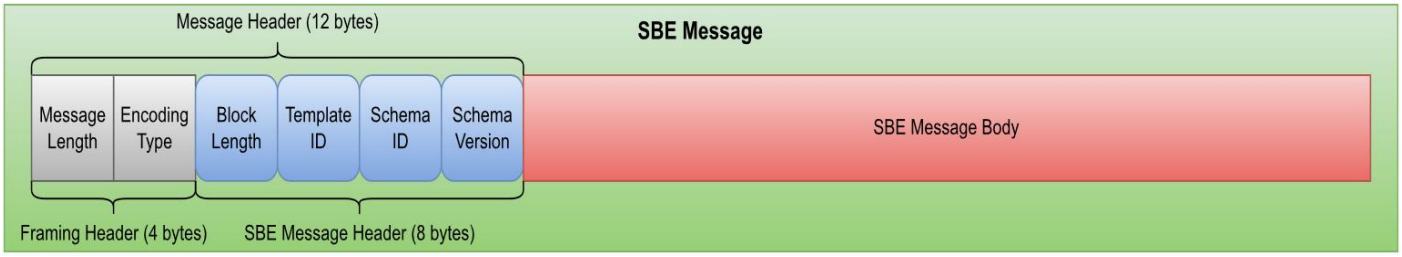
Trying to re-negotiate with new session version identification mid-session will be rejected. In this case, the rejection informsw the current session version identification (*currentSessionVerID* field). The expected next sequence number will only be informed while trying to establish a connection with unexpected sequence number. In this case, the gateway will publish *EstablishReject* message, and the expected sequence number is informed in the *lastIncomingSeqNo* field.

4.3 Authentication

Binary Entrypoint sessions require that the user has access credentials (user and password) to authenticate on order entry gateway on *Negotiate* and *Establish* messages. Those credentials (user and password) are generated by B3 and sent to the brokerage firms.

4.4 Packet composition

Each message in the packet starts with a Message Header that consists of the Framing Header and the SBE Message Header. The message header is in little-endian format (the least significant values come first). The total size is 12 bytes. After that comes the SBE message itself (the body of the message).



Framing Header is based on **Simple Open Framing Header (SOFH)**, but *MessageLength* field occupy only 2 bytes (support maximum message size of 16384 bytes): https://www.fixtrading.org/standards/fix-sofh/

The encoding type is SBE 1.0 little-endian = 0xEB50 (Because the header is in little endianness: 0x50, 0xEB in the wire).

NOTE

In some situations, a TCP packet may contain more than one SBE message, but all messages start with the framing header following with SBE header and the SBE message body.

4.5 Session-Level Messages

Binary EntryPoint uses the FIX Performance (FIXP) protocol to establish and manage bi-directional sessions.

Binary Entrypoint session messages are summarized as follows:

Stage	Msg Sent (Client to B3)	Msg Received (B3 to Client)	Purpose
Initialization	Negotiate	-	Initiates Connection
	-	NegotiateResponse	Accepts Connection
	-	NegotiateReject	Rejects Connection
Binding	Establish	-	Binds Connection
	-	EstablishAck	Accepts Binding
	-	EstablishReject	Rejects Binding
Transferring	Sequence	Sequence	Initiates a Sequenced Flow
			(Keep Alive/Heartbeat)
	Retransmit Request	-	Requests Replay
	-	Retransmission	Accepts Replay
	-	Retransmit Reject	Rejects Replay
	-	Not Applied	Missed Messages Warning in Idempotent flow
Unbinding	Terminate	Terminate	Kills the TCP connection

4.5.1 Daily Reset

B3 will internally reset its inbound and outbound sequence numbers to “1” at the start of each trading session (day). The customer should also reset their sequence numbers in both directions prior to negotiating and establishing a FIXP session at the beginning of the day.

When FIXP sessions are established at the start of each day, the customer must initiate a session by sending the Establish message (after Negotiation) with value of *nextSeqNo* field as “1”.

B3 will respond with an *Establishment Acknowledgment* confirmation with value of *nextSeqNo* field as “1”.

4.5.2 Negotiate

Negotiate message acts as the handshake to the order entry gateway. It must be the first message sent at every trading session, like the FIX protocol *Logon* message.

The *Negotiate* message contains specific identifiers of the FIXP session and security credentials. The credentials must be informed at the credentials field (tag 35512). The credentials are a JSON document containing two keys:

- **auth_type:** Describe the type of authentication to be applied. Until now, the only mechanism supported is "**basic**".
- **username:** The security username provided by B3 to authenticate the FIXP session. In case of *auth_type* = *basic*, the gateway uses the *sessionID* field as username.
- **access_key:** Token provided by B3 for the FIXP session.

Next, a valid example of the credentials key:

```
{
        "auth_type": "basic",
        "username": "some_username",
        "access_key": "some_password"
}
```

For each trading session, clients must send a *Negotiate* message, always using increased *sessionVerID* values during the week.

Once the *Negotiate* message is accepted by the order entry gateway, it will reply with an *NegotiateResponse* message.

semanticVersion field in the *NegotiateResponse* message is used to inform the current semantic version of the SBE schema supported in the session, so clients can identify if they are using the latest schema available.

If the *Negotiate* message is rejected by the order entry gateway, it replies with an *NegotiateReject* message containing the rejection reason, and after that, also sends *Terminate* message and closes the socket.

Next, see most probable reasons that may cause a rejection:

- **AlreadyNegotiated:** The Negotiate attempt was redundant in the current trading session. The gateway will send your current, active *sessionVerID* so that you can send a new *Establish* message.
- **SessionBlocked:** Session is not authorized to operate on the order entry gateway.
- **InvalidTimestamp:** Timestamp in the message is too far from the gateway expected value.
- **Credentials:** Authentication could not be done with provided credentials.
- **InvalidFirm:** The entering firm provided in the Negotiate message is not the owner of the provided *sessionID*.
- **DuplicateSessionConnection:** The session is already negotiated and in use by another TCP socket.
- **AuthenticationInProgress:** The session is currently being negotiated by another TCP socket.

4.5.3 Establish

After the Negotiate, clients are expected to send an *Establish* message so that both parties can keep track of the current sequence numbers. In lost connection scenarios, if the client system knows the current value of *sessionVerID* field, it does not need to go to a new session negotiation process, it simply needs to send a new *Establish* message.

The *Establish* message also contains important data regarding the connection's behavior:

- **keepAliveInterval:** Interval defined in milliseconds to which the client's heartbeat timing is governed by.
- **nextSeqNo:** The expected sequential number on the next application message.
- **cancelOnDisconnect:** Mechanism to cancel working orders on disconnect events. More details can be found in section 0.

Once the Establish message is accepted by the order entry gateway, it will reply with an *EstablishAck* message informing the value of *nextSeqNo*. All sessions start with a default value of 1.

If the *Establish* message is rejected by the order entry gateway, it replies with an *EstablishmentReject* message containing the rejection reason, and after that, also sends *Terminate* message and closes the socket.

Below are the possible reasons for an *EstablishmentReject* message:

- **Unnegotiated:** Establish request was not preceded by a Negotiate message, or the current *sessionVerID* was finalized, requiring a new *Negotiate* with a higher number than the previous one.
- **AlreadyEstablished:** The session has already sent an Establish message. Establish attempt was redundant.
- **SessionBlocked:** Session is not authorized to operate on the order entry gateway.
- **InvalidTimestamp:** The timestamp provided is too far from the gateway's expected value.
- **KeepaliveInterval:** The interval provided is out of the accepted range (1 to 60000).
- **Credentials:** Authentication could not be done with provided credentials.
- **DuplicateSessionConnection:** The session is already negotiated and in use by another TCP socket. Ensure the last socket's connection was effectively closed before trying a reconnection to not receive a rejection with this status.
- **AuthenticationInProgress:** The session is currently being negotiated/established by another TCP socket.
- **Unspecified:** Any other reason that prevents the establishment of the session.

4.5.3.1 Reestablishing after a critical failure

As detailed in the previous section, B3's order entry gateway will only allow one *Negotiate* Message during the current trading session.

However, if your application suffers a critical failure and loses all state, it might not know it already sent a *Negotiate* message previously.

Binary Entrypoint allows you to re-establish a valid connection and recover the previous state in these scenarios. Whenever an already negotiated FIXP session sends another *Negotiate* message, it will trigger a *NegotiateReject* with these specific tags:

Tag	Tag name	Data Type	Comment
35522	negotiationRejectCode	NegotiationRejectCode Enum (uint8)	This field will contain the reject code NegotiationRejectCode.**ALREADY_NEGOTIATED**
35523	currentSessionVerID	sessionVerID (uint64)	This field will contain the current session's *sessionVerID* of the already negotiated FIXP session.

These fields allow you to recover your current value of *sessionVerID*. With this value, you can send a new *Establish* message and resume sending orders to the gateway or recover the lost state via Retransmit messages.

4.5.4 Sequence

The Sequence message serves two purposes in the FIXP session:

4.5.4.1 Heartbeat

When no messages are exchanged between a client and the order entry gateway, *Sequence* messages need to be periodically sent as a heartbeat to prevent connection termination. A **client**'s heartbeat timing is governed by the *keepAliveInterval* value sent in the ***Establish*** message, and the **gateway**'s heartbeat timing is governed by the *keepAliveInterval* value sent in ***EstablishAck*** message.

Each party should check whether it has received any message from its peer in the expected interval. Silence is taken as evidence that the transport is no longer valid, and the session should be terminated in that event.

Recommendation in case of silence more than *keepAliveInterval*

It is recommended that client systems wait up to 3 (three) times the defined *keepAliveInterval* value in *EstablishAck* message before sending the *Terminate* message and close the connection.

4.5.4.2 Sequence reset

For recoverable and idempotent flows, *Sequence* messages could be exchanged after the connection is established to synchronize the next expected sequence number. It also be used between real-time stream and recovery stream, setting the value of *nextSeqNo* field accordingly.

If the client sends a *Sequence* message with the purpose to move the next sequence number higher than what the gateway expects, the gateway will update the next sequence number it expects to receive from and will send *NotApplied* message to notify the client the gap and the last sequence number not applied (that was the expected sequence number before the update).

4.5.4.3 After the end of retransmission

When the gateway responds to a retransmission, it sends a *Retransmit* message, after that, the requested messages. At the end of the successful retransmission, a *Sequence* message is sent to "return the context" back to the “live feed”, informing current “live feed” sequence number in the *nextSeqNo* field.

4.5.5 NotApplied

Since the message flow from client to exchange is **idempotent**, when the exchange recognizes a sequence number gap, it will send a *NotApplied* message, but **accept** messages with a higher-than-expected sequence number, updating the expected sequence number for the next incoming messages.

The sender on an idempotent flow (client system) uses the *NotApplied* message to discover which its requests have not been acted upon. It has a responsibility to decide about recovery at an application layer. It may decide to resend the transactions with new sequence numbers, to send different transactions, or to do nothing.

In *Binary Entrypoint*, the *NotApplied* message is handled as a session-level message, thus it does **not** consume a sequence number.

Tag	Tag Name	Presence	Data Type (Primitive Type)	Offset (Size)	Comments
35	messageType	C	MessageType Enum (uint8)		Message type = NotApplied. Constant: 9
35529	fromSeqNo	R	SeqNum (uint32)	0 (4)	The first applied sequence number.
35530	count	R	MessageCounter (uint32)	4 (4)	How many messages haven´t been applied?

4.5.6 Retransmit

Retransmit messages allow clients to replay the flow of messages in the active trading session. Unlike in standard FIX session specification, the *Binary Entrypoint* protocol allows clients to rejoin the trading session without recovering previously processed messages. *Binary Entrypoint* only allows the retransmission of messages from the current active trading session.

Advised behavior is to avoid doing retransmissions while sending new orders to the gateway, as these messages can be replayed later in the active session. Optimally, a retransmission batch should not exceed the size of a datagram, even on a TCP stream.

Only one outstanding retransmission request with a maximum count of **1000** messages is allowed at a time. Violation of this limit culminates in a termination process.

Once a *RetransmitRequest* message is accepted by the order entry gateway, a *Retransmission* message will be sent followed by the replayed messages. A *Sequence* message will be sent to notify the switch back to the live feed.

4.5.7 Terminate

Terminate message is used to signal to the counterparty that the TCP connection will be closed. This unbinds the transport from the session, but it does not end a logical session. The Terminate message initiates and confirms the end of the TCP socket connection.

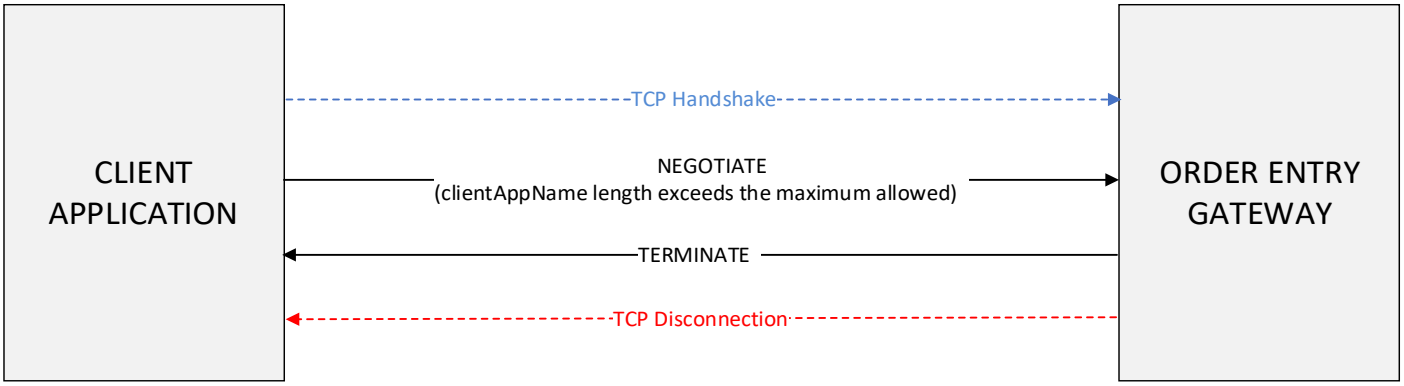
After the termination process is complete, another *Establish* with the same *sessionVerID* must be done if you intend to send more messages on the active trading session.

Terminate message sent by the gateway is also used to notify an unrecoverable error generated by an incoming message. See the table below some common errors in this category:

Offending Condition	Inbound Message	Rejection Message	Error Code
SOFH *messageLength* exceeds 512 bytes.	All	*Terminate*	*terminationCode:* 16 - INVALID_SOFH.
SOFH (4 bytes) + SBE header (8 bytes) + *blockLength* in the SBE header + variable length field header (1-byte) + length of variable field exceeds value of SOFH *messageLength*.	*Negotiate/Establish*	*Terminate*	*terminationCode:* 17 - DECODING_ERROR.

Offending Condition	Inbound Message	Rejection Message	Error Code
blockLength in the SBE header < minimum allowed length for the message.	*Negotiate/Establish*	*Terminate*	*terminationCode:* 17 - DECODING_ERROR.
Messages without the variable field header (*length*).	*Negotiate/Establish*	*Terminate*	*terminationCode:* 17 - DECODING_ERROR.
Length value for variable field exceeds maximum length for the type declared in the schema for fields: *credentials*, *clientIP*, *clientAppName*, *clientAppVersion*.	*Negotiate/Establish*	*Terminate*	*terminationCode:* 17 - DECODING_ERROR.

Example of the message flow where the client does not send *clientAppName* field exceeds 30 characters:



4.5.8 Example – Establish message with credentials

Here is an example of a HEX Dump of a complete *Establish* message with credentials in one TCP packet for better understanding on how to work with fixed root block and one variable-length data (access key changed for security reasons):

```
          +-------------------------------------------------+
          |  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f |
+--------+-------------------------------------------------+----------------+
|00000000| 8c 00 50 eb 2a 00 04 00 01 00 06 00 01 e1 f5 05 |..P.*...........|
|00000010| 66 70 f3 1c 89 01 00 00 40 ce 48 9a 01 6e 6e 17 |fp......@.H..nn.|
|00000020| 60 ea 00 00 00 00 00 00 01 00 00 00 03 00 f4 01 |`...............|
|00000030| 00 00 00 00 00 00 55 7b 20 20 20 22 61 75 74 68 |......U{   "auth|
|00000040| 5f 74 79 70 65 22 3a 20 22 62 61 73 69 63 22 2c |_type": "basic",|
|00000050| 20 20 20 22 75 73 65 72 6e 61 6d 65 22 3a 20 22 |   "username": "|
|00000060| 31 30 30 30 30 30 30 30 31 22 2c 20 20 20 22 61 |100000001",   "a|
|00000070| 63 63 65 73 73 5f 6b 65 79 22 3a 20 22 31 32 33 |ccess_key": "123|
|00000080| 34 35 36 37 38 39 41 42 43 22 20 7d             |456789ABC" }....|
+--------+-------------------------------------------------+----------------+
```

Offset	Length	Field	Hex bytes	Decoded value
0000	2	messageLength	8c 00	0x008c = 140
0002	2	encodingType	50 eb	0xeb50 = SBE 1.0 Little-Endian
0004	2	blockLength	2a 00	0x002a = 42
0006	2	templateID	04 00	0x0004 = 4 (Establish)
0008	2	schemaID	01 00	0x0001 = 1 (Schema ID)
000A	2	schemaVersion	06 00	0x0006 = 6 (Schema Version)
000C … 0035	42	SBE Message Root Block	01 e1 f5 05 …	
0036	1	CredentialsEncoding.length	55	0x55 = 85 (credentials length)
0037 … 008B	85	CredentialsEncoding.varData	7b 20 20 20 …	credentials

4.6 Application-level Messages

4.6.1 Messages and directions

More details, see *Binary Entrypoint* Message Reference document.

Message Type	TemplateID	Application Message	From B3 to Client	From Client to B3
14	206	BusinessMessageReject	✓	
15	100	SimpleNewOrder		✓
16	101	SimpleModifyOrder		✓
17	102	NewOrderSingle		✓
18	104	OrderCancelReplaceRequest		✓
19	105	OrderCancelRequest		✓
20	106	NewOrderCross		✓
21	200	ExecutionReport - New Order/Restated	✓	

Message Type	TemplateID	Application Message	From B3 to Client	From Client to B3
22	201	ExecutionReport - Modify	✓	
23	202	ExecutionReport - Cancel	✓	
25	204	ExecutionReport - Reject	✓	
24	203	ExecutionReport - Trade	✓	
26	205	ExecutionReport - Forward	✓	
27	300	SecurityDefinitionRequest		✓
28	301	SecurityDefinition	✓	
31	401	QuoteRequest	✓	✓
32	402	QuoteStatusReport	✓	
33	403	Quote	✓	✓
34	404	QuoteCancel	✓	✓
35	405	QuoteRequestReject	✓	✓
36	501	PositionMaintenanceCancelRequest		✓
37	502	PositionMaintenanceRequest		✓
38	503	PositionMaintenanceReport	✓	
39	601	AllocationInstruction		✓
40	602	AllocationReport	✓	
29	701	OrderMassActionRequest		✓
30	702	OrderMassActionReport	✓	

4.6.2 Message Sequence numbers

The FIXP protocol recommends maintaining separate sequence numbers for incoming and outgoing messages between the customer and the exchange. This ensures that all messages to and from the exchange are in the correct order however in *Binary Entrypoint* defines that only business messages are explicitly sequenced, session-level messages don´t affect the implicit sequence in the related flow.

To guarantee message delivery, both the customer and exchange must maintain inbound and outbound sequence numbers. The customer's responsibility for maintaining inbound and outbound sequence numbers includes:

- Update inbound sequence number from the value of *msgSeqNum* in the header for each business message received from the exchange for a particular value of *sessionVerID* field.
- Increment outbound sequence number by one for each message sent to the exchange for a particular value of *sessionVerID* field.

- Issue a *RetransmitRequest* message when a sequence gap is detected from the exchange for a particular value of *sessionVerID* field.
- If the exchange detects a sequence gap from customer, then a *NotApplied* message will be sent, and the customer can take action as needed.
- This can include sending a *Sequence* message instructing the exchange to ignore the gap and proceed ahead with the value of *nextSeqNo* field (recommended behavior).
- Sequence numbers are maintained for a particular value of *sessionVerID* field among the primary and backup processes during fail-over scenario.
- The *Sequence*, *Establish*, *EstablishmentAck* and *Retransmission* messages are sequence forming since they contain the *nextSeqNo* field which indicates the sequence number of subsequent business messages.
- All business messages will explicitly be assigned a sequence number that is located at the business header composite type (more details in the next section).
- Since there is no heartbeat message, the Sequence message itself serves as the de-facto heartbeat and at a minimum it must be sent after the lapse of each *keepAliveInterval* if no other messages are being sent.

4.6.3 Business Header

There are 3 types of business headers:

4.6.3.1 InboundBusinessHeader

This header is related to all messages coming exclusively from the client to B3. Order entry gateways uses the *marketSegmentID* field from this header to redirect it to the correct matching engine.

Field	Type	Required	Comments
sessionID	SessionID	Y	Client connection identification on the gateway assigned by B3. It is present to help with measuring latencies on the wire from B3 monitoring team.
msgSeqNum	SeqNum	Y	Inbound flow sequence number of a given SessionID/SessionVerID.
sendingTime	SendingTime	Y	Time of message transmission, expressed in UTC (elapsed nanoseconds from epoch).
marketSegmentID	MarketSegmentID	Y	Identifies the market segment whose message goes to.

4.6.3.2 OutboundBusinessHeader

This header is related to all messages exclusively from B3 to client. If *PossResend* flag in *eventIndicator* field is true, probably an internal component has a takeover process in progress, some messages can be published in duplication with different outbound sequence number. Client systems can relate those messages using the unique business identification field.

The *LowPriority* flag in the *eventIndicator* field indicates a low priority in risk evaluation. When set to true (bit 1), it means the order has lost its priority in the FIFO queue to reach the matching engine. This can occur due to specific risk evaluation scenarios, such as when risk parameters are loaded on demand (when the related account does not trade for a defined number of consecutive days), protected or synchronous mode is activated for the account or document, or risk evaluation of a cross-order. For further details please check "LiNe - Concept Guide" documentation in https://www.b3.com.br/en_us/solutions/platforms/risk-management/line-5/specification/.

Field	Type	Required	Comments
sessionID	SessionID	Y	Client connection identification on the gateway assigned by B3. It is present to help with measuring latencies on the wire from B3 monitoring team.
msgSeqNum	SeqNum	Y	Outbound flow sequence number of a given SessionID/SessionVerID.
sendingTime	SendingTime	Y	Time of message transmission, expressed in UTC (elapsed nanoseconds from epoch).
eventIndicator	EventIndicator	Y	Set of indicators for a given event. Currently there are two use cases: indicates possible retransmission of message during recovery process and low priority flag. Since version 4.
marketSegmentID	MarketSegmentIDOptional	N	Optional identifier of the market segment. Since version 4.

The following table describes the business identification field that can be used to identify duplications with different sequence numbers when the value of *possResend* flag in *eventIndicator* field is **true**:

Outbound messages type	Business identification field	Tag number
ExecutionReport - New Order/Restated	execID	17
ExecutionReport – Modify	execID	17
ExecutionReport – Cancel	execID	17

Outbound messages type	Business identification field	Tag number
ExecutionReport – Reject	execID	17
ExecutionReport – Trade	execID	17
ExecutionReport - Forward	execID	17
SecurityDefinition	securityReqID	320
QuoteRequest	quoteID	117
QuoteStatusReport	quoteID	117
Quote	quoteID	117
QuoteCancel	quoteID	117
QuoteRequestReject	quoteID	117
PositionMaintenanceReport	posMaintRptID	721
AllocationReport	allocID	70
OrderMassActionReport	massActionReportID	1369

4.6.3.3 BidirectionalBusinessHeader

This header is related to all messages that can be send to or comes from B3 to client. Those messages are:

- QuoteRequest
- QuoteStatusReport
- Quote
- QuoteCancel
- QuoteRequestReject
- QuoteRequestReject

Field	Type	Required	Comments
sessionID	SessionID	Y	Client connection identification on the gateway assigned by B3. It is present to help with measuring latencies on the wire from B3 monitoring team.
msgSeqNum	SeqNum	Y	Inbound or outbound flow sequence number of a given SessionID/SessionVerID.
sendingTime	SendingTime	Y	Time of message transmission, expressed in UTC (elapsed nanoseconds from epoch).

Field	Type	Required	Comments
eventIndicator	EventIndicator	Y	Set of indicators for a given event. Currently there are two use cases: indicates possible retransmission of message during recovery process and low priority flag. Since version 4.
marketSegmentID	MarketSegmentIDOptional	N	Optional identifier of the market segment.

4.6.4 Received Timestamps

From **version 3** of the schema, B3 included an optional *receivedTime* (tag 35544) field in the *ExecutionReport_New*, *ExecutionReport_Modify*, *ExecutionReport_Cancel* messages. Received time means the instant when inbound message arrived in the B3's binary order entry gateway. It is optional because some scenarios that the event is not related to any incoming messages such as restating the order from iceberg orders, or from stop orders, cancelation from pre trade risk evaluation, from market ops or from *IoC* orders that do not provoke a matching, the value is not returned.

4.6.5 Example – SimpleNewOrder message

Here is an example of a HEX Dump of a complete *SimpleNewOrder* message in one TCP packet for better understanding on how to work with fixed root block (with *investorID* field filled) and one variable-length data filled (memo field):

```
          +-------------------------------------------------+
          |  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f |
+--------+-------------------------------------------------+----------------+
|00000000| 75 00 50 eb 54 00 64 00 01 00 06 00 01 e1 f5 05 |..P.U.d.........|
|00000010| 05 00 00 00 80 11 49 0a 04 6e 6e 17 50 00 01 00 |......I..nn.P...|
|00000020| 6b 70 f3 1c 89 01 00 00 0f 00 00 00 54 41 44 41 |kp..........TADA|
|00000030| 00 00 00 00 00 00 54 41 44 41 00 00 55 4f f0 90 |......TADA..UO..|
|00000040| 2e 00 00 00 31 32 30 00 64 00 00 00 00 00 00 00 |....120.d.......|
|00000050| 08 43 0f 00 00 00 00 00 2C 01 00 00 40 E2 01 00 |.C......,...@...|
|00000060| 14 53 49 4d 50 4c 45 4e 45 57 4f 52 44 45 52 20 |.SIMPLENEWORDER |
|00000070| 42 55 59 20 35                                  |BUY 5...........|
+--------+-------------------------------------------------+----------------+
```

Offset	Length	Field	Hex bytes	Decoded value
0000	2	messageLength	76 00	0x0075 = 117
0002	2	encodingType	50 eb	0xeb50 = SBE 1.0 Little-Endian
0004	2	blockLength	55 00	0x0054 = 84
0006	2	templateID	64 00	0x0064 = 100 (SimpleNewOrder)
0008	2	schemaID	01 00	0x0001 = 1 (Schema ID)
000A	2	schemaVersion	06 00	0x0006 = 6 (Schema Version)
000C … 0060	84	SBE Message Root Block	01 e1 f5 05 …	
0062	1	MemoEncoding.length	14	0x14 = 20 (memo length)
0063 ... 0076	20	MemoEncoding.varData	53 49 4d 50 ...	memo

4.7 Cancel on Disconnect (*CoD*)

Cancel on Disconnect (*CoD*) is a feature that provides the ability to automatically cancel all open orders after a disconnection event. *CoD* is configured per session.

When the situation expected for the *CoD* handler happens, the order entry gateway will attempt to cancel all non-GT orders, (i.e., all orders with validity different than “Good till Date” and “Good till Cancel”).

Orders associated with the FIXP Session entered on behalf will also be canceled if *CoD* is enabled for that FIXP session.

After reestablishing connection, FIXP sessions will receive *ExecutionReport_Cancel* messages from the orders canceled by *CoD* mechanism.

To enable *CoD*, two optional fields must be provided at the Establish message: *CancelOnDisconnectType* field (tag 35002) and *CODTimeoutWindow* field (tag 35003).

4.7.1 *CoD* Type

To detail *CoD* behavior, clients must include the *CancelOnDisconnectType* field (tag 35002) on every *Establish* message, indicating the criteria used to initiate it. *CoD* is not enabled by default.

Tag	Tag name	Required	Data Type	Comment
35002	*CancelOnDisconnectType*	N	*CancelOnDisconnectType* Enum (uint8)	Criteria used to initiate *CoD* by the order entry gateway. If this Tag is not present, then *CoD* will not be enabled. Valid values: 0 - Do Not Cancel on Disconnect or Logout 1 - Cancel on Disconnect Only 2 - Cancel on Logout Only 3 - Cancel on Disconnect or Logout

4.7.1.1 Do not Cancel on Disconnect or Logout (default)

When *CancelOnDisconnectType* field is not present in the Establish message or has a value of 0, *CoD* functionality will not be enabled, and no orders will be automatically canceled after disconnection.

4.7.1.2 Cancel on Disconnect Only

CancelOnDisconnectType = 1 triggers *CoD* in an abrupt disconnection event (i.e., no orders will be canceled after a graceful shutdown).

4.7.1.3 Cancel on Logout Only

CancelOnDisconnectType = 2 triggers *CoD* after a graceful shutdown (i.e., no orders will be canceled after an abrupt disconnection event).

4.7.1.4 Cancel on Disconnect or Logout

CancelOnDisconnectType = 3 triggers *CoD* after any type of disconnection, graceful or abrupt.

4.7.2 *CoD* Timeout Window

Cancel on Disconnect allows a timeout window configuration, setting a time window the order entry gateway must wait before trying to cancel active orders for that specific FIXP session.

The window must be defined in milliseconds, defined by the Establish message (*CODTimeoutWindow* field – tag 35003). A value of 0 (default) causes immediate activation upon the detection of connection loss.

Tag	Tag name	Data Type	Comment
35003	CODTimeoutWindow	DeltaInMillis	Order entry gateway will not trigger *CoD* if the customer reconnects within the specific timeout window (In milliseconds) which starts when the triggering event is detected. The default value is 0, and the max allowed value is 60000.

4.7.3 Triggering *CoD* mechanism on gateway-forced disconnections

The gateway will trigger Cancel on Disconnect (*CoD*) for a session whenever it forcibly disconnects a client, including (but not limited to) the following cases:

- Invalid or out-of-order business sequence numbers.
- Inactivity: no messages received within the configured timeout window.
- Administrative actions by market surveillance due to inappropriate behavior.
- SOFH decoding errors (e.g., invalid encoding type or body length).
- Unsupported schema version or unsupported template IDs.

In all such cases, *CoD* is triggered if *CoD* is enabled in the *Establish* message for the session. **It is not triggered** if *CancelOnDisconnectType* = 0 (*Do not Cancel on Disconnect or Logout*). This behavior protects B3 and the participants from conditions outside their control by proactively removing potentially stale or unmanaged working orders.

Note: Any new gateway-enforced disconnection types introduced in future releases will adhere to the same *CoD* behavior unless explicitly documented otherwise.

4.7.4 *CoD* Limitations

CoD is intended to mitigate potential losses caused by unexpected disconnections. There is no guarantee that all open orders will be successfully canceled since certain conditions such as:

- Orders are in-flight and arrive in the matching engine after the mass cancel. Mass cancel requests originating from the *"Cancel on Disconnect"* mechanism follow a different internal path compared to order messages that first go to the pre-trade risk validation engine before going to the matching engine of the related market segment.
- The state or phase of the related instrument does not allow cancelation.
- Orders have already been traded.

For example, markets or instruments in states such as Pre-Close or Close do does not allow cancellations. Likewise, *CoD* cannot cancel an order that is participating in an auction or that has been successfully filled during the timeout window.

In a disconnection scenario, it is advised that firms contact market operations if there is uncertainty about the status of given orders.

4.8 Mass Cancel

With Mass Cancel, clients can quickly and effectively cancel multiple orders in a single way, rather than manually canceling each order individually. This feature is particularly useful in situations where there is a need to respond swiftly to market conditions, manage risks, or implement specific trading strategies across many orders.

By utilizing the Mass Cancel functionality, traders and market participants can handle and manage their orders efficiently, enabling faster decision-making and execution. This tool is essential in trading systems to handle high volumes of orders and facilitate automated trading strategies.

There is no guarantee that all open orders that satisfy the given filter criteria will be successfully canceled since certain conditions such as:

- Orders are in-flight and arrive in the matching engine after the mass cancel. Client's mass cancel requests follow a different internal path compared to order messages that first go to the pre-trade risk validation engine before going to the matching engine of the related market segment.
- The state or phase of the related instrument does not allow cancellation. For example, GTC, GTD and MOA orders participating in Theoretical Opening Price (TOP).
- Orders have already been traded.

It provides filters that can be combined to allow a more specific, targeted behavior of the mass cancel action:

- ***Side (Buy or Sell):*** specifies the side or direction of the orders to be canceled. It indicates whether the orders to be canceled are buy orders (Side = 1) or sell orders (Side = 2).
- ***SecurityID:*** specifies the security identification number for which the cancellation is requested.
- ***OrdTagID:*** specifies the order tag identification of the orders to be canceled.
- ***Asset:*** is used to identify the specific asset for which you want to cancel orders.
- ***InvestorID:*** specifies the unique identifier of investor used for self-trade prevention mechanism.

The mass cancel contains *massActionType* field to indicate the action type. Following are the possible options:

- **2 – Release orders from suspension:** It unblocks the investor or the order entry session. If the *investorID* field is sent in the request, the unblocking is applied only to the specified investor code. If the *investorID* field is not sent, the order entry session will be unblocked.
- **3 – Cancel orders:** only cancel the orders.
- **4 - Cancel and Block:** It cancels active orders and prevents the submission of new orders. If the *investorID* field is sent in the cancellation request, the suspension is applied only to the specified investor code. However, if the *investorID* field is not sent, the order entry session will be blocked. All subsequent incoming messages will be rejected until a new *OrderMassActionRequest* message with *massActionType* is set to release orders from suspension.

One behavior that is important to clarity is that the value of clOrdID field from execution reports that informs cancelled orders will not have the same value of clOrdIDs of the mass order action request that triggered those cancellations. The clients will know those orders were cancelled by Mass Order Cancel triggered by an *OrderMassActionRequest* message because those Execution Report messages will have the value of *execRestatementReason* field equals to 202 (Order Mass Action from Client Request) and they can correlate them accordingly with the value of *massActionReportID* field.

4.8.1 Mass Cancel on Behalf

Mass Cancel on Behalf refers to the ability for a session to initiate a mass cancellation of orders on behalf of multiple sessions **that belong to the same brokerage firm** from the same client.

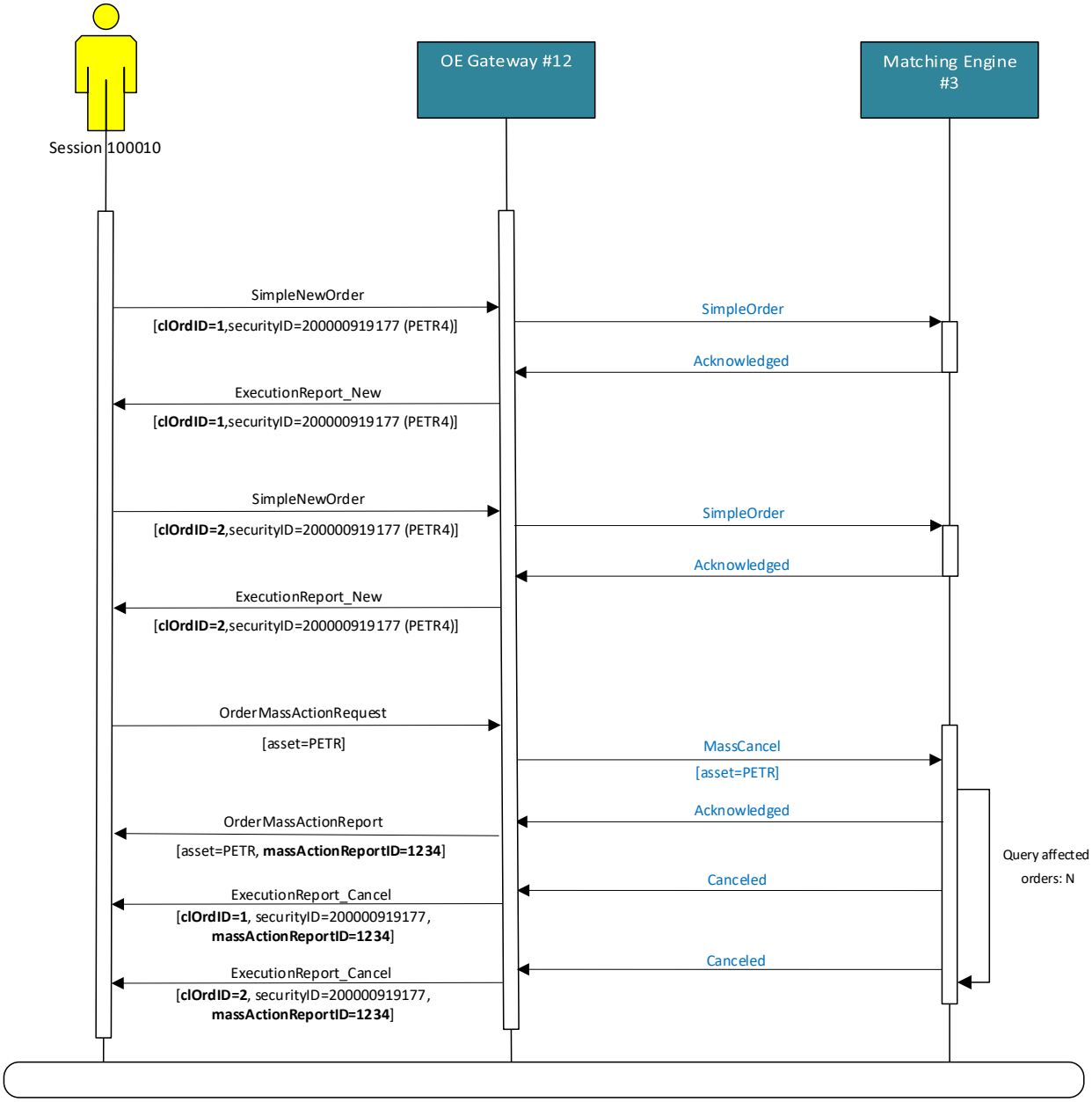
The client will be identified by the *investorID* field, the same identifier used for self-trade prevention mechanism. The request message may also be composed with one or more of the following fields as a filter: *asset*, *securityID, side* and/or *ordTagID*.

Only the original session will receive the *ExecutionReport_Cancel* messages that will include *actionRequestedFromSessionID* field that informs the mass cancel requester session identification. The Mass Cancel acknowledgement (*OrderMassActionReport* message) message will be sent to the requester session.

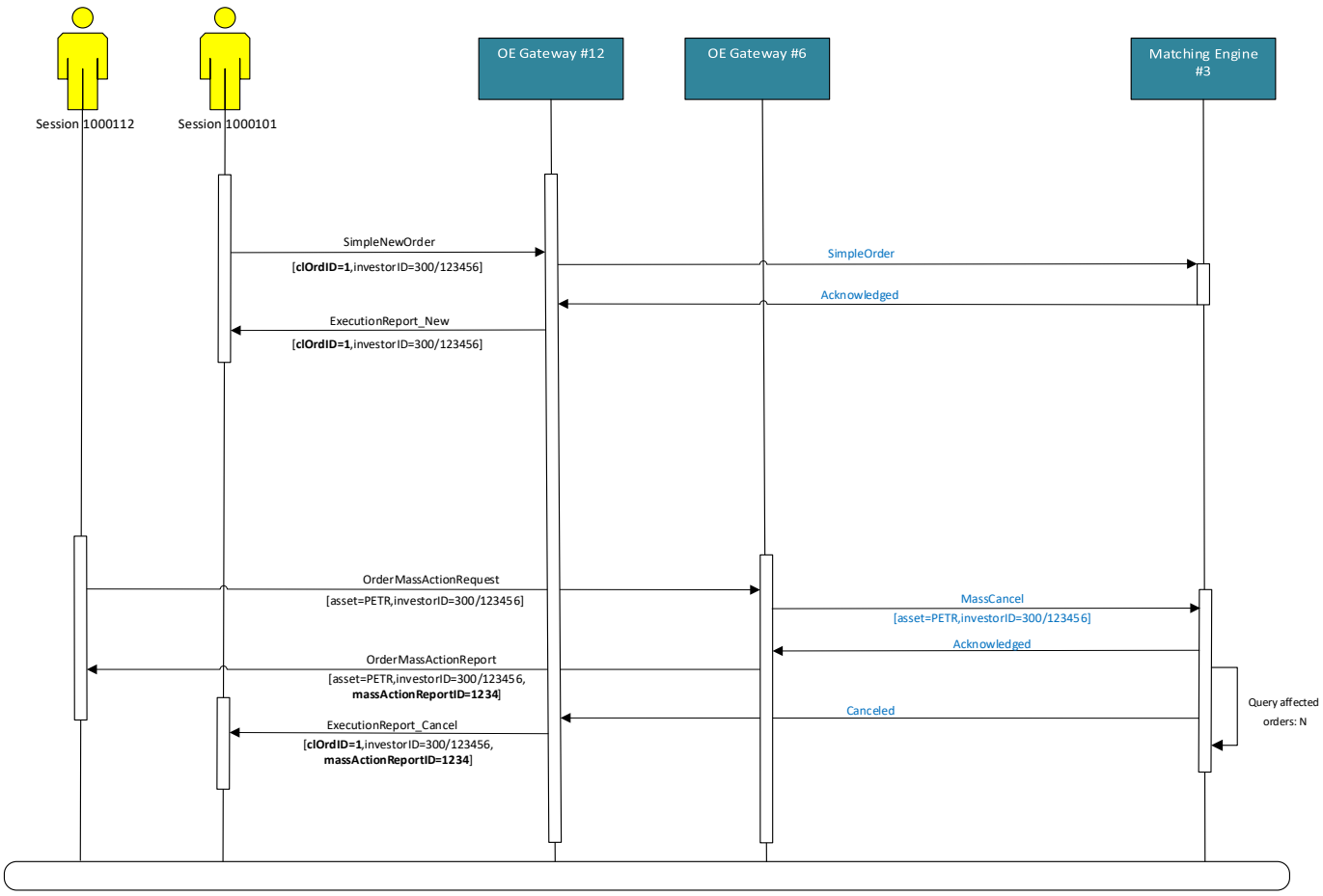
By utilizing Mass Cancel on Behalf, the designated session can swiftly and effectively cancel orders across multiple other sessions that **need to belong to the same brokerage firm (identification associated to the session that sent the possible affected orders)**, ensuring consistency and adherence to predetermined rules or guidelines. This capability streamlines the cancellation process, minimizes manual intervention, and allows for prompt risk mitigation or adjustment of trading strategies.

4.8.2 Mass Cancel Flows

4.8.2.1 Mass cancel in the same session



4.8.2.2 Mass cancel on behalf from different session /gateway



4.9 Throttle

Throttling is a mechanism that allows the order entry gateway to control the rate of inbound messages from FIXP sessions. This allows the gateway to keep the inbound rate controlled, optimizing performance.

Throttle configuration is applied per FIXP session, and consists of two parameters:

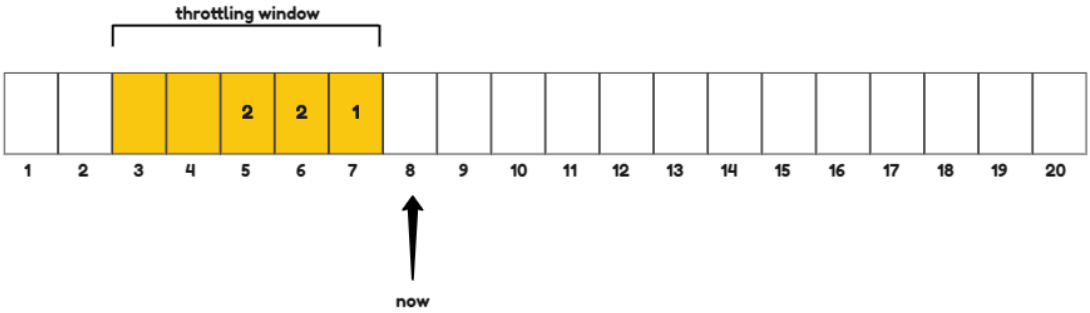
- **Time window** – The millisecond-sliding window in which the number of messages is computed. When a new message is sent, the throttling mechanism will sum the number of messages sent in the last N milliseconds to determine if the message will be accepted or rejected.
- **Number of messages** – The max number of messages that can be sent within the current millisecond time window.

Messages that violate the throttle settings will always be rejected. The *BusinessRejectReason* code will be provided as “Throttle limit exceeded”.

4.9.1 Example scenarios

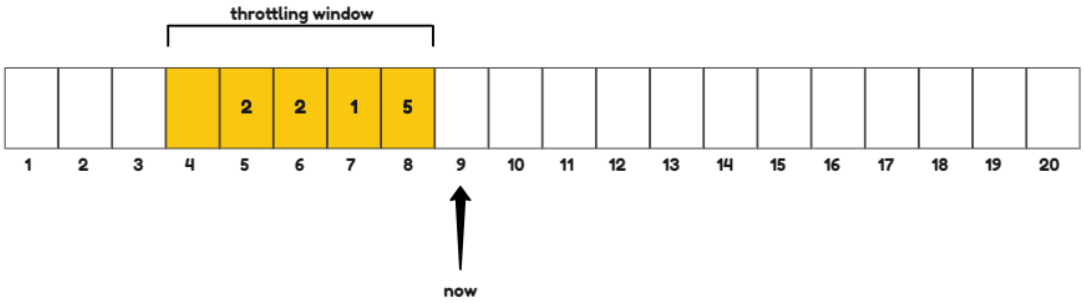
For the example scenario, consider that a FIXP session has a throttle configuration of 10 messages per 5 milliseconds. The images below depict a 20-millisecond window.

In the first scenario, when a message is sent at the 8th millisecond, the sliding window evaluates the number of messages in the last 5 milliseconds like this:



The message will be accepted since the session has sent 5 messages in the last 5 milliseconds.

In the second scenario, the messages sent at the 9th millisecond will be rejected, since the number of messages sent on the current window (now – 5 milliseconds) is already 10.



This specific session will be able to send messages only at the 10th millisecond, where the first 2 messages will be out of the active window.

4.10 Basic validations from the gateway applied for inbound messages

Regarding basic validations on binary order entry gateways, in case of violations mentioned below, the gateway will reject the offending message by sending a rejection message depending on the context indicated in the table below:

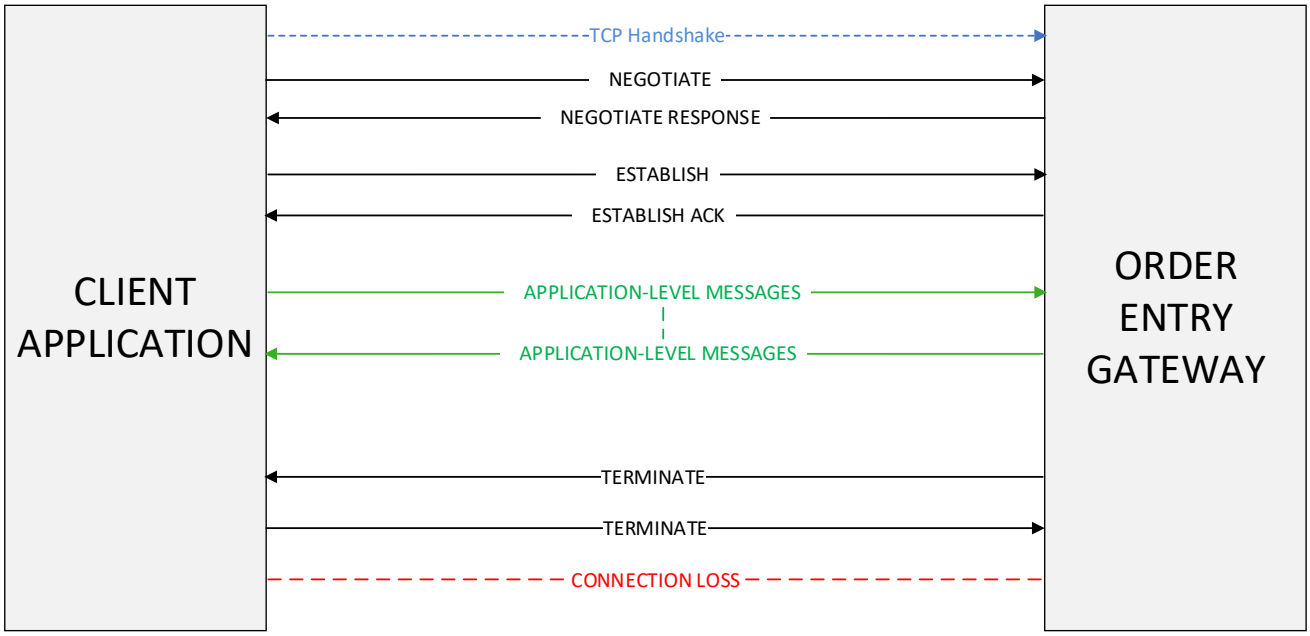
Offending Condition	Inbound Message	Rejection Message	Error Code
SOFH *messageLength* exceeds 512 bytes.	All	*Terminate*	*terminationCode:* 16 - INVALID_SOFH.
SOFH (4 bytes) + SBE header (8 bytes) + *blockLength* in the SBE header + variable length field header (1-byte) + length of variable field exceeds value of SOFH *messageLength*.	*Negotiate/Establish*	*Terminate*	*terminationCode:* 17 - DECODING_ERROR.
blockLength in the SBE header < minimum allowed length for the message.	*Negotiate/Establish*	*Terminate*	*terminationCode:* 17 - DECODING_ERROR.
Messages without the variable field header (*length*).	*Negotiate/Establish*	*Terminate*	*terminationCode:* 17 - DECODING_ERROR.
Length value for variable field exceeds maximum length for the type declared in the schema for fields: *credentials*, *clientIP*, *clientAppName*, *clientAppVersion*.	*Negotiate/Establish*	*Terminate*	*terminationCode:* 17 - DECODING_ERROR.
<CR> ou <LF> character present in the following fields: *deskID*, *senderLocation*, *enteringTrader* or *executingTrader*.	Business-level inbound messages	*Business Message Reject*	*businessRejectReason:* 33003 *text :*"Line breaks not supported in *<field>*".
Incorrect value of *InboundBusinessHeader.sessionID*	Business-level inbound messages	*Business Message Reject*	*businessRejectReason:* 33003 *text :*"Wrong sessionID in businessHeader".
Length value for variable field exceeds maximum length for the type declared in the schema for fields: *memo* and *deskID*.	Business-level inbound messages	*Business Message Reject*	*businessRejectReason:* 33003 *text:* "*<field>* too long".

5 MESSAGE-FLOW EXAMPLES

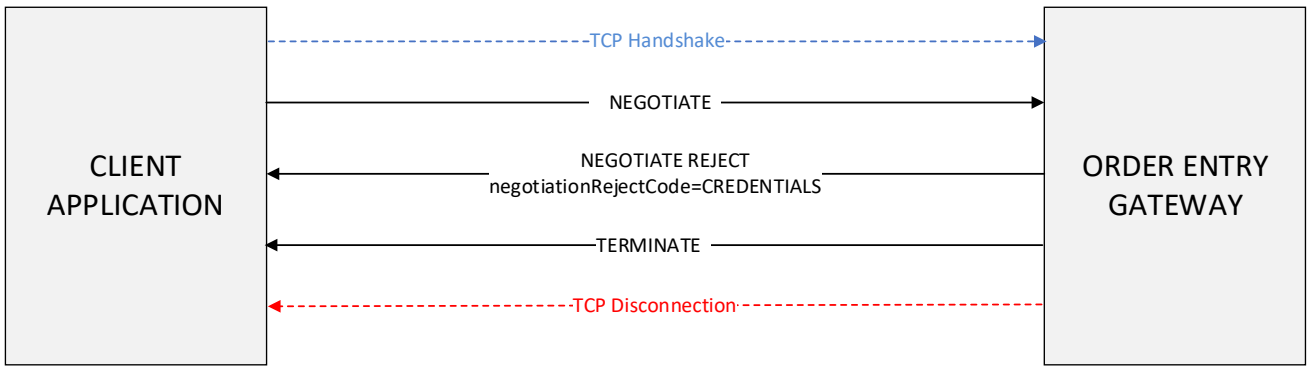
This section shows some scenarios and the expected behavior from the client side.

5.1 Normal connectivity

This is the normal, everyday scenario where everything happens as expected according to the protocol specification.

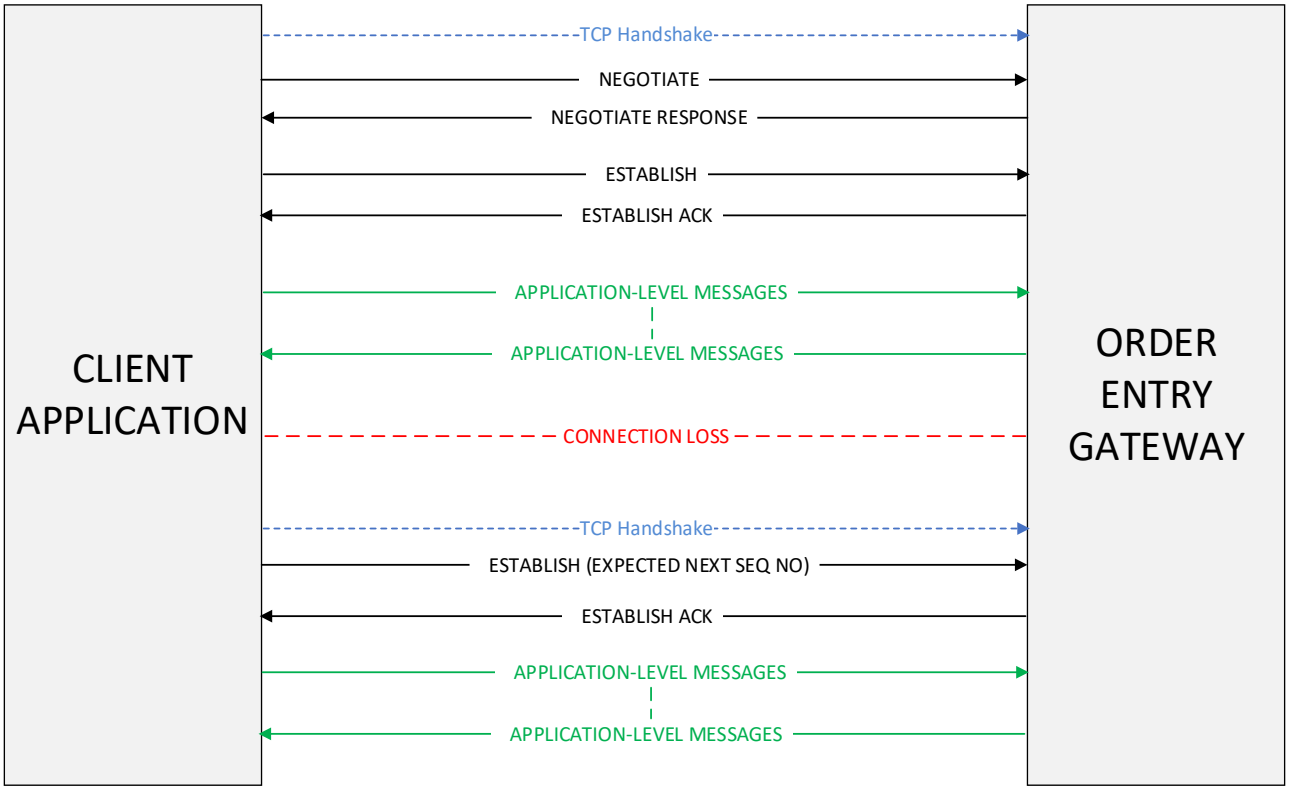


5.2 Invalid credentials in Negotiation Process



5.3 Loss of connectivity during the session

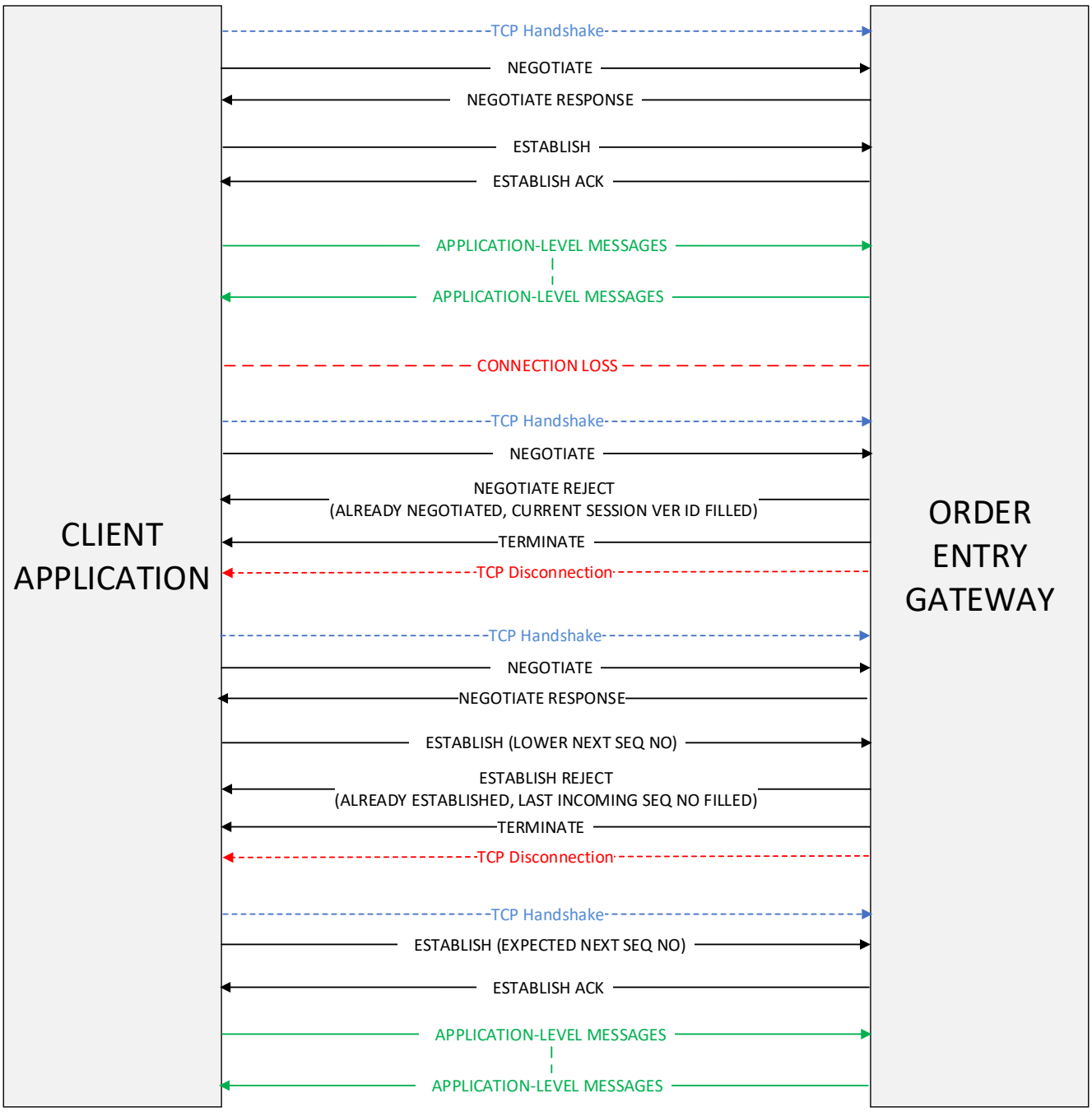
This is the scenario where the client suffers a sudden disconnection during the trading session. Since the protocol specified only one *Negotiate* message must be sent during the day to each gateway, clients are expected to send a new *Establish* message, being able to resume the order flow after receiving a *EstablishAck* message from the Gateway.



5.4 Client-side failure

This scenario depicts the event of a total failure from the client, where all state is lost (*sessionVerID* and *seqNum*). A recovery can be executed by sending a new Negotiate message. The Gateway will reject it sending a *NegotiateReject* message with an error code 3 (**ALREADY_NEGOTIATED**), since the session has already sent a *Negotiate* earlier.

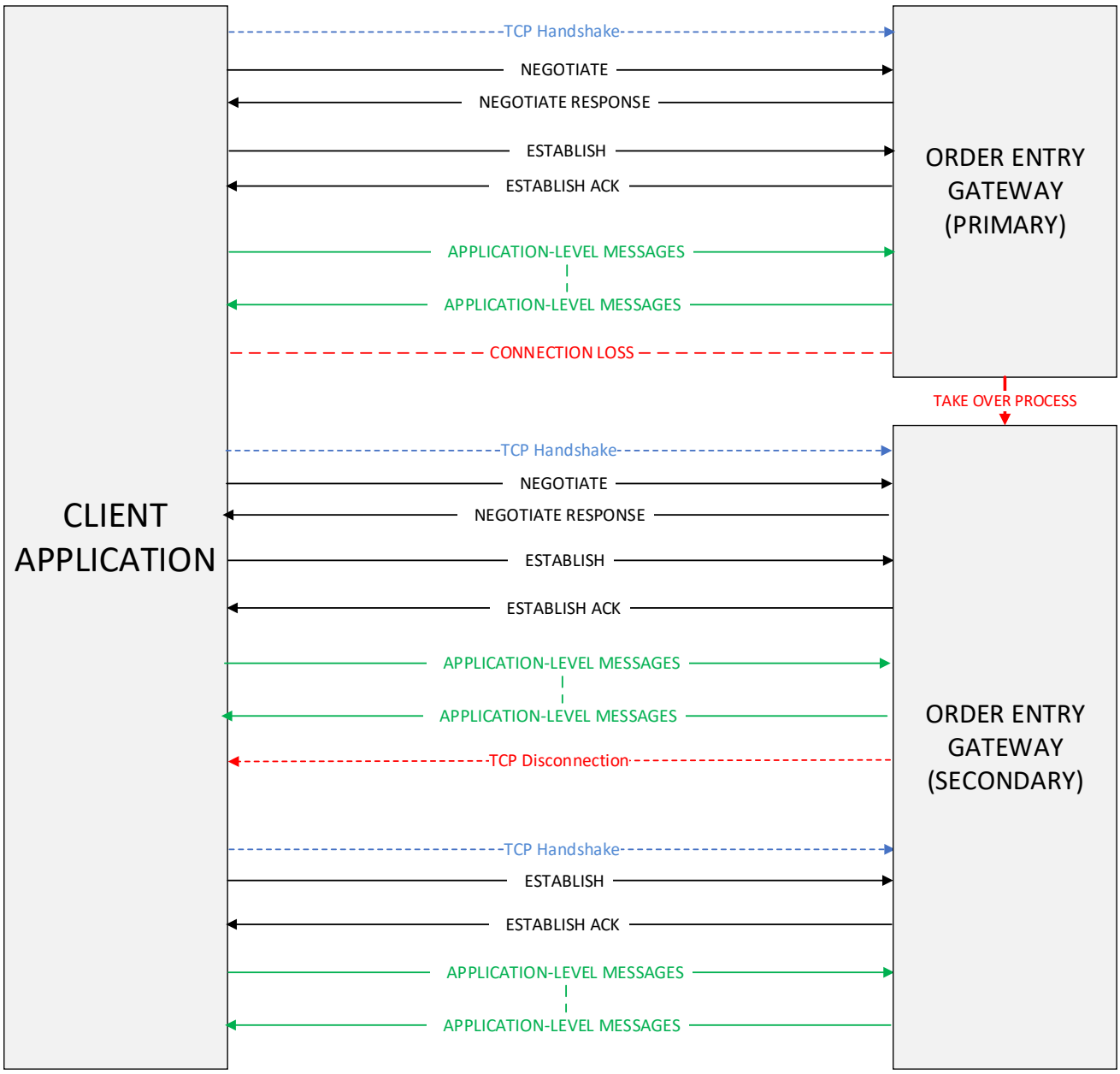
The point is that the *NegotiateReject* will contain the current *sessionVerID* for that session. The client then can then send a new *Establish* message with the correct *sessionVerID*, reenabling the message flow again.



5.5 Gateway failures

This scenario depicts the event of a total failure from the gateway. At the first failure, connection must be switched to the backup server. At this reconnection a Negotiate must be made, since it's done once per gateway. We recommend clients to use the same *sessionVerID* that were being used at the session.

At the second failure, B3 will attempt to reestablish the secondary instance instead of switching back to the primary. In this scenario only a *Establish* is necessary.



6 CERTIFICATION

B3 has a certification environment for testing and certification purposes of their software before accessing the productive environment of the exchange.

The validation and the tests on acquired or under-development solutions can be conducted during working days from 9:00 to 18:00 (local time), with no follow-up needed from the certification team.

For certification arrangements, email the Trading Certification at mailto:tradingcertification@b3.com.br.

7 ORDER CHARACTERISTICS

The behavior of an order can be affected by parameters such as order type and validity. This section describes the types and modifiers which can be applied to a given order.

Although *Binary Entrypoint* strives to maintain a consistent interface across all market segments, sometimes it is not possible to achieve this goal due to constraints such as market regulation. Thus, order concepts are described in the broadest way possible, and market specifics are noted when appropriate.

7.1 Order Types

Order types are determined by the *OrdType* field (tag 40). The following table depicts order type availability at the various segments.

Order Type	Domain value	Market Segments		
		Equities	Fixed Income	Derivatives
Market with Protection	1	✓	✓	
Limit	2	✓	✓	✓
Stop with Protection	3	✓	✓	
Stop Limit	4	✓	✓	✓
Market to Limit	K	✓		✓
Retail Liquidity Provider	W	✓		✓
Pegged Midpoint	P	✓		

7.1.1 Market Orders with Protection (ordType = 1)

Market orders (tag 40 = 1) do not have an associated price. Upon entry, they "sweep" the order book, potentially generating multiple fills at different price levels until one of the following happens:

- The order is filled.
- The order reaches a protection price level (*ProtectionPrice* field - tag 35001), which is automatically calculated by the matching engine. The protection price level represents the worst price that an order can fill. If there is still an unfilled quantity and the next fill would occur at a price beyond protection, the market order rests as a limit order with a price equal to the protection level price.

For bids, the protection price is calculated by adding an offset to the last trade price. For asks (offer), the offset is subtracted from the last trade. The protection price cannot be specified in the incoming order.

The following tables depict how Market orders behave when the protection price is hit:

Starting Order Book (Last Trade Price: 10, Offset: 2 Protection Price (bid): 12)			
Bids		**Asks**	
Qty	Price	Price	Qty
		10	500
		11	300
		13	200

A buy market order for one thousand contracts is received. The following takes place:

Msg sent	Msg received	clOrdId	orderID	Price	orderQty	protection Price	ordStatus	last Qty	last Price	Comment
New Order		202211141516000		N/A	1000					
	ER_New	202211141516000	123	N/A	1000	12	New			Execution Report confirming receipt is sent back to the customer.
	ER_Trade	202211141516000	123	N/A	1000	12	Partially Filled	500	10	Order is partially filled for 500 @ 10.
	ER_Trade	202211141516000	123	N/A	1000	12	Partially Filled	300	11	Second Execution.

7.1.2 Limit Orders (ordType = 2)

Limit orders (tag 40 = 2) specify the worst price at which the order may execute, i.e., an order to buy a security at or below a stated price (defined in tag 44 – Price) or to sell a security at or above a stated price. If the order does not execute, it will remain in the order book.

7.1.3 Stop Orders with Protection (ordType = 3)

Stop orders (tag 40 = 3) have an associated trigger price but no limit price. Whenever a trade crosses the trigger price, the order is automatically inserted in the order book as a limit order. The order price is automatically determined by the trading engine by the addition or subtraction of an offset price in the same spirit of the Market Order with protection.

The following tables depict how Market orders behave when the protection price is hit:

Starting Order Book (Last Trade Price: 10, Offset: 2 Protection Price (bid): 12)			
Bids		**Offers**	
Qty	Price	Price	Qty
		10	500
		11	300
		13	200

A buy market order for one thousand contracts is received. The following takes place:

Msg sent	Msg received	clOrdID	orderID	price	orderQty	stopPx	protection Price	ordStatus	working Indicator	Comment
New Order		202211141516000		N/A	1000	10				
	ER_New	202211141516000	123	N/A	1000	10	12	New	N	Execution Report confirming receipt is sent back to the customer.
A trade crossed (quantity = 100) the trigger price. Thus, the order will rest as a limited order at price 12. Subsequent Execution Reports will have OrdType = Limit.										

Msg received	clOrdID	orderID	price	orderQty	stopPx	protectionPrice	ordStatus	working Indicator	lastQty	lastPrice	Comment
ER_New	202211141516000	123	12	1000		12	New	Y			Execution Report is sent back to the customer.
ER_Trade	202211141516000	123	12	1000		12	Partially Filled		400	10	Order is partially filled for 400 @ 10.
ER_Trade	202211141516000	123	12	1000		12	Partially Filled		300	11	Second Execution:300 @ 11.
The next price level (13) is higher than the protection price (12). Thus, the order will rest as a limited order at price 12. Subsequent Execution Reports will have OrdType = Limit.											

7.1.4 Stop Limit Orders (ordType = 4)

Stop limit orders (tag 40 = 4) are associated with a trigger price – or stop price – at which it becomes a limit order. The order will be inserted into the order book as soon as the first trade occurs at the specified stop price (tag 99 – StopPx). From this point on, it will behave as a regular limit order, with the price defined in *Price* field (tag 44). The order type behavior is like that of Market to Limit Orders: after triggering, it will explicitly change to a limited order (tag 40 = 2).

In the case of illiquid instruments, if a GT Stop Limit Order is accepted, the price variation might cause this order to be triggered outside of the hard limits. In such a situation, the order price will be adjusted according to the values of the intraday limits.

Example: If an Ibovespa buy Stop Limit Order is triggered at 70,200 points, but the high hard limit is 70,089 points, the order will become a limit order at 70,085 points, which is an acceptable price according to instrument tick size (5 points for Ibovespa). This prevents off-tick orders.

7.1.5 Market with Leftover as Limit (orderType = K)

Market with leftover as limit orders (tag 40 = K, also known as Market to limit) will behave like a regular market order, and any unexecuted quantity will become a limit order in the order book, with its limit price set at the last executed price. In a partial fill scenario, the *OrdType* field (tag 40) field is changed to ‘2’ (Limit) to reflect the new order behavior.

7.1.5.1 Scenario: Market with Leftover as Limit order is filled twice

Msg sent	Msg received	clOrdId	ordType	orderID	price	orderQty	ord Status	last Qty	last Price	Comment
New Order		202301	K		N/A	7000				
	ER_New	202301	K	123	N/A	7000	New			Execution Report confirming receipt is sent back to the customer.
	ER_Trade	202301	K	123	N/A	7000	Partially Filled	2000	10.58	Order is partially filled for 2000@10.58
	ER_Trade	202301	2	123	10.58	7000	Partially Filled	1000	10.58	Second Execution: 1000@10.58

7.1.6 RLP – Retail Liquidity Provider (orderType = W)

The Retail Liquidity Provider – RLP (tag 40=W) is a new type of order within the B3 PUMA Trading System with the following aims:

- To enable intermediaries to supply liquidity for part of the flow of aggressing orders from retail customers.
- To assure compliance with best execution principles.
- To preserve the adequate functioning of the price formation process.

The new type of order would have the following characteristics:

Only intermediaries who meet the requirements established by B3 (transparency with customers, opt-in, and opt-out mechanisms etc.) would be able to use the RLP feature.

1. RLP orders could be aggressed only by orders from customers of the same intermediary who were flagged as retail customers (tag 35487=1).
2. RLP orders would be market-pegged orders (the intermediary would indicate the buy and/or sell quantity and the order price would be automatically adjusted by PUMA to the best bid plus configured offset or best ask minus configured offset).
3. If the spread between the best ask and the best bid were two or more tick sizes, the RLP price would improve by one tick size or more at the intermediary's discretion (as is the case under the existing rule for cross orders in the derivatives segment).
4. Considering the top price level of the order book at a given time and the arrival of an aggressing retail order from an intermediary's customer:

a. Except for the scenario of item b, the RLP would pass in front of all offers from all other intermediaries (by price-firm-time priority instead of price-time priority). In this case, the aggressing order would be forwarded to the RLP book. Any remainder left after execution in the RLP book would be sent to the central orders book.
b. The RLP would not be ranked ahead of orders from customers of the same intermediary that would match the aggressing order (no preemption for customers of an intermediary). Thus, if an order from another customer of the same intermediary would be aggressed on the opposite side of the book, the aggressing order would be routed to the order book and would match existing orders from all brokerages houses up to the last order from the same intermediary's customer (inclusive). After executing with the last customer of the same brokerage firm at the top of the book and in case there is a balance, the aggressor order will be sent to the RLP book. Any remainder after execution in the RLP book would be sent to the central order book.

5. The aggregate volume of RLP orders in the market would not be allowed to exceed Y% of the total volume of the instrument (will be defined and disclosed on the B3 website).
6. Because no orders in the overall market would be able to aggress them, RLP orders would not have pre-trade transparency but would be disclosed via the market data feed immediately after the execution of the trade.
7. All RLP orders are valid for the day.
8. RLP orders and orders flagged as retail customers (tag 35487=1) do not support disclosed quantity (tag 111).
9. RLP orders and orders flagged as retail customers (tag 35487=1) do not support minimum quantity (tag 110).
10. Orders flagged as retail customers (tag 35487=1) can only be limit (tag 40=2), stop with protection (tag 40=3), stop limit (tag 40=4), or market with leftover as limit (tag 40=K) orders.
11. Orders flagged as retail customers (tag 35487=1) support only day (tag 59=0), immediate or cancel (tag 59=3) and fill or kill (tag 59=4) validities.

7.1.6.1 Retail Liquidity Provider (RLP) Functionality Scenarios

General considerations on scenarios 1-7:

- The order book spread is closed and the tick size for the instrument is equivalent to five (5) points.
- Pegged prices of hidden buy and sell orders (RLPs) are 74,995 and 75,000, respectively.
- Given that a customer of brokerage house A is submitting a retail order, the hidden order (RLP) from brokerage house B is inactive and therefore cannot be executed.

7.1.6.1.1 Scenario 1: narrow spread without an order from the brokerage house's customer on the order book

- A retail customer of brokerage house A sends the trading platform a bid for 10 at a limit price of 75,000.

➢ A hidden ask (RLP) from brokerage house A is active and can be aggressed by retail bids from the same brokerage house since there are no visible asks from customers of brokerage house A at the top price level of the order book.

	BID			ASK		
	Firm	Qty	Price	Firm	Qty	Price
Hidden	**RLP A**	**1000**	**Pegged**	**Pegged**	**1000**	**RLP A**
	RLP B	1000	Pegged	-	-	-
Visible	C	5	74.995	75.000	20	D
	D	10	74.990	75.005	10	F
	E	5	74.995	75.010	5	G

A bid from a retail customer of brokerage house A for 10@75,000 aggresses a hidden ask (RLP) from brokerage house A.

The trade is published in the market data feed.

Trade			
Buy firm	**Sell firm**	**Qty**	**Price**
A	RLP A	10	75.000

Resulting book: balance of 990 quantities in the RLP from brokerage house A.

7.1.6.1.2 Scenario 2: narrow spread with an order from a brokerage house's customer the at top price level of the order book

➢ A retail customer of brokerage house A sends the trading platform a bid for 10 at a limit price of 75,000.

➢ A hidden ask (RLP) from brokerage house A is inactive and cannot be aggressed by retail bids from the same brokerage house since there is a visible ask from a customer of brokerage house A at the top price level of the order book and this ask matches the total quantity of the RLP.

BID			ASK		
Firm	**Qty**	**Price**	**Firm**	**Qty**	**Price**
RLP A	**1000**	**Pegged**	Pegged	1000	RLP A
RLP B	1000	Pegged	-	-	-

BID			ASK		
C	5	74.995	75.000	10	A
D	10	74.990	75.005	10	F
E	5	74.995	75.010	5	G

- A bid from a retail customer of brokerage house A for 10@75,000 aggresses a visible ask from a customer of brokerage house A at the top price level of the order book.
- The trade is published in the market data feed.

Trade			
Buy firm	**Sell firm**	**Qty**	**Price**
A	A	10	75.000

- Resulting book: Hidden ask (RLP) from brokerage house A is altered to active since there are no longer any visible asks from customers of brokerage A at the top price level of the order book.

7.1.6.1.3 Scenario 3: narrow spread with an order from the brokerage house's customer the at top price level of the order book (an alternative scenario)

- A retail customer of brokerage house A sends the trading platform a bid for 10 at a limit price of 75,000.
- A hidden ask (RLP) from brokerage house A is inactive and cannot be aggressed by retail bids from the same brokerage house since there is a visible ask from a customer of brokerage house A at the top price level of the order book, and the total quantity of visible orders at the top price level of the order book matches the quantity of the RLP.

BID			ASK		
Buy firm	**Qty**	**Price**	**Price**	**Qty**	**Sell firm**
RLP A	**1000**	**Pegged**	Pegged	1000	RLP A
RLP B	1000	Pegged	-	-	-
C	5	74.995	75.000	5	A
D	10	74.990	75.000	5	F
E	5	74.985	75.010	5	G

- A bid from a retail customer of brokerage house A for 10@75,000 aggresses an ask from a customer of brokerage house F and then aggresses an ask from a customer of brokerage house A at the second price level of the order book.

- The trade is published in the market data feed.

Trade			
Buy firm	**Sell firm**	**Qty**	**Price**
A	F	5	75.000
A	A	5	75.000

7.1.6.1.4 Scenario 4: narrow spread with an order from the brokerage house's customer at the top price level of the order book and RLP for a higher quantity than the quantity available at a top price level

A retail customer of brokerage house A sends the trading platform a bid for 15 at a limit price of 75,000.

	BID			ASK		
Hidden	**Buy firm**	**Qty**	**Price**	**Price**	**Qty**	**Sell firm**
	RLP A	**1000**	**Pegged**	**Pegged**	**1000**	**RLP A**
	RLP B	1000	Pegged	-	-	-
Visible	C	5	74.995	75.000	10	A
	D	10	74.990	75.005	10	F
	E	5	74.995	75.010	5	G

- A bid from a retail customer of brokerage house A for 15@75,000 aggresses the disclosed offer from brokerage house A, and the balance is matched with the non-disclosed sell offer (RLP) of brokerage house A.

- The trades are published in the market data feed:

Trades			
Buy firm	**Sell firm**	**Qty**	**Price**
A	A	10	75.000
A	RLP	5	75.000

- Resulting book: balance of 995 quantities in the hidden order (RLP) from brokerage house A.

7.1.6.1.5 Scenario 5: Narrow spread with the client of the brokerage firm in the 1st price level and that accepts to be deprecated.

- A retail customer of brokerage house A sends the trading platform a bid for 10 at a limit price of 75,000.
- A hidden ask (RLP) from brokerage house A is active because even if the sell order of client of brokerage A in the book (marked as A*) could be aggressed, this one accepts to be deferred in the analysis of the RLP matching algorithm.

	BID			ASK		
	Buy firm	**Qty**	**Price**	**Price**	**Qty**	**Sell firm**
Hidden	**RLP A**	**1000**	**Pegged**	**Pegged**	**1000**	**RLP A**
	RLP B	1000	Pegged	-	-	-
Visible	C	5	74.995	75.000	5	A
	D	10	74.990	75.000	5	F
	E	5	74.985	75.000	5	G

- A bid from a retail customer of brokerage house A for 10@75,000 aggresses the RLP ask of the brokerage house A.

- The trades are published in the market data feed:

Trade			
Buy firm	**Sell firm**	**Qty**	**Price**
A	RLP A	10	75.000

Resulting book:

	BID			ASK		
	Buy firm	**Qty**	**Price**	**Price**	**Qty**	**Sell firm**
Hidden	**RLP A**	**1000**	**Pegged**	**Pegged**	**990**	**RLP A**
	RLP B	1000	Pegged	-	-	-
Visible	C	5	74,995	75	5	F
	D	10	74,99	75	5	A*
	E	5	74,985	75	5	G

7.1.6.1.6 Scenario 6: Narrow spread with the client of the brokerage firm in the 1st price level and who accepts to be deprecated and another that does not accept.

- A retail customer of brokerage house A sends the trading platform a bid for 30 at a limit price of 75,000.
- A hidden ask (RLP) from brokerage house A is inactive because a sell order of brokerage A in the book (marked as A*) even though waived its priority on the RLP analysis, there is a second ask of brokerage A, at the same price level, which did not waive its priority.

	BID			ASK		
	Buy broker	**Qty**	**Price**	**Price**	**Qty**	**Sell broker**
Hidden	**RLP A**	**1000**	**Pegged**	Pegged	990	RLP A
	RLP B	1000	Pegged	-	-	-
Visible	C	5	74,995	75	5	F
	D	10	74,99	75	5	A*
	E	5	74,985	75	5	G

- A bid from a retail customer of brokerage house A for 30@75,000 aggresses the ask of the brokerage houses F, A*, G, A, and RLP A.
- The trades are published in the market data feed:

Trade			
Buy firm	**Sell firm**	**Qty**	**Price**
A	F	5	75.000
A	A*	5	75.000
A	G	5	75.000
A	A	5	75.000
A	RLP A	10	75.000

Resulting book:

	BID			ASK		
	Buy firm	**Qty**	**Price**	**Price**	**Qty**	**Sell firm**
Hidden	**RLP A**	**1000**	**Pegged**	Pegged	990	RLP A
	RLP B	1000	Pegged	-	-	-

	BID			ASK		
Visible	C	5	74,995	75	5	H
	D	10	74,990			
	E	5	74,985			

7.1.7 Order Validity Types (Time in Force)

The following table depicts Validity Type availability at the various segments.

Market Segments			
Validity Type	Equities	Fixed Income	Derivatives
Day	✓	✓	✓
Immediate or Cancel	✓	✓	✓
Fill or Kill	✓	✓	✓
Good Till Date	✓	✓	
Good Till Cancel	✓	✓	
Good for Auction	✓	✓	
At the Close	✓	✓	

7.1.8 Day (TimeInForce = 0)

Day orders (tag 59 = 0) are available in the order book during the day until they execute or are canceled (either by the customer who submitted it or B3 market operations). It is considered the default validity when none is specified.

7.1.9 Good Till Cancel (GTC) (TimeInForce = 1)

Good till cancel orders (tag 59 = 1) never expire except on a corporate action day when all GTC orders are canceled. They are inserted in the order book and remain until cancellation by the customer or market surveillance, or until it is fully executed. Until these conditions are met, GTC orders are restated to client systems at the start of each trading session.

7.1.10 Immediate or Cancel (IOC) (TimeInForce = 3)

The Immediate or Cancel (IOC) validity (tag 59 = 3), also known as Fill and Kill (FAK), indicates that the order requires immediate execution, and the unexecuted quantity is automatically canceled. If there is no counterparty to execute against, the order is acknowledged and then canceled.

The following table depicts the scenario where the incoming order can be matched:

Msg sent	Msg received	clOrdID	orderID	price	orderQty	ordStatus	last Qty	last Price	Comment
NewOrder		20230201		10.58	7000				
	ER_New	20230201	123	10.58	7000	New			Execution Report confirming receipt is sent back to the customer.
	ER_Trade	20230201	123	10.58	7000	Partially Filled	4000	10.58	Order is partially filled for 4000 @ 10.58.
	ER_Cancel	20230201	123	10.58	7000	Canceled			The order is canceled.

7.1.11 Fill or Kill (FOK) (TimeInForce = 4)

Fill Or Kill (FOK) orders (tag 59 = 4) require that the full amount stated in the order is executed upon entering the order book. If there is not enough quantity on the opposite side to fill the order, the order is acknowledged and then canceled. It is also known as All or Nothing (AON).

Msg sent	Msg received	clOrdID	orderID	price	orderQty	ordStatus	last Qty	Comment
New Order		20230201		10.58	7000			The instrument is in continuous trading mode.
	ER_New	20230201	123	10.58	7000	New		Execution Report acknowledging the order.
	ER_Cancel	20230201	123	10.58	7000	Canceled		Execution Report canceling the order, since there is no opposite side.

7.1.12 Good Till Date (GTD) (TimeInForce = 6)

The Good till Date (tag 59 = 6) validity causes the order to expire at the end of the trading session of the date stated in the *expireDate* field (tag 432) field of the original order submitted by the customer, except on a corporate action day when all GTD orders are canceled once before the predefined expiry date. Until these conditions are met, GTD orders are restated to client systems at the start of each trading session.

At the end of the trading session of the expiration date, the orders are canceled by the matching engine, and customers who submitted the orders receive Execution Reports expiring the orders [*ordStatus* field (tag 39) = Expired (C)].'

7.1.13 At the Close (MOC) (TimeInForce = 7)

The At the Close (tag 59 = 7) validity allows the participants to place orders that participate in the closing auction in advance. For example, the order can be placed during continuous trading but will become active just when the closing auction starts. It is also known as Market on Close (MOC).

When placed, if the order is not rejected, participants will receive an acknowledgment (*ER_New* message, *ordStatus*=New) indicating that the order is accepted but not active (*WorkingIndicator*=N).

When the closing auction starts, an Execution Report is sent, notifying that the order is active (*ER_New* message, *ordStatus*=New, *WorkingIndicator*=Y).

7.1.14 Good for Auction (MOA) (TimeInForce = A)

The Good for Auction validity (tag 59 = A) indicates that an order is valid for the ongoing auction only. It is also known as Market on Auction (MOA).

Thus, it is only accepted during an auction. Whenever the auction finishes, the order expires.

7.1.15 Order Quantities

EntryPoint supports different order quantities which can be used to accomplish determinate trading strategies, such as minimum guaranteed execution and partial order disclosure. Order quantities are discussed in the following sections.

7.1.16 Disclosed Quantities (Iceberg Orders)

Disclosed Quantity allows participants to trade a large lot of as given security without exposing the whole lot in the market at once. The *maxFloor* field (tag 111) determines the largest amount which is shown in the order book at a time. For example, an order with *orderQty* = 10000 and *maxFloor* = 500 will show in the order book as a 500 contract (shares) order. After the order is filled for 500 contracts, the matching engine will replenish the quantity back to 500 contracts, until all *orderQty* is consumed, or the order is canceled.

In the PUMA Trading System, to preserve the hidden nature of iceberg orders, the matching engine will assign a new order identifier (*secondaryOrderID* field) each time the order is replenished. It also enters in the open book with the least priority in the same price-level order list (same behavior as a new order).

Msg sent	Msg received	orderID	secondary OrderID	orderQty	max Floor	lastQty	leavesQty	ordStatus
NewOrder				10000	500			

Msg sent	Msg received	orderID	secondary OrderID	orderQty	max Floor	lastQty	leavesQty	ordStatus
	ER_New	1	201	10000	500		10000	New
	ER_Trade	1	202	10000	500	200	9800	Partially Filled
	ER_Trade	1	202	10000	500	300	9500	Partially Filled
	ER_New	1	**203**	10000	500		9500	**Restated**

7.1.17 Minimum Quantity

Orders with minimum quantity must execute at least the quantity stated in the *minQty* field (tag 110) in every transaction. Orders whose minimum quantity may not be satisfied upon entry in the order book are canceled.

The minimum quantity can also be specified on modification messages. The behavior is equivalent to the order entry scenario.

7.1.18 Trade-Related Quantities

EntryPoint provides fields that can be used to track quantity state through executions. Those fields are sent in every Execution Report message:

- *cumQty* field (tag 14): indicates the accumulated quantity of all trades involving the order. Its value grows as subsequent fills take place. If a given trade for a filled order is canceled (trade bust), the *cumQty* value sent in the Execution Report message is set to zero. If a trade of a partially filled order is canceled, the *cumQty* value will decrease in the same value as the busted quantity. Participants can track the actual executed quantity by subtracting *lastQty* value from the cumulative quantity maintained by the participant's application.
- *leavesQty* field (tag 151): conveys the amount which is still open for execution. This quantity decreases with every fill. Please note that trade busts do not roll back the busted quantity. Thus, *leavesQty* value does not increase when a bust takes place.
- *lastQty* field (tag 32): contains the traded quantity from the last execution. *lastQty* value changes with every fill and will stay the same between fills.

7.1.19 Canceled Quantity

If the order is canceled (upon specific single or mass action request or by exchange's surveillance team), the session whose order was entered receives a related execution report message (*ExecutionReport_Cancel*). The canceled quantity of the related order is not explicitly published but may be calculated by subtract the values of *orderQty* with *cumQty* (fields that come with *ExecutionReport_Cancel* message).

7.1.20 In-Flight Modification

The *orderQty* field is interpreted by B3 as the total investor quantity, i.e., the total size of the order. That stands true for order modification requests as well. Hence, the connecting counterparty must consider this when implementing cancel/modification logic, especially regarding in-flight modification scenarios (where the order is executed at the exchange at the same time the counterparty issues a modification request for that same order).

The following scenarios represent a high-level overview of the messages exchanged (with the most relevant fields only) for different situations that represent the interpretation of *orderQty*:

7.1.20.1 Scenario 1: Plain modification of previously sent order

In this scenario, an order is sent (BUY 1000 @ 12) and has its quantity increased to 1400 due to a modification request.

Msg sent	Msg received	clOrdID	origClOrdID	orderID	origClOrdID	price	orderQty	ordStatus	cumQty	leavesQty	Comment
NewOrder		20230301				12.00	1000				New Order from a trader.
	ER_New	20230301		1		12.00	1000	New	0	1000	Order is packed by the exchange.
ModifyOrder		20230322	20230301	1	20230301	12.00	1400				Order qty is increased to 1400 by the trader.
	ER_Modify	20230322	20230301	1	20230301	12.00	1400	Replaced	0	1400	Modification is ack' ed by the exchange.

7.1.20.2 Scenario 2: In-flight modification of previously sent order

In this scenario, an order is sent (BUY 1000 @ 12), which partially executes for 200. Concurrently (i.e., before receiving the Execution Report notifying the partial fill of 200), the counterparty issues a modification request to increase its quantity to 1300.

Msg sent	Msg received	clOrdID	origClOrdID	orderID	price	orderQty	ordStatus	cumQty	leavesQty	lastQty	lastPrice	Comment
New Order		20230301			12.00	1000						New Order from the trader.
	ER_New	20230301		1	12.00	1000	New	0	1000			Order is ack' ed by the exchange.

Msg sent	Msg received	clOrdID	origClOrdID	orderID	price	orderQty	ordStatus	cumQty	leavesQty	lastQty	lastPrice	Comment
Modify Order		20230322	20230301	1	12.00	1300						Order qty is increased to 1300 by the trader.
	ER_Trade	20230301		1	12.00	1000	Partially Filled	200	800	200	12.00	Partial fill is received by the trader.
	ER_Modify	20230322	20230301	1	12.00	1300	Replaced	200	1100			Modification is ack' ed by the exchange.

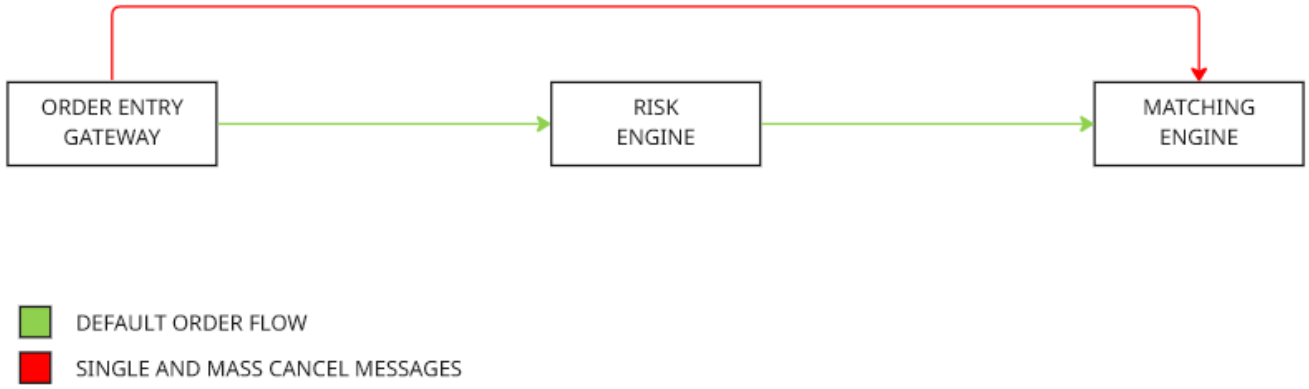
7.1.20.3 Scenario 3: In-flight modification of previously sent order

In this scenario, an order is sent (BUY 1000 @ 12.00), which partially executes for 800 (remaining quantity = 200). Concurrently, the counterparty issues a modification request trying to decrease the order quantity from the original 1000 to 700.

Msg sent	Msg received	clOrdID	origClOrdID	orderID	price	orderQty	ordStatus	CumQty	leavesQty	lastQty	lastPrice	Comment
NewOrder		20230301			12.00	1000						New Order from the trader.
	ER_New	20230301		1	12.00	1000	New	0	1000			Order is ack' ed by the exchange.
	ER_Trade	20230301		1	12.00	1000	Partially Filled	800	200	800	12.00	Partial fill is received by the trader.
ModifyOrder		20230322	20230301	1	12.00	700						Order qty is decreased to 700 by trader
	ER_Cancel	20230322	20230301	1	12.00	1000	Canceled	200	N/A			Since the order is being modified with orderQty < CumQty, it is canceled

7.2 Cancellation behavior

One important thing to keep in mind when implementing cancellation logic is the message path. After the implementation of LiNe Affinity in the PUMA Trading System, cancellation messages (*OrderCancelRequest* and *OrderMassActionRequest*) follow a different path from the order entry gateway to the matching engine as depicted in the next diagram:



Cancel messages take precedence, bypassing the risk engine (*LiNe*) and going directly to the matching engine with a lower latency time.

It is important to know that this architecture change can cause scenarios where a single cancellation is rejected by the matching engine because it arrived before the actual order, which is passing through risk analysis at the risk engine.

7.3 Simple Order Messages

Simple Order Messages are designed to have a smaller size and slightly less latency to be encoded and decoded.

Binary EntryPoint supports two types of simple order messages: *SimpleNewOrder* and *SimpleModifyOrder*. They have less feature options and domains. You can see details at the Message Specification document.

7.3.1 Simple New Order

- *OrdType* field only accepts Market (1) and Limit (2) values.
- *TimeForce* field only accepts Day (0), ImmediateOrCancel (3) and FillOrKill (4).

7.3.2 Simple Modify Order

- Can only modify *orderQty* (tag 38) and/or *Price* (tag 44).

7.4 Order Characteristics Modification/Removal/Cancel

The various order characteristics behave differently regarding modification. Characteristics like order cannot be modified, while others can be altered subject to market rules. The following table summarizes how and when various order characteristics can be changed:

Characteristic	Field	Tag	Add	Remove	Change
Order Type	OrdType	40			✓
Validity	TimeInForce	59			✓
Expire Date	ExpireDate	432	✓ (if modifying to GTD)	✓ (if modifying from GTD)	✓ (if TimeInForce is GTD)
Price	Price	44	✓ (if modifying to Limit or Stop Limit)	✓ (if modifying from Limit or Stop Limit)	✓ (if OrdType is Limit or Stop Limit)
Security ID	Security ID	48			
Order Side	Side	54			
Stop Price	StopPx	99	✓ (if modifying to stop limit)	✓ (if modifying from the stop limit)	✓ (if OrdType is Stop Limit)
Memo	Memo	5149	✓		✓

Characteristic	Field	Tag	Add	Remove	Change
Minimum Quantity	MinQty	110	✓		
Order Quantity	orderQty	38			✓
Disclosed Quantity	MaxFloor	111	✓	✓ (set value of tag MaxFloor to 0)	✓
Account	Account	1	✓	✓ (set value of tag *AccountType* to 38)	✓ [2]
Entering Trader					
ExecutingTrader			✓		
enteringFirm					
SenderLocation					
InvestorID			✓		✓

Only the fields that are being changed need to be sent in the replacement. Fields that are not sent will be considered the same as the original order.

[2] Account Change/Removal/Cancel might be unavailable to some participants due to specific market rules.

7.5 Impact of the changes on the order's priority.

The changes in the orders may impact the priorities. The changes and the respective impacts are shown in the table below:

Characteristic	Impact on the priority
Price	Priority Loss
Stop price triggering	Priority Loss
Stop price limit	Priority Loss
Increase Quantity	Priority Loss
Decrease Quantity	Keep Priority
Minimum quantity	Priority Loss
Order Type	Keep Priority
Validity	Keep Priority
Account [1]	Keep Priority

7.6 Order Identification

EntryPoint supports order identifiers for participants to keep track of order events.

7.6.1 Participant-Issued Identifiers

Participant-issued identifiers are assigned by market participants through the order lifecycle. They are discussed in the following subsections.

7.6.1.1 clOrdID

The *clOrdID* field (tag 11) field is the primary client-side order identifier. It is initially assigned at order entry and can be subsequently changed through the order lifecycle in modifications and cancellation requests. It must be unique among all active orders on a given instrument sent via a specific FIXP session.

7.6.1.2 origClOrdID

The *origClOrdID* field (tag 41) field is used in conjunction with the *clOrdID* field and allows the client to implement client-side order chaining, that is, to keep a history of client-initiated order events. Each Modification/Cancellation

request must have an associated *origClOrdID* field. The following table depicts how chaining maintain order relation across subsequent operations.

Msg sent	Msg received	clOrdID	orig ClOrdID	order ID	price	order Qty	ord Status	cum Qty	leaves Qty
New Order		202211141530			12.00	1000			
	ER_New	202211141530		1	12.00	1000	New	0	**1000**
Modify Order		202211141531	202211141530	1	12.00	1200			
	ER_Modify	202211141531	202211141530	1	12.00	1200	Replaced	0	**1200**
Modify Order		202211141532	202211141531	1	12.00	1400			
	ER_Modify	202211141532	202211141531	1	12.00	1400	Replaced	0	1400

7.6.1.3 *clOrdID/origClOrdID* Chaining Rules

On outbound messages, *EntryPoint* echoes the *clOrdID*/*origClOrdID* fields assigned in the inbound message:

Msg sent	Msg received	clOrdID	origClOrdID	price	orderQty
NewOrder		202211141530		12.00	1000
	ER_New	202211141530		12.00	1000
ModifyOrder		202211141531	202211141530	12.00	1200
	ER_Reject	202211141531	202211141530	12.00	1200
ModifyOrder		202211141532	202211141530	13.00	1400
	ER_Modify	202211141532	202211141530	13.00	1400

This behavior also allows participants to send the same *clOrdID/origClOrdID* fields on all messages in each order chain:

Msg sent	Msg received	clOrdID	origClOrdID	price	orderQty
NewOrder		202211141530		12.00	1000
	ER_New	202211141530		12.00	1000
ModifyOrder		202211141530	202211141530	12.00	1200
	ER_Reject	202211141530	202211141530	12.00	1200
ModifyOrder		202211141530	202211141530	12.00	1400
	ER_Modify	202211141530	202211141530	12.00	1400
CancelOrder		202211141530	202211141530	12.00	1400
	ER_Reject	202211141530	202211141530	12.00	1200
CancelOrder		202211141530	202211141530	12.00	1400
	ER_Cancel	202211141530	202211141530	12.00	1400

7.6.1.4 OrdTagID

The *ordTagID* field (tag 35505) field is a user-defined criterion for mass action like Mass Cancel messages.

7.6.2 Exchange-Issue Identifiers

Exchange-Issued identifiers are discussed in the following subsections.

7.6.2.1 MarketSegmentID

Identifies the market segment. Required for all tradable instruments. Not present in equity indexes, ETF indexes, BTB, and Option Exercise, which are non-tradable instruments.

7.6.2.2 orderID

The orderID (tag 37) field is one of the exchange-issued order identifiers. It is assigned by the matching engine on successful order entry, and it remains the same during the entire order lifecycle. The identifier is guaranteed to be globally unique across all parameters.

7.6.2.3 secondaryOrderID

The secondaryOrderID (tag 198) field is an alternate identifier issued by the exchange systems. Uniqueness is guaranteed to be globally unique across all parameters. Differently from the *orderID* field, the value of *secondaryOrderID* field changes for every client-initiated event, and when a disclosed quantity order is replenished. The nature of this behavior allows hiding disclosed orders through multiple replenishments in the market data perspective. The following tables present the *secondaryOrderID* behavior in various scenarios:

Example: Order is received, partially filled, and totally filled (no change to the value of *secondaryOrderID*).

Msg sent	Msg received	clOrdID	securityID	orderID	secondary OrderID	ordStatus	Comment
NewOrder		202211141600	10030432				New Order from the trader.
	ER_New	202211141600	10030432	1	1	New	Order is ack 'ed by the exchange.
	ER_Trade	202211141600	10030432	1	1	Partially filled	Partial fill is received by the trader.
	ER_Trade	202211141600	10030432	1	1	Filled	The order is filled.

Example: Disclosed order is received, partially and then completely filled (value of *secondaryOrderID* changes upon replenishment).

Msg sent	Msg received	clOrdID	securityID	orderID	secondary OrderID	ordStatus	Comment
NewOrder		202211141600	10030432				New Order from the trader.

Msg sent	Msg received	clOrdID	securityID	orderID	secondary OrderID	ordStatus	Comment
	ER_New	202211141600	10030432	1	1	New	Order is ack 'ed by the exchange.
	ER_Trade	202211141600	10030432	1	1	Partially filled	Partial fill is received by the trader.
	ER_Trade	202211141600	10030432	1	2	Partially filled	Partial fill is received by the trader.
	ER_Trade	202211141600	10030432	1	3	Filled	The order is filled.

Example: Order is received, and then modified (*secondaryOrderID* changes upon modification).

Msgsent	Msg received	clOrdID	securityID	order ID	secondary OrderID	ordStatus	Comment
NewOrder		202211141600	10030432				New Order from the trader.
	ER_New	202211141600	10030432	1	1	New	Order is ack 'ed by the exchange.
	ER_Modify	202211141600	10030432	1	2	Replaced	The order was modified.
	ER_Trade	202211141600	10030432	1	2	Filled	The order is filled.

7.6.3 Order Identifier Rules

The following rules apply to the order identifiers:

- *clOrdID/origClOrdID* should be unique per FIXP session and matching engine. Uniqueness is checked among standing orders only.
- If an order is rejected upon entry, the rejection Execution Report (ExecutionReport_Reject) message contains a newly created *orderID*.

7.7 Strategy Identification

If participants need to identify their Order Entry Strategy, messages in *Binary Entrypoint* have been equipped with a customized field called "*strategyID*", so that participants can use it to identify their strategies.

strategyID field is only present in the full new or amendment messages and the value is returned in the following related *ExecutionReport* messages.

8 EXECUTION REPORT

8.1 Aggressor Indicator

The *aggressorIndicator* field (tag 1057) boolean type is returned in the *ExecutionReport_Trade* message to indicate whether the order initiator is an aggressor or not in the trade.

Values are:

- Y = Order initiator is the aggressor
- N = Order initiator is the passive

8.2 Rejection Codes

B3 will issue an *ExecutionReport_Reject* message for orders rejected by order entry gateway, pre-trade risk control, or the match engine. The code for the rejection is stated in *ordRejReason* field (tag 103) and the actual text of the rejection is informed in *text* field (tag 58).

To make it easier for client systems to handle the internationalization of error messages, B3 compiles a list of all possible rejection codes on the *EntryPoint* website, available here:
http://www.b3.com.br/data/files/6F/97/A8/0F/6AEFD610BB692DD6AC094EA8/EntryPointErrorCodesV1.0.13.pdf

8.3 Daily GTC/GTD Restatement

To ensure greater synchronization and integrity of the GT (GTC and GTD) order state in the PUMA system through daily republication at the beginning of the day. This initiative mainly addresses the demand from retail brokerage firms.

Currently, the management of GT orders by customers presents challenges related to the maintenance, updating, and consistency of information. Daily republication simplifies this process, reducing operational risks and inconsistencies that can impact customer experience and market reliability.

8.3.1 How it works

On every trading day, at a predefined time in the early hours of the day, all matching engines that have GT (GTC/GTD) orders, publish ***ExecutionReport_Modify*** messages with the following characteristics:

Field Name	Field Id	Value
ordStatus	39	**0** (*New*) or **1** (*Partially Filled*)
execRestatementReason	378	**1** (*GT Restatement*)

If the restated GTC/GTD order has already been partially filled, the *leavesQty* and *cumQty* fields will be populated accordingly and the value of *ordStatus* field reflects it: **1** (*Partially Filled*). Otherwise, if no matches occurred previously in the order, the value of *ordStatus* field is **0** (*New*).

There are two scenarios that clients must deal with in the FIXP sessions:

1. **The session is established before the restatement process occurs**

   It is straightforward – the client receives all of restatement directly the moment they are published.

2. **The session will only be established after the restatement process occurs**

   When the client is establishing the session, sending the *Establish* message, the gateway publishes back the *EstablishAck* message whose *nextSeqNo* field takes account the number of outstanding restatement execution report messages. Therefore, to receive those restatements, the client needs to request retransmission, taking into account the received value of nextSeqNo field. See section 4.5.6.

9 PARTICIPANT IDENTIFICATION

In *Binary Entrypoint*, participants are identified using the *senderLocation, enteringTrader* and *executingTrader* in the related messages, and *enteringFirm* informed in the *Negotiate* message.

The following table illustrates an example of a DMA provider (e.g., "XYZ") sending a New Order Single to B3, entered by trader "ABC":

Msg	clOrdId	Price	Qty	Entering Trader	Sender Location	EnteringFirm (from Negotiate)
NewOrderSingle	1205	10.58	7000	ABC	XYZ	127

The following table describes all available participant domains.

Tag	Field	Description	Applicable Segments	
			Equities	Derivatives
35501	enteringFirm	Brokerage firm identifier as assigned by B3.	in the Negotiate msg	in the Negotiate msg
35502	enteringTrader	Trader identifier. B3 assigned for desk traders, free format for DMA.	Required	Required
35503	senderLocation	Identifies the order originator and DMA category. Desk traders always set this to 'BVMF,' DMA connections must use the value assigned by B3.	Required	Required
35504	investorID	Used for Self-Trading prevention at the customer level or mass cancel on behalf.	Optional	Optional
35505	ordTagID	Identifies the order tag identification.	Optional	Optional
35506	executing Trader	Identifies the trader which is acting on behalf of another. Only used when trading on behalf.	Optional	Optional
35507	custodianInfo	Identifies the custodian as described below:	Optional	Optional
→ 35508	custodyAccount	Identifies the custody account.	Optional	Optional
→ 35509	custody AllocationType	Identifies the custody portfolio.	Optional	Optional
35510	deskID	Participants may use this field to identify the client associated with the given account number. This information may be used to correlate the order entry messages with the messages at the back-office and clearing systems.	Optional	Optional
35531	contraBroker	Brokerage firm identifier as assigned by B3 used to indicate the counterparty brokerage firm in a Forward deal.	Required for forward deals, N/A for other securities	N/A

10 SECURITY IDENTIFICATION

In the order entry messages, the only required field to uniquely identify security is Security ID (tag 48). This tag conveys the human-readable security identifier, and it is available in the Security List message.

To trade securities listed in other venues (e.g., Globex); the *securityExchange* (tag 207) tag should be used to specify the market to which the security belongs, using its proper MIC[3] code. If *securityExchange* is not provided, BVMF is assumed as a default venue and it is a constant type.

To maintain compatibility with previous implementations of the order entry interface, *securityIDSource* (tag 22) is still present as a constant type.

11 ACCESS CATEGORIES

All four DMA categories are available in the equities and derivative segments. Each category has a new alpha-numeric code which indicates that a given connection belongs to a specific category and replaces the former Bovespa numeric ranges. The new alpha-numeric code must be sent in every message as a Sender Location Party Role.

[3] Market Identifier Code, an International Standard (ISO 10383) used to uniquely identify Exchanges.

The following table depicts all access categories, their associated codes, and the correspondent Bovespa three-digit numbers:

Access Category	Market Segment	Sender Location
Desk traders	Equities	BVMF
	Derivatives	BVMF
Orders Conveyor	Equities	REPS
	Derivatives	REPS
DMA1	Equities	DMA1
	Derivatives	DMA1
DMA2	Equities	Provider Code
	Derivatives	Provider Code
DMA3	Equities	DMA3
	Derivatives	DMA3
DMA4	Equities	COLO0/COLO1
	Derivatives	COLO0/COLO1

12 MEMO

To provide a field that participants can use to submit a comment or a description about the current request, most messages in *EntryPoint* has a customized field called *memo* (tag 5149).

This tag is defined as a free-format text field (limited to 40 characters) that may be used to convey the client's relevant information.

The use of the *memo* field (tag 5149) field is convenient because its content is always echoed in the reports. Additionally, as the information might have meaning only to its publisher, the content entered in this field is not visible to the counterparty.

Observe that the scope of *memo* field (tag 5149) is restricted to the Order Entry scenario, which means that the information may be available around the Order Entry and Drop Copy gateways only. The *memo* field (tag 5149) will not reach other systems, such as in the clearing or post-trading areas. In this aspect, it is not recommended to use *memo* field (tag 5149) as a key to correlate messages from the trading with data collected in the post-trading systems, for example.

If the participant needs to have the information reflected outside the scope of the trading environment, it is advised to consider using the *deskID* field (tag 35510)"[4], which is also a free format text field, but which content can be used to add a description or comment to the client's account number and therefore offered to external systems.

[4] Refer to the Participant Identification section, in this document, for more information about the use of *deskID* (tag 35510).

13 CLIENT IDENTIFICATION

13.1 Account Number

Client Identification should be sent in the Account field (tag 1). Exchange replies such as Execution Reports and Order Cancel Rejects will also echo the client identification in the Account field (tag 1).

The *accountType* field (tag 581) acts as a qualifier of the account. For example, setting value of *accountType* to '38' in a modification indicates that the account information should be removed from the order[5]. If *accountType* field is null, the account tag is interpreted as a regular account.

Tag	Tag name	Required	Data Type	Comment
1	account	N	uint32 (4)	Account mnemonic.
581	accountType	N	enum	Type of Account associated with an order. The absence of this Tag indicates it is a Regular Account. Valid values: 38 – Remove Account Information 39 – Regular Account

13.2 Account Annotation

There are scenarios where participants need to include an annotation in the order entry message exclusively to identify the client associated with a given account number. In most cases, this information is used to correlate the order entry messages with the data in the back-office and clearing systems.

In *Binary EntryPoint*, the Account field (tag 1) only accepts numeric values. Therefore, users are advised to take advantage of the *deskID* field (tag 35510) which provides a powerful and consistent method to allow participants to annotate the account.

13.3 Trading Sub-Account Number

The trading sub-account field (*tag 35121*) is a new optional field with the same type as account field (*tag 1*). The initial purpose of the sub-account is, if present, the risk evaluation takes that sub-account instead of the account declared in account field (*tag 1*) that is still used for allocation/settlement purposes in the clearing house.

If the sub-account field (*tag 35121*) is not present (null value), the account field (*tag 1*) is used for the risk evaluation rand allocation/settlement as it has been done before the introduction of the sub-account number concept.

It is important to know that once the sub-account field is declared in order entry messages (*NewOrderSingle*, *NewOrderCross*), it is immutable during the order life cycle: trying to change the value in *OrderCancelReplaceRequest* message will provoke rejection from the B3's risk evaluation system.

[5] This functionality might be unavailable due to Market Rules (e.g., DMA participants cannot remove the account information). Please, refer to section Market Segment Specific Rules for market specific rules regarding account handling.

The sub-account number, once declared in the new entry message (*NewOrderSingle*), needs to be declared in the subsequent modifications with the same value, at the risk of provoking rejection from the B3's risk evaluation system.

14 MARKET-SEGMENT SPECIFIC RULES

The following sections describe market-specific rules.

14.1 BOVESPA Segment (Equities)

14.1.1 Trading Hours

For a list of equities trading hours, sessions, and holidays, please visit www.b3.com.br/en_us/, Solutions, Platforms, PUMA Trading System, Participants and Traders, Trading hours, Equities.

14.1.2 Orders triggering Instrument Freeze (frozen orders)

Orders that trigger an instrument freeze will be accepted by using the "suspended" state. The execution report indicating the suspension should be considered as an acknowledgment of order acceptance, i.e., it has the semantics of a 'New' execution report. When the instrument freeze is lifted, *EntryPoint* will send one or more fill notifications, or a cancellation execution report if the order is rejected as the instrument 'thaws. The following table illustrates the first scenario:

Msg Received	Msg Sent	ordStatus	orderQty	cumQty	Leaves Qty	Last Qty	Comment
New Order			10				Order is sent by the client system.
	ER_New	New	10	0	10		Order is accepted by the exchange, causing an instrument freeze.
	ER_Trade	Partially filled	10	6	4	6	Order is partially filled (quantity executed = 6,000)
	ER_Trade	Filled	10	10	0	4	The order is filled.

Order sent followed by freezing, partial fill, and subsequent thawing of the instrument.

14.2 BM&F Segment (Derivatives)

This section describes specific trading rules for the derivatives market.

14.2.1 Trading Hours

For a list of derivatives trading hours, sessions, and holidays, please visit www.b3.com.br/en_us/, Solutions, Platforms, PUMA Trading System, Participants and Traders, Trading hours, and Derivatives.

14.2.2 Trade Give-Ups

A trade may be given up to another firm, i.e., the firm that carries the position is a firm other than the executing firm (the firm that puts the order in the market). Give-ups may be done post-trade (via the “B3 Services” website).

The give-up indication is done in the order message by providing the account source number in the Account field (tag 1). Once the trade is made, the take-up firm (the firm the trade was given up to) will receive notification and will accept or reject the trade. If the take-up firm rejects the trade, the position is carried by the executing firm.

Example:

Customer “1234” was assigned, at the B3 system, as the source account for entering firm = “GHI” which is linked to a give-up firm “DEF”.

The target account number at firm “DEF” is “9898”. Trader “ABC” is the trader that puts the order in the market.

Msg sent	clOrdID	price	orderQty	account	enteringFirm	enteringTrader
D	ABC1	10.58	7000	1234	GHI	ABC

Since the target account number is not provided in the order message, the pre-trade limit validation is performed on the source account “1234”.

15 ADVANCED FUNCTIONALITIES

15.1 User-Defined Spreads (UDS)

User-Defined Spreads provide users of the electronic trading platform the ability to create strategies composed of their choice of leg instruments, leg ratio, and leg side.

The main purposes of implementing this functionality are:

1. All or Nothing semantics, which guarantees that the participant will get fills on all strategy legs (respecting the leg ratios) or none.
2. Allow customers to create customized spreads and have them immediately active (vs. calling operations staff to create the spread and have them available the next day).
3. Reduce the number of pre-listed instruments since market participants can create spreads on demand, based on actual needs, as opposed to the exchange trying to foresee what instruments are needed.
4. Improves the quality of the instrument listing since spreads without activity are eliminated.

15.1.1 Creation Rules

Constraints are enforced by the system and must be observed during the creation of a UDS, such as:

- Instruments used to create a UDS must reside on the same engine.
- The first listed instrument will be used to determine the engine. In addition, instruments listed in the Blacklist cannot be used as strategy legs.
- The system does not allow user-defined strategies that use the same instrument as different legs.
-

  For example:

  - UDS-1: Buy ACME4; Sell ACME4
  - UDS-2: Buy ACME4; Sell 2 ACME3; Buy ACME4
- The system also does not allow a strategy to be created as the opposite of an existing one. For example:
  - UDS-1: Buy ACME4; Sell ACME3
  - UDS-2: Sell ACME4; Buy ACME3

  In this case, instead of creating UDS-2, the brokerage firm should sell UDS-1 to achieve the same effect.
- All strategies are created from a buyer's perspective. In the previous example, selling UDS-1 means selling ACME4 and buying ACME3, the opposite of the strategy creation.
- The system does not allow strategies with underlying legs of different Tick Sizes.

15.1.2 Expiration Date

A UDS is valid for at least seven days. If no activity occurs during this window, the instrument is deleted.

15.1.3 Security Strategy Types

Several strategies may be structured using the UDS functionality, such as Box, Straddle, and Butterfly. Please refer to **Appendix B:** for examples and a detailed list of strategy types supported by the trading platform.

15.2 Exercise & Blocking

This session will be updated in the future.

15.3 Forward Declaration/Acceptance (“Termo”)

The Forward (also known as “Termo”) Declaration/Acceptance model allows participants to record an out-of-band, pre-arranged deal, in the exchange environment.

The Quote Request FIX message is used within the context of this Forward transaction in which two parties have completed a deal outside the Exchange and are initiating the negotiation process to formalize and execute this operation on the Exchange.

This is done privately between these two counterparties so the Quote Request submitted by the Initiator will be directed to the Respondent.

DMA participants can only initiate the negotiation but cannot be the counterparty to Forward contacts. Desk Traders can either declare or accept Forward deals.

15.3.1 Forward Types

Four types of forward contracts can be entered. The following table describes each one of these contracts:

Type	Description
Common forward	Forward trade to be physically and financially settled at the agreed face value.
Flexible forward	Forward trade that has a specific feature that differentiates it from common forward; the possibility of enabling the forward purchaser to replace the underlying stock of the initially agreed contract.
Dollar forward	Forward trade whose contractual price will be corrected daily by the variation of the average foreign exchange rate of the Brazilian Real (BRL) against the US dollar (USD), as of the trading day to the closing day, excluding first and last.
Index points forward	Forward trade allows the secondary trading of forward contracts, in which the financial settlement amount is calculated by converting the value of the index points into local currency.

15.3.2 Forward + Cash ("Termo Vista")

An alternative modality of a forward contract is the Forward + Cash, also known as "Termo Vista," which is a type of transaction that involves an operation in the forward market with its inclusion on the cash market, inverting the buyer and seller, i.e., the forward buyer becomes the cash seller, and the forward seller becomes the cash buyer.

15.3.3 Forward + Registered Cash ("Termo Vista Registered")

Similarly, Forward + Registered Cash also involves an operation in the cash market. The difference is that, in the Forward + Registered Cash modality, the customer indicates the id of the previous cash trade in the declaration message.

At the end of the forward negotiation, a cash trade is executed inverting the buyer and seller, i.e., the forward buyer becomes the cash seller, and the forward seller becomes the cash buyer. Exactly how it happens in the Forward + Cash modality.

15.3.4 Security Code

Every security allowed to be traded at the exchange forward market has one correspondent non-tradable symbol: e.g., ACME4 has a non-tradable instrument ACME4T ("T" means Termo).

It is not possible to buy or sell it, except by sending and receiving a declaration. The security codes used in the forward market are:

Type of forward	Letter	Example
Common	T	ACME4T
Flexible	S	ACME4S
Dollar forward	D	ACME4D
Index points	T	ACME51T

15.3.5 Instrument States

Forward instruments are in Forward-specific groups. The group schedule has the following states:

- Closed
- Open (scheduled to open simultaneously with the underlying)
- Forbidden

- A declaration can only be accepted if the Forward instrument is in the Open state.

15.3.6 Quote Lifecycle

Every action taken in a Forward deal such as Declaration, Acceptance, Cancellation, Refusal, or Expiration is confirmed by a Quote Status Report (tag 35=AI) message.

The outcome of the operation is expressed in the *quoteStatusReportType* field (tag 35005) which can be New, Accept, Reject, or Expired. The actual status of the Quote is conveyed by the *quoteStatus* field (tag 297) and can assume values such as Pending, Accepted, Canceled, Quote Not Found, Pass, or Expired.

The following diagram depicts the Quote state transition according to the agent that triggered the action and its respective outcome:

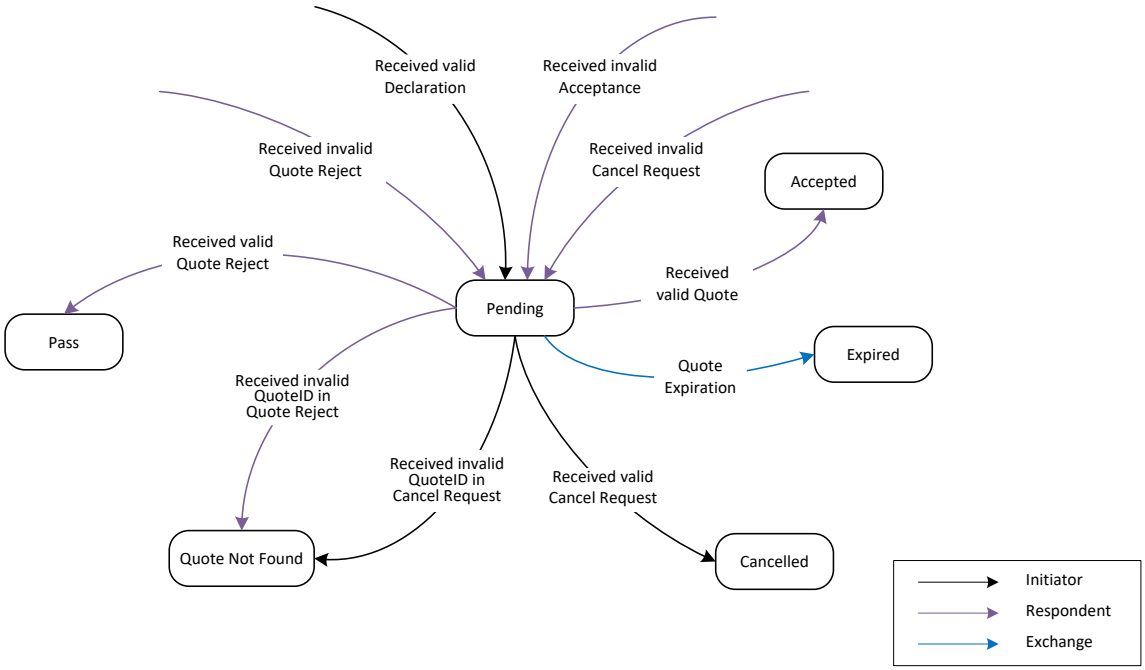


Figure 1 - Quote Status Transitions

15.3.6.1 Declaration

In the Declaration, the Forward deal is entered by the Initiator and received by the counterparty. It contains all the information (e.g., interest rate, price, quantity) that needs to be analyzed by the Respondent. Either buyer or seller can enter the declaration.

15.3.6.2 Cancellation

Before the Forward is accepted by the Respondent, the Initiator can send a cancellation and terminate the deal.

15.3.6.3 Acceptance

In the Acceptance, the Respondent can either accept or reject the Forward deal. After the respondent accepts the Forward deal, B3 generates trade execution reports for the Initiator and Respondent on their respective FIX sessions. To fulfill the deal, the system might generate more than one trade execution report per counterparty.

15.3.6.4 Refusal

If the Respondent does not agree with the terms presented in the Declaration, the Forward can be refused, the Initiator will be notified. The Initiator will need to send a new Declaration to reintroduce the deal.

15.3.6.5 Expiration

By the end of the day, all Declarations that have not been accepted are expired and the Forward deal is automatically terminated by the Exchange.

15.3.7 Contract Details

The following table describes fields used in the FIX messages. These fields represent important concepts in a Forward deal and their meaning are presented below:

Field	Description
PrivateQuote (1171)	When trading a Forward contract, this field must be set as "Y" to specify that this quote is private, i.e., available to a specified counterparty only.
SettlType (63)	Determines who has the power to anticipate the settlement of the deal. Can be Buyer (Regular), Seller, or Mutual. Mutual means agreement between the parties.
DaysToSettlement (5497)	Deadline for completing the Forward deal.
FixedRate (5706)	Interest is to be paid by the forward buyer and received by the forward seller, in proportion to the agreed days to settlement.
ExecuteUnderlyingTrade (35004)	Specifies if a simultaneous trade of the underlying security is to be performed. Used to indicate Termo Vista.

15.4 Self-Trading Prevention

Self-Trading Prevention at customer level is a functionality that aims to restrict matching between buying and selling orders from the same customer, regardless of brokerage firm.

For this purpose, the customer must be identified with a unique Investor ID, included within the order message. The use of this unique identifier is optional, and it is up to the customer to provide this information within the order-entry messages.

Note that Investor ID is not the same as the customer's account number nor is there necessarily a one-to-one relationship mapping between Account and Investor ID values.

Self-Trading Prevention gives the opportunity to choose which order should be canceled when identifying a potential match between an aggressor and a resting order. The options available are: cancel aggressor order, cancel resting order and cancel both orders.

In all cases, the system sends an *ExecutionReport_Cancel* message and provides the reason for the order elimination identified by the value 103 – "Self Trading Prevention" in *execRestatementReason* field (tag 378).

15.4.1 Self-Trading Prevention Instruction

As presented below, *selfTradePreventionInstruction* field (tag 35539) field must be used to indicate which order should be canceled.

Tag	Tag Name	Data Type	Value
35539	SelfTradePreventionInstruction	uint8	0 – None. 1 – Cancel Aggressor Order. 2 – Cancel Resting Order. 3 – Cancel Both Orders.

Self-Trading Prevention rules at the customer level do not apply for auction, match events that trigger s an auction, orders entered on behalf by GSN UDS legs, cross orders, RLP orders, MOC and MOA orders. Also, the usage of Self-Trading

15.4.2 Investor ID

To guarantee the uniqueness of Investor IDs, B3 adopted the following convention to define the customer identifier for each participant:

The Investor ID is composed of 2 parts:

Name	Data Type	Size	Offset	Description
prefix	uint16	2	0	Prefix is a user-defined value and needs to be within pre-defined range. Max value: 999. Max value: 999.
document	uint32	4	4	Document number whose value depends on the class of the investor (corporate resident, individual resident or non-resident investor). Max value: 999999999.

And depends on the type of the participant:

- Brazilian residents and corporate investors must use the 8 leftmost digits of their CNPJ.
- A Brazilian resident and individual investors use the 9 leftmost digits of their CPF as a unique customer identifier.
- Non-resident investors with CVM code (code 99) must use a six-digit code extracted from their individual investor ID. The whole code is formatted as AAAAA.BBBBBB.**CCCCCC**.X-Y, where only the **CCCCCC** part is used. Also, client can use 3-digit prefix before the CVM code in the document free of any pre-defined rules.
- Non-resident investors without CVM code (code 88) must use the eight-digit code extracted from their individual investor ID from the B3 operational code. The whole code is formatted as AAAAAA.**BBBBBBBB**-Y, where only the **BBBBBBBB** part is used. Also, client can use 1-digit prefix before the operational code in the document free of any pre-defined rules.
- Any investor may use a numeric code generated by B3 for the investor, and investors with an alphanumeric document must use a B3-generated code.

The table below presents some examples of Investor IDs:

Investor	Document Type	Document Number	Unique Customer Identifier
Corporate resident	CNPJ	**01.234.567**/0001-23	Prefix part: from **100** to **199**. Document part: base number of the CNPJ, corresponds to the first 8 digits of the investor's CNPJ, without special formatting characters. Examples: • **100, 01234567** • **110, 87654321**
Individual resident	CPF	**012.345.678**-90	Prefix part: from **200** to **299**.

Investor	Document Type	Document Number	Unique Customer Identifier
			Document part: the first 9 digits of the investor's CPF, without special formatting characters. Examples: • **200, 012345678** • **201, 157666432**
Non-resident investor with CVM code (code 99)	CVM operational code	01234.567890.**123456**.7-8	Prefix part: from **300** to **499**. Document part: the individual 6-digit code of the non-resident investor, given by the CCCCCC part of their CVM operational code, whose format is AAAAA.BBBBBB.**CCCCCC**.X-Y. There are **optional 3-digit prefix** before the operational code in the document free of any pre-defined rules. Examples: • **300, 000123456** (without 3-digit optional prefix). • **499, 987654321** (3-digit optional prefix = 987).
Non-resident investor without CVM code (code 88)	B3 operational code	012345.**67890123**-9	Prefix part: from **500** to **899**. Document part: the individual 8-digit code of the non-resident investor, given by the BBBBBBBB part of their B3 operational code, whose format is AAAAAA.**BBBBBBBB**-Y. There is an optional **1-digit prefix** before the operational code in the document free of any pre-defined rules. Examples: • **700, 067890123** (without 1-digit optional prefix). • **899, 987654321** (1-digit optional prefix = 9).
Any type of Investor or Investor with an alphanumeric document	Code generated by B3	123456789	Prefix part: from **900** to **999**. Document part: numeric code generated by B3 for the investor. Participants need to request this code from B3. Examples: • **900, 000123456**. • **999, 987654321**.

Note:

- The ID is not to be confused with the client's account code.

- A client who holds multiple accounts and/or is associated with more than one firm will always use the same ID to identify the offers on which STP should act.
- *InvestorID* field can also be used as criteria for mass cancel on behalf across sessions.
- Clients with alphanumeric documents require a numeric code that will be generated by B3. It is the participant's responsibility to request this code from B3.

15.5 Market Protections

Market Protections are parameters selected by the participants to help them to reduce their risk. These parameters are set for an account inside a firm.

When participants choose to use this feature, the PUMA Trading System begins to monitor a set of cash and/or options instruments or assets configured as references and when a given limit is reached (or exceeded in certain scenarios) the trading platform triggers the Protected Mode, rejecting new messages and canceling the remaining orders for cash and/or options instruments set to be canceled in the configuration, except those which cannot be canceled due to auction rules.

It is important to observe that although this functionality is intended to mitigate potential losses, by establishing a protection threshold, conditions may prevent the correct operation of the feature and there is no guarantee that all resting orders will be successfully canceled.

For example, in case the market or the instrument is in a state that does not allow order cancellations, such as Pre-Close and Close states, orders for that instrument will not be canceled.

For orders that remain in the book due to events that prevent cancellation, the platform will allow modifying of these orders.

In addition, if the user intends to modify any order to set a given firm and account that is already with protection mode enabled, the platform will allow performing this modification.

Moreover, the Market Protections can be activated mid-execution but any previous condition (resting orders, trades, etc.) that happened before the activation will not be included in further monitoring or canceling. In section 15.5.1.2 we present an example where a protection threshold is exceeded.

The Market Protection functionality allows monitoring, for example, only cash instruments and canceling only option orders associated with the monitored instruments, or vice versa.

15.5.1 Protection Types

There are protection types available for market makers to use. The table below summarizes each one:

Protection	Description
New Order Fill Protection	Several orders fill for protected instruments/asset(s), firms, and account during the time interval.
Execution Protection	The number of actual fills, including partial fills, fills for protected instruments/asset(s), firms, and account during the interval.
Traded Quantity Protection	Gross quantity of all instruments traded fills for protected instruments/asset(s), firm, and account during the interval.
Buy/Sell Protection	Net count of buys (+1) and sells (-1) traded for protected instruments/asset(s) during the interval.
Delta Protection	Aggregate (combined + and -) delta values from each execution (Fill) are validated against the specified delta protection value.

As indicated in the table above, a Time Interval is in place for each protection type. The Time Interval only starts after a trade takes place; it does not continuously run throughout the session. If the elapsed time from the Time Interval start is greater than the assigned Time Interval Value, the counters of the enabled protections are automatically reset to zero.

Following, we describe the Market Protections in detail and present examples for each different type.

15.5.1.1 New Order Fill

The customer selects a threshold for the number of new order fills for a protected instrument(s) / asset(s), firm, and account, within a Time Interval.

The Protection calculation is based on the whole order and not the number of executions the order may generate. An entire order that is filled will be shown as one in the New Order Fill Protection count. Buys and asks both increases the New Order Fill counter by '1'.

Example:

Consider the situation described below:

- Based on the customer requests, B3 sets the New Order Fill protection limit = 5 for firm 100 and account 123 when monitoring cash instrument XPTO3 and its options and cash instrument XPTO4 (without considering its options)

- B3 sets the Time Interval = 15 seconds
- A customer has multiple resting orders with account 123.
- No new orders are entered during the 15 seconds time interval.

Within the 15-seconds time interval, the following events occur:

#	Event	Comment
1	A single resting order's bid in XPTO4 is matched in 5 separate executions (partial fills)	New Order Fills counter increments to 1
2	A single resting order's ask in XPTO4 is matched in 3 separate executions	New Order Fills counter increments to 2
3	A single resting order's bid in XPTO3A (option of cash instrument XPTO3) is matched in 1 execution	New Order Fills counter increments to 3
4	A single resting order's bid in XPTOA4 (option of cash instrument XPTO4) is matched in 1 execution	New Order Fills are not incremented. Options of XPTO4 are not being monitored for the specified account
5	A single resting order's ask in XPTO3 is matched in 2 separate executions	New Order Fills counter increments to 4
6	A single resting order's bid in XPTO3 is matched in 5 separate executions	New Order Fills counter increments to 5

As a result:

- Protection mode would be enabled for firm 100 and account 123 after the last match event ends in XPTO3.
- System attempts to cancel all remaining resting orders for account 123 for the monitored instruments and their respective canceling settings (only orders for cash instrument, only orders for options instruments of that cash instruments, or both)
- Any new incoming orders for account 123 related to monitored instruments without reset tag would be rejected.
- The customer will have to send a reset message to begin submitting orders for monitored instruments to that account again.

15.5.1.2 Execution Protection

The customer specifies a threshold for the number of executions, or actual fills for a protected instrument(s) / asset(s), firm, and account, within a Time Interval. As soon as the Execution Protection threshold is met or exceeded, the PUMA platform initiates protection of the customer's orders for the protected account and instruments associated with the protection.

The PUMA platform allows this protection to be exceeded in instances where a single inbound order matches with multiple resting orders. The PUMA Trading System does not stop the match process during a single match event.

Example 1

Consider the situation described below:

- An Execution Protection value of 10 for firm 100 and account 123 when monitoring cash instrument XPTO3 and its options and cash instrument XPTO4 (without considering its options)
- Time Interval Value = 15 seconds
- A customer has multiple resting orders with firm 100 and account 123.
- No new orders are entered during the 15 seconds time interval.

Within the 15-seconds time interval, the following events occur:

#	Event	Comment
1	A single resting order's bid in XPTO4 for a quantity of 500 is matched in 5 separate executions (partial fills)	New Order Fills counter increments to 5
2	A single resting order's ask in XPTO4 for a quantity of 300 is matched in 3 separate executions	New Order Fills counter increments to 8
3	A single resting order's bid in XPTO3A (option of cash instrument XPTO3) for a quantity of 100 is matched in 1 execution	New Order Fills counter increments to 9
4	A single resting order's bid in XPTOA4 (option of cash instrument XPTO4) for a quantity of 100 is matched in 1 execution	New Order Fills counter is not incremented. Options of XPTO4 are not being monitored for the specified firm and account
5	A single resting order's ask in XPTO3 for a quantity of 100 is matched in 1 execution	New Order Fills counter increments to 10

As a result:

- Protection mode would be enabled for firm 100 and account 123 after the last match event ends in XPTO3.
- System attempts to cancel all remaining resting orders for firm 100 and account 123 for the monitored instruments and their respective canceling settings (only orders for cash instrument, only orders for options instruments of cash instruments or both)
- Any new incoming orders for firm 100 and account 123 related to monitored instruments without reset tag would be rejected.
- The customer will have to send a reset message to begin submitting orders for monitored instruments to that account again.

Example 2: Triggered Stop Order Canceled at Protection Mode Activation

Consider the situation described below:

- An Execution Protection value of 10 for firm 100 and account 123 when monitoring cash instrument XPTO3 and its options and cash instrument XPTO4 (without considering its options)
- Time Interval Value = 15 seconds
- A customer has multiple resting orders with firm 100 and account 123, including a stop order for instrument XPTO3 with this account.
- No new orders are entered during the 15 seconds time interval.

Within the 15-seconds time interval, the following events occur:

#	Event	Comment
1	A single resting order's bid in XPTO4 for a quantity of 500 is matched in 5 separate executions (partial fills)	New Order Fills counter increments to 5
2	A single resting order's ask in XPTO4 for a quantity of 300 is matched in 3 separate executions	New Order Fills counter increments to 8
3	A single resting order's bid in XPTO4 for a quantity of 100 is matched in 1 execution	New Order Fills counter increments to 9
4	A single resting order's bid in XPTOA4 (option of cash instrument XPTO4) for a quantity of 100 is matched in 1 execution	New Order Fills counter is not incremented. Options of XPTO4 are not being monitored for the specified account
5	A single resting order's ask in XPTO3 for a quantity of 100 is matched in 1 execution at a price good to trigger the stop order registered for the instrument	New Order Fills counter increments to 10

As a result:

- Protection mode would be enabled for firm 100 and account 123 after the last match event ends in XPTO3.
- System attempts to cancel all remaining resting orders for account 123, including the stop order triggered after the last execution in XPTO3, for the monitored instruments and their respective cancelling settings (only orders for cash instrument, only orders for options instruments of that cash instruments or both).
- Any new incoming orders for firm 100 and account 123 related with monitored instruments without reset tag would be rejected.
- The customer will have to send a reset message to begin submitting orders for monitored instruments to that account again.

15.5.1.3 Traded Quantity Protection

The customer specifies a threshold for the number of traded quantities of contracts related with monitored instruments/assets for a protected firm and account, within a Time Interval. As soon as the Traded Quantity Protection threshold is met or exceeded, the PUMA platform initiates protection of the customer's orders for the protected firm, account and instruments associated with the protection.

The PUMA platform allows this protection to be exceeded in instances where a single inbound order matches with multiple resting orders. The PUMA Trading System does not stop the match process during a single match event.

Example 1

Consider the situation described below:

- A Traded Quantity Protection value of 1000 for firm 100 and account 123 when monitoring cash instrument XPTO3 and its options and cash instrument XPTO4 (without consider its options)
- Time Interval Value = 15 seconds
- A customer has multiple resting orders with account 123.
- No new orders are entered during the 15 seconds time interval.

Within the 15-seconds time interval, the following events occur:

#	Event	Comment
1	A single resting order's bid in XPTO4 for a quantity of 200 is matched in 2 separate executions (partial fills)	Traded Quantity Counter increments to 200
2	A single resting order's ask in XPTO3 for a quantity of 700 is matched in 7 separate executions	Traded Quantity Counter increments to 900
3	A single resting order's bid in XPTOA4 (option of cash instrument XPTO4) for a quantity of 100 is matched in 1 execution	Trade Quantity Counter is not incremented. Options of XPTO4 are not being monitored for the specified account.
4	A single resting order's ask in XPTO4 for a quantity of 100 is matched in 1 execution	Traded Quantity Counter increments to 1000
5	Protection mode is enabled for firm 100, account 123 and monitored instruments of that configuration.	All remaining resting orders for firm 100, account 123 and cancellable instruments of that configuration are canceled

As a result:

- Protection mode would be enabled for firm 100 and account 123 after the last match event ends in XPTO4.

- System attempts to cancel all remaining resting orders for firm 100 and account 123 for the monitored instruments and their respective cancelling settings (only orders for cash instrument, only orders for options instruments of that cash instruments or both)
- Any new incoming orders for firm 100 and account 123 related with monitored instruments without reset tag would be rejected.
- The customer will have to send a reset message to begin submitting orders for monitored instruments to that account again.

Example 2: Iceberg Order Filled and Protection Value Exceeded

Consider the situation described below:

- A Traded Quantity Protection value of 1000 for firm 100 and account 123 when monitoring cash instrument XPTO3 and its options and cash instrument XPTO4 (without consider its options)
- Time Interval Value = 15 seconds
- A customer has multiple resting orders with firm 100 and account 123.
- There is an iceberg order for firm 100 and account 123 of 1000 (50000) shares for the instrument XPTO3.
- No new orders are entered during the 15 seconds time interval.

Within the 15-seconds time interval, the following events occur:

#	Event	Comment
1	A single resting order's bid in XPTO4 for a quantity of 200 is matched in 2 separate executions (partial fills)	Traded Quantity Counter increments to 200
2	A single resting order's ask in XPTO3 for a quantity of 700 is matched in 7 separate executions	Traded Quantity Counter increments to 900
3	A single resting order's bid in XPTOA4 (option of cash instrument XPTO4) for a quantity of 100 is matched in 1 execution	Traded Quantity Counter is not incremented. Options of XPTO4 are not being monitored for the specified account
4	The iceberg order in XPTO3 for a quantity of 10000 (50000) is matched by a single order of 10000 shares in 1 execution	Traded Quantity Counter increments to 10900
5	Protection mode is enabled for firm 100, account 123 and monitored instruments of that configuration.	All remaining resting orders for firm 100, account 123 and cancellable instruments of that configuration are canceled

As a result:

- Protection mode would be enabled for firm 100 and account 123 after the last match event ends in XPTO3.

- System attempts to cancel all remaining resting orders for firm 100 and account 123 for the monitored instruments and their respective cancelling settings (only orders for cash instrument, only orders for options instruments of that cash instruments or both)
- Any new incoming orders for firm 100 and account 123 related with monitored instruments without reset tag would be rejected.
- The customer will have to send a reset message to begin submitting orders for monitored instruments to that account again.

15.5.1.4 Buy/Sell Protection

The PUMA Trading System triggers Buy/Sell Protection when the absolute value of the Buy/Sell Protection parameter is greater than or equal to the value defined by the customer.

The Buy/Sell Protection parameter counts the number of contracts traded related with monitored instruments for a protected account, within a Time Interval.

All instrument types (cash, futures, options) are counted equally.

Example

Consider the situation described below:

- A Buy/Sell Protection value of +/- 1000 (absolute value) for firm 100 and account 123 when monitoring cash instrument XPTO3 and its options and cash instrument XPTO4 (without consider its options)
- Time Interval Value = 15 seconds
- A customer has multiple resting orders with firm 100 and account 123.
- No new orders are entered during the 15 seconds time interval.

Within the 15-seconds time interval, the following events occur:

#	Event	Comment
1	A single resting order's bid in XPTO4 for a quantity of 200 is matched in 2 separate executions (partial fills)	Traded Quantity Counter changes counter to 200
2	A single resting order's ask in XPTO3 for a quantity of 700 is matched in 7 separate executions	Traded Quantity Counter changes counter to -500
3	A single resting order's bid in XPTOA4 (option of cash instrument XPTO4) for a quantity of 100 is matched in 1 execution	Trade Quantity Counter is not changed. Options of XPTO4 are not being monitored for the specified account.
4	A single resting order's ask in XPTO4 for a quantity of 500 is matched in 1 execution	Traded Quantity Counter changes to -1000
5	Protection mode is enabled for firm 100, account 123 and monitored instruments of that configuration.	All remaining resting orders for firm 100, account 123 and cancellable instruments of that configuration are canceled

As a result:

Protection mode would be enabled for firm 100 and account 123 after the last match event ends in XPTO4.

System attempts to cancel all remaining resting orders for firm 100 and account 123 for the monitored instruments and their respective cancelling settings (only orders for cash instrument, only orders for options instruments of that cash instruments or both)

Any new incoming orders for firm 100 and account 123 related with monitored instruments without reset tag would be rejected.

The customer will have to send a reset message to begin submitting orders for monitored instruments to that account again.

15.5.1.5 Delta Protection

Delta measures the rate of change of an option premium concerning a price change in the underlying contract. Delta is a measure of price sensitivity at any given moment.

Not all options move point-for-point with their underlying futures contracts. If a futures contract moves .50 points and the option only moves .25 points, its delta is 50%; i.e., the option is only 50% as sensitive to the movement of underlying futures contract.

The delta will change as an option moves from out-of-the money to at-the-money to in-the-money, approaching 100%. Deltas range from 0% to 100%. The delta of the underlying futures contract is 100%.

The PUMA Trading System triggers Delta Protection when the absolute value of the Delta Protection parameter is greater than or equal to the value defined by the customer.

This protection assumes that all contracts have a defined delta. In absence of a delta, it will be considered as 1.0.

Delta Protection compares a **Delta Counter Value (W)** with a **Delta Static Value**. If the absolute value of W increments or decrements to a value great than or equal to the Delta Static Value within **Time Interval (N**), then the Delta Protection is triggered.

Delta	Description
Delta Static Value	It is a minimum/maximum delta protection value defined by the customer for a set of instruments by firm and account. One value is assumed to be positive and negative, i.e., 300 means +300 and -300 deltas.

Delta Counter Value (W)	Increments and decrements deltas per protected account.
Time Interval (N)	Resets W to zero every N seconds, unless the protection is triggered.

Example:

Consider the situation described below:

- Delta Static Value of +/- 60000 for firm 100 and account 123 when monitoring cash instrument XPTO10 and its options (XPTOA10, XPTOM10, XPTOA11, …)
- Time Interval Value = 15 seconds
- A customer has multiple resting orders with firm 100 and account 123.
- XPTOA10 Call Bid Delta value = +50
- XPTOA10 Call Ask Delta value = -50
- XPTOM10 Put Bid Delta value = - 60
- XPTOM10 Put Ask Delta value = +60
- XPTOA11 Call Bid Delta value = +55
- XPTOA11Call Ask Delta value = -55
- XPTOM11 Put Bid Delta value = - 65
- XPTOM11 Put Ask Delta value = +65
- No new orders are entered during the 15 seconds time interval.

Within the 15-seconds time interval, the following events occur:

#	Event	Comment
1	A single resting order's Call bid in XPTOA10 for a quantity of 1000 is matched in 2 separate executions (partial fills)	Delta Counter Value = 0 + (+50 x 1000) = 50000
2	A single resting order's Call ask in XPTOA10 for a quantity of 200 is matched in 1 execution	Delta Counter Value = 50000 + (-50 x 200) = 40000
3	A single resting order's Put ask in XPTOM11 for a quantity of 400 is matched in 1 execution	Delta Counter Value = 40000 + (+65 x 400) = 66000

4	Protection mode is enabled for firm 100 and account 123.	All remaining resting orders for firm 100, account 123 and cancellable instruments of that configuration are canceled

As a result:

Protection mode would be enabled for firm 100 and account 123 after the last match event ends in XPTOM11.

System attempts to cancel all remaining resting orders for firm 100 and account 123 for the monitored instruments and their respective cancelling settings (only orders for cash instrument, only orders for options instruments of that cash instruments or both)

Any new incoming orders for firm 100 and account 123 related with monitored instruments without reset tag would be rejected.

The customer will have to send a reset message to begin submitting orders for monitored instruments to that account again.

15.5.2 Protection Counters

It is important to observe that the Market Protections counters might diverge in the way they are incremented depending on the type of operation been executed.

15.5.2.1 Counters settings

Different settings of counters can be configured for a protected account.

In a same protection configuration, a list of instruments/assets belonging to a same matching engine can be combined as references for counting and each element of this list can count in a specific way, i.e., each reference can have a different configuration.

For Equities market, it is possible to count only cash instruments, only options instruments related to given cash instruments or both together and for Derivatives market, instead of use a cash instrument as reference, it is possible to use an asset (BGI – “Live cattle,” DOL – “U.S. Dollar”, etc.) as reference.

15.5.2.2 Counters settings restrictions

For a given protected account, when an associated configuration already has an instrument/asset as reference, it cannot be used anymore in the same configuration. i.e., it is not possible to have different counting and cancelling settings for a same reference.

15.5.2.3 Cross Order

A direct operation registered via *NewOrderCross* message will affect the Market Protections counters as if there were two separated executions.

15.5.2.4 User Defined Spreads (UDS)

Executions of User Defined Spreads are not considered for monitoring, i.e., for counting. The same applies for their legs.

15.5.3 Orders cancellation on protection activation

In the same way that for Equities is possible to count only cash instruments, only option instruments related to given cash instruments or both together and for Derivatives market, instead of use a cash instruments as reference, it is possible to use an asset (BGI – “Boi Gordo”, DOL – “Dólar Comercial”, etc.) as reference, it is possible to configure to cancel orders of a protected account only for the instruments configured as references, only for their options or both.

As mentioned before, it is not possible to have different counting and cancelling settings for a same reference.

15.5.3.1 Order types allowed for cancellation.

Besides Limit orders that are always canceled on protection activation, a customer can request to configure the following order types for cancellation on protection activation:

- Good Till orders (GTC/GTD)
- Stop orders.
- MOC orders

15.5.3.2 Configuration update

When an update is performed on a Market Protection configuration for a given firm and account, the monitoring is reset for that firm and account, having the same effect of activating a new configuration where only any previous condition (resting orders, order executions, etc.) that happened before the update will not be considered for monitoring and cancelling.

15.5.4 Automatic Reset

There are specific circumstances where Market Protections values are automatically reset by the PUMA Trading System.

15.5.4.1 Next Trading Session

When trading session is closed for the day, the Market Protections are reset. During the next trading day, the instrument groups will start in the Monitoring Mode accepting new orders normally, even if the Protection Mode was enabled at the end of the previous day.

15.5.4.2 Configuration update

When an update is performed on a Market Protection configuration for a given firm and account, the monitoring is reset for that firm and account, having the same effect of activating a new configuration where only any previous condition (resting orders, order executions, etc) that happened before the update will not be considered for monitoring and cancelling.

15.5.5 FIXP Tags Usage

Binary EntryPoint supports the Market Protections functionality by providing the set of tags and error codes that allow the trading platform to communicate with the clients and inform them about the events triggered by the functionality.

Additionally, the order entry interface allows the clients to reset the *monitoring mode* once they are ready to trade again. Find below the changes made to the order entry interface.

15.5.5.1 Protected Mode

In Protected Mode, the trading platform will cancel remaining orders and prevent the entry of new orders for all instruments associated with the protected group.

After canceling the orders, the trading platform sends *ExecutionReport_Cancel* (35=8) of cancellation to the participant with field: *execRestatementReason* (tag 378) = 200.

Tag	Tag name	Required	Data Type	Comment
378	execRestatementReason	N	uint8	Indicates reason of restatement, if available. **Valid values:** 8 - Market Option 100 - Cancel on Hard Disconnection 101 - Cancel on Logout 102 - Cancel on Disconnect and Logout 103 - Self Trading Prevention 105 – Cancel from Firmsoft 200 - Market Protections 201 – RiskManagementCancel 202 – Mass Cancel from Client Request

15.5.5.2 Resetting Monitoring Mode

Once Market Protections are triggered, the Exchange will not accept new orders of instruments/assets being monitored from that firm and account being protected.

When the client system is ready to re-submit orders, it is necessary to notify the PUMA Trading System to restart the Monitoring Mode by sending *mmProtectionReset* field (tag 9773) = true in the *NewOrderSingle/SimpleNewOrder* messages.

Although modifications of existing orders do not require the use of *mmProtectionReset* field (tag 9773), one may include this tag in the OrderCancelReplaceRequest/*SimpleModifyOrder* messages to reset the monitoring mode.

This tag makes the platform to accept new orders for the protected group again.

Tag	Tag name	Required	Data Type	Comment
9773	mmProtectionReset	Y	Boolean	Resets the Market Protections.

15.5.5.3 Rejection Message

After Market Protection is triggered, any new order for the monitored instruments/assets for an account being protected, without *mmProtectionReset* field (tag 9773) = Y, will be refused with an *ExecutionReport_Reject* message.

Fields *ordRejReason* (tag 103) = 2600 and *text* (tag 58) = "`Market Protection in effect for configuration ID <ID> with product <PRODUCT> of participant <FIRM>/<ACCOUNT>`" will help to identify the cause of rejection.

15.6 Implied Order

The Implied functionality allows the interaction of orders from individual books of a product (*outrights*) with orders from spreads/strategies of that same product generating orders synthetically.

An implied order can be created from orders from outright books in a strategy book (*Implied In*) or from orders from an outright book and orders from a spread/strategy book in an outright book (*Implied Out*).

<u>Implied orders have these main characteristics:</u>

- Implied orders only trade against regular orders
- They will have lower priority than the regular orders at the same price.
- Implied orders are not available when the spread/strategy and its legs are not in a matching state.

- Self-trade Prevention is not applied to implied orders.
- If there is a partial execution in one of the orders that generated the implied order, the implied is updated with the new calculated quantity.

Instruments that are implied eligible have the *impliedMarketIndicator* field (tag 1144) enabled (value 1) in the *SecurityDefinition* message (*Binary UMDF*).

15.6.1 Implied In orders

These orders are derived from existing outright orders in individual contracts (legs). It creates a synthetic order at the spread/strategy book.

This means that an regular order in a spread/strategy can be matched with other regular orders in the spread/strategy OR with a combination of orders in the legs of the spread/strategy based on price and the rules corresponding to the method of allocation being used.

Example:

Moment #1: Client X inserts a buy order with 10 @ 6.99 in DI1F24:

DI FUT – DI1F24 (maturity: Jan/24)			
Bid Qtd	Bid Price	Offer Price	Offer Qtd
10	6.99 (A)		

Moment #2: Client Y inserts a sell order with 10 @ 7.34 in DI1N24:

DI FUT – DI1N24 (maturity: Jul/24)			
Bid Qtd	Bid Price	Offer Price	Offer Qtd
		7.34 (B)	10

Moment #3: the combination of 2 orders above generates an implied (IN) order in offer book with 10 @ 0.35 in DIIF24N24 (spread of maturities of F24 and N24):

Spread of Sell DI1F24 and Buy DI1N24 (DIIF24N24)			
Bid Qtd	Bid Price	Offer Price	Offer Qtd
		0.35 (B - A)	10

Moment #4: Client Z inserts a sell order with 15 @ 0.35 in DIIF24N24:

Message	Instrument	MDEntry Type	MDUpdate Action	MDEntry Px	MDEntry Size	MDEntryPosition No	MatchEvent Indicator
Order_MBO	DIIF24N24	1 (Offer)	0 (NEW)	0.35	15	1	0

The resulted book will be:

Spread of Sell DI1F24 and Buy DI1N24 (DIIF24N24)			
Bid Qtd	Bid Price	Offer Price	Offer Qtd
		0.35 (Normal)	15

Spread of Sell DI1F24 and Buy DI1N24 (DIIF24N24)			
		0.35 (Implied)	10

15.6.2 Implied Out orders

These orders are derived from a combination of an existing regular order in a spread/strategy book and an existing outright order in one of the underlying individual legs. It creates a synthetic order on the other underlying leg book.

This means that a regular order in a leg can be matched with other regular orders for this specific leg OR with a combination of orders from any spread/strategy composed of this leg and orders of the other corresponding leg of the spread. **This feature is not yet available and the availability will be announced through appropriate channels in time.**

15.6.3 Matching involving an implied order

The **trades** and **trade busts** related to the same implied order event (spread/strategy and its legs) will have the identification "7" (Implied Order) at *orderCategory* field (tag 1115) and they will have the same value in *impliedEventID* field (tag 35540) in *ExecutionReport_Trade* message.

One aspect to be highlighted is that all *ExecutionReport_Trade* messages related to the legs of the match that involves an implied order has always the "FILLED" value in the *ordStatus* field because of internal algorithm used to match real orders for each leg that composes that spread/strategy.

Examples:

1) EDS DI1 – DV01 Neutral – Implied In:

Strategy type	DI1 - DV01 Neutral
Strategy symbol	DIIF25F27
Strategy round lot	5
Short leg	DI1F25
Short leg ratio	- 2.779610
Short leg round lot	1
Long leg	DI1F27
Long leg round lot1	1

Sequence of messages:

#	Message	Symbol	Side	Price	Qty	Firm
1	New Order	DI1F25	Sell	13.716	20	A
2	New Order	DIIF25F27	Sell	0.384	5	C
3	New Order	DI1F27	Buy	14.100	5	B

- An implied order is created in the DIIF25F27 book with:
  - Price = long leg price – short leg price = 0,384.
  - Qty = *min* {*round down* [short leg qty / absolute (ratio)]; long leg qty} = *min* [*round down*(20/2.779610); 5] = 5.

Book DI1F25							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
				13.716	20	A	1

Book DI1F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
3	B	5	14.100				

Book DIIF25F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
-	-	***5$_i$***	***0.384***	0.384	5	C	2

- The matching occurs:

Symbol	Qty	Preço	#Buy	Buy Firm	#Sell	Sell Firm
DIIF25F27	**5**	**0.384**	**(Implied)**	**(Implied)**	**2**	**C**
DI1F25	**14**	**13.716**	**2'**	**C**	**1**	**A**

Symbol	Qty	Preço	#Buy	Buy Firm	#Sell	Sell Firm
DI1F27	**5**	**14.100**	**3**	**B**	**2'**	**C**

- Execution Report messages:

Symbol	OrderID	LastQty	LastPx	Side	Entering Firm	Order Category	MultiLeg ReportingType	Implied EventID
DIIF25F27	2	5	0.384	Sell	C	Implied	Multi Leg Security	12345678900003
DI1F25	2'	14	13.716	Buy	C	Implied	Individual Leg of a Multi Leg security	12345678900003
DI1F25	1	14	13.716	Sell	A	Implied	-	12345678900003
DI1F27	3	5	14.100	Buy	B	Implied	-	12345678900003
DI1F27	2'	5	14.100	Sell	C	Implied	Individual Leg of a Multi Leg security	12345678900003

- Books after the matching:

Book DI1F25							
BID				**ASK**			
#	**Firm**	**Qty**	**Price**	**Price**	**Qty**	**Firm**	**#**
				13.716	6	A	1

Book DI1F27							
BID				**ASK**			
#	**Firm**	**Qty**	**Price**	**Price**	**Qty**	**Firm**	**#**

Book DIIF25F27							
BID				**ASK**			
#	**Firm**	**Qty**	**Price**	**Price**	**Qty**	**Firm**	**#**

Book DIIF25F27							

2) EDS DI1 – DV01 Neutral – Implied In – with Iceberg order:

Strategy type	DI1 - DV01 Neutral
Strategy symbol	DIIF25F27
Strategy round lot	5
Short leg	DI1F25
Short leg ratio	-2.779610
Short leg round lot	1
Long leg	DI1F27
Long leg round lot1	1

Initial Books:

Book DI1F25							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
10	C	50	13.714	13.716	20/100 (iceberg)	A	1
				13.716	10	B	6
				13.717	20	D	3

Book DI1F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#

Book DIIF25F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
12	F	10	0.382	0.385	10	C	13

New messages:

#	Message	Symbol	Side	Price	Qty	Firm
21	New Order	DI1F27	Buy	14.100	12	B

- An implied order is created in the DIIF25F27 book with:
  - Price = long leg price – short leg price = 0.384.
  - Qty = *min* {*round down* [short leg qty / *absolute* (ratio)]; long leg qty} = *min* [*round down* (30 / 2.779610); 12] = 10.

Book DI1F25							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
10	C	50	13.714	13.716	20/100 (iceberg)	A	1
				13.716	10	B	6
				13.717	20	D	3

Book DI1F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
21	B	12	14.100				

Book DIIF25F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
-	-	*10$_i$*	*0.384*	0.385	10	C	13
12	F	10	0.382				

New messages:

#	Message	symbol	Side	Price	Qty	Firm
22	New Order	DIIF25F27	Sell	0.384	10	C

⇨ The implied order is aggressed by the new order:

Book DI1F25							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
10	C	50	13.714	13.716	20/100 (iceberg)	A	1
				13.716	10	B	6
				13.717	20	D	3

Book DI1F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
21	B	12	14.100				

Book DIIF25F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
-	-	*10$_i$*	*0.384*	0.384	10	C	22
12	F	10	0.382	0.385	10	C	13

- The trades are generated:

Symbol	Qty	Preço	#Buy	Buy Firm	#Sell	Sell Firm
DIIF25F27	10	0.384	(Implied)	(Implied)	22	C
DI1F25	20	13.716	22'	C	1	A
DI1F25	8	13.716	22'	C	6	B
DI1F27	10	14.100	21	B	22'	C

- Execution Report messages:

Symbol	OrderID	LastQty	LastPx	Side	Entering Firm	Order Category	MultiLeg ReportingType	Implied EventID
DIIF25F27	22	10	0.384	Sell	C	Implied	Multi Leg Security	12345678900003
DI1F25	22'	20	13.716	Buy	C	Implied	Individual Leg of a Multi Leg security	12345678900003
DI1F25	1	20	13.716	Sell	A	Implied	-	12345678900003
DI1F25	22'	8	13.716	Buy	C	Implied	Individual Leg of a Multi Leg security	12345678900003
DI1F25	6	8	13.716	Sell	B	Implied	-	12345678900003
DI1F27	21	10	14.100	Buy	B	Implied	-	12345678900003
DI1F27	22'	10	14.100	Sell	C	Implied	Individual Leg of a Multi Leg security	12345678900003

- Books after the matching:

Book DI1F25							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
10	C	50	13.714	13.716	2	B	6
				13.716	20/80 (iceberg)	A	1
				13.717	20	D	3

Book DI1F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
21	B	2	0.384				

Book DIIF25F27							
BID				ASK			
#	Firm	Qty	Price	Price	Qty	Firm	#
12	F	10	0.382	0.385	10	C	13

⇨ No implied is created because the quantity (2) is less than the spread/strategy round lot (5).

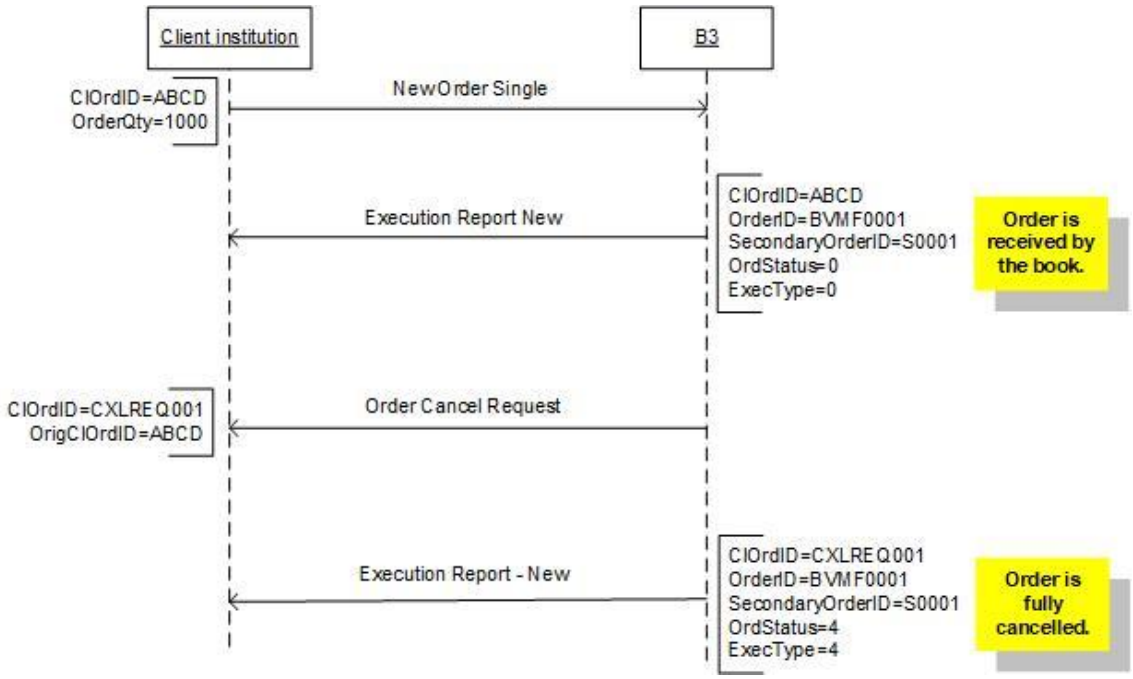
16 APPLICATION MESSAGE SCENARIOS

The following sections provide examples of the most common application message scenarios. In all scenarios, if a message is malformed or fails specific business level conditions, it will be rejected with *BusinessMessageReject* message (e.g., *enum* field out of range).

16.1 Order management scenarios

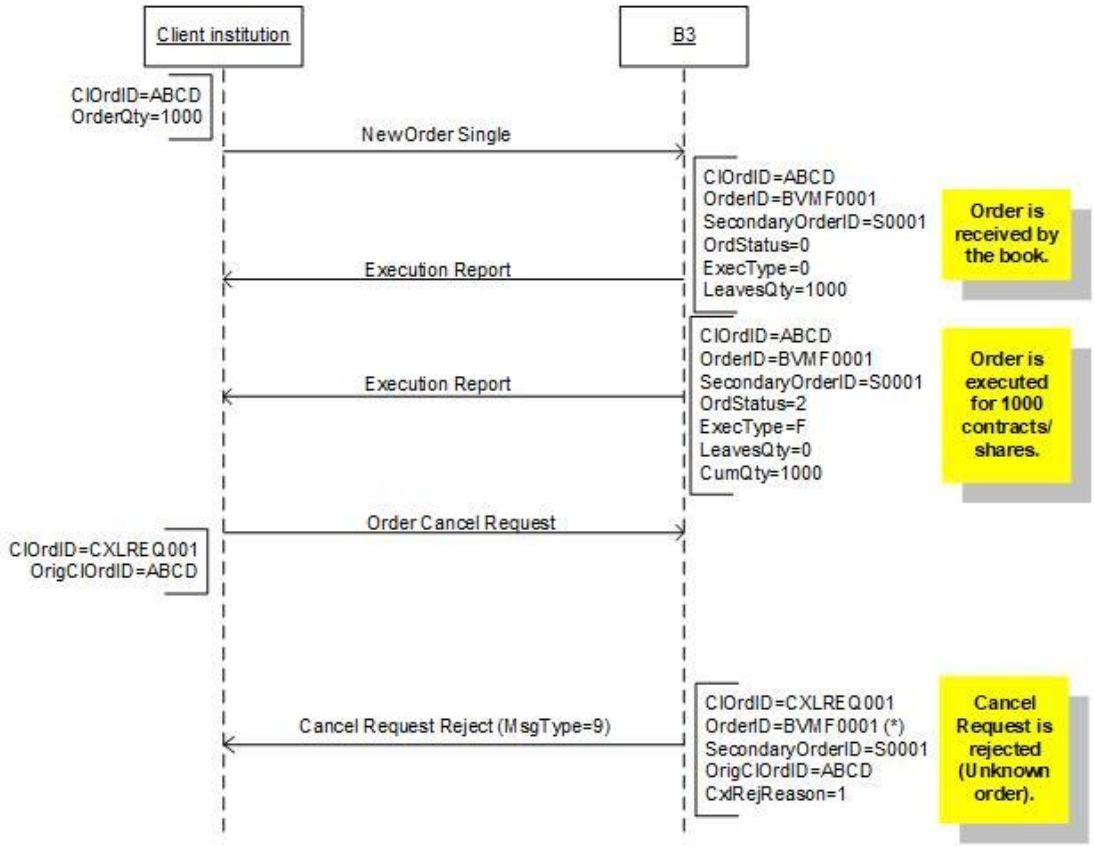
16.1.1 Order Cancellation by *clOrdID*

In this example, the client institution issues an order, and cancels it afterwards referring to its *clOrdID* field. The value of *clOrdID* field was generated by the issuer of the order and must be unique for that FIXP session and instrument. B3 correlates the value of *clOrdID* field issued by the client with its internal order identification per instrument, sent to the client in the *orderID* field in the *ExecutionReport_Cancel* message.



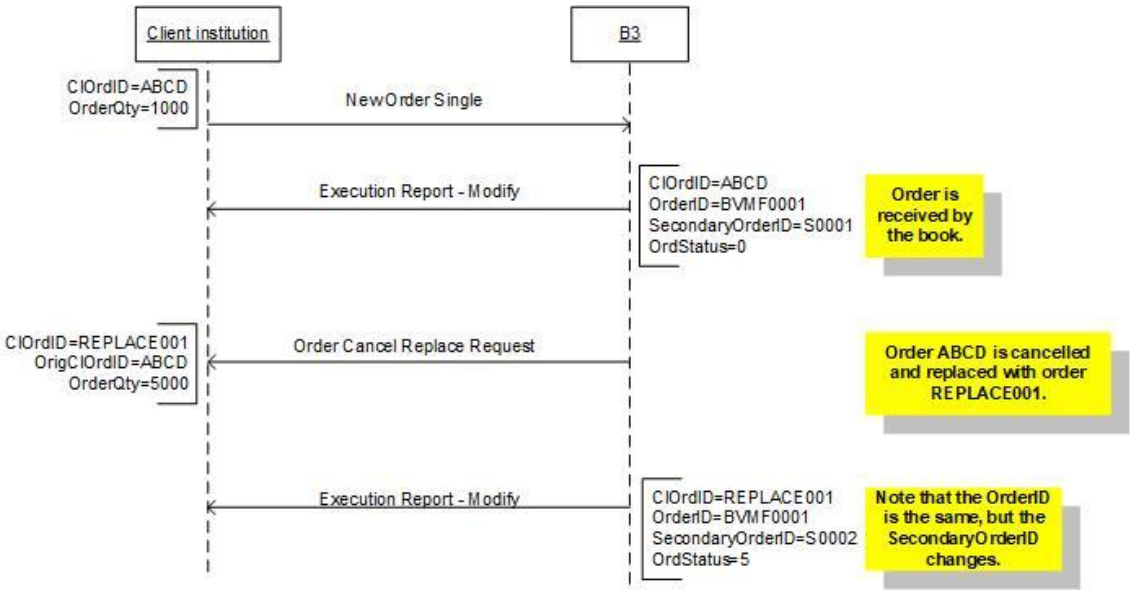
16.1.2 Order Cancellation Attempt of Filled Order

In this example, the client issues a new order, this order is filled, and the client attempts to cancel the filled order. The cancel request will be rejected.



16.1.3 Order Modification

This example illustrates the modification of an order issued by the client. Notice that an order that is modified keeps the B3 order ID (*orderID* field) of the canceled order.



16.1.4 Order Mass Action

16.1.4.1 Request

This is the message which is sent by client systems for a specified matching engine in the *marketSegmentID* field (tag 1300) or by order entry gateway to each matching engine instance when "Cancel on Disconnect" mechanism is triggered.

There are also several filters to be used to select which orders belonged to the session:

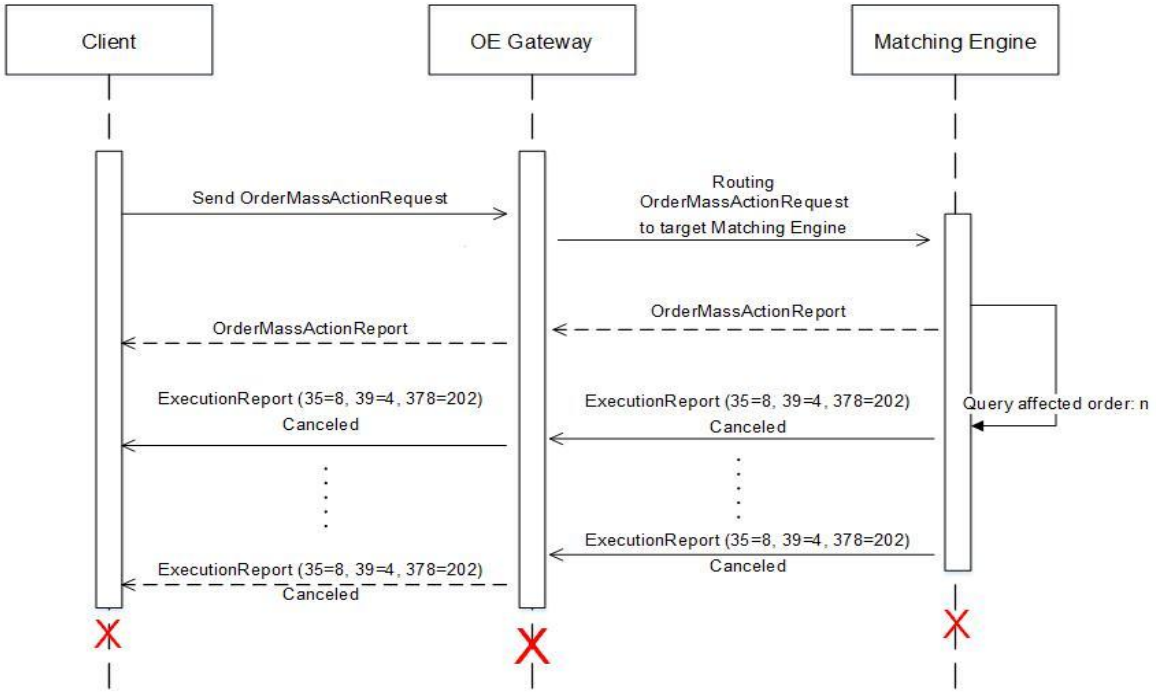
- By ***OrdTagID***: cancel all orders that were previously tagged with the informed value.
- By ***Side***: specify if buy or sell orders are desired to be canceled.
- By ***SecurityID***: specify the desired instrument whose orders are to be canceled.

The Order Mass Action Request will not result in the cancellation of GTC, GTD and MOA orders participating in TOP. Only Day, MOC and MOA orders not participating in TOP will be canceled. Also, the Order Mass Action Request is both an external (from client) and internal system message between the OE Gateway and matching engine. Only requests sent by clients will generate response.

16.1.4.2 Report

This is the message which is sent by each matching engine instance to the order entry gateway in response to an Order Mass Action Request when "Cancel on Disconnect" is triggered or sent by the client. This message is both an external and internal system message and will be sent back to the customer in order entry sessions. The Order Mass Action Request will always be acknowledged positively by the matching engine except when the request itself is malformed.

One behavior that is important to clarity is that the value of *clOrdID* field (tag 11) from execution reports that informs canceled orders will not have the same value of *clOrdID* field (tag 11) of the mass order action request that triggered those cancellations, so clients cannot correlate them accordingly, but they will know that those orders were canceled by Mass Order Cancel triggered by an *OrderMassActionRequest* message because those *ExecutionReport_Cancel* messages will have the *execRestatementReason* field (tag 378) = 202 (Order Mass Action From Client Request).



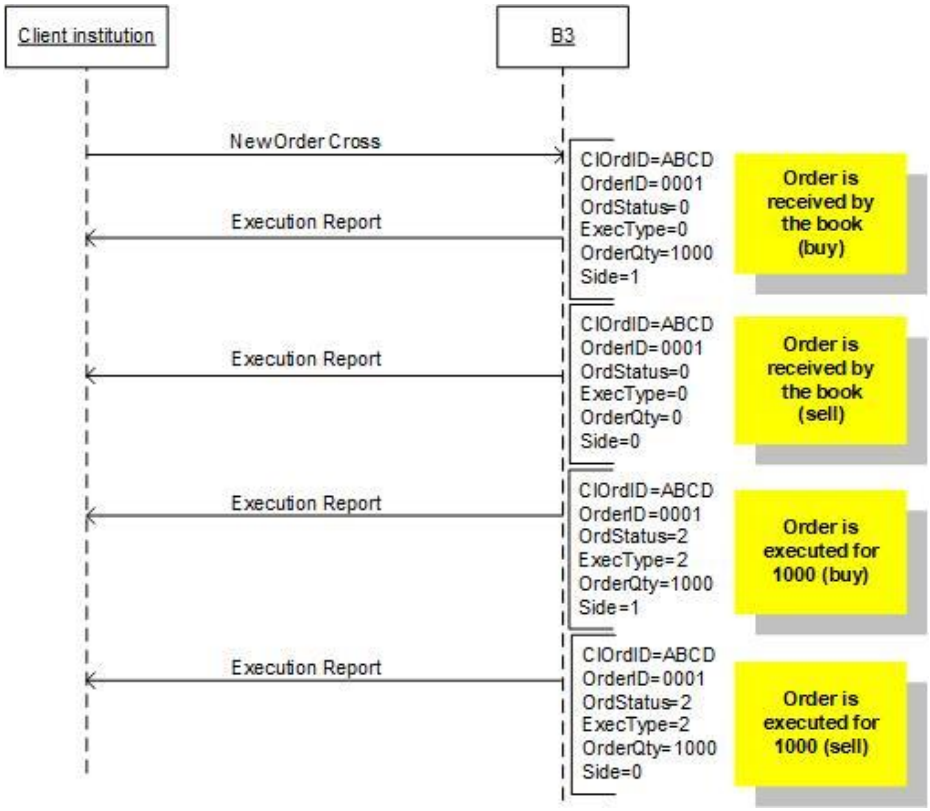
16.1.5 Cross Order

The New Cross Order message is used by institutions to electronically submit orders to buy and sell the same security for different investors through the register of direct operation in the trading system.

In *EntryPoint*, the use of cross orders is available not only to desk traders, but to all participants, independently the type of access used to connect to the Exchange. Such scenario must be evaluated during the customer's system certification process.

The acknowledgment of receipt of a New Cross Order message is issued by B3 in the form of two Execution Report - New messages. The order may be accepted (*ExecutionReport_New*) or rejected (*ExecutionReport_Reject*) according to B3 rules.

If the cross trading meets any of parameters determined for cross trade auctions, the security will be submitted to a regular auction. If there are any valid asks at better prices (buying or selling) the cross order will be rejected.



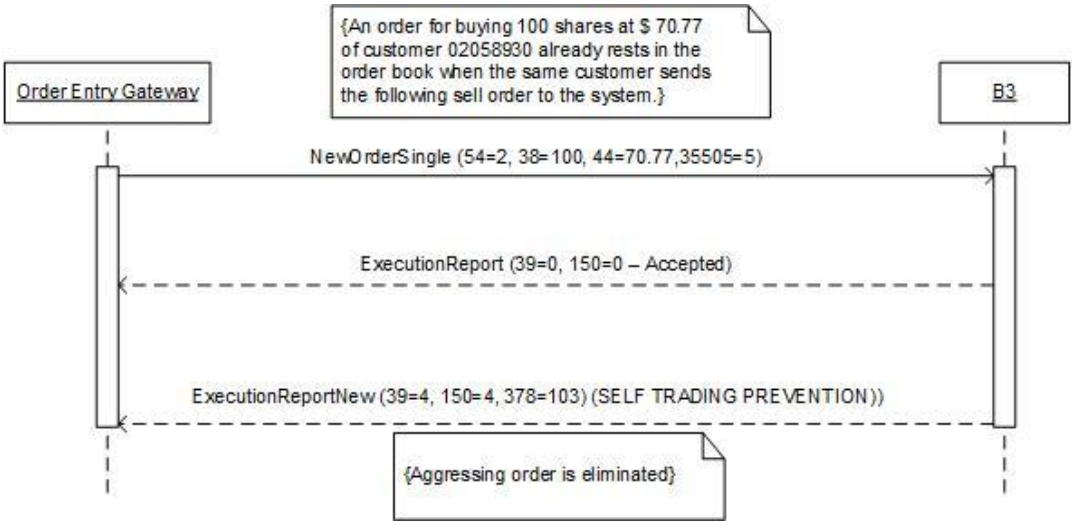
16.2 Self-Trading prevention scenarios

16.2.1 Self-Trading prevention on aggressing order

In this scenario, the customer already has an order in the book tagged with their unique Investor ID and the new order that is being placed can potentially match with the one in the book.

In this case, the new order is accepted and then canceled upon entry. *ordStatus* field (tag 39) in the *ExecutionReport_Cancel* message sent to the participant indicates that the order has been canceled (tag 39 = 4 - Canceled) and *execRestatementReason* field (tag 378) provides a self-explanatory reason for the elimination (tag 378 = 103 - Self-Trading Prevention).

Note that the system does not run Self-Trading validations at customer level for orders with null value for *investorID* field (tag 35504).



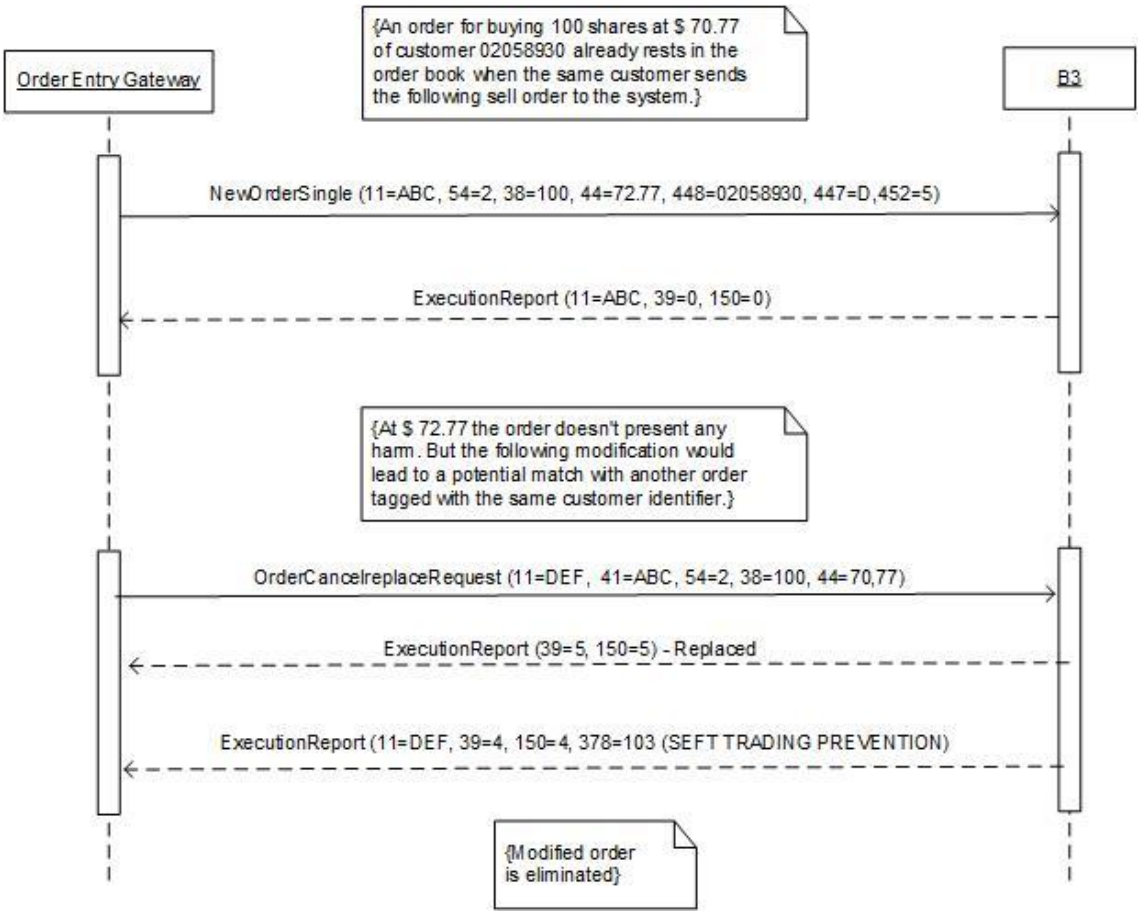
The following table shows the sequence of messages received and sent by the Exchange and sample values are assigned to key fields to demonstrate their usage:

	Message Sent	Message Received	ordType	investorID	orderQty	price	ordStatus	exec RestatementReason	Comment
1	NewOrder		2	2058930	100	70.77	--	--	
2		ER_New	2	2058930	100	70.77	0	--	New
3		ER_Cancel	2	2058930	100	70.77	4	103	Canceled

16.2.2 Self-Trading prevention on Order Modification

When an order modification leads to a potential match with another order, tagged with the same Investor ID, the modification will be accepted but it will be followed by an immediate elimination of the order.

ordStatus field (tag 39) in the *ExecutionReport_Cancel* message sent to the participant indicates that the order has been canceled (39 = 4 - Canceled) and *execRestatementReason* field (tag 378) provides a self-explanatory reason for the elimination (378 = 103 - Self-Trading Prevention).



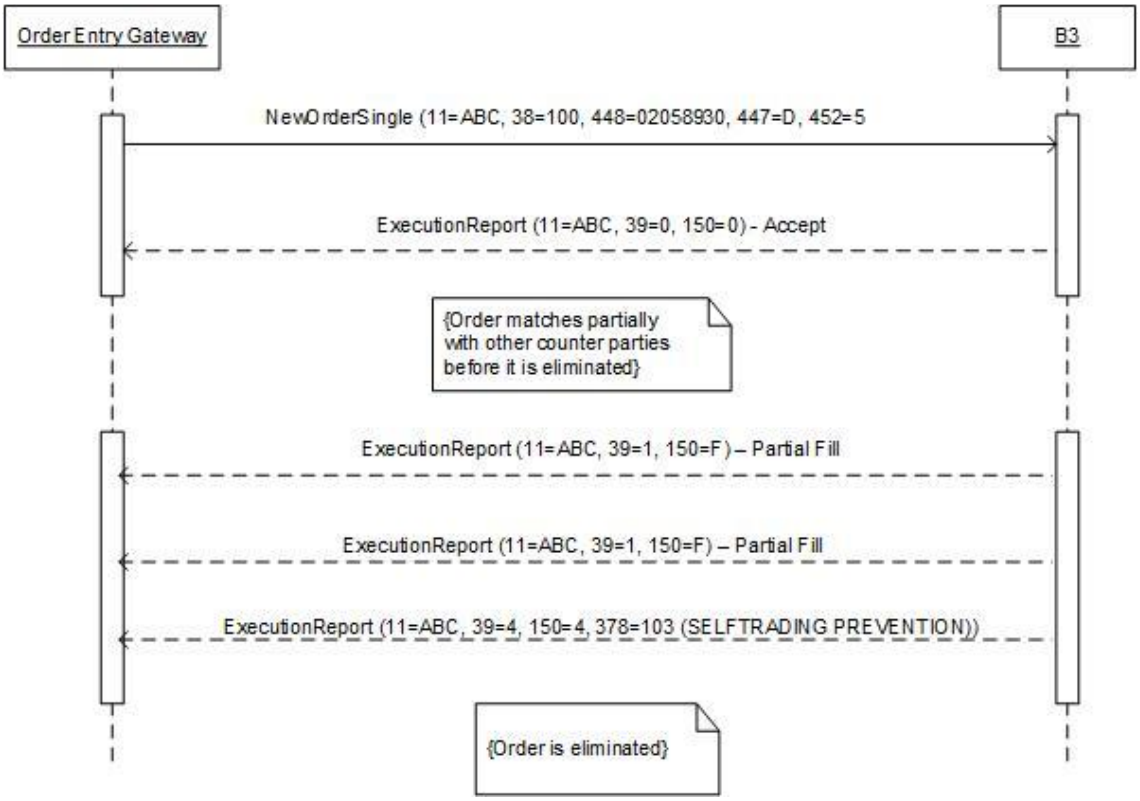
The following table shows the sequence of messages received and sent by the Exchange and sample values are assigned to key fields to demonstrate their usage:

Message Sent	Message Received	clOrdID	ord Type	investorID	order Qty	price	ord Status	exec Restatement Reason	Comment
New Order		202203001	2	2058930	100	72.77	--	--	
	ER_New	202203001	2	2058930	100	72.77	0	--	New
Modify Order		202203022	2	2058930	100	70.77	--	--	
	ER_Modify	202203022	2	2058930	100	70.77	5	--	Replaced
	ER_Cancel	202203022	2	2058930	100	70.77	4	103	Canceled

16.2.3 Self-Trading prevention and Partial Fills

This scenario presents a situation in which the order is partially executed in 200 shares and the remaining amount of 800 is eliminated because the next aggressed order has the same unique Investor ID.

ordStatus field (tag 39) in the *ExecutionReport_Cancel* message sent to the participant indicates that the order has been canceled (39 = 4 - Canceled) and *execRestatementReason* field (tag 378) provides a self-explanatory reason for the elimination (378 = 103 - Self-Trading Prevention).



For incoming Minimum quantity orders or FOK orders, the match engine analyzes the book to assure the minimum required quantities can be achieved without self-trade. Otherwise, the incoming (aggressing) order must be eliminated upon entry.

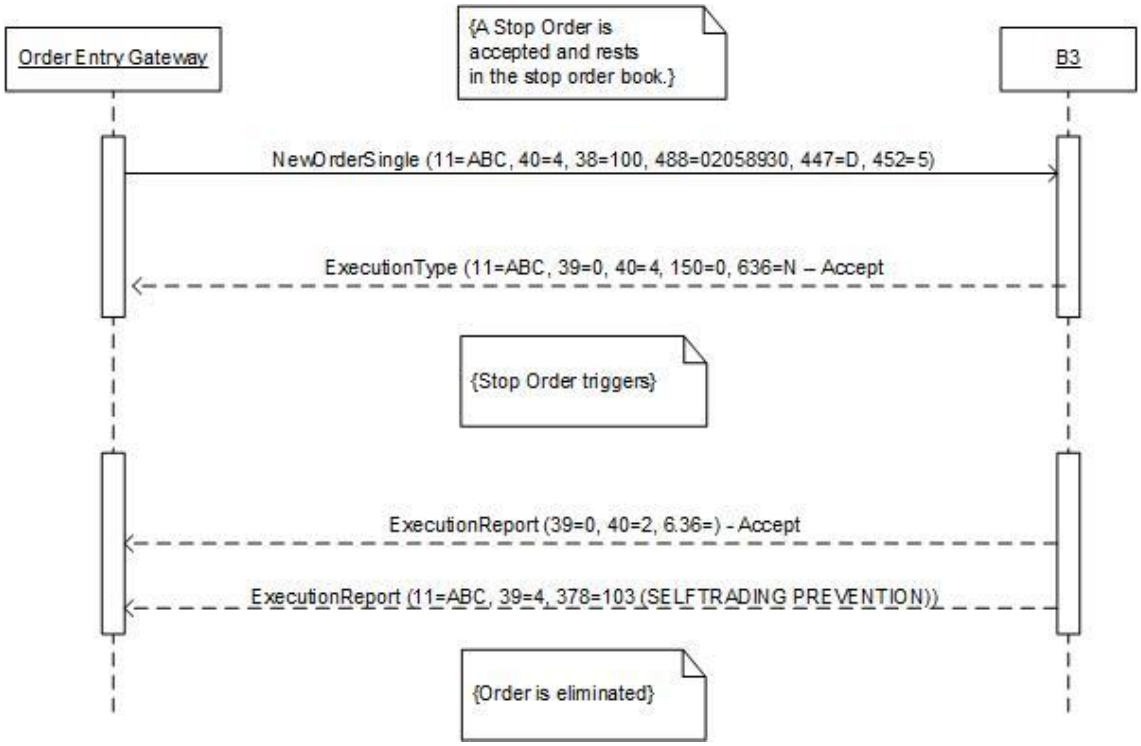
The following table shows the sequence of messages received and sent by the Exchange and sample values are assigned to key fields to demonstrate their usage:

Msg sent	Msg received	clOrdID	investorID	order Qty	last Qty	leaves Qty	cum Qty	ord Status	exec Restatement Reason	Comment
New Order		20230301	2058930	1000	--	--	--	--	--	
	ER_New	20230301	2058930	1000	--	1000	--	0	--	New
	ER_Trade	20230301	2058930	1000	100	900	100	1	--	Partial Fill
	ER_Trade	20230301	2058930	1000	100	800	200	1	--	Partial Fill
	ER_Cancel	20230301	2058930	1000	--	800	200	4	103	Canceled

16.2.4 Self-Trading prevention on stop orders

In the following scenario, a Stop order becomes a Limit order, and it is tagged with the same Investor ID as another order in the book. If such condition leads to a self-trade, the triggered order will be immediately eliminated.

WorkingIndicator field (tag 636) points when the order becomes active and available for trading. Upon activation, the *ordType* field (tag 40) changes from “Stop Limit” (40 = 4) to “Limit” (40 = 2) and the order is eliminated by the Self-Trading Prevention functionality.



16.3 Market Protections

In the following examples, we assume that instruments XPTO1, XPTO2 and XPTO3 do not need be part of same product group. Additionally, consider that the Market Protections have already been triggered for a firm’s protected account due an action on instrument XPTO1.

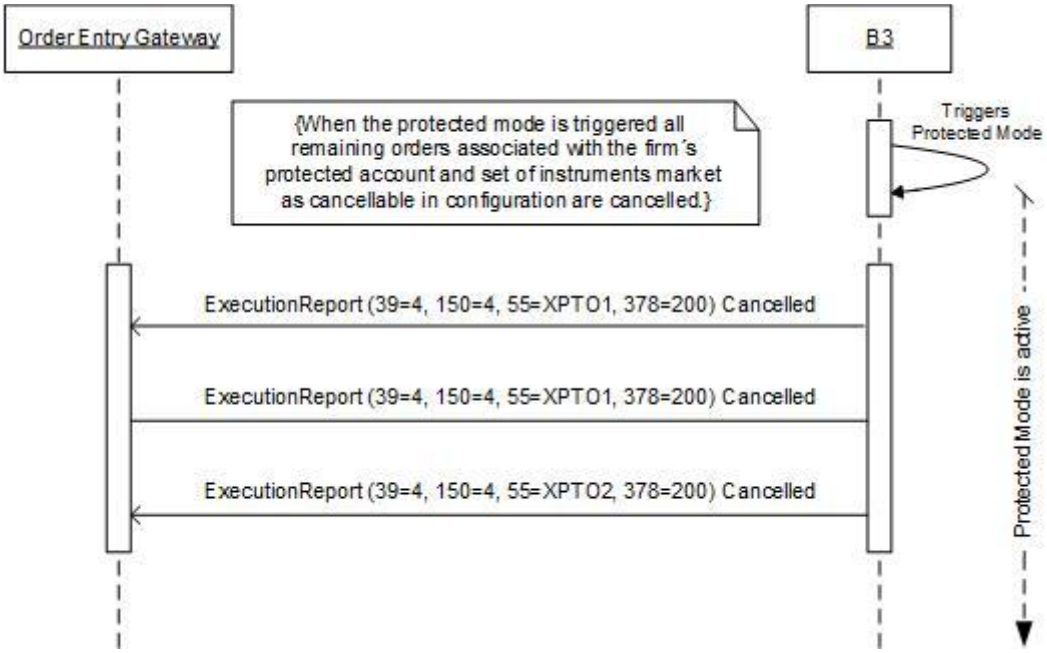
In such scenario, the system will attempt to cancel all resting orders associated with instruments XPTO1, XPTO2 and XPTO3. After that, the customer will need to reset the protection before new orders can be accepted for those instruments again.

16.3.1 Protected Mode

The Protected Mode is triggered for a set of instruments where the protection threshold is reached or, in cases, exceeded.

16.3.1.1 Automatic Order Cancellation

When the protected mode is triggered, all remaining orders associated with the firm's protected account and set of instruments market as cancellable in configuration are canceled, except orders related to instruments that are in a state that does not allow order cancellations.

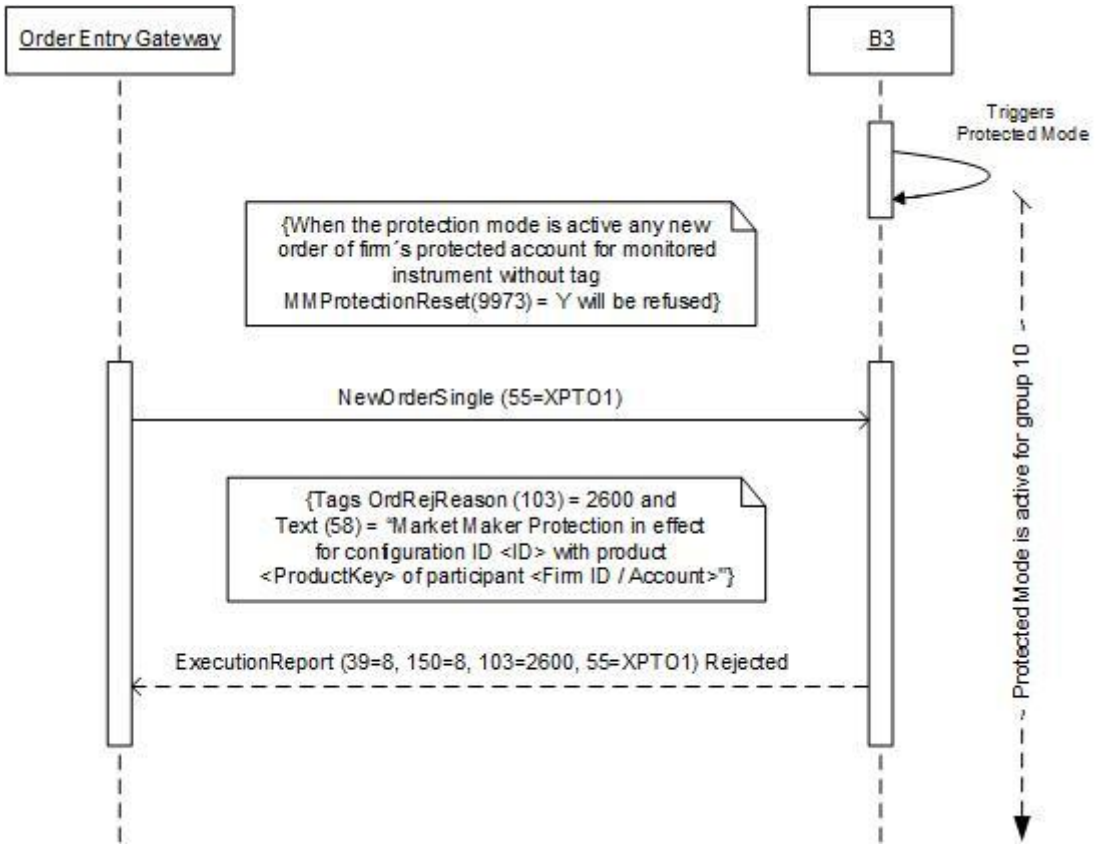


The events are described in the table below:

#	Description
1	The Protected Mode is triggered by instrument XPTO1
2	System attempts to cancel all remaining resting orders from configuration related with XPTO1, i.e., it will cancel remaining orders for XPTO2 and XPTO3 if any order exists.

16.3.1.2 Rejection Message

In Protected Mode, the trading platform will prevent the entry of new orders for any instrument associated with the firm's protected account and set of instruments market as cancellable in configuration.



The events are described in the table below:

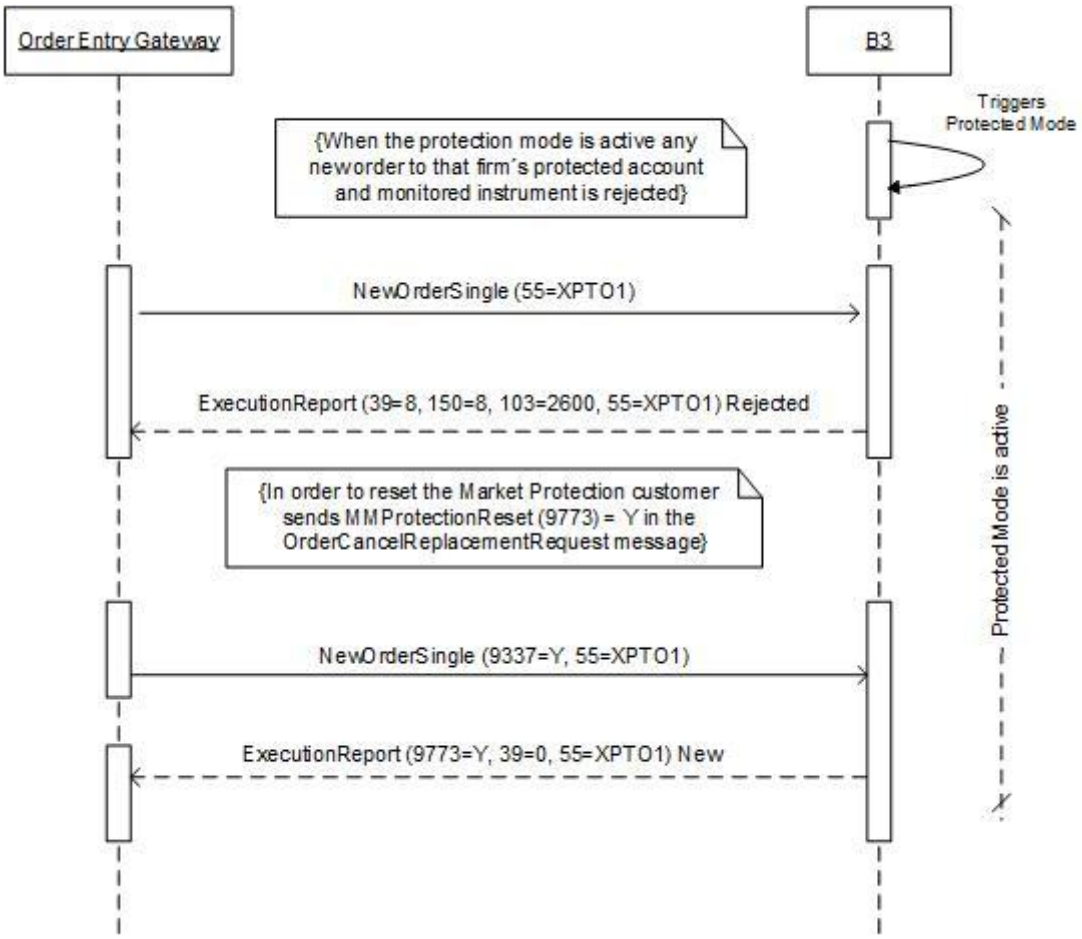
#	Description
1	Market Protections have already been triggered by instrument XPTO1
2	Customer sends a New Order Single (35=D) without tag (9773=Y)
3	Since a given firm, account, and instrument (XPTO1) is running in Protected Mode, the new order is rejected

16.3.2 Resetting Monitoring Mode

To reset the protection, the customer must send *mmProtectionReset* field (tag 9773) = true in either a *NewOrderSingle* message or *OrderCancelReplaceRequest* message. The following examples depict these two scenarios.

16.3.2.1 Sending MMProtectionReset in NewOrderSingle

In this example, the customer resets the protection by sending a *NewOrderSingle* message with *mmProtectionReset* field (tag 9773) = true.

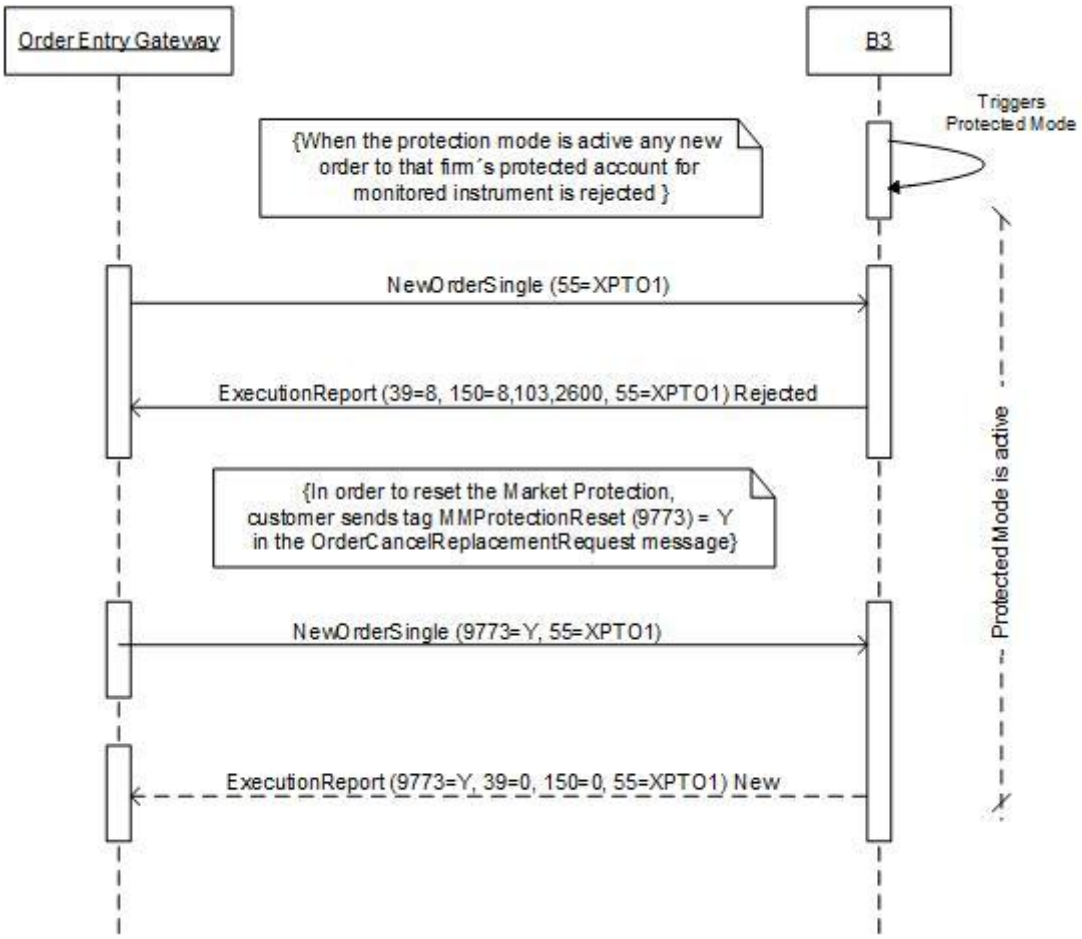


The events are described in the table below:

#	Description
1	Market Protections have already been triggered by instrument XPTO1
2	Customer sends a New Order Single (35=D) without tag (9773=Y)
3	Since the firm, account and instrument provided is running in Protected Mode, the new order is rejected
4	Customer sends a New Order Single (35=D) with the specific tag (9773=Y) to notify the trading platform to leave the Protection Mode and reset the Monitoring Mode
5	Order is registered
6	New orders for firm's protected account and monitored instruments start being accepted again

16.3.2.2 Sending MMProtectionReset in OrderCancelReplaceRequest

Alternatively, customer may include *mmProtectionReset* field (tag 9773) = true in an *OrderCancelReplaceRequest* message sent to modify an existing order of a firm's protected account and monitored instrument.

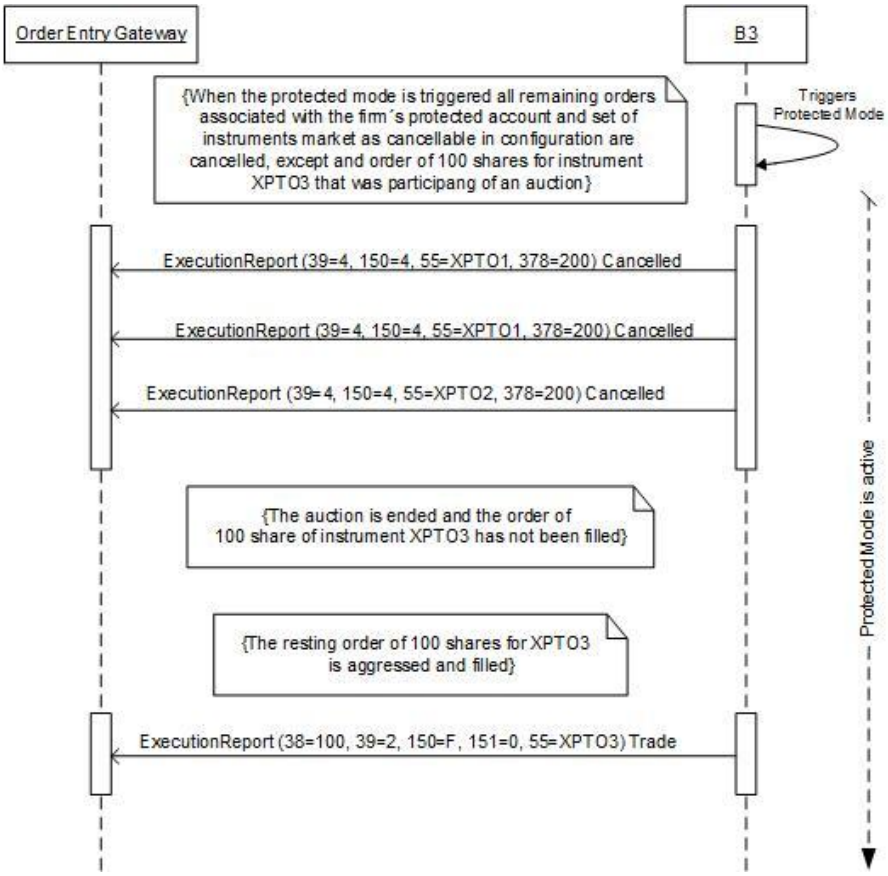


The events are described in the table below:

#	Description
1	Market Protections have already been triggered by firm's protected account and monitored instruments
2	Customer sends a New Order Single (35=D) without tag (9773=Y)
3	Since the firm's protected account and instrument provided is running in Protected Mode, the new order is rejected
4	Customer sends an Order Cancel Replace Request (35=G) with the specific tag (9773=Y) to notify the trading platform to leave the Protection Mode and reset the Monitoring Mode
5	Modification is registered
6	New orders for firm's protected account and monitored instruments start being accepted again

16.3.3 Order Filled During the Protected Mode

For this example, consider there is an order of 100 shares for the instrument XPTO3 that is participating in the auction when the Protected Mode is triggered.

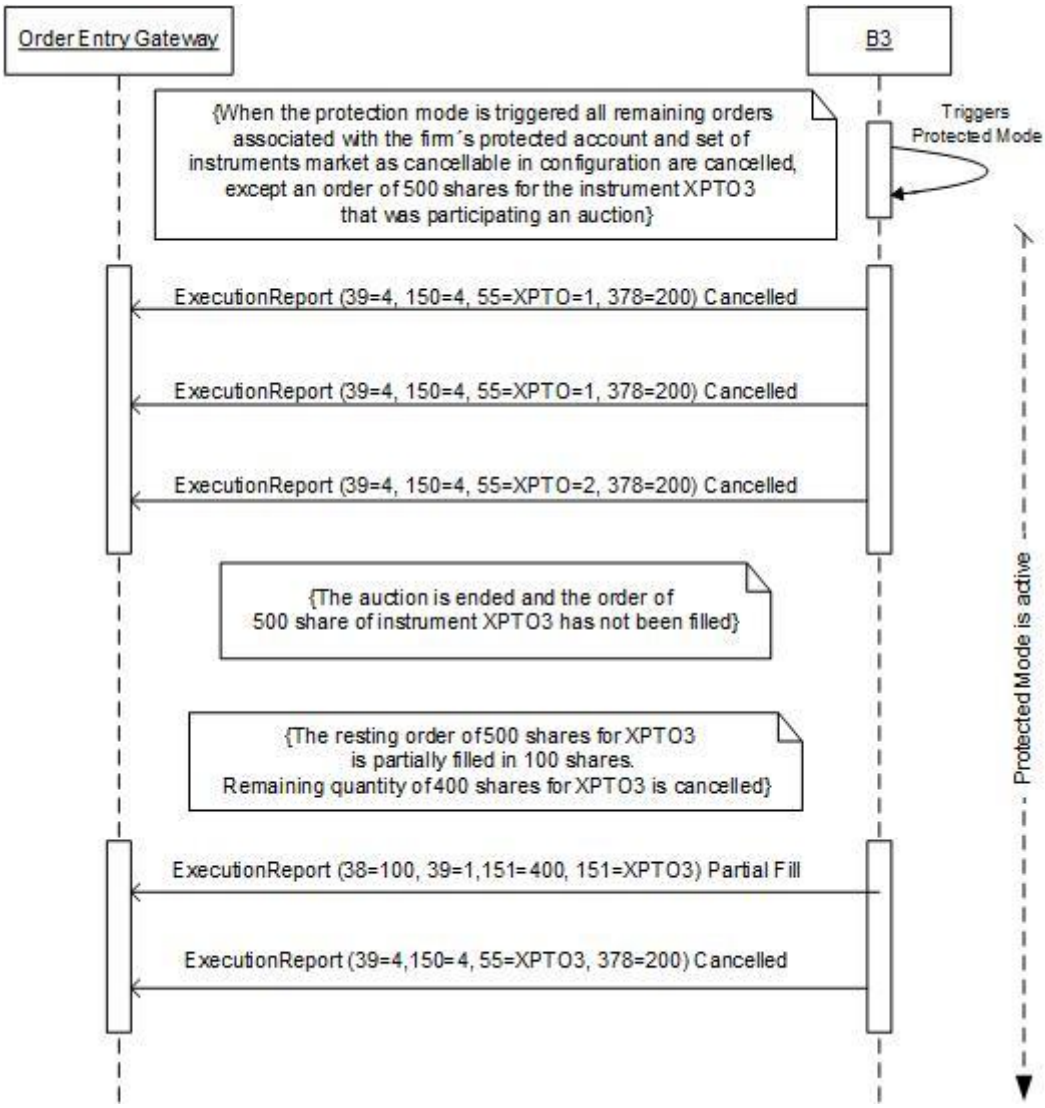

Order Entry Gateway
B3
{When the protected mode is triggered all remaining orders associated with the firm´s protected account and set of instruments market as cancellable in configuration are cancelled, except and order of 100 shares for instrument XPTO3 that was participang of an auction}
Triggers Protected Mode
ExecutionReport (39=4, 150=4, 55=XPTO1, 378=200) Cancelled
ExecutionReport (39=4, 150=4, 55=XPTO1, 378=200) Cancelled
ExecutionReport (39=4, 150=4, 55=XPTO2, 378=200) Cancelled
{The auction is ended and the order of 100 share of instrument XPTO3 has not been filled}
{The resting order of 100 shares for XPTO3 is aggressed and filled}
ExecutionReport (38=100, 39=2, 150=F, 151=0, 55=XPTO3) Trade
Protected Mode is active

The events are described in the table below:

#	Description
1	The Protected Mode is triggered by one protected instrument for a firm's protected account
2	All remaining resting orders of set of instruments configured are canceled, except an order of 100 shares for the instrument XPTO3 that is participating in the auction
3	The auction for instrument XPTO3 is ended
4	The order of 100 shares for XPTO3 was not filled during the auction
5	Protected Mode continues active for configuration related with firm's protected account and a set of monitored instruments
6	The resting order of 100 shares for XPTO3 is aggressed and filled

16.3.4 Order Partially Filled during Protected Mode and Remaining Quantity Canceled

For this example, consider there is a resting order of 500 shares for instrument XPTO3. The instrument is in Reserved state, participating in the auction's theoretical price formation.



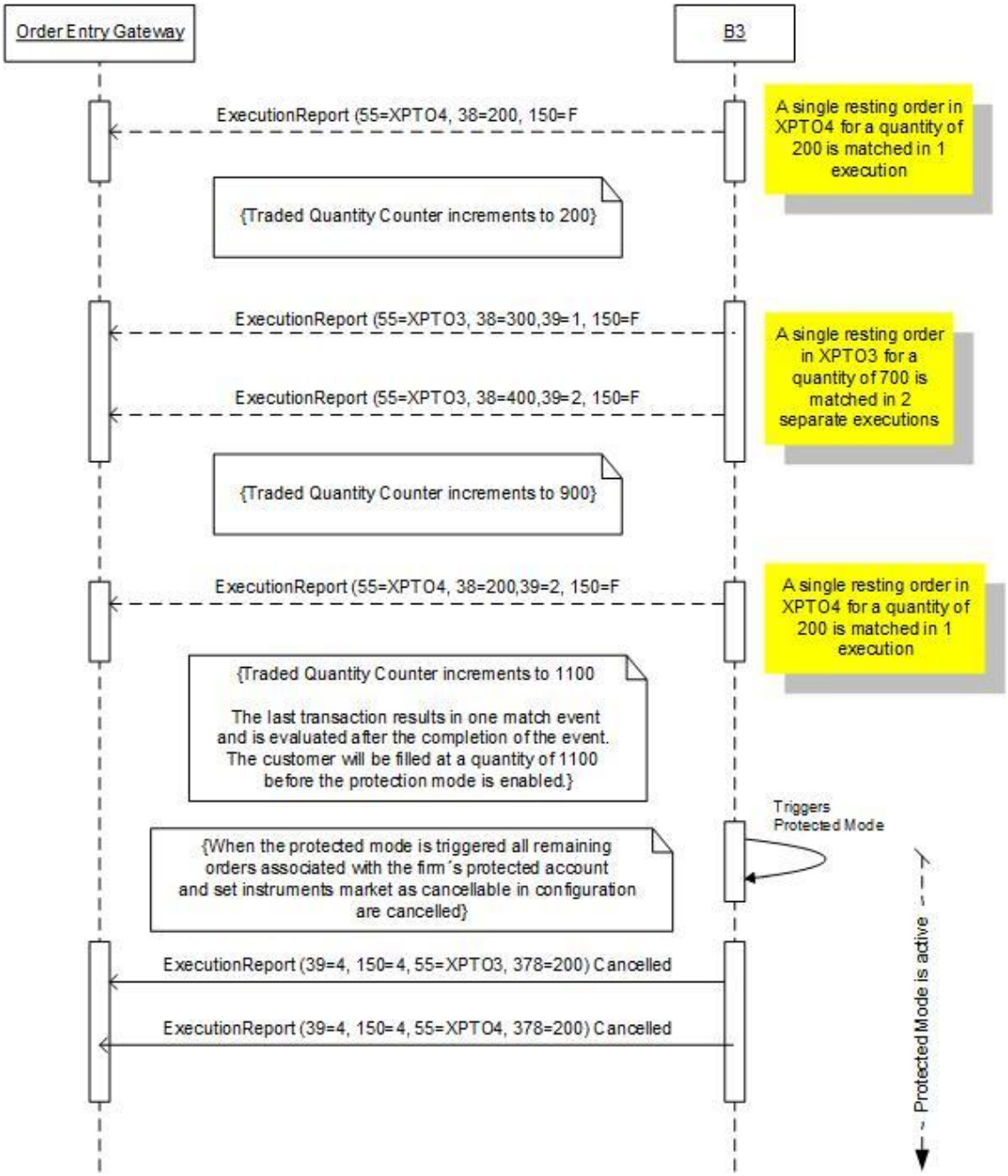
The events are described in the table below:

#	Description
1	The Protected Mode is triggered by one protected instrument for a firm's protected account
2	All remaining resting orders of configuration are canceled, except an order of 500 shares for the instrument XPTO3 that is participating in the auction
3	The auction for instrument XPTO3 is ended
4	The order of 500 shares for XPTO3 was not filled during the auction
5	Protected Mode continues active for firm's protected account and a set of monitored instruments
6	The resting order of 500 shares for XPTO3 is aggressed and partially filled in 100 shares
7	Remaining quantity of 400 shares for XPTO3 is canceled

16.3.5 Order Filled and Protection value Exceeded.

This is example depicts a situation where the Protection value is exceeded before the Protected Mode is triggered.

Consider a scenario where a Traded Quantity Protection a configuration is set to 1000.



The events are described in the table below:

#	Event
1	A single resting order's bid in XPTO4 for a quantity of 200 is matched in 1 execution.
2	A single resting order's ask in XPTO3 for a quantity of 700 is matched in 7 separate executions.
3	A single resting order's ask in XPTO4 for a quantity of 200 is matched in 1 execution.
4	Customer is filled at a quantity of 1100 before the Protection Mode is enabled for the group.
5	Protection mode is enabled for firm's protected account and set of monitored instruments. All remaining resting orders within a set of instruments market as cancellable in configuration are canceled.

16.3.6 Stop Order Triggered after Auction Not Canceled at Protection Mode Activation

For this example, consider there is a stop order of 100 shares for the instrument XPTO3. The instrument XPTO3 is in auction.

16.4 Forward

This session will be updated in the future.

16.5 Exercise

This session will be updated in the future.

16.6 Sweep & Cross

Sweep and cross is a mechanism through which the participant combines the sweeping of orders in the central book with the subsequent registration of a cross order on the platform. Normally this is done to "open the spread" and allow the cross order to be registered following the rules of improving prices.

It makes operations (which are currently done manually) possible in an automated way for customers through a cross order where one side is executed against the orders in the central book before closing against the other side of the cross.

The *NewOrderCross* message will be used to send Sweep & Cross.

The acknowledgment of receipt of a New Order Cross message is issued by B3 in the form of two Execution Report type messages. The order may be accepted (*ExecutionReport_New*) or rejected (*ExecutionReport_Reject*) according to B3 rules.

The *crossType* field identifies the model:

- **4 – Sweep and Cross (Cross executed against book from Client)**
  A cross order that can be executed against existing orders in the order book up to a limited price and a sweep quantity followed by a cross order. Basically, the client selects one side of the cross order that will be executed against existing orders on the other side of the book, as long as the price in the order book is better or the same as the price in the cross order. Any remaining or total amount that is not executed in the order book will be executed by the cross order.

The *crossPrioritization* field indicates the side of the cross order that should be prioritized (sweep aggressor):

- **1 - Buy side is prioritized.**
- **2 - Sell side is prioritized.**

The optional field *maxSweepQty* indicates the maximum sweep quantity. If it is not informed, the cross quantity will be used for the sweep operation.

Any side with remaining quantity will be cancelled with reason: "*210 - Cancel remaining from Sweep and Cross*" in the *execRestatementReason* field.

Here are some possible scenarios:

16.6.1 Max sweep quantity not informed

Book:

PETR4 open book					
BUY			SELL		
Firm	Qty	Price	Price	Qty	Firm
A	200	19.85	19.88	200	B
B	500	19.84	19.90	500	C
C	500	19.82	19.91	300	D
D	300	19.78	19.95	200	F
E	500	19.75	19.96	500	G

PETR4 open book					
			19.98	700	H
			19.99	500	A
			20.00	100	B
			20.10	100	C
			20.10	100	D

Sweep Cross:

Instrument	Firm	Buy account	Sell account	Price	Quantity	Sweep Model	Cross Side prioritized
PETR4	E	123	1000	20.00	100,000	4 - Cross executed against book from Client	Buy

As the maximum sweep quantity was not informed, the cross quantity (100,000) was used for the sweep operation.

Trades generated:

Instrument	Price	Quantity	Buy firm	Buy account	Sell firm	Sell account	Type
PETR4	19.88	200	E	123	B		Sweep
PETR4	19.90	500	E	123	C		Sweep
PETR4	19.91	300	E	123	D		Sweep
PETR4	19.95	200	E	123	F		Sweep

Instrument	Price	Quantity	Buy firm	Buy account	Sell firm	Sell account	Type
PETR4	19.96	500	E	123	G		Sweep
PETR4	19.98	700	E	123	H		Sweep
PETR4	19.99	500	E	123	A		Sweep
PETR4	20.00	100	E	123	B		Sweep
PETR4	20.00	97.000	E	123	E	1000	Sweep Cross

Book after Sweep:

PETR4 open book					
BUY			SELL		
Firm	Qty	Price	Price	Qty	Firm
A	200	19.85	20.10	100	C
B	500	19.84	20.10	100	D
C	500	19.82			
D	300	19.78			
E	500	19.75			

16.6.2 Max sweep quantity informed

Book:

PETR4 open book					
BUY			SELL		
Firm	Qty	Price	Price	Qty	Firm
A	200	19.85	19.88	200	B
B	500	19.84	19.90	500	C
C	500	19.82	19.91	300/1,500 (Iceberg)	D
D	300	19.78	19.95	200	F
E	500	19.75	19.96	500	G
			19.98	700	H
			19.99	500	A
			20.00	100	B
			20.10	100	C
			20.10	100	D

Sweep Cross:

Instrument	Firm	Buy account	Sell account	Price	Quantity	Sweep Model	Cross Side prioritized	Max Sweep Qty
PETR4	E	123	1000	19.92	2,000	4 - Cross executed against book from Client	Buy	1500

As the sweep doesn't reach the cross price and 19.92 is worse than 19.91 the cross is not generated.

Trades generated:

Instrument	Price	Quantity	Buy firm	Buy account	Sell firm	Sell account	Type
PETR4	19.88	200	E	123	B		Sweep
PETR4	19.90	500	E	123	C		Sweep
PETR4	19.91	300	E	123	D		Sweep
PETR4	19.91	300	E	123	D		Sweep
PETR4	19.91	200	E	123	D		Sweep

Book after Sweep:

PETR4 open book					
BUY			SELL		
Firm	Qty	Price	Price	Qty	Firm
A	200	19.85	19.91	100/700 (Iceberg)	D
B	500	19.84	19.95	200	F
C	500	19.82	19.96	500	G
D	300	19.78	19.98	700	H
E	500	19.75	19.99	500	A
			20.00	100	B
			20.10	100	C
			20.10	100	D

Appendix A: **Glossary**

Term	Definition
B3	Securities, Commodities and Futures Exchange, based in São Paulo, Brazil. For more information, visit the web site: http://www.b3.com.br/en_us/
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.
Correspondent Broker	Identifies a correspondent broker (broker/firm who originates the order to B3 from a DMA provider or order routing solution).
Counterparty	Party to a trade.
Derivatives	A financial security (such as option or future) whose characteristics and value are derived from the characteristics and the value of another asset.
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.
Entering Firm	Broker who has recorded or reported an execution. This term is particularly useful where the trade is entered into a trading system by a broker who is not a party to the trade, as it allows any inquiries or problem resolution to be directed to the appropriate source.
Entering Trader	The individual identified by a trading badge number or initials that enters an order to a market (especially in open outcry markets). Usually, the Entering Trader is the same as the Executing Trader. However, there are scenarios where the Entering Trader will have the trade executed by another trader who is then identified as the Executing Trader.
EntryPoint	B3's solution for accessing its electronic trading platform
Executing Firm	Identifies executing / give-up broker.
Executing Trader	Trader or broker id associated with Executing Firm who executes the trade.
FIX	Financial Information Exchange Protocol
FIX Gateway	Service that provides connectivity to third-party clients and brokerages using the FIX protocol.
Futures	Contracts covering the sale of financial instruments or physical commodities for future delivery on a commodity exchange.
Give-up firm	Firm to which a trade is given up, i.e., firm which carries the trade.
Holder	The investor holding the right to exercise an option.
Instrument	Financial capital in a readily tradable form.
IP	Internet Protocol
Issuer	An entity that puts a financial asset in the marketplace. The grantor of an options contract who assumes the obligation, if the holder exercises the option, to sell the underlying asset to or buy it from the holder.
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 trade 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.

Order Entry Gateway	The service provided by B3 which relays FIX or FIXP messages from a third-party client, usually a vendor, to the PUMA system.
Order Origination Firm	Firm that originates the order.
Position	Balance resulting from one or more operations with options from the same series, executed on behalf of the same investor, through the same brokerage firm.
SBE	Simple Binary Encoding to produce fast and compact encodings of FIX messages.
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.
SISBEX	The system used by B3 to negotiate public bonds.
SSL	Secure Socket Layer
Strike Price	Price at which the holder will be entitled to buy or sell the option's underlying asset.
TCP	Transport Control Protocol
UDS	User-Defined Spread.
Vendor	An 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.
TOP	Theoretical Opening Price (in auctions)
ER_*	Exec-specific ExecutionReport. For example: ER_New refers to ExecutionReport_New message.
NewOrder	It refers to NewOrderSingle or SimpleNewOrder messages.
ModifyOrder	It refers to OrderCancelReplaceRequest or SimpleModifyOrder messages.
CancelOrder	It refers to OrderCancelRequest message.

Appendix B: **Security Strategy types**

Spread Types		
Spread	**Type**	**Description**
3-Way	3W	A 3-Way (3W) option spread is constructed of calls and puts on the same contract and expiry month with three different strike prices. A Call 3-way consists of buying the call for the middle strike price, selling the call for the high strike price, and selling the put for the low strike price. A Put 3-way consists of buying the put for the middle strike price, selling the put for the low strike price, and selling the call for the high strike price. **Spread ratio: (Buy 1: Sell 1: Sell 1)** **3-Way Call Spread** Construction: • Buy 1 Call at strike2exp1. • Sell 1 Call at strike3exp1. • Sell 1 Put at strike1exp1. Example: • Buy 1 ACMEG11C002100. • Sell 1 ACMEG11C002150. • Sell 1 ACMEG11P002050. **3-Way Put Spread** Construction: • Buy 1 Put at strike2exp1. • Sell 1 Put at strike1exp1. • Sell 1 Call at strike3exp1. Example: • Buy 1 ACMEG11P002100. • Sell 1 ACMEG11P002050. • Sell 1 ACMEG11C002150.
3-Way: Straddle versus a Call	3C	A 3-way: Straddle versus Call (3C) option spread consists of buying a Straddle and (versus) selling a Call in the same expiry month. The Straddle component consists of buying a Call and buying a Put in the same contract, expiration, and strike price. The opposing (versus) component is to sell a Call for the same contract and expiration but at a different strike price. **Spread ratio: (Buy 1: Buy 1: Sell 1)** Construction: • Buy 1 Call at strike1exp1. • Buy 1 Put at strike1exp1. • Sell 1 Call at strike2exp1. Example: Buy the 3-way: Straddle versus Call • Buy 1 ACMEG11C001900. • Buy 1 ACMEG11P001900. • Sell 1 ACMEG11C002100.

3-Way: Straddle versus a Put	3P	A 3-way: Straddle versus Put (3P) option spread consists of buying a Straddle and (versus) selling a Put in the same expiry month. The Straddle component consists of buying a Call and buying a Put in the same contract, expiration, and strike price. The opposing (versus) component is to sell a Put for the same contract and expiration but at a different strike price. **Spread ratio: (Buy 1: Buy 1: Sell 1)** Construction: • Buy 1 Call strike1exp1. • Buy 1 Put at strike1exp1. • Sell 1 Put at strike(?)exp1. Example: Buy the 3-way: Straddle versus Put • Buy 1 ACMEG11C001900 • Buy 1 ACMEG11P001900 • Sell 1 ACMEG11P001700
Box	BX	A Box (BX) option spread consists of buying the call and selling the put at the same lower strike price and buying the put and selling the call at the same higher strike all within the same contract and expiry month. **Spread ratio: (Buy 1: Sell 1: Buy 1: Sell 1)** Construction: • Buy 1 Call at strike1exp1. • Sell 1 Put at strike1exp1. • Buy 1 Put at strike2exp1. • Sell 1 Call at strike2exp1. Example: • Buy 1 ACMEG11C001900. • Sell 1 ACMEG11P001900. • Buy 1 ACMEG11P002100. • Sell 1 ACMEG11C002100.
Butterfly	BO	A Butterfly is an options strategy involving three strike prices that are of equal distance apart with all having the same expiration date. **Call Butterfly** Involves buying one call at the lowest strike price, selling two calls at the middle strike price, and buying one call at the highest strike price. **Put Butterfly** Involves buying one put at the highest strike price, selling two puts at the middle strike price, and buying one put at the lowest strike price. Examples: **Call Butterfly** (all Call options) • Leg 1 = Buy 1 ACMEG11C001800. • Leg 2 = Sell 2 ACMEG11C001900. • Leg 3 = Buy 1 ACMEG11C002000. **Put Butterfly** (all Put options) • Leg 1 = Buy 5 ACMEG11P002000. • Leg 2 = Sell 10 ACMEG11P001900. • Leg 3 = Buy 5 ACMEG11P001800. Butterflies are always done in a 1x2x1 ratio. The three Strike prices are always of equal distance apart. All options have the same expiration date.

Christmas Tree (Xmas Tree)	XT	A Christmas Tree (XT) option spread is constructed of all calls (Call Xmas Tree), or all puts (Put Xmas Tree). The Call Xmas Tree consists of buying a call at one strike, selling a call at a higher strike, and selling yet another call at a higher strike, all within the same contract and expiration month. The Put Xmas Tree consists of buying a put at a higher strike and selling a put at a lower strike and selling yet another put at a still lower strike, all within the same contract and expiration month. The Xmas Tree requires a specific symmetry in the strikes in that the difference between the strike prices is the same for all legs. **Spread ratio: (Buy 1: Sell 1: Sell 1)** **Call Xmas Tree** Construction: • Buy 1 Call at strike1exp1. • Sell 1 Call at strike2exp1. • Sell 1 Call at strike3exp1. Example: • Buy 1 June 2008 Eurodollar 9800 Call. • Sell 1 June 2008 Eurodollar 9850 Call. • Sell 1 June 2008 Eurodollar 9900 Call. **Put Xmas Tree** Construction: • Buy 1 Put at strike3exp1. • Sell 1 Put at strike2exp1. • Sell 1 Put at strike1exp1. Example: • Buy 1 June 2008 Eurodollar 9900 Put. • Sell 1 June 2008 Eurodollar 9850 Put. • Sell 1 June 2008 Eurodollar 9800 Put.
Conditional Curve	CC	Conditional Curve (CC) option spreads are unique to CME Interest rate products and consist of buying a call (put) at a strike in one instrument group and selling a call (put) at a strike in another instrument group. Additionally, it is possible to have a Conditional Curve spread with a single strike (i.e., same for each leg) or two different strikes, where both strikes are listed. **Call Conditional Curve** Construction: • Buy1callstrikeexp1 instrument1. • Sell1callstrikeexp1 instrument2. Example: • Buy 1 December Eurodollar 9800 Call. • Sell 1 December 1-year Mid-Curve 9800 Call. **Put Conditional Curve** Construction: • Buy1putstrikeexp1 instrument 1. • Sell1putstrikeexp1 instrument 2. Example: • Buy 1 December Eurodollar 9800 Put. • Sell 1 December 1-year Mid-Curve 9800 Put.

Condor	CO	A Condor is an options strategy that has four legs (either all Calls or all Puts) and four strikes that are an equal distance apart. **Call Condor** • Involves buying one Call at the lowest strike, selling one Call at the second strike, selling one Call at the third strike, and finally buying one Call at the highest strike. **Put Condor** • Involves buying one Put at the highest strike, selling one Put at the third strike, selling one Put at the second strike, and finally buying one Put at the lowest strike. Example: Buying Call Condor • Leg 1 = Buy 1 ACMEG11C001800. • Leg 2 = Sell 1 ACMEG11C001850. • Leg 3 = Sell 1 ACMEG11C001900. • Leg 4 = Buy 1 ACMEG11C001950. All legs are done in equal quantities. All four Strike prices are an equal distance apart. All options have the same expiration date.
Diagonal	DG	A strategy involving the simultaneous buying and selling of two options of the same type that have different strike prices and different expiration dates. • Two different strikes are used. • Option with the longer expiration date is always bought. • Option with the shorter expiration date is always sold. • Either Strike price (higher or lower) can be bought as long as it has the longer expiration. • Same rules for Calls or Puts. Example: • Leg 1 = Buy 1 ACMEG11C001800 (Buying Feb. 2011 expiration). • Leg 2 = Sell 1 ACMEV10C002100 (Selling Oct. 2010 expiration). Or • Leg 1 = Buy 1 ACMEG11C002100 (Buying Feb. 2011 expiration). • Leg 2 = Sell 1 ACMEV10C002000 (Selling Oct. 2010 expiration). The same trades above but showing the order of the strike prices does not matter as long as they are buying the further out expiration and selling the closer term expiration.
Guts	GT	A Guts (GT) option spread consists of buying a Call at a strike price and buying a Put at a higher strike price in the same expiry. **Spread ratio: (Buy 1: Buy 1)** Construction: • Buy 1 Call at strike1exp1. • Buy1 Put at strike2exp1. Example: Buy the Guts • Buy 1 ACMEG11C002100. • Buy 1 ACMEG11P002200.

Horizontal	HO	A horizontal (HO) option spread consists of buying a call (put) at a strike in the far month and selling a call (put) at the same strike in the near month. **Spread ratio: (Buy 1: Sell 1)** Example • Leg 1 = Buy 1 ACMEG11C001800. • Leg 2 = Sell 1 ACMEV10C001800. Both legs use same quantity and strike price but have different expirations.
Double	DB	A Double (DB) option spread is constructed of all calls (Call Double), or all puts (Put Double). The **Call Double** consists of buying a call at a strike price and buying another call at a higher strike price within the same contract and expiry month. The **Put Double** consists of buying a put at a strike price and buying another put at a lower strike price within the same contract and expiry month. **Spread ratio: (Buy 1: Buy 1)** **Call Double** Construction: • Buy1 Call at strike1exp1. • Buy1 Call at strike2exp1. Example: • Buy 1 ACMEG11C002100. • Buy 1 ACMEG11C002150. **Put Double** Construction: • Buy 1 Put at strike2exp1. • Buy 1 Put at strike1exp1. Example: • Buy 1 ACMEG11P002150. • Buy 1 ACMEG11P002100.
Iron Butterfly	IB	An Iron Butterfly contains four legs as is an equivalent strategy to a regular butterfly spread which contains only three legs. Uses three different strikes with both inside legs using the same strike. Construction: • Sell 1 Put at lowest strike. • Buy 1 Put at middle strike. • Buy 1 Call at middle strike. • Sell 1 Call at highest strike. Example: Buying Iron Butterfly • Leg 1 = Sell 1 ACMEG11P001800. • Leg 2 = Buy 1 ACMEG11P001850. • Leg 3 = Buy 1 ACMEG11C001850. • Leg 4 = Sell 1 ACMEG11C001900.

Iron Condor	IC	An Iron Condor (IC) option spread consists of buying a put spread and buying a call spread at higher strike prices. More specifically this consists of selling a put at one strike price, buying a put at a higher strike price, buying a call at a higher strike price, and selling a call at an even higher strike price, all within the same contract and expiration. **Spread ratio: (Sell 1: Buy 1: Buy 1: Sell 1)** Construction: • Sell 1 Put at strike1exp1. • Buy 1 Put at strike2exp1. • Buy 1 Call at strike3exp1. • Sell 1 Call at strike4exp1. Example: • Sell 1 ACMEG11P001800. • Buy 1 ACMEG11P001850. • Buy 1 ACMEG11C001900. • Sell 1 ACMEG11C001950.
Horizontal Straddle	HS	A Horizontal Straddle (HS) option spread consists of buying a straddle at one strike price in the deferred month and selling a straddle at the same or different strike in the near month. More specifically, a Horizontal Straddle (HS) consists of buying a call and buying a put at the same strike price in the deferred month and selling a call and selling a put at the same lower strike price in the near month, all within the same contract and expiry month. **Spread ratio: (Buy 1: Buy 1: Sell 1: Sell 1)** Construction: • Buy 1 Call at strike1exp2. • Buy 1 Put at strike1exp2. • Sell 1 Call at strike1exp1. • Sell 1 Put at strike1exp1. Example: Horizontal Straddle • Buy 1 ACMEG11C002100. • Buy 1 ACMEG11P002100. • Sell 1 ACMEV10C002100 (same or different strike in near month) • Sell 1 ACMEV10P002100 (same or different strike in near month) Or • Buy 1 Sept 2008 Eurodollar 98000 Call. • Buy 1 Sept 2008 Eurodollar 98000 Put. • Sell 1 June 2008 Eurodollar 98500 Call. • Sell 1 June 2008 Eurodollar 98500 Put.

Jelly Roll	JR	A Jelly Roll is created by entering two separate positions simultaneously. One position involves buying a put and selling a call with the same strike price and expiration. The second position involves selling a put and buying a call. The strike prices in the far expiry (second position) can be, but need not be, equal to the strike price in the near expiry. The strike prices of the put and call in the second position are identical but different from the previous position, and the duration of the second position is longer than the previous position. **Jelly Roll (all same strikes, different expirations)** • Sell 1 Call and Buy 1 Put at exp1 • Buy 1 Call and Sell 1 Put at exp2 **Jelly Roll (different strikes, different expirations)** • Sell 1 Call and Buy 1 Put at strike1 at exp1 • Buy 1 Call and Sell 1 Put at strike2 at exp2 Or • Sell 1 Call and Buy 1 Put at strike2 at exp1 • Buy 1 Call and Sell 1 Put at strike1 at exp2
Ratio 1x2	12	A Ratio 1x2 is the purchase of an option(s), call or put, and the selling of a greater number of the same type of options that are out-of-the-money for those purchased. All options involved have the same expiration date. **Call** • Buy 1 Call at strike1 and exp1. • Sell 2 Calls at strike2 and exp1. **Put** • Buy 1 Put at strike2 and exp1. • Sell 2 Puts at strike1 and exp1. Trades can be done in larger quantities if in a 1x2 ratio.
Ratio 1x3	13	A Ratio 1x3 is the purchase of an option(s), call or put, and the selling of a greater number of the same type of options that are out-of-the-money for those purchased. All options involved have the same expiration date. **Call** • Buy 1 Call at strike1 and exp1. • Sell 3 Calls at strike2 and exp1. **Put** • Buy 1 Put at strike2 and exp1. • Sell 3 Puts at strike1 and exp1. Trades can be done in larger quantities if in a 1x3 ratio.
Ratio 2x3	23	A Ratio 2x3 is the purchase of an option(s), call or put, and the selling of a greater number of the same type of options that are out-of-the-money for those purchased. All options involved have the same expiration date. **Call** • Buy 2 Calls at strike1 and exp1. • Sell 3 Calls at strike2 and exp1. **Put** • Buy 2 Puts at strike2 and exp1. • Sell 3 Puts at strike1 and exp1. Trades can be done in larger quantities if in a 2x3 ratio.

Risk Reversal	RR	An option strategy combining the simultaneous purchase of out-of-the-money Calls (Puts) with the sale of out-of-the money Puts (Calls). The options will have the same expiration date and are done in the same quantities. **Call** • Buy 1 Call at strike2 (out-of-money) and exp1. • Sell 1 Put at strike1 (also out-of-money) and exp1.
Straddle Strips	SS	Straddle Strips are options strategies which are buying
Straddle	ST	A Straddle (ST) option spread consists of buying both a call and put option on the same contract, strike price, and expiration date. **Spread ratio: (Buy 1: Buy 1)** Construction: • Buy 1 Call at strike1 and exp1. • Buy 1 Put at strike1 and exp1. Example: • Buy 1 ACMEG11C001900. • Buy 1 ACMEG11P001900.
Strangle	SG	A Strangle (SG) option spread consists of buying a Put at a lower strike price and buying a Call at a higher strike price within the same contract and expiration. **Spread ratio: (Buy 1: Buy1)** Construction: • Buy 1 Put at strike1exp1. • Buy 1 Call at strike2exp1. Example: • Buy 1 ACMEG11P001900. • Buy 1 ACMEG11C001950.

Strip	SR	A Strip (SR) option spread is constructed of all Calls (Call Strip), or all Puts (Put Strip). The Call Strip consists of buying calls within the same contract and strike price for each of four consecutive quarterly expiry months, resulting in a total of four (4) Calls purchased. The Put Strip consists of buying puts within the same contract and strike price for each of four consecutive quarterly expiry months, resulting in a total of four (4) Puts purchased. The Strip requires a specific symmetry in the expiry months in that the time difference between the expiry months is the same for all legs. **Spread ratio: (Buy 1: Buy 1: Buy 1: Buy 1)** **Call Strip** Construction: • Buy 1 Call at strike1exp1. • Buy 1 Call at strike1exp2. • Buy 1 Call at strike1exp3. • Buy 1 Call at strike1exp4. Example: Call • Buy 1 June 2008 Eurodollar 9800 Call. • Buy 1 Sept 2008 Eurodollar 9800 Call. • Buy 1 Dec 2008 Eurodollar 9800 Call. • Buy 1 March 2009 Eurodollar 9800 Call. **Put Strip** Construction: • Buy 1 Put at strike1exp1. • Buy 1 Put at strike1exp2. • Buy 1 Put at strike1exp3. • Buy 1 Put at strike1exp4. Example: Put • Buy 1 June 2008 Eurodollar 9800 Put. • Buy 1 Sept 2008 Eurodollar 9800 Put. • Buy 1 Dec 2008 Eurodollar 9800 Put. • Buy 1 March 2009 Eurodollar 9800 Put.

Vertical	VT	A Vertical (VT) option spread is made up of all calls or all puts and consists of buying a call at a strike price and selling a call at a higher strike price or buying a put at a strike price and selling a put at a lower strike price within the same contract and expiration date. **Spread ratio: (Buy 1: Sell 1)** **Call Vertical** Construction: • Buy 1 Call at strike1exp1. • Sell 1 Call at strike2exp1. Example: Call Spread • Buy 1 ACMEG11C001900. • Sell 1 ACMEG11C001950. **Put Vertical** Construction: • Buy 1 Put at strike2exp1. • Sell 1 Put at strike1exp1. Example: Put Spread • Buy 1 ACMEG11P001950. • Sell 1 ACMEG11C001900.
Cash / Cash	VV	A strategy involving the simultaneous buying and selling of two cash instruments (Equity vs. Equity, Equity vs. ETF, or ETF vs ETF) Example: • Leg 1 = Buy 1 ACME3 (Buying ACME common stock). • Leg 2 = Sell 1 ACME4 (Selling ACME preferred stock). Or • Leg 1 = Buy 1 ACME3 (Buying ACME common stock). • Leg 2 = Sell 1 BOVA11 (Selling IBovespa ETF).
Cash / Option	VO	A strategy involving buying or selling any number (greater than 0) of Cash instruments (Equity or ETF) and selling or buying any number (greater than 0) of equity options. No more than 40 total instruments are allowed in a single strategy.
Single Stock Future / Option	FO	A strategy involving buying or selling any number (greater than 0) of Single Stock Futures (Equity or ETF) and selling or buying any number (greater than 0) of equity options. No more than 40 total instruments are allowed in a single strategy.
Single Stock Futures / Single Stock Future	FF	A strategy involving the simultaneous buying and selling of two Single Stock Future instruments. Example: • Leg 1 = Buy 1 ACMEEFX (Buying ACME May Single Stock Future). • Leg 2 = Sell 1 ACMEFFX (Selling ACME June Single Stock Future).
Cash / Single Stock Future	FV	A strategy involving the simultaneous buying a Cash Equity and selling a Single Stock Future instrument. Example: • Leg 1 = Buy 1 ACME3 (Buying ACME common stock). • Leg 2 = Sell 1 ACMEEFX (Selling ACME May Single Stock Future).
Generic	GN	Any UDS that does not fit any of the above-listed strategy types will be assigned the type of Generic.