FIX Specification for Datatec Systems

SET ICAP: Set-FX

Version 3.12

18 July 2017

Contents

Figures

Tables

1. Document History

Fecha	Autor	Descripción
2016-09-14	Benito Velasco	Version 1.00
2017-05-03	Benito Velasco	Version 2.00: Improvements and corrections.
2017-06-02	Alexe Bahamonde	Version 2.1: Format adjustments.
2017-07-12 2017-07-17	Benito Velasco	Version 3.01: Adjustments and corrections after DFIX Gateway development.
2017-07-18	Alexe Bahamonde	Version 3.12: Format adjustments.

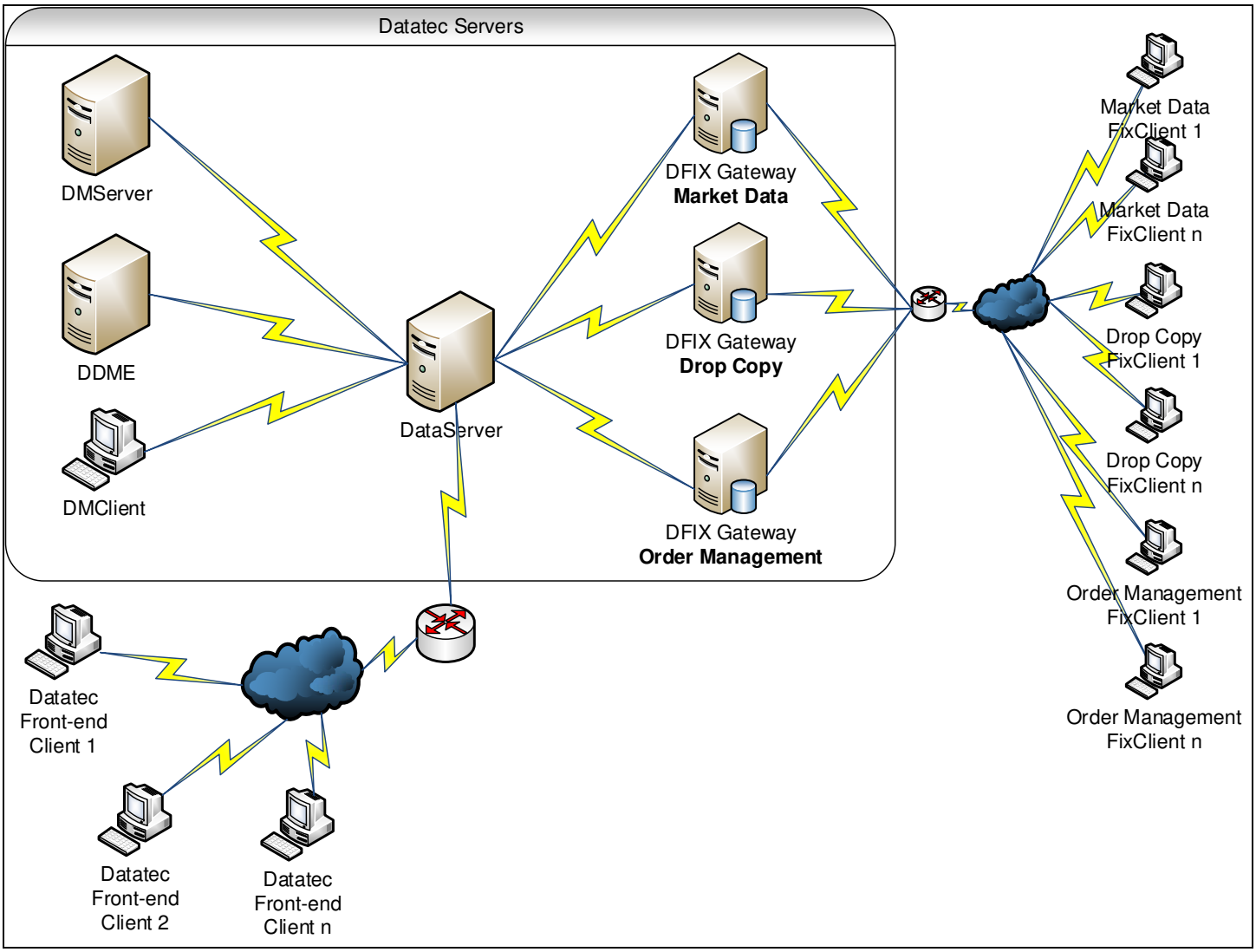
2. Introduction

This document details the specifications of the FIX Protocol for connecting to FX Spot markets on the DATATEC platform. The FIX Client should connect using FIX Protocol 5.0SP2with the latest Extension Pack.

The implementation supports the FIX Protocol standard conventions. The FIX session will always be started and terminated by the FIX Client. The DATATEC FIX (DFIX) Gateway which will act as a server will always be the session listener. There may be multiple instances of the DFIX Gateway, and each instance can allow multiple FIX client connections.

The following diagram shows the architecture of the Datatec environment including FIX Client connections:

Figure 1–Architecture including FIX Client connections



The DFIX Gateway maintains a TCP/IP connection with the DS (Data Server) using the Datatec internal message protocol and multiple FIX Sessions with FIX Clients. The main job of the DFIX Gateway is to translate Datatec messages received from the DS into FIX messages to send to the appropriate FIX Clients; and to translate FIX messages received from FIX clients to send them to the DS. In some cases the DFIX Gateway

will complete other tasks like maintaining the central order book and when necessary preparing messages like Snapshot and Incremental messages for Market Data.

3. FIX Messages Supported

The following table lists the FIX messages that will be supported by the DFIX Gateway; the direction of the messages is indicated with respect to the DFIX Gateway.

Table 1 – FIX Messages Supported

Message Name	Message Type	Direction	Message Function
Administrative Messages			
Logon	A	Inbound Outbound	Identifies and authenticates a user establishing a FIX session with the gateway. The logon message must be the first message sent by the application requesting to initiate a FIX session.
Logout	5	Inbound Outbound	Initiates or confirms the termination of a FIX session. Disconnection without the exchange of logout message should be interpreted as an abnormal disconnection.
Reject	3	Inbound Outbound	Response message issued when a message is received but cannot be processed by the DFIX Gateway or FIX Client, e.g. when the FIX message is malformed, or missing standard required fields.
ResendRequest	2	Inbound Outbound	Sent by either party to initiate a re-transmission of missing messages from the other party.
SequenceReset	4	Inbound Outbound	Used by either party to respond to the ResendRequest message to reset the incoming sequence number on the opposing side. This message has two modes: Sequence Reset - Gap Fill and Sequence Reset-Reset.
TestRequest	1	Inbound Outbound	Checks sequence numbers or verifies communication line status. The opposite application responds to the TestRequest with a Heartbeat message containing the TestReqID field with the value from the same field in the TestRequest message.
Heartbeat	0	Inbound Outbound	Monitors gateway status during periods of inactivity.
BusinessMessageReject	j	Outbound	Rejects any application message that cannot be processed by the DFIX Gateway and cannot be rejected via another message.
Reference Data Messages			
MarketDefinitionRequest	BT	Inbound	Request for markets structure information.
MarketDefinition	BU	Outbound	Respond to MarketDefinitionRequest. It will provide the initial snapshot of the market

Message Name	Message Type	Direction	Message Function
			structure information, one message for each market which the FIX Client has access.
MarketDefinitionUpdateReport	BV	Outbound	Used by DFIX Gateway to send subsequent updates in market structure information.
PartyRiskLimitsRequest	CL	Inbound	Used to request for risk information for specific parties, specific party roles or specific instruments.
PartyRiskLimitsReport	CM	Outbound	Used to communicate party risk limits. The message can either be sent as a response to the PartyRiskLimitsRequest message or can be published unsolicited.
PartyRiskLimitsUpdateReport	CR	Outbound	Used to convey incremental changes to risk limits.
Market Data Messages			
MarketDataRequest	V	Inbound	Requests for a market data on specific market.
MarketDataRequestReject	Y	Outbound	Rejects MarketDataRequest messages that cannot be honored due to business or technical reasons.
MarketDataSnapshotFullRefresh	W	Outbound	Responds to the MarketDataRequest message with the current market data on specific market.
MarketDataIncrementalRefresh	X	Outbound	Used to convey subsequent changes on specific market.
Order Management Messages			
NewOrderSingle	D	Inbound	Used by institutions wishing to electronically submit orders for execution. This could include hit/take orders.
OrderCancelRequest	F	Inbound	Request to cancel all of the remaining quantity of an existing order.
OrderCancelReplaceRequest	G	Inbound	Request message to change the details of an existing order that is not been fully filled.
OrderCancelReject	9	Outbound	Reject message for an OrderCancelReplaceRequest or OrderCancelRequest that cannot be honored.
ExecutionReport	8	Outbound	Responds with the action that matching engine has taken in response to a new or existing order. Include: - Confirm the receipt of an order. - Confirm changes to an existing order (i.e. accept cancel and replace requests). - Relay order status information. - Relay fills information on working orders.

Message Name	Message Type	Direction	Message Function
OrderMassActionRequest	CA	Inbound	Used to request the cancellation or status of a group of orders that match the criteria specified in the request.
OrderMassActionReport	BZ	Outbound	Acknowledgement to an OrderMassActionRequest.
OrderStatusRequest	H	Inbound	Used by the institution to generate an order status message back from DFIX Gateway.
Trade Capture Reporting and Post Trade Messages			
TradeCaptureReportRequest	AD	Inbound	Request all trades belonging to requesting institution, and to subscribe or unsubscribe for trade capture reports.
TradeCaptureReport	AE	Inbound Outbound	This message will be used to: - Sent as a reply to a TradeCaptureReportRequest. - Report trades between counterparties. - Report trades to a trade matching system (Trade Registration). - Reject a trade registered. - Confirm a trade registered. - Report unmatched and matched trades. - Report a trade annulled. - Report trades switched in or switched out clearing house.
TradeCaptureReportRequestAck	AQ	Outbound	Used to indicate that no trades matched the selection criteria specified in the TradeCaptureReportRequest or the TradeCaptureReportRequest was invalid.
TradeCaptureReportAck	AR	Outbound	This message will be used to: - Acknowledge trade capture reports received from counterparty. - Reject a trade capture report received from counterparty.

4. DFIX Gateway Functionality

4.1. FIX Support for DFIX Functional Areas

The DFIX Gateway functionality has been broken down into several areas, each area will use a specific set of FIX messages, as indicated in the following table:

Table 2 – DFIX Gateway Functionality Areas

Area	Sent by FIX Client	FIX Message Type	Sent by DFIX Gateway	FIX Message Type
Logon and Authentication	FIX Session Logon and Authentication Request	A	FIX Session Authenticated	A
			Authentication Failure	5
	FIX Session Logoff Request	5	FIX Session Logoff Response	5
Markets Definition	Market Definition Subscription Request	BT	Market Definition Response	BU
	N/A		Market Definition Update	BV
Market Data	Market Data Subscription Request	V	Subscription Reject	Y
			Subscription Response (snapshot/full refresh)	W
	N/A		Incremental Updates	X
	Market Data Unsubscribe Request	V	N/A	
Credit Limits	Credit Limits Subscription Request	CL	Credit limits Response	CM
			Credit limits Update	CR
Order Management	Submit Order Request	D	Execution Report	8
	N/A		Execution Report (changes in status of an order)	8
	Order Amend Request	G	Execution Report	8
			Order Amend Reject	9
	Order Interrupt Request	F	Execution Report	8
			Order Interrupt Reject	9
	Interrupt All Orders Request	CA	Interrupt All Orders Accepted plus Execution Report (changes in status of orders)	BZ 8
			Interrupt All Orders Reject	BZ
	Hit/Take Order Request – specific amount	D TimeInForce=3	Hit/Take Reject	8
			Execution Report (Partial fill or filled)	8
	Order Status Request	H	Execution Report (order status)	8
			Order Status Reject	8
Post Trade	Trade Capture Report Request (Subscription)	AD	Trade Capture Reports	AE
			Request Reject	AQ
	Trade Registration Request	AE	Trade Registration Request	AR

Area	Sent by FIX Client	FIX Message Type	Sent by DFIX Gateway	FIX Message Type
			rejected or accepted	
			Trade Registration Request sent to counterpart	AE
	Confirm Trade Registration	AE	Confirm Trade Registration rejected	AR
			Trade Capture Report	AE
	Reject Trade Registration	AE	Reject Trade Registration sent to counterpart	AE

Each of the FIX messages specified in the above table will be described in the following sections of this document.

4.2. User profiles

The Datatec platform will support three user profiles for FIX Clients. FIX Clients will be setup with the appropriate profiles based on their needs.

1. Order and Trade Management, known as Order Management - This profile allows order submission, modification and cancelation of orders as well as receiving execution reports from DFIX Gateway. This profile will be able to support trade registration.

2. Market Data - This profile provides real-time information of OrderBook (current bids and offers), market trades, trade summary and statistics.

3. Drop Copy - This profile provides carbon copies of all Execution Reports and Trade Capture Reports where the firm is involved in the current business day. Drop Copy is a read-only interface that does not allow a recipient to enter, cancel or replace any orders on Datatec platform.

On the platform there will be at least one instance of DFIX Gateway dedicated to serving each user profile with its respective DFIX Gateway for fail over. The DFIX Gateway architecture is scalable. The exchange could have 2 or 3 or more DFIX Gateways serving each kind of profile. If an exchange client wants both Order Management and Market Data, two separate FIX sessions should be established with two different DFIX Gateway servers each serving one of the profiles.

The OrderBook is maintained in the Matching Engine and is streamed to the DataServers using a fast technique native to Datatec called Global Tables which are maintained in memory in each Datatec subscriber to a given table. The DFIX Gateways that are serving Order Management and Market Data profiles will be connected to DataServers using the Datatec protocol and will receive a copy of the OrderBook. The DFIX Gateway which is serving Drop Copy profile will also be connected to the Data Servers but this does not subscribe to the OrderBook.

4.3. Required Functionality by Profile

The following table shows the functionality areas for each profile, some of which are required to be implemented by the FIX Client and some of which are optional.

Table 3 – Required Functionality by Profile

Functionality Area \ Profile		Order Management	Market Data	Drop Copy
Logon and Authentication		R	R	R
Markets Definition		R	R	
Market Data			R	
Credit Limits	Credit limits Subscription	C		
Order Management		R		
Post Trade	Trade Capture Report Subscription	C		RT
	Trade Registration	C		

C	Conditionally implemented – Based on customer needs
R	Implementation required as part of profile
RT	Implementation required for third parties

5. Identifiers

An identifier can be a word, number, letter, symbol, or any combination of those, which are used to uniquely identify something. The words, numbers, letters, or symbols may follow an encoding system, or they may simply be arbitrary.

All IDs must use valid characters that include:

- Numbers 0-9
- Letters (A-Z, a-z)
- - Hyphen (decimal 45)
- _ Underscore (decimal 95)

The maximum length for an ID varies depending on the type or purpose of ID.

5.1. Messaged IDs

In FIX Protocol there are different message fields that are handled as identifiers.

5.1.1. Request IDs

All FIX messages that are used to request information include a request ID field which should be created by the FIX Client and used to track other messages associated with the request. The maximum length for request IDs that is supported by the DFIX Gateway is 32 characters.

5.1.2. Order and FirmTrade IDs

IDs used for orders and trades, like ClOrdID(11), OrderID(37), OrigClOrdID(41), ExecID(17), FirmTradeID(1041) must be up to a maximum length of 10 characters. Some of these IDs are created by FIX Clients and some other by the DFIX Gateway. The uniqueness of these values is guaranteed within a single trading day.

5.1.3. Trade IDs

IDs used for TradeID(1003) must be up to a maximum length of 32 characters. These IDs are created by the DFIX Gateway. The uniqueness of these values is guaranteed within a single trading day.

5.2. Party IDs

These kinds of IDs are all created on the Datatec by the Platform administrator, and are currently being sent to and used by the native Datatec GUI. The platform has identifiers for Firms (Institutions), Branches, Users (Traders, Brokers, etc.).

Firms are identified by a short 3-character ID and by a name of maximum 30 characters. In some cases the FIX Client needs to send the short ID, the exchange will provide the list of all firms that participate in markets segments supported by DFIX Gateway. The list of firms will be delivered by Datatec to the clients in a CSV file with columns as shown in the following:

```
ID,Name,Class,Function
BCL,Bancolombia,B,P
BGT,Banco de Bogotá,B,P
CIT,Citibank,B,P
CCD,Cámara de Compensación de Divisas,C,C
STN,Banco Santander,B,P
```

Branches belong to firms, branches are identified by a short 4-character ID plus 3-character of the city code where branch was created plus 1-character assigned by the system to distinguish branches in the same city; i.e. a complete ID for branches is an 8-character ID; this will be send by the DFIX Gateway in message where the branch ID is required. Normally the first 4 characters are assigned by the system administrator ensuring that this code is not assigned to any other branch in the system, so the native GUI normally only use the first 4 characters. Branches also have a full name of maximum 50 characters. The list of branches will also be delivered by the exchange in a CSV file with columns as shown follow:

```
Branch ID,Firm ID,Name,City,Agency
BCLBBOGA,BCL,Bancolombia  Bogotá,BOG,A
BCLCCLIA,BCL,Bancolombia Cali,CLI,A
BCLGBOGB,BCL,Bancolombia  Bogotá 1,BOG,B
BGTBBOGA,BGT,Banco de Bogotá  Bogotá,MDL,A
BGTCCLIA,BGT,Banco de Bogotá Cali,CLI,A
CITBBOGAA,CIT,Citibank Bogotá,BOG,A
CITCCLIA,CIT,Citibank Cali,CLI,A
CITMMDLA,CIT,Citibank Medellín,BOG,A
CCDBBOGA,CCD,Cámara de Compensación de Divisas  Bogotá,BOG,A
SNTBBOGA,SNT,Banco Colpatria Bogotá,BOG,A
SNTKBOGB,SNT,Banco Colpatria Bogotá Respaldo,BOG,B
```

Users belong to branches, and they are identified by a short name of maximum 10 characters and also by a full name of maximum 30 characters.

The DFIX Gateway will send these identifiers, names and full names in the Parties component along with the role of each, when this information is needed.

Beneficiary Codes

Normally trades are between two market participants like banks, investment houses, etc. All these counterparties use IDs as described before. There are two other sorts of trades:1) between two market participants but one of them or both parties traded on behalf of a final customer on their respective sides; and 2) between a market participant and a final customer. Customers do not use the same kind of IDs as market participants; these use their personal or institution codes as follows:

Code	ID Type	Sample values
As used in Colombia		
C	Identification (Cédula)	79265455, 06074698
D	New Identification (Cédula nueva)	1000149153, 1010038336
N	NIT	8605344581, 8001322424
P	Passport	06360020297, 10531039208
T	Identity Card	97022204673, 96101709865
I	New Identity Card	

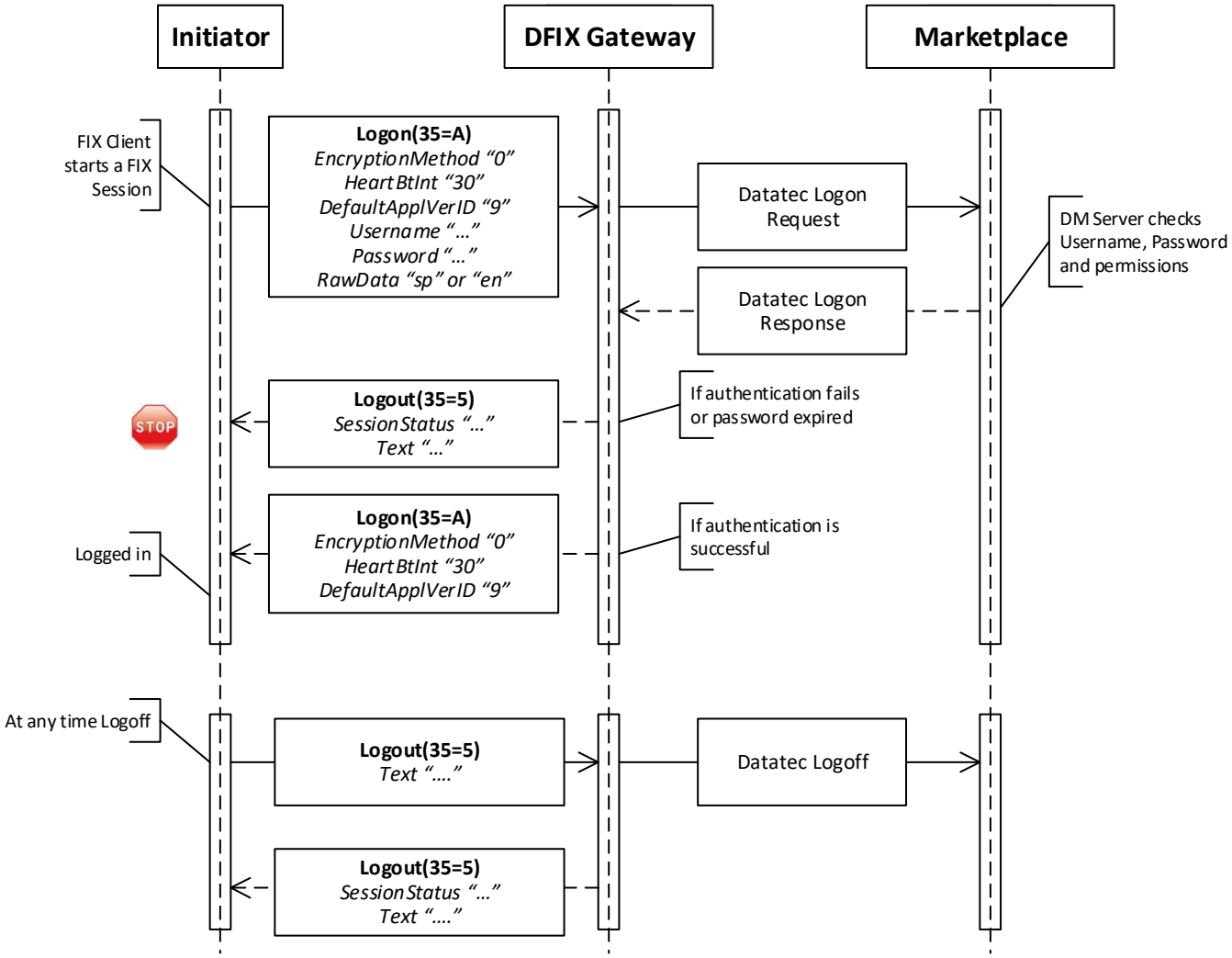
Code	ID Type	Sample values
E	Foreign Identification	263092, 288454
R	Civil Registration Code	9804195822, 1014875766
X	Foreign Company Code	20472315154, 1896767
S	Swift Code	CSFBGB2LXX1, SBOSGB2XXXX
As used in Peru		
R	RUC	20130098488, 20219960256
D	DNI	434518267, 06669871
A	Cavali Code	910697, 707725
C	Card abroad (Foreign Identification)	
O	Other ID	CONTIFONDOS, CONTISAB

When the DFIX sends one of these beneficiary IDs, The DFIX will concatenate the code and the value into the field PartyID(448). (e.g. C06074698, N8605344581).

6. Logon and Authentication

This section describes the Logon and Authentication function, and the FIX messages used to support this function. The flow of the messages involved in the Logon and Authentication process is as follows:

Figure 2 – Session Logon and Authentication Workflow



6.1. Standard Header & Trailer

All FIX messages will have header, body and trailer. The following represents the format of the FIX Standard Header and Standard Trailer, as used by the DFIX Gateway. Any additional tags that the FIX Client may include in the header will be ignored.

6.1.1. FIX Message Header

Each administrative or application message is preceded by a standard header. The header identifies the message type, length, destination, sequence number, origination point and time.

Table 4 – Standard Message Header

Tag	Field Name	DataType	Reqd	Comments
8	BeginString	String	Y	Identifies the beginning of the message and version of the protocol used in the transmission Protocol version. Must be the FIRST TAG IN THE MESSAGE. (Always unencrypted) Valid values: FIXT.1.1
9	BodyLength	Length	Y	Message length (number of bytes) from MsgType up to but not including the CheckSum tag. ALWAYSSECOND TAG IN MESSAGE. (Always unencrypted)
35	MsgType	String	Y	Defines message type. ALWAYS THIRD TAG IN MESSAGE. (Always unencrypted)
34	MsgSeqNum	SeqNum	Y	Integer message sequence number.
49	SenderCompID	String	Y	Code used to identify message sender. For messages FIX Client sends it must be set with the value assigned by the Datatec administrator. For messages Datatec sends it will have the value DFIX_GW
52	SendingTime	UTCTimeStamp	Y	Time of message transmission (always expressed in UTC (Universal Time Coordinated, also known as "GMT"). Format: YYYYMMDD-HH:MM:SS.mmm Example: 20100122-20:46:12.805
56	TargetCompID	String	Y	Code used to identify message receiver. For messages FIX Client sends this field must have the value DFIX_GW. For messages Datatec sends it will contain the value assigned by Datatec administrator.

SenderCompID and TargetCompID values in the table above are used by the FIX Client to send a message; when the DFIX Gateway sends a message to the FIX Client, the values of these fields are switched around.

6.1.2. FIX Message Trailer

Each administrative or application message is terminated by a standard trailer. The trailer is used to segregate messages and contains the three digit character representation of the Checksum value.

Table 5 – Standard Message Trailer

Tag	Field Name	DataType	Reqd	Comments
10	CheckSum	String	Y	Three byte, simple checksum. ALWAYS LAST TAG IN MESSAGE. (Always unencrypted)

6.2. FIX Session Logon and Authentication

The DFIX Gateway application after connecting to DataServer, is ready to accept FIX sessions; it's waiting for a FIX Session Logon Request sent by a customer to establish a FIX session. The session is established using the Logon(35=A) message and part of the process to establish a session includes the authentication of the initiator. This requires a valid SenderCompID(49) which defines the party initiating the session, a Username(553) and a Password(554). The FIX session will not be established if authentication processing fails.

The FIX Logon(35=A) as detailed below also contains information related to the "default" version of FIX being used and optionally the Extension Pack. The heart beat interval is also established and DFIX Gateway accepts only an interval of 30 seconds. Any different value in HeartBtInt(108) will cause a session reject.

Table 6 – Logon Message

Logon(35=A)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **A**
98	EncryptMethod	int	Y	Method of encryption Supported values: 0 = None/Other
108	HeartBtInt	int	Y	Heartbeat interval. Supported values: 30 = 30 seconds
141	ResetSeqNumFlag	Boolean		Indicates both sides of FIX session should reset sequence numbers. Supported values: N = No (Default) Y = Yes, reset sequence numbers
553	Username	String		Required for logon initiator. FIX Client ID provided by ICAP and is formed by the 8-character branch ID and a short name. This must be authorized by the exchange for one and only one of the following profiles: Order Management, Market Data or Drop Copy.
554	Password	String		Required for logon initiator. Associated password for the Client ID, 8-16 alphanumeric characters, case sensitive. If it´s a password change this is the old password.
925	NewPassword	String		Required for logon initiator if it is sending a password change. New password must comply the rules of the password policy. 8-16 alphanumeric characters. Case sensitive. Must be different from the previous client's 12 passwords. Must have at least one alphabetic character. Must have at least one numeric character.

Logon(35=A)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
				Cannot have more than four sequential characters. For example “abcde”, “12345” are not allowed. Cannot have more than two consecutive equal characters. For example “AAA” is not allowed.
1137	DefaultApplVerID	String	Y	The default FIX version for the entire session and applies to all messages on this session. Supported values: 9 = FIX 5.0 SP2
1407	DefaultApplExtID	int		The default Extension Pack for the entire FIX session and applies to all messages on this session. Supported values: 214 = EP214 - Default value if not specified
1408	DefaultCstmApplVerID	String		The default DFIX FIX Client version number for the entire session and applies to all messages on this session. Supported values: 1.0
1409	SessionStatus	int		Session status at time of logon. Field is used when the logon messages is sent as an acknowledgment from the acceptor of the FIX session. Supported values: 0 = Session active (Default Logged in) 1 = Session password changed (Logged in)
58	Text	String		Available to provide a response to logon when used as a logon acknowledgement from acceptor back to the logon initiator.
	Standard Trailer		Y	

6.3. FIX Session Logout

The Logout(35=5) message initiates or confirms the termination of a FIX session. Disconnection without the exchange of logout messages should be interpreted as an abnormal condition. Either the FIX Client or DFIX Gateway can initiate a logout of the FIX session.

When the DFIX Gateway receives the Logout(35=5) message, it cancels all client orders in the OrderBook which are not fully executed, by sending Datatec withdrawal messages to the Matching Engine. The DFIX Gateway also sends to the FIX Client a Logout(35=5) message to indicate session logout complete and the connection is closed. **DFIX Gateway will not be sending ExecutionReport(35=8) messages to the client who is disconnecting, indicating their orders are cancelled. The client is responsible for ensuring their systems reflect this status of their outstanding orders; but DFIX Gateway will keep sending the ExecutionReport messages to the other FIX Clients in the same firm and clients with drop copy profile.**

If the FIX session terminates abruptly, the DFIX Gateway will also cancel all client orders in the OrderBook which are not fully executed, by sending Datatec withdrawal messages to the Matching Engine. **The DFIX Gateway will not be sending ExecutionReport(35=8) messages to the client who is disconnecting,**

indicating their orders are cancelled. The client is responsible for ensuring their systems reflect this status of their outstanding orders; but DFIX Gateway will keep sending the ExecutionReport messages to the other FIX Clients in the same firm and clients with drop copy profile. The FIX Client also has to try to reestablish a new FIX session.

Table 7 – Logout Message

Logout(35=5)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **5**
1409	SessionStatus	int		Session status at time of logout. Field is used when the logon messages is sent as a logon failure from the acceptor of the FIX session. Supported values: 3 = New session password does not comply with policy 4 = Session logout complete (Default) 5 = Invalid username or password 6 = Account locked 7 = Logons are not allowed at this time 8 = Password expired 9 = Received MsgSeqNum(34) is too low. 10 = Received Next Expected MsgSeqNum(789) is too high 1001 = Duplicated user (already connected) 1002 = Not Authorized 1003 = Wrong Profile 1004 = Wrong SenderCompID 1099 = Other
58	Text	String		Free format text string
	Standard Trailer		Y	

6.4. Logon Failures and Account Locked

All logon failures return a Logout(35=5) message with an appropriate reason code in the field SessionStatus(1409) and may include additional explanation in the field Text(58) which provides information regarding the failure. If the session initiator fails to authenticate (SessionStatus(1409) with values: 3 – New session password does not comply with policy, or 5 – Invalid username or password, or 8 - Password expired) with the DFIX Gateway within three attempts, on the third failed attempt the account will be locked and all subsequent logon attempts will be rejected. To unlock the account requires marketplace operations to reset the account. Any other causes for failure will not cause the account to be locked after the defined number of failed attempts.

6.5. Password Expired and Password Change

If the FIX Client's password has expired or the user has been authorized to connect using a temporary password, FIX Client will receive a Logout(35=5) message where the field SessionStatus(1409)=8. The client must start a new FIX session, sending the old password in the field Password(554) and the new in the field NewPassword(925).

A voluntary password change can only be done at logon time, and is done in the same way as an expired password is changed, sending the old password and the new password on Logon message.

When the DFIX Gateway receives a Logon(35=A) message with NewPassword(925), and the UserName(553) and current Password(554) are valid, the new password is checked against the password policy for compliance. If the new password complies, the FIX session is accepted by sending Logon(35=A) message with SessionStatus(1409)=1, and the new password is updated in the Datatec database and becomes the password to be used for subsequent session logons. If the new password does not comply with the password policy then an error status (SessionStatus(1409)=3) is returned in the Logout(35=5) message.

6.6. Logon Submitted Tags

The following table shows tags in Logon(35=A) message that must be submitted based on various scenarios where this message is sent.

Table 8 – Logon Submitted Tags

Tag	Logon without password change	Logon with Password Change	Logon Acknowledgment
EncryptMethod(98)	R	R	R
HeartBtInt(108)	R	R	R
Username(553)	R	R	
Password(554)	R	R	
NewPassword(925)		R	
DefaultApplVerID(1137)	R	R	R
DefaultApplExtID(1407)	C	C	C
DefaultCstmApplVerID(1408)	C	C	C
SessionStatus(1409)			C
Text(58)			C

C	Conditionally required – Based on Input or Error condition
R	Required as part of the message

6.7. Scenarios to establish FIX Sessions

There are three types of scenarios to establish FIX sessions:

1. At the start of the day, a new FIX session should be used to connect to the DFIX Gateway, with MsgSeqNum (34) set to 1 in the Logon(35=A) message. The FIX session should expect MsgSeqNum (34) in the reply message to start from 1.

2. If for any reason the FIX session is disconnected, the DFIX Gateway will remember all client subscriptions except Market Data subscriptions. The FIX Client should try to reconnect and establish the FIX session to the same DFIX Gateway. The Logon(35=A) message should set MsgSeqNum (34)

to the last MsgSeqNum (34) sent plus 1. A FIX session can only be reconnected during the same day. The FIX session should also expect the MsgSeqNum (34) from the DFIX Gateway to be the last MsgSeqNum (34) received plus 1. The standard ResendRequest(35=2) message can be used to retrieve messages missed during disconnection if a higher than expected MsgSeqNum (34) is received. After reconnect the DFIX Client does not need to send all subscription messages again, only subscriptions to Market Data need to be sent.

3. If the FIX session cannot be established with the same DFIX Gateway, it should fail over to a different DFIX Gateway. Connecting to a different DFIX Gateway is similar to a reconnection. If it's the first time the FIX Client is connecting to this failover DFIX Gateway the sequence numbers will begin from 1. If the FIX Client has already connected to this failover DFIX Gateway today, when the FIX Client fails over and connects again, inbound and outbound sequence numbers will be maintained consistent with the previous connection with this failover DFIX Gateway and resetting of sequence numbers is not required.

   When a FIX Client connects successfully to a specific DFIX Gateway, all other FIX Gateways receive a notification and remove any subscription that they may have received in any prior connection from this same FIX Client. For this reason, every time a FIX Client which was connected to one DFIX Gateway connects to another DFIX Gateway the FIX Client should send all the subscription messages again.

DFIX Gateway will adhere to the FIX Session behavior and recovery as prescribed by the FIX Session standard for FIXT.1.1 Session Layer Protocol[1].

6.8. Error Handling Messaging

If a message fails a session-level rule (e.g. body length is incorrect), a session-level Reject(35=3) message will be issued by the recipient. The Reject format is as follows.

Table 9 – Reject Message

Reject(35=3)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **3**
45	RefSecNum	SeqNum	Y	MsgSeqNum of rejected message.
371	RefTagID	int		The tag number of the FIX field being referenced
372	RefMsgType	String		The MsgType of the FIX message being referenced
373	SessionRejectReason	int		Code to identify reason for a session-level reject message. Supported values. 0 = Invalid Tag Number 1 = Required Tag Missing 2 = Tag not defined for this message type 3 = Undefined tag 4 = Tag specified without a value 5 = Value is incorrect (out of range) for this tag

[1] See http://www.fixtradingcommunity.org/pg/structure/tech-specs/session-level-specs

Reject(35=3)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
				6 = Incorrect data format for value 7 = Decryption problem 8 = Signature problem 9 = CompID problem 10 = SendingTime Accuracy Problem 11 = Invalid MsgType 12 = XML Validation Error 13 = Tag appears more than once 14 = Tag specified out of required order 15 = Repeating group fields out of order 16 = Incorrect NumInGroup count for repeating group 17 = Non "Data" value includes field delimiter (<SOH> character) 18 = Invalid/Unsupported Application Version 99 = Other
58	Text	String		Free format text string.
	Standard Trailer		Y	

Rejected messages should be logged and the incoming sequence number incremented.

For a FIX message which fulfills session-level rules and cannot be rejected via any other means, the BusinessMessageReject(35=j) message will be issued. The BusinessMessageReject format is as follows.

Table 10 – BusinessMessageReject Message

BusinessMessageReject(35=j)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **j** (lowercase)
45	RefSecNum	SeqNum		MsgSeqNum of rejected message.
372	RefMsgType	String	Y	The MsgType of the FIX message being referenced
379	BusinessRejectRefID	String		The value of the business-level "ID" field on the message being referenced.
380	BusinessRejectReason	Int	Y	Code to identify reason for a BusinessMessageReject message. Supported values. 0 = Other 1 = Unknown ID 2 = Unknown Security 3 = Unknown Message Type 4 = Application not available 5 = Conditionally required field missing 6 = Not Authorized
58	Text	String		Free format text string.
	Standard Trailer		Y	

6.9. Resend Request

The ResendRequest(35=2) message is sent by the receiving application to initiate the retransmission of messages. This function is utilized if a sequence number gap is detected, if the receiving application lost a message, or as a function of the initialization process.

The resend request can be used to request a single message, a range of messages or all messages subsequent to a particular message.

The ResendRequest format is as follows.

Table 11 – ResendRequest Message

ResendRequest(35=2)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **2**
7	BeginSeqNo	SeqNum	Y	Message sequence number of first message in range to be resent.
16	EndSeqNo	SeqNum	Y	Message sequence number of last message in range to be resent. If request is for a single message BeginSeqNo = EndSeqNo. If request is for all messages subsequent to a particular message, EndSeqNo = 0 (representing infinity).
	Standard Trailer		Y	

6.10. Sequence Reset

In the event that either system detects an error in sequence numbers, the SequenceReset(35=4) message is sent to rectify the situation.

The SequenceReset format is as follows:

Table 12 – SequenceReset Message

SequenceReset(35=4)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **4**
36	NewSeqNo	SeqNum	Y	New sequence number.
123	GapFillFlag	Boolean		Indicates that the Sequence Reset message is replacing administrative or application messages that will not be resent. Supported values: Y = Gap Fill message, MsgSeqNum field valid N = Sequence Reset, ignore MsgSeqNum

SequenceReset(35=4)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Trailer		Y	

6.11. Test Request

The TestRequest(35=1) message forces a heartbeat from the opposing application. The test request message checks sequence numbers or verifies communication line status. The opposite application responds to the Test Request with a Heartbeat containing the TestReqID.

Table 13 – TestRequest Message

TestReset(35=1)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **1**
112	TestReqID	String	Y	Identifier included in Test Request message to be returned in resulting Heartbeat.
	Standard Trailer		Y	

6.12. HeartBeat

The Heartbeat(35=0) message monitors the status of the communication link and identifies when the last of a string of messages was not received.

Table 14 – Heartbeat Message

Heartbeat(35=0)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **0**
112	TestReqID	String		Required when the heartbeat is the result of a Test Request message.
	Standard Trailer		Y	

7. Markets Definition

Once the FIX Client has established a FIX session, the FIX Client using MarketData or OrderManagement profiles must send a message requesting the definition of the markets to which the user is authorized to access. The request is made using MarketDefinitionRequest(35=BT) message with the following fields:

Table 15 – MarketDefinitionRequest Message

MarketDefinitionRequest(35=BT)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **BT**
1393	MarketReqID	String	Y	Unique identifier for the request, created by client
263	SubscriptionRequestType	char	Y	1 = Snapshot + Updates (Subscribe)
	Standard Trailer		Y	

When the DFIX Gateway receives the MarketDefinitionRequest message, it will send one MarketDefinition (35=BU) message for each Datatec market the user is authorized to view or participate. The format of the message is:

Table 16 – MarketDefinition Message

MarketDefinition(35=BU)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **BU**
1393	MarketReqID	String	Y	Unique identifier value carried from request
1394	MarketReportID	String	Y	Unique identifier created by the DFIX Gateway
1301	MarketID	Exchange	Y	This is the Market Identification Code (MIC) conforming to ISO-10383. Supported values: XBOG = Bolsa de Valores de Colombia XBCL = Bolsa Electrónica de Chile XLIM = Bolsa de Valores de Lima
1300	MarketSegmentID	String	Y	Identifies the market segment, assigned by Datatec Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
1396	MarketSegmentDesc	String		Short description or name of the Market Segment, assigned by Datatec
<InstrumentScopeGrp> component			Y	Repeating group used to specify the instruments that belong to the market segment
1656	NoInstrumentScopes	NumInGroup	Y	Number of instrument scopes. Datatec has defined one instrument for each market segment currently supported, and then this field comes with 1.

MarketDefinition(35=BU)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
→	1535	InstrumentScopeOperator	int	Y	Operator to perform on the instrument(s) specified Supported values: 1 = Include
→	**<InstrumentScope>** component			Y	Block used to specify the instrument
→	1536	InstrumentScopeSymbol	String	Y	Currency pair in CCY/CCY format. The supported values will be used in all messages that include the field Symbol(55). Supported values: USD/COP USD/CLP USD/PEN
→	End **<InstrumentScope>** component				
End **<InstrumentScopeGrp>** component					
15	Currency		Currency	Y	Identifies currency used for price. Supported values: COP = For market 71 and 76 CLP = For market 39 PEN = For market 51
<BaseTradingRules> component					Trading rules that are applicable to a market segment of trading session.
<TickRules> component					Repeating group, specifies the rules for determining how a price ticks.
1205	NoTickRules		NumInGroup		Number of tick rules. If the market on Datatec has defined basic fraction for price, this field comes with 1.
→	1206	StartTickPriceRange	Price		Required if NoTickRules(1205)> 0 It should always be 0 (None). Required by the FIX Protocol. FIX clients should not use the contents of this tag.
→	1208	TickIncrement	Price		Required if NoTickRules(1205)> 0 Basic fraction for price.
End **<TickRules>** component					
<PriceLimits> component					Specifies the price limits that are valid for trading. These are specified as ticks above the best sell order in the market, and ticks below the best buy order in the market. This range in ticks may be modified by the exchange during market hours, in which case a MarketDefinitionUpdateReport(35=BV) message with the new tick values will be sent.
1306	PriceLimitType		int		Describes how the price limits are expressed Supported values: 1 = Ticks
1148	LowLimitPrice		Price		Ticks down from the reference price used to determine whether the price of a new order is valid.

MarketDefinition(35=BU)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
1149	HighLimitPrice		Price		Ticks up from the reference price used to determine whether the price of a new order is valid.
End **<PriceLimits>** component					
1786	TradeVolType		int		Defines the type of units for MinTradeVol(562) and MaxTradeVol(1140) Supported values: 0 = Number of units (e.g. share, par, currency, contracts)
562	MinTradeVol		Qty		The minimum order or trade quantity/amount e.g. 250000
1140	MaxTradeVol		Qty		The maximum order or trade quantity/amount e.g. 50000000
1245	TradingCurrency		Currency	Y	Identifies currency used for amount. Supported values: USD = US Dollar
423	PriceType		int	Y	Defines the type of price used in the market. Supported values: 20 = Normal rate representation
End **<BaseTradingRules>** component					
<OrdTypeRules> component					Repeating group, specifies the order types that are valid for trading on market segment
1237	NoOrdTypeRules		NumInGroup		Number of order types, always 1.
→	40	OrdType	char		Required if NoOrdTypeRules(1237)> 0 Type of orders allowed in the market. Supported values: 2 = Limit - With amount and price, match against orders better or equal to the specified price
End **<OrdTypeRules>** component					
<TimeInForceRules> component					Repeating group, specifies the time in force rules that are valid for trading on market segment
1239	NoTimeInForceRules		NumInGroup		Number of time in force techniques
→	59	TimeInForce	char		Required if NoTimeInForceRules(1239)> 0 Specifies how long an order remains into the market. Supported values: 1 = Good Till Cancel (GTC) 3 = Immediate Or Cancel (IOC)
End **<TimeInForceRules>** component					
< ClrInstGrp > component					Repeating group, specifies the clearing conditions that are valid for trading on market segment * FIX 5.0 component added to message

MarketDefinition(35=BU)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
576	NoClearingInstructions	NumInGroup		Number of clearing instructions
→	577 ClearingInstruction	int		Required if NoClearingInstructions(576)> 0 Indicates clearing instructions allowed in the market segment. Supported values: 0 = Process normally 6 = Clear against central counterparty
End **<ClrInstGrp>** component				
	Standard Trailer		Y	

If there are subsequent changes in the definition of markets, these changes are sent to the FIX Clients using MarketDefinitionUpdateReport(35=BV) message with the following fields:

Table 17 – MarketDefinitionUpdateReport Message

MarketDefinitionUpdateReport(35=BV)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **BV**
1393	MarketReqID	String	Y	Unique identifier value carried from request
1394	MarketReportID	String	Y	Unique identifier created by the DFIX Gateway
1395	MarketUpdateAction	char	Y	Specifies the action taken for the specified MarketID(1301) + MarketSegmentID(1300). Supported values: A = Add D = Delete M = Modify
1301	MarketID	Exchange	Y	This is the Market Identification Code (MIC) conforming to ISO-10383. Supported values: XBOG = Bolsa de Valores de Colombia XBCL = Bolsa Electrónica de Chile XLIM = Bolsa de Valores de Lima
1300	MarketSegmentID	String	Y	Identifies the market segment, assigned by Datatec Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
1396	MarketSegmentDesc	String		Short description or name of the Market Segment, assigned by Datatec
<InstrumentScopeGrp> component			Y	Repeating group used to specify the instruments that belong to the market segment
1656	NoInstrumentScopes	NumInGroup	Y	Number of instrument scopes. Datatec has defined one

MarketDefinitionUpdateReport(35=BV)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					instrument for each market segment currently supported, and then this field comes with 1.
→	1535	InstrumentScopeOperator	int	Y	Operator to perform on the instrument(s) specified Supported values: 1 = Include
→	**<InstrumentScope>** component			Y	Block used to specify the instrument
→	1536	InstrumentScopeSymbol	String	Y	Currency pair in CCY/CCY format. The supported values will be used in all messages that include the field Symbol(55). Supported values: USD/COP USD/CLP USD/PEN
→	End **<InstrumentScope>** component				
End **<InstrumentScopeGrp>** component					
15	Currency		Currency	Y	Identifies currency used for price. Supported values: COP = For market 71 and 76 CLP = For market 39 PEN = For market 51
<BaseTradingRules> component					Trading rules that are applicable to a market segment of trading session.
<TickRules> component					Repeating group, specifies the rules for determining how a price ticks.
1205	NoTickRules		NumInGroup		Number of tick rules. If the market on Datatec has defined basic fraction for price, this field comes with 1.
→	1206	StartTickPriceRange	Price		Required if NoTickRules(1205)> 0 It should always be 0 (None). Required by the FIX Protocol. FIX clients should not use the contents of this tag.
→	1208	TickIncrement	Price		Required if NoTickRules(1205)> 0 Basic fraction for price.
End **<TickRules>** component					
<PriceLimits> component					Specifies the price limits that are valid for trading. These are specified as ticks above the best sell order in the market, and ticks below the best buy order in the market. This range in ticks may be modified by the exchange during market hours, in which case a MarketDefinitionUpdateReport(35=BV) message with the new tick values will be sent.
1306	PriceLimitType		int		Describes how the price limits are expressed Supported values: 1 = Ticks

MarketDefinitionUpdateReport(35=BV)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
1148	LowLimitPrice		Price		Ticks down from the reference price used to determine whether the price of a new order is valid.
1149	HighLimitPrice		Price		Ticks up from the reference price used to determine whether the price of a new order is valid.
End **<PriceLimits>** component					
1786	TradeVolType		int		Defines the type of units for MinTradeVol(562) and MaxTradeVol(1140) Supported values: 0 = Number of units (e.g. share, par, currency, contracts
562	MinTradeVol		Qty		The minimum order or trade quantity/amount e.g. 250000
1140	MaxTradeVol		Qty		The maximum order or trade quantity/amount e.g. 50000000
1245	TradingCurrency		Currency	Y	Identifies currency used for amount. Supported values: USD = US Dollar
423	PriceType		int	Y	Defines the type of price used in the market. Supported values: 20 = Normal rate representation
End **<BaseTradingRules>** component					
<OrdTypeRules> component					Repeating group, specifies the order types that are valid for trading on market segment
1237	NoOrdTypeRules		NumInGroup		Number of order types, always 1.
→	40	OrdType	char		Required if NoOrdTypeRules(1237)> 0 Type of orders allowed in the market. Supported values: 2 = Limit - With amount and price, match against orders better or equal to the specified price
End **<OrdTypeRules>** component					
<TimeInForceRules> component					Repeating group, specifies the time in force rules that are valid for trading on market segment
1239	NoTimeInForceRules		NumInGroup		Number of time in force techniques
→	59	TimeInForce	char		Required if NoTimeInForceRules(1239)> 0 Specifies how long an order remains into the market. Supported values: 1 = Good Till Cancel (GTC) 3 = Immediate Or Cancel (IOC)
End **<TimeInForceRules>** component					
< ClrInstGrp > component					Repeating group, specifies the clearing conditions that are valid for trading on market segment

MarketDefinitionUpdateReport(35=BV)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					* FIX 5.0 component added to message
576	NoClearingInstructions		NumInGroup		Number of clearing instructions
→	577	ClearingInstruction	int		Required if NoClearingInstructions(576)> 0 Indicates clearing instructions allowed in the market segment. Supported values: 0 = Process normally 6 = Clear against central counterparty
End **<ClrInstGrp>** component					
	Standard Trailer			Y	

8. Market Data

8.1. Introduction

This part of the document was included to try to give the reader a clear idea of the common practices on handling orders, used in FIX specifications for different markets. The text in this sub sections 8.1was taken from the "Recommended Practices for Book Management" written by the FIX Protocol Limited Market Data Working Group[2].

The FIX Market Data Working Group has defined a set of messages and protocols with the objective of creating a standard that allows FIX Clients to connect quickly and reliably across a range of markets.

This Working Group recognizes two general categories of markets. The first one is the style of market where orders and quotes are maintained in a central order book and is typical of the automated equity markets. The second one is a Quote market where quotes are streamed from many dealers and cannot be consolidated. The latter is typical of the bond and currency markets.

8.1.1. Markets with OrderBook

The use of a Central OrderBook is typical of electronic equity markets where executable orders are placed into a book, sorted by price, then time. A client can expect to execute at the price of the best order in the book or better. The markets are usually anonymous, where the displayed price is firm and available to all parties.

A Central OrderBook is characterized by two fundamental concepts; Precision and Depth.

Precision refers to the level of detail at which a book is kept and may be summarized by price level or comprised of individual orders. These are referred to as ***Price-Depth (Aggregate Order Book)*** and ***Order-Depth (Non-aggregated Order Book Detail)*** respectively.

Price-Depth is used to refer to the case where the aggregated quantity is provided for each side of each price level.

Order-Depth is used to refer to the case where every order is described and the client can recreate a copy of the OrderBook. There is limited information about each order. Details such as Identity, Account and Free Text are not available, except in a hit and take market where identity of the opposite party may be divulged.

Depth is the second aspect of book management, which refers to the number of entries that will be kept in a book. Book Depth may range from ***Top-Of-Book*** which represents only the best price level to ***Full Book Depth*** which represents all orders currently open in the market.

Different markets refer to the views with different names. The Working Group has selected a name that is generally acceptable so that the different views are not confused. A common name for the aggregate view is Price-Depth while the common name for the OrderBook Detail is Order-Depth. Alternate names for Top-Of-Book are Top-Of-File and Best Bid Offer or BBO.

[2] Version 2.0 dated January 2007 as found on the FIX website

The Working Group has defined a standard for these views that can be used across multiple markets. Given the fundamental characteristics of Aggregation and Depth, the following permutations are possible and are recommended as the most common book management practices:

- Top-Of-Book
- Partial Price-Depth
- Partial Order-Depth
- Full Price-Depth
- Full Order-Depth

Trades are sent as a separate instructions within a FIX Market Data Message (MDEntryType = "Trade"). Trades are only used to update the Ticker, or any display where Last, High, Low, Open, Close, Volume, and Cumulative Volume are displayed. **The Trade instruction is not used to update the view of the OrderBook.**

FIX provides two message formats that will be referred to within this document.

1. The Market Data Snapshot Full Refresh is used to provide a full snapshot of the current book. For this reason it is appropriate for top-of-book updates in which both sides of the book are sent. The message may also be used in a recovery situation where a number of order books need to be reestablished.

2. The Market Data Incremental Refresh message is used to apply instructions to a book in an incremental manner. This message is best suited for the maintenance of price-depth and order-depth books but may also be used for top-of-book.

8.1.2. Top-Of-Book

Server provides the current best bid and offer in the market. Trade details and Instrument Status are provided with separate Market Data Instructions.

Bid	Offer
250 @ 2348.90	500 @ 2349.10

Server may distribute other information beside Top-Of-Book, like trade information

Last	Qty	Open	High	Low	Chg	Tick	CumQty
2348.95	250000	2348.25	2349.10	2348.10	-0.15	-	10500000

Server updates the Top-Of-Book view on FIX Clients with the OVERLAY instruction in the Market Data Incremental Refresh message. Overlay means that client should replace values sent previously. The Bid and Ask Sides are updated independently with separate instructions. Many existing BBO (Best Bid Offer) feeds have fixed formats that provide both the bid and offer sides in all updates. The practice of sending separate instructions can provide efficiencies by allowing only the bid or ask to be sent, based on which side has changed, rather than both sides. Top-Of-Book only has one price level so there is no need to use multiple instructions when the best price changes. An action of *overlay* is used to indicate a new price or a change in quantity at an existing price. When the best price changes the server does not send a delete of one price level followed by a New for the next price level.

8.1.3. Price-Depth

Server provides the aggregate quantity available at each price level. This view can be visualized as a number of rows in a table for each of the bid and offer sides. On each side there are a number of rows showing the quantity available at a number of price levels.

Example:

Bid	Offer
100 @ 2348.90	300 @ 2349.10
65 @ 2347.85	1000 @ 2349.20
450 @ 2347.80	250 @ 2349.30

The Price-Depth view is an expansion of the Top-Of-Book view. A Price-Depth Book is sequenced by Price, descending for bid and ascending for ask.

Server maintains the Price-Depth view with the following entries:

New, to create/insert a new price level
Delete, to remove a price level
Change, to update quantity at a price level

The Bid and Ask Sides are updated with separate instructions.

Most markets do not show the Price-Depth view for the entire book. The Price-Depth view is limited to a number of levels, typically three to five. Equity Markets that trade in pennies tend to show at least ten price levels. The Recipient must know how many price levels are being supplied by the vendor and must delete the bottom row when the number of rows is exceeded. The server will not send a delete instruction when the number of rows is exceeded. The server will send the bottom row again when a higher level row is deleted.

If a trade occurs in the market then the server will send Delete or Change instructions to update the Price-Depth. The trade instruction itself is not used to update the order book as the order may not have executed against the book.

MDPriceLevel

The addition of the tag PriceLevel is used for markets that are not Price priority or in any situation where there can be confusion on the sequencing of the rows. PriceLevel is also beneficial for recovery after a gap in the data feed. If PriceLevel is present then the client can tell if a new price is the top price level and can delete higher price levels.

Indexing the Entry

Price-Depth entries can be referenced for maintenance purposes using several techniques. The first is to create a composite index which consists of the Security Identifier, MDEntryType (bid/Offer) and MDEntryPx. This set of fields acts as a composite key that allows the entry to be accessed and subsequently deleted.

The other approach is to use MDEntryID and assign a unique identifier to act as a key. Every active entry carries a unique ID which can be used for reference purposes. When the entry is no longer active, the ID can be re-used and assigned to the next active entry.

Both approaches have their advantages. The 'composite key' approach does not require that an external identifier to be carried on the initial Add or Delete instruction nor does it require the sending system to maintain a matrix of external identifiers. The MDEntryID approach provides a single field for referencing an entry which may improve system performance.

Note that use of MDEntryID does not mean that MDEntryType, MDEntryPx, or MDPriceLevel can be dropped on an Update or Delete instruction. These fields are necessary to ensure the integrity of the book when applying the instruction.

PriceLevel is a technique for conveying the position to which a given incremental instruction should be applied. The purpose is to reinforce the integrity of the instruction being received in relation to the current state of the book. The table below shows how PriceLevel can be useful in detecting a book inconsistency due to dropped message:

			OrderBook on client			
Current status			Level **1** **2** **3** **4** **5**	Bid 20@10 10@9 15@8 12@7 7@6	Offer 1@11 3@12 5@13 4@14 2@15	Level **1** **2** **3** **4** **5**
	Delete Bid@10 Level=1 Add Bid@5 Level=5, Amt=8	Dropped Message	Level **1** **2** **3** **4** **5**	Bid 20@10 10@9 15@8 12@7 7@6	Offer 1@11 3@12 5@13 4@14 2@15	Level **1** **2** **3** **4** **5**
Case 1	Change Bid@9Level=1, Amt=3 Change Bid@7 Level=3, Amt=9	User recognizes inconsistency of Bid@9 Level=1, removes Bid@10 and shifts all levels up by 1. Bid@7 can now be applied. Book is correct other than missing Bid@5	Level **1** **2** **3** **4**	Bid 3@9 15@8 9@7 7@6	Offer 1@11 3@12 5@13 4@14 2@15	Level **1** **2** **3** **4** **5**
Case 2	Change Bid@7 Level=3, Amt=9 Change Bid@6 Level=4, Amt=8	User recognizes inconsistency of Bid@7 Level=3 but does not know which levels above should be removed. Must Request miss message or Snapshot.	Level **1** **2** **3** **4** **5**	Bid 3@9 15@8 9@7 8@6 8@5	Offer 1@11 3@12 5@13 4@14 2@15	Level **1** **2** **3** **4** **5**

8.1.4. Order-Depth

The server provides limited information about each individual orders at each price level. Information is sent about each order at the price level rather that stating that a total quantity is available at each price level. The Client can summarize the information to create the Price-Depth view if desired.

It is possible in some markets to assume that orders can be sorted by price then time, however many markets do not prioritize all orders by time. The OrderBook can be viewed as a table with multiple orders at each price

level. Each order has a unique Entry ID. When adding orders to the book the Add instruction contains the EntryPositionNumber within the PriceLevel at which the order is to be inserted. Change and Delete instructions will reference the order by its MDEntryID. The MDEntryID is not to be confused the ClOrdID which is the confidential order reference assigned by the client when the order was entered. MDEntryID is an identifier assigned by the server to track this order in the market data feed and may be unique across the entire feed. FIX requires that the MDEntryID be unique among all other active entries,

Note that MDEntryID is constant for an entry until that entry is removed from the book due either to deletion or dropping out of the stated market depth.
Example:

Bid			Offer		
1	9740	ID=213, Qty=2	1	9760	ID=230, Qty=5
2	9730	ID=223, Qty=4; ID=227, Qty=1	2	9770	ID=231, Qty=3
3	9720	ID=973, Qty=1	3	9780	ID=232, Qty=5
			4	9790	ID=233, Qty=4; ID=234, Qty=3

The server must send the following instructions to maintain the OrderBook:

New, to insert a new order
Delete, to delete an order
Change, to alter the details of an order

New is used to insert an order, identified by MDEntryID, into a position within a price level. The new entry is sequenced by Price Level Number within the book and then Entry Position Number within that PriceLevel.

Change is used to change the quantity or other information related to an order. Change should not be used to alter the PriceLevel or EntryPositionNumber of an entry as this is the responsibility of the client maintaining the book. The trading function of Cancel & Replace that changes the price of an order would result in a delete of the old order and an insert of the updated order into the new price. 'Change' should not be used when the price of the order is changing due to a cancel/replace or when the order is being re-ranked for any reason. Any action which causes the order to move to a different price level or position should be accomplished through use of a 'delete' followed by a 'new'.

Delete and Change Instructions use MDEntryID as the key to reference the order. They can optionally include the PriceLevel and EntryPositionNumber of the order. This can aid client systems when the market has deep OrderBooks as it narrows the search for the subject order. It also provides a constant check regarding the integrity of the book.

The Recipient must know how many price levels are being supplied by the vendor and must delete the bottom row when the number of rows is exceeded. The server will not send a delete instruction when the number of rows is exceeded. The server will send the bottom row again when a higher level row is deleted.

8.1.5. Quote Markets

In a Quote Market there are dealers that provide quotes. There is no central market and the concept of a Central Order book does not exist. The prices displayed are not necessarily firm or available to all participants. A customer must contact the dealer to trade and the price is often negotiated as a points discount to the

published price. Orders are sent directly to the dealer. In some equity OTC markets, the orders can be sent to a central routing hub which then routes the orders to the end dealers.

In the debt markets the dealers provide quotes for different products based upon the popularity of the product. Some markets provide unsolicited quotes while others are solicited and require a call to the dealer to obtain a quote. The dealer may provide discounts to the quoted rate depending upon the rating of the client or other negotiated arrangements.

The dealer in a quote market will publish its prices to multiple vendors. The vendors seek to pass the dealer quotes to the client as a montage of prices showing the price and quantity available from each dealer. Some Vendors will take consolidated feeds from other vendors and further consolidate the quotes into higher level montages such as adding markets from multiple countries.

A standard to cover this style of market has the same requirements as the Price-Depth view on the OrderBook concept. The vendor will pass on quotes that have size and price however there can be many rows that have the same price, representing the different dealers. An argument can be made that it is also like an OrderBook view with multiple orders at each price level. It is considered to be closer to a Price-Depth view than an Order Depth view since the number of quotes in the market is static with the prices constantly changing rather than the number of orders changing.

The Vendor must issue instructions to maintain the view of each table:

New, to create a new price level
Delete, to remove a price level
Change, to update quantity at a price level

The Bid and Offer Sides are updated independently. A message may contain updates to many books and can update the bid and offer sides in the same message.

Each Dealer will be represented as its own price-level row and there can be many at the same price level.

Each price-level row must be assigned a unique MDEntryID, price-level, a price and a price-position along with the quantity available at that position. Change and Delete must then reference MDEntryID and Dealer reference to update the book. With this approach it is possible to show multiple rows at the same price show the rows in any sequence and minimize errors.

8.2. Market Data on DFIX Gateway

The DFIX Gateway currently supports Top-Of-Book and Full Order-Depth views, thus maintaining consistency between what FIX clients will receive and what native users currently receive on Datatec platform, where all users of the Datatec GUI front end who have access to the Spot markets are able to see all orders in the market.

For each order sent as part of Order-Depth view, the DFIX Gateway will calculate the trade condition depending on the user's credit conditions.

8.3. Market Data Subscription

The FIX Client must send a MarketDataRequest(35=V) message for the instruments that trade in the market segments whose definition has already been received. The user can only send one market segment and one instrument in each subscription request.

A subscription message where the NoRelatedSym(146) is “0” will result in a protocol violation and DFIX Gateway will reject the subscription.

A subscription message where the NoMarketSegments(1310) is not equal to “1” will result in an invalid subscription and DFIX Gateway will reject the subscription. DFIX Gateway will only support NoMarketSegments(1310)=1.

The format of the request message is as follows:

Table 18 – MarketDataRequest Message

MarketDataRequest(35=V)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
	Standard Header			Y	MsgType = **V**
262	MDReqID		String	Y	Unique identifier for the request, created by client
263	SubscriptionRequestType		Char	Y	Type of market data subscription request. Supported values: 1 = Snapshot + Updates (Subscribe) 2 = Disable previous Snapshot + Update Request (Unsubscribe)
264	MarketDepth		Int	Y	Depth of market for Book Snapshot / Incremental updates. Supported values: 0 = Full book depth 1 = Top of book
265	MDUpdateType		Int		Required if SubscriptionRequestType(263) = 1(Snapshot + Updates). Type of market data update. Supported values: 1 = Incremental refresh
266	AggrgatedBook		Boolean		Required if SubscriptionRequestType(263) = 1(Snapshot + Updates). Indicates whether the order book entries are aggregated or not. Supported values: N = Orders not aggregated by price level
<MDReqGrp> component				Y	Repeating group of MDEntryType in the subscription
267	NoMDEntryTypes		NumInGroup	Y	Number of MDEntryType fields in the request
→	269	MDEntryType	char	Y	The type of market data entry. Supported values: 0 = Bid 1 = Offer 2 = Trade 4 = Opening price 5 = Closing price

MarketDataRequest(35=V)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					7 = Trading session high price 8 = Trading session low price 9 = Trading session weighted average price B = Trade volume N = Session High Bid O = Session Low Offer * = All supported entry types - To indicate all available data
End **<MDReqGrp>** component					
<MarketSegmentScopeGrp> component				Y	Repeating group, used to limit the result set to the specified market segment
1310	NoMarketSegments		NumInGroup	Y	Number of market segments in the subscription. It must be 1.
→	1301	MarketID	Exchange	Y	Carried from MarketDefinition. This is the Market Identification Code (MIC) conforming to ISO-10383. Supported values: XBOG = Bolsa de Valores de Colombia XBCL = Bolsa Electrónica de Chile XLIM = Bolsa de Valores de Lima
→	1300	MarketSegmentID	String	Y	Carried from MarketDefinition. Identifies the market segment, it's assigned by Datatec Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
End **< MarketSegmentScopeGrp>** component					
<InstrmtMDReqGrp> component				Y	Repeating group used to limit the result set to a specific instrument. It must be 1.
146	NoRelatedSym		NumInGroup	Y	Number of instruments in the request
→	55	Symbol	String	Y	Currency pair in CCY/CCY format, from MarketDefinition InstrumentScopeSymbol(1536)
End **< InstrmtMDReqGrp>** component					
	Standard Trailer			Y	

If DFIX Gateway receives a new subscription request with the same or different MDReqID(262), the new request replaces the previous one. Each user is only allowed to have one active request.

Examples:

Time	MDReqID (262)	MarketDepth (264)	NoMDEntryTypes (267)	MDEntryType (269)					Comment
1	1	0	2	0	1				All current bids and current offers will be sent
2	1	0	3	0	1	2			All current bids, current offers and market trades will be sent
3	2	0	1	2					Stop sending bids and offers and continue sending market trades
4	2	0	1	*					All available data will be sent
5	3	0	2	0	1				All current bids and current offers will be sent

Market Data subscription is used to receive real-time market data; in the event of a FIX session disconnection, all active subscriptions will be cancelled. On a Market Data profile, FIX Client must reset sequence numbers on every logon using ResetSeqNumFlag(141)=Y and client needs to send another MarketDataRequest(35=V) message.

8.4. Market Data Subscription Rejected

The DFIX Gateway analyzes the request and may send a rejection message if the user does not have access to the requested information or if any field of the request is incorrect or not supported. Although several currency pairs may be included in the subscription request, one rejection message will be sent for each market segment and currency pair rejected.

The rejection is sent using the following message:

Table 19 – MarketDataRequestReject Message

MarketDataRequestReject(35=Y)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **Y**
262	MDReqID	String	Y	MDReqID value carried from MarketDataRequest
281	MDReqRejReason	char	Y	Reason for the rejection. Supported values: 0 = Unknown symbol 1 = Duplicated MDReqID 3 = Insufficient permissions 4 = Unsupported SubscriptionRequestType 5 = Unsupported MarketDepth 6 = Unsupported MDUpdateType 7 = Unsupported AggrgatedBook 8 = Unsupported MDEntryType A = Unsupported Scope
58	Text	String		Text of the reason for rejection, including the instrument that was rejected
	Standard Trailer		Y	

8.5. Market Data Subscription Accepted

If the subscription request is valid and the user is authorized to access the requested data, for each instrument and market in the subscription the DFIX Gateway sends one or more **Market Data – Snapshot/Full Refresh**(35=W)messages. The TotNumReports(911)field tells the client how many of these messages will be sent. All messages resulting from one request message will contain the same MDReqID carried from the request.

Although multiple instruments (currency pairs) can be contained in a single Subscription Request, a response Snapshot/Full Refresh message will be sent individually for each successfully subscribed market segment and currency pair for each type of view.

A Market Data Snapshot message may contain several trades, opening, closing, high, low, VWAP (Volume Weighted Average Price) prices, traded volume, as well as high bid and low offer; but only for one instrument per message.

Messages containing bids and/or offers cannot contain trades, opening, closing, high, low, VWAP prices, traded volume, high bid or low offer.

Table 20 – MarketDataSnapshotFullRefresh Message

MarketDataSnapshotFullRefresh(35=W)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **W**
911	TotNumReports	int		Total number of reports returned in response to a request. Only present in the last message sent.
262	MDReqID	String	Y	MDReqID value carried from MarketDataRequest
1301	MarketID	Exchange	Y	Identifier of the market which data belongs. This is the Market Identification Code (MIC) conforming to ISO-10383. Supported values: XBOG = Bolsa de Valores de Colombia XBCL = Bolsa Electrónica de Chile XLIM = Bolsa de Valores de Lima
1300	MarketSegmentID	String	Y	Identifier of the market segment to which data belongs. Identifies the market segment, it's assigned by Datatec Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component			Y	Identifies the instrument to which data belongs
55	Symbol	String	Y	Currency pair
End **<Instrument>** component				
1021	MDBookType	int		Required if this message include bid or offer entry types. Describes the type of book (view) for which the

MarketDataSnapshotFullRefresh(35=W)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					feed is intended Supported value: 1 = Top of Book 3 = Order-Depth
<MDFullGrp> component				Y	Repeating group with requested data
268	NoMDEntries		NumInGroup	Y	Number of repeating blocks to follow. Must be >= 1.
→	269	MDEntryType	char	Y	Type of market data entry. Supported values: 0 = Bid 1 = Offer 2 = Trade 4 = Opening price 5 = Closing price 7 = Trading session high price 8 = Trading session low price 9 = Trading session weighted average price B = Trade volume N = Session High Bid O = Session Low Offer J = Empty book
→	278	MDEntryID	String		Required when MDEntryType is 0 (Bid), 1 (Offer), N (Session High Bid) or O (Session Low Offer). Unique MarketData Entry identifier. Allows subsequent incremental changes to be applied using this field
→	270	MDEntryPx	Price		Required when MDEntryType is not B (Trade Volume) or J (Empty book). Price of the market data entry
→	271	MDEntrySize	Qty		Required when MDEntryType is 0 (Bid), 1 (Offer), 2 (Trade) or B (Trade Volume) Quantity or volume of the market data entry
→	272	MDEntryDate	UTCDateOnly		If MDEntryType is 2 (Trade), N (Session High Bid) or O (Session Low Offer), this field contains the trade date or price date (UTC Date only)
→	273	MDEntryTime	UTCTimeOnly		If MDEntryType is 2 (Trade), this field contains the match time from the Matching Engine. For a registered trade it contains the time entered by initiator. (UTC Time only). If MDEntryType is) N (Session High Bid) or O (Session Low Offer), this field contains the time when the price was the best.
→	1070	MDQuoteType	Int		Calculated when MDEntryType is 0 (Bid), 1 (Offer) and MDBookType = 3; client configuration could specify if this field is or not calculated. Also for MDBookType = 1, this field is not calculated. Identifies quote type.

MarketDataSnapshotFullRefresh(35=W)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					Supported values: 1010 = Tradable with bilateral credit 1011 = Tradable with bilateral credit and Clearing house credit 1013 = Not tradable, order from same firm 1014 = Not tradable, insufficient credit
→	110	MinQty	Qty		If MDEntryType is 0 (Bid) or 1 (Offer), this field contains minimum quantity of an order to be executed
→	37	OrderID	String		If MDEntryType is 0 (Bid) or 1 (Offer), this field contains the ClOrdID(11) that was submitted by the order owner. This field is sent only for order belonging to the FIX Client, otherwise omitted.
→	1003	TradeID	String		If MDEntryType is 2 (Trade), This field contains the unique trade identifier generated by Datatec.
→	828	TrdType	Int		If MDEntryType is 2 (Trade), this specifies the type of trade. Supported values: 0 = Regular trade (Default) 22 = Privately negotiated trade – Registered trade
→	829	TrdSubType	Int		If MDEntryType is 2 (Trade), this may contain additional trade type qualifiers. Used in conjunction with TrdType(828). Supported values: 36 = SWAP converted 1000 = Next Day converted 1001 = FIX converted
→	1023	MDPriceLevel	Int		If MDEntryType is 0 (Bid) or 1 (Offer), this field contains the display position of a bid or offer, numbered from most competitive to least competitive, per market segment an side, beginning with 1.
→	483	TransBkdTime	UTCTimestamp		If MDEntryType is 2 (Trade), this field contains the time of trade agreement for privately negotiated trades * FIX 5.0 field added to message
→	2445	AggressorTime	UTCTimestamp		If MDEntryType is 2 (Trade), this field contains the timestamp of aggressive order.
→	2446	AggressorSide	char		If MDEntryType is 2 (Trade), this field contains the side of aggressive order. Supported values: 1 =Buy 2 =Sell
→	1925	TradeClearingInstruction	int		If MDEntryType is 0 (Bid) or 1 (Offer), this field contains the clearing condition of the order. * FIX 5.0 field added to message

MarketDataSnapshotFullRefresh(35=W)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					Supported values: 0 = Process normally - Default if not specified. Not cleared against central counterparty. 6 = Clear against central counterparty
End **<MDFullGrp>** component					
	Standard Trailer			Y	

8.6. Market Data Incremental Refresh

The Market Data – Snapshot/Full Refresh(35=W)message may be followed by **Market Data – Incremental Refresh**(35=X) messages that can be sent in two ways: 1. Sent at predetermined time intervals with the latest changes that have happened in market data; and 2. Every time there is a change in market data. In the second of these, there may be an excess load of messages depending on how frequent are the changes in market data.

Separate Market Data – Incremental Refresh messages for each type of view will be sent; but the message may contain multiple instruments in a single message.

Table 21 – MarketDataIncrementalRefresh Message

MarketDataIncrementalRefresh(35=X)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **X**
1021	MDBookType	int		Required if this message include bid or offer entry types. Describes the type of book (view) for which the feed is intended 1 = Top of Book 3 = Order-Depth
262	MDReqID	String	Y	MDReqID value carried from MarketDataRequest
1301	MarketID	Exchange	Y	Identifier of the market which data belongs. This is the Market Identification Code (MIC) conforming to ISO-10383. Supported values: XBOG = Bolsa de Valores de Colombia XBCL = Bolsa Electrónica de Chile XLIM = Bolsa de Valores de Lima
1300	MarketSegmentID	String	Y	Identifier of the market segment to which data belongs. Identifies the market segment, it's assigned by Datatec Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot

MarketDataIncrementalRefresh(35=X)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
<MDIncGrp> component				Y	Repeating group with updated data
268	NoMDEntries		NumInGroup	Y	Number of repeating blocks to follow. Must be >= 1.
→	279	MDUpdateAction	char	Y	Must be the first field of this repeating group. Types of market data update action. Supported values: 0 = New 1 = Change 2 = Delete 3 = Delete Thru 4 = Delete from 5 = Overlay
→	269	MDEntryType	char	Y	Market data entry type. Supported values: 0 = Bid 1 = Offer 2 = Trade 4 = Opening price 5 = Closing price 7 = Trading session high price 8 = Trading session low price 9 = Trading session weighted average price B = Trade volume N = Session High Bid O = Session Low Offer J = Empty book
→	278	MDEntryID	String		Required when MDEntryType is 0 (Bid), 1 (Offer), N (Session High Bid) or O (Session Low Offer). Unique MarketData Entry identifier. Allows subsequent incremental changes to be applied using this field
→	**<Instrument>** component			Y	Component used to identify the “Instrument”
→	55	Symbol	String	Y	Currency pair
→	End **<Instrument>** component				
→	270	MDEntryPx	Price		Required when MDEntryType is not B (Trade Volume) or J (Empty book). Price of the market data entry
→	271	MDEntrySize	Qty		Required when MDEntryTypeis 0 (Bid), 1 (Offer), 2 (Trade) or B (Trade Volume) Quantity or volume of the market data entry
→	272	MDEntryDate	UTCDateOnly		If MDEntryType is 2 (Trade), this field contains the trade date (UTC Date only)
→	273	MDEntryTime	UTCTimeOnly		If MDEntryType is 2 (Trade), this field contains the match time from the Matching Engine. For a registered trade it contains the time entered by initiator. (UTC Time only)

MarketDataIncrementalRefresh(35=X)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
→	1070	MDQuoteType	int		Calculated when MDEntryType is 0 (Bid), 1 (Offer) and MDBookType = 3; client configuration could specify if this field is or not calculated. Also for MDBookType = 1, this field is not calculated. Identifies quote type. Supported values: 1010 = Tradable with bilateral credit 1011 = Tradable with bilateral credit and Clearing house credit 1013 = No tradable, order from same firm 1014 = No tradable, insufficient credit
→	110	MinQty	Qty		If MDEntryType is 0 (Bid) or 1 (Offer), this field contains minimum quantity of an order to be executed
→	37	OrderID	String		If MDEntryType is 0 (Bid) or 1 (Offer), this field contains the ClOrdID(11) that was created by the order owner
→	1003	TradeID	String		If MDEntryType is 2 (Trade), This field contains the unique trade identifier generated by Datatec.
→	828	TrdType	int		If MDEntryType is 2 (Trade), this specifies the type of trade. Supported values: 0 = Regular trade 22 = Privately negotiated trade – Registered trade
→	829	TrdSubType	int		If MDEntryType is 2 (Trade), this may contain additional trade type qualifiers. Used in conjunction with TrdType(828). Supported values: 36 = SWAP converted 1000 = Next Day converted 1001 = FIX converted
→	290	MDEntryPositionNo	int		If MDEntryType is 0 (Bid) or 1 (Offer), this field contains the position of a bid or offer inside of a price level. The DFIX Gateway always send 1
→	1023	MDPriceLevel	int		If MDEntryType is 0 (Bid) or 1 (Offer), this field contains the display position of a bid or offer
→	483	TransBkdTime	UTCTimestamp		If MDEntryType is 2 (Trade), this field contains the time of trade agreement for privately negotiated trades
→	2445	AggressorTime	UTCTimestamp		If MDEntryTypeis 2 (Trade), this field contains the Timestamp of aggressive order.
→	2446	AggressorSide	char		If MDEntryTypeis 2 (Trade), this field contains the side of aggressive order. Supported values: 1 =Buy 2 =Sell

MarketDataIncrementalRefresh(35=X)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
→	1925	TradeClearingInstruction	int		If MDEntryType is 0 (Bid) or 1 (Offer), this field contains the clearing condition of the order. * FIX 5.0 field added to message Supported values: 0 = Process normally (default, not cleared against central counterparty) 6 = Clear against central counterparty
End **<MDIncGrp>** component					
	Standard Trailer			Y	

8.7. Market Data Unsubscribe

A FIX Client can unsubscribe from the current active subscription by sending a MarketDataRequest(35=V) message setting fields MDReqID(262) equal to the last value sent on subscription and SubscriptionRequestType(263) equal 2 (Disable previous Snapshot + Update Request).Sending update messages related to the subscription being disabled will be discontinued.

Other required fields in the MarketDataRequest message must be sent to satisfy the FIX Protocol, but these will be ignored by the DFIX Gateway.

9. Credit Limits

Bilateral Limits

FX Spot markets on Datatec like USD/COP, USD/CLP and USD/PEN use bilateral credit limits, where each firm that wants to trade in a market must define the amount of credit limits against their counterparties. The amounts are defined in three ways:

1) **Buy/Sell Credit Limits**: Defined as separate limit amounts, one for purchases(buy limit) and other for sales (sell limit).For each one the remaining credit balance is calculated as the credit limit minus traded amounts(utilized amount) for the given side. The side is from the perspective of the party setting the limit against counterparties.

   Example:

   Customer A extends 10 million buy credit and 12 million sell credit to CustomerB
   If Customer A buys 4 million from CustomerB, the resulting buy credit limit balance will be 6 million. Subsequently, if Customer A sells 1 million to CustomerB, the resulting sell credit limit balance will be 11 million.

2) **Netted Credit Limits**: Defined as separate limit amounts, one for purchases and other for sales, where the remaining credit balance is calculated as the limit minus traded amounts for one side plus traded amounts of the other side.

   Example:

   Customer A extends 10 million buy credit and 12 million sell credit to CustomerB.
   If Customer A buys 4 million from CustomerB, the resulting netted credit limit balance will be 6 million for buy credit and 16 million for sell credit. Subsequently, if CustomerA sells 1 million to CustomerB, the resulting netted credit limit balance will now be 7 million for buy credit and 15 million for sell credit.

   Note the difference compared to Buy/Sell Credit Limits. With netted credit limits, buys affect the balance of the buy credit limit **and** the balance of the sell credit limit. Similarly, sells also affect both sell and buy credit limit balances.

3) **Gross Credit Limits**: Defined as a single gross amount, the remaining credit balance is calculated as the limit minus all traded amounts (purchases and sales).

   Example:

   CustomerA extends 10 million gross credit to CustomerB.
   If CustomerA buys 4 million from CustomerB, the resulting gross credit limit balance will be 6 million. Subsequently, if Customer A sells 1 million to CustomerB, the resulting gross credit limit balance will be 5 million.

For bilateral credit limits there is a credit limit owner (the party setting the limit) and the counterparty. Only the owner can see and change the amounts assigned to their counterparties. The counterparty can never see the amount of credit that others have assigned to it. Only authorized users within a firm can change the amounts of

credit limits for their own firm and only on a schedule that is set in each market. It's normally configured to allow changes to credit limits only outside of market hours.

Clearinghouse Limits

For USD/COP FX Spot market, there is a Clearinghouse (Cámara de Compensación de Divisas – CCD) which also assigns a netted buy and sell credit limit for each participant to trade with all the other counterparties. Each counterparty is only entitled to see the amount of their own clearinghouse assigned credit limit and the current balance.

For both bilateral and clearinghouse limits, a party's credit balance is used to determinate if orders displayed in the market are tradable or not tradable for each counterparty. This condition is indicated for each order in the MDQuoteType(1070) field.

The messages involved in this section are:

- PartyRiskLimitsRequest(35=CL)
- PartyRiskLimitsReport(35=CM)
- PartyRiskLimitsUpdateReport(35=CR)

All these messages are defined in 5.0SP2 FIX Protocol.

9.1. Credit Limits Request

The **PartyRiskLimitsRequest**(35=CL) message is used to request credit limits information for specific parties, specific party roles or specific instruments. As stated above, a firm may only request information about its own credit limits extended to other counterparties in the market segment or request credit limits extended to its own firm by the clearing house.

Table 22 – PartyRiskLimitsRequest Message

PartyRiskLimitsRequest(35=CL)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **CL**
1666	RiskLimitRequestID	String	Y	Unique identifier for the request, created by client
1760	RiskLimitRequestType	int	Y	Type of risk limits information. Supported values: 1 = Definitions 3 = Definitions and utilization
263	SubscriptionRequestType	char	Y	Subscription request type. Supported values: 1 = Snapshot + Updates (Subscribe) 2 = Disable previous Snapshot + Update Request (Unsubscribe)
<RequestingPartyGrp> component				Required when SubscriptionRequestType(263) = 1. Repeating group used to identify the party making the request and their role.

PartyRiskLimitsRequest(35=CL)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
1657	NoRequestingPartyIDs		NumInGroup		If specified, always 1.
→	1658	RequestingPartyID	String		Required when NoRequestingPartyIDs > 0 Party identifier of the requester – It should contain a 3-character Firm ID.
→	1659	RequestingPartyIDSource	char		Required when NoRequestingPartyIDs > 0 Identifies the source of RequestingPartyID value. Supported values: C = Generally accepted market participant identifier - Datatec mnemonic
→	1660	RequestingPartyRole	int		Required when NoRequestingPartyIDs > 0 Identifies the role of RequestingPartyID Supported values: 13 = Order Origination Firm 21 = Clearing House
End **<RequestingPartyGrp>** component					
<Parties> component					Repeating group used to delimit the scope of the request for specific party(-ies)
453	NoPartyIDs		NumInGroup		If the client wants to receive the credit limits defined against specifics counterparties the client transacts with, this field contains the number of counterparties.
→	448	PartyID	String		Required when NoPartyIDs > 0. Party identifier – It should contain a 3-character Firm ID
→	447	PartyIDSource	char		Required when NoPartyIDs > 0. Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier - Datatec mnemonic
→	452	PartyRole	int		Required when NoPartyIDs > 0. Identifies the role of RequestingPartyID Supported values: 17 = ContraFirm - This will be the Credit Limits Counterparty Firm 21 = Clearing House
End **<Parties>** component					
1533	RiskLimitPlatform		String	Y	The area to which credit limits is applicable. It identifies the market segment which credit limit values belong, Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
	Standard Trailer			Y	

If the market segment only uses bilateral credit limits, FIX Clients can request the amounts of credit limits that their own institution has extended to all counterparties by sending the PartyRiskLimitsRequest(35=CL) message with its own 3-character Firm ID in RequestingPartyID(1658) and RequestingPartyRole(1660) = 13, and the Parties component is omitted.

If the market segment uses clearing house credit limits, the owner of the clearing house credit limits is the clearing house and the counterparties are the other institutions (market participants). Market participants requesting information on the credit limits extended by clearing house to them should send in the PartyRiskLimitsRequest(35=CL) with its own 3-character Firm ID in RequestingPartyID(1658), RequestingPartyRole(1660) = 13, PartyID(448) with the clearing house 3–character Firm ID and PartyRole(452) = 21.

If the FIX Client is the clearing house and wants to request the amounts of credit limits that the clearing house has extended to market participants, the request should include its own 3-character Firm ID in RequestingPartyID(1658) and RequestingPartyRole(1660) = 21. The Parties component is not needed in this case.

9.2. Credit Limits Request Rejected

When the DFIX Server receives PartyRiskLimitsRequest(35=CL)message, it analyzes the information received and checks permissions, then using the **PartyRiskLimitsReport**(35=CM) message will send a rejection of the request or will send all information requested. If the request is rejected, all required fields of the message will be sent and RequestResult(1511) will carry the code of the rejection and RejectText(1328) a text explaining the reason for rejection.

The complete structure of the PartyRiskLimitsReport(35=CM) message is included on the next sub section.

9.3. Credit Limits Request Accepted

If the request was not rejected, then the DFIX Gateway will send the information of credit limits in one or more messages, the client should use the value of the field TotNoParties(1512) to find out how many counterparties in total will be received and the field LastFragment(893) to find out which the last party is.

The PartyRiskLimitsReport(35=CM) message with credit limits information has the following structure:

Table 23 – PartyRiskLimitsReport Message

PartyRiskLimitsReport(35=CM)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **CM**
1667	RiskLimitReportID	String	Y	Unique identifier for the report created by DFIX Gateway
1666	RiskLimitRequestID	String	Y	The RiskLimitRequestID value carried from PartyRiskLimitsRequest
1760	RiskLimitRequestType	int	Y	The type of request. Supported values:

PartyRiskLimitsReport(35=CM)						
Tag	**Field Name**			**DataType**	**Reqd**	**Comments**
						1 = Definitions 3 = Definitions and utilization
1511	RequestResult			int	Y	The result of the request. Supported values: 0 = Valid request 1 = Invalid or unsupported request 2 = No data found that match selection criteria – but the request is valid 3 = Not authorized to receive data 4 = Data temporarily unavailable 5 = Request for data not supported 99 = Other
1512	TotNoParties			int		Required when RequestResult(1511) = 0 (Valid request). Total number of counterparties as a result of the request
893	LastFragment			Boolean		Required when RequestResult(1511) = 0 (Valid request). Indicates whether this message is the last N = Not last message Y = Last message
<PartyRiskLimitsGrp> component						Required when RequestResult(1511) = 0 (Valid request). Repeating group of parties and the risk limits for the party
1677	NoPartyRiskLimits			NumInGroup		Number of counterparties included in this message
→	**<PartyDetailGrp>** component					Required if NoPartyRiskLimits(1677) > 0 Repeating group, contains details for a party
→	1671	NoPartyDetails		NumInGroup		If specified, always 1
→	→	1691	PartyDetailID	String		Required if NoPartyDetails(1671) > 0 Party identifier – 3-character Firm ID
→	→	1692	PartyDetailIDSource	char		Required if NoPartyDetails(1671) > 0 Identifies the source of PartyDetailID Supported values: C = Generally accepted market participant identifier - Datatec mnemonic
→	→	1693	PartyDetailRole	int		Required if NoPartyDetails(1671) > 0 Identifies the role of PartyDetailID Supported values: 17 = Contra Firm - This will be the Counterparty Firm 21 = Clearing House
→	→	**<RelatedPartyDetailGrp>** component				Required if the credit limits owner is using credit limits by branches. Repeating group that include the owner branch.

PartyRiskLimitsReport(35=CM)							
Tag	**Field Name**				**DataType**	**Reqd**	**Comments**
→	→	1562	NoRelatedPartyDetailID		NumInGroup		If specified, always 1
→	→	→	1563	RelatedPartyDetailID	String		Required if NoRelatedPartyDetailID(1562) > 0 Branch identifier – 8-character Branch ID
→	→	→	1564	RelatedPartyDetailIDSource	char		Required if NoRelatedPartyDetailID(1562) > 0 Identifies the source of Related PartyDetailID Supported values: C = Generally accepted market participant identifier - Datatec mnemonic
→	→	→	1565	RelatedPartyDetailRole	int		Required if NoRelatedPartyDetailID(1562) > 0 Supported values: 75 = Location ID
→	→	End **<RelatedPartyDetailGrp>** component					
→	→	1672	PartyDetailStatus		int		Indicates the status of the party in credit limits definition Supported values: 0 = Active - Default if not specified. 1 = Suspended
→	End **<PartyDetailGrp>** component						
→	**<RiskLimitsGrp>** component						Required if NoPartyRiskLimits(1677) > 0 Repeating group, contains credit limits for the party.
→	1669	NoRiskLimits			NumInGroup		If specified, always 1
→	→	**<RiskLimitTypesGrp>** component					Required if NoRiskLimits(1669) > 0 Repeating group of credit limit types and values
→	→	1529	NoRiskLimitTypes		NumInGroup		There may be "1" or "2" instances specified. If the firm is configured for gross limits, this field will be "1". If firm is configured for Buy/Sell credit limits or Buy/Sell net limits, this field will be "2".
→	→	→	1530	RiskLimitType	int		Required if NoRiskLimitTypes(1529) > 0 Specifies the type of credit limit If the firm is configured for gross limit, only RiskLimitType=1 (Gross limit) is specified. If the firm is configured for buy/sell limits or buy/sell net limit, the first instance will represent the buy limit when RiskLimitType=0 (Credit limit) or 2 (Net limit); the second instance will represent the sell limit when RiskLimitType=0 (Credit limit) or 2 (Net limit). Supported values: 0 = Credit limit 1 = Gross limit 2 = Net limit
→	→	→	1531	RiskLimitAmount	Amt		Required if NoRiskLimitTypes(1529) > 0 Specifies the amount of credit limit that was setup. If there is no amount setup "0" will be sent

PartyRiskLimitsReport(35=CM)							
Tag	**Field Name**				**DataType**	**Reqd**	**Comments**
→	→	→	1766	RiskLimitUtilization Amount	Amt		Required if NoRiskLimitTypes(1529) > 0 Contains the current utilized amount. If there is no utilized amount "0" will be sent
→	→	→	1533	RiskLimitPlatform	String		Required if NoRiskLimitTypes(1529) > 0 The area to which credit limits is applicable. It Identifies the market segment which credit limit values belong, Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
→	→	→	1633	LimitAmtRemaining	Amt		Required if NoRiskLimitTypes(1529) > 0 The balance amount remaining calculated after each trade. * FIX 5.0 field added to message
→	→	End **<RiskLimitTypesGrp>** component					
→	End **<RiskLimitsGrp>** component						
→	1670	RiskLimitID			String		Required if NoPartyRiskLimits(1677) > 0 Unique reference identifier of the credit limits entry assigned by Datatec.
End **<PartyRiskLimitsGrp>** component							
1328	RejectText				String		Identifies the reason for rejection
	Standard Trailer					Y	

9.4. Credit Limits Update Report

After the DFIX Gateway has sent all the credit limit information requested by the client, DFIX Gateway will send **PartyRiskLimitsUpdateReport**(35=CR) messages at the time that any change either by new limit definitions, modifications or deletion of limit definitions, or by market transactions that change the utilized amount.

The PartyRiskLimitsUpdateReport message has the following structure:

Table 24 – PartyRiskLimitsUpdateReport Message

PartyRiskLimitsUpdateReport(35=CR)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **CR**
1667	RiskLimitReportID	String	Y	Unique identifier for the report created by DFIX Gateway

PartyRiskLimitsUpdateReport(35=CR)							
Tag	**Field Name**				**DataType**	**Reqd**	**Comments**
1666	RiskLimitRequestID				String	Y	The RiskLimitRequestID value carried from PartyRiskLimitsRequest
<PartyRiskLimitsUpdateGrp> component						Y	Repeating group of parties and the risk limits for the party
1677	NoPartyRiskLimits				NumInGroup		Number of counterparties included in this message
→	1324	ListUpdateAction			char		Required if NoPartyRiskLimits(1677) > 0 The type of update. Supported values: A = Add D = Delete M = Modify
→	**<PartyDetailGrp>** component						Required if NoPartyRiskLimits(1677) > 0 and ListUpdateAction(1324) is A or M; when ListUpdateAction(1324) = D, this component is not included in the **PartyRiskLimitsUpdateGrp** component. Repeating group, contains details for a party
→	1671	NoPartyDetails			NumInGroup		If specified, always 1
→	→	1691	PartyDetailID		String		Required if NoPartyDetails(1671) > 0 Party identifier – 3-character Firm ID
→		1692	PartyDetailIDSource		char		Required if NoPartyDetails(1671) > 0 Identifies the source of PartyDetailID Supported values: C = Generally accepted market participant identifier - Datatec mnemonic
→	→	1693	PartyDetailRole		int		Required if NoPartyDetails(1671) > 0 Identifies the role of PartyDetailID Supported values: 17 =Contra Firm– This will be the Counterparty Firm 21 = Clearing House
→	→	**<RelatedPartyDetailGrp>** component					Required if the credit limits owner is using credit limits by branches. Repeating group that include the owner branch.
→	→	1562	NoRelatedPartyDetailID		NumInGroup		If specified, always 1
→	→	→	1563	RelatedPartyDetailID	String		Required if NoRelatedPartyDetailID(1562) > 0 Branch identifier – 8-character Branch ID
→	→	→	1564	RelatedPartyDetailIDSource	char		Required if NoRelatedPartyDetailID(1562) > 0 Identifies the source of PartyDetailID Supported values: C = Generally accepted market participant identifier - Datatec mnemonic
→	→	→	1565	RelatedPartyDetailRole	int		Required if NoRelatedPartyDetailID(1562) > 0 Supported values:

PartyRiskLimitsUpdateReport(35=CR)							
Tag	**Field Name**				**DataType**	**Reqd**	**Comments**
							75 = Location ID
→	→	End **<RelatedPartyDetailGrp>** component					
→	→	1672	PartyDetailStatus		int		Indicates the status of the party in credit limits definition Supported values: 0 = Active -Default if not specified 1 = Suspended
→	End **<PartyDetailGrp>** component						
→	**<RiskLimitsGrp>** component						Required if NoPartyRiskLimits(1677) > 0 Repeating group, contains credit limits for the party
→	1669	NoRiskLimits			NumInGroup		If specified, always 1
→	→	**<RiskLimitTypesGrp>** component					Required if NoRiskLimits(1669) > 0 Repeating group of credit limit types and values
→	→	1529	NoRiskLimitTypes		NumInGroup		If the firm is configured for gross limits, this field will be "1". If firm is configured for Buy/Sell credit limits or Buy/Sell net limits, this field will be "2".
→	→	→	1530	RiskLimitType	int		Required if NoRiskLimitTypes(1529) > 0 Specifies the type of credit limit If the firm is configured for gross limit, only RiskLimitType=1 (Gross limit) is specified. If the firm is configured for buy/sell credit limit or buy/sell net limit, the first instance will represent the buy limit when RiskLimitType=0 (Credit limit) or 2 (Net limit); the second instance will represent the sell limit when RiskLimitType=0 (Credit limit) or 2 (Net limit). Supported values: 0 = Credit limit 1 = Gross limit 2 = Net limit
→	→	→	1531	RiskLimitAmount	Amt		Required if NoRiskLimitTypes(1529) > 0 Specifies the amount of credit limit that was setup. If there is no amount setup"0" will be sent.
→	→	→	1766	RiskLimitUtilization Amount	Amt		Required if NoRiskLimitTypes(1529) > 0 Contains the current utilized amount. If there is no utilized amount"0" will be sent
→	→	→	1533	RiskLimitPlatform	String		Required if NoRiskLimitTypes(1529) > 0 The area to which credit limits is applicable. Identifies the market segment which credit limit values belong, Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day

PartyRiskLimitsUpdateReport(35=CR)							
Tag	**Field Name**				**DataType**	**Reqd**	**Comments**
							39 = CL Dollar Spot 51 = PE Dollar Spot
→	→	→	1633	LimitAmtRemaining	Amt		Required if NoRiskLimitTypes(1529) > 0 The balance amount remaining calculated after each trade. * FIX 5.0 field added to message
→	→	End **<RiskLimitTypesGrp>** component					
→	End **<RiskLimitsGrp>** component						
→	1670	RiskLimitID			String		Required if NoPartyRiskLimits(1677) > 0 Unique identifier of the credit limit. This will contain a new identifier when ListUpdateAction=A, otherwise it will contain the same identifier as established in the PartyRiskLimitsReport message if updating or deleting an existing credit limit "record".
End **<PartyRiskLimitsGrp>** component							
	Standard Trailer					Y	

9.5. Credit Limits Unsubscribe

A FIX Client can unsubscribe from the previous subscription sending a PartyRiskLimitsRequest(35=CL) message setting fields RiskLimitRequestID(1666) equal to the last value sent on subscription and SubscriptionRequestType(263) equal 2 (Disable previous Snapshot + Update Request). Sending update messages related to the subscription being disabled will be discontinued.

Other required fields in the PartyRiskLimitsRequest message must be sent to satisfy the FIX Protocol, but these will be ignored by the DFIX Gateway.

10.Order Management

The order management section consists of a description of how messages are used for entering, modifying and canceling (withdraw) orders. This section also describes how messages are used for hit and take operations.

The messages involved in this section are:

- NewOrderSingle(35=D)
- OrderCancelRequest(35=F)
- OrderCancelReplaceRequest(35=G)
- OrderCancelReject(35=9)
- OrderMassActionRequest(35=CA)
- OrderMassActionReport(35=BZ)
- OrderStatusRequest(35=H)
- ExecutionReport(35=8)

10.1. Uniqueness of ClOrdID(11)

The DFIX Gateway will check the uniqueness of ClOrdID(11)in the NewOrderSingle(35=D), OrderCancel/ReplaceRequest(35=G) and OrderCancelRequest(35=F) messages. FIX Clients submitting these messages must ensure uniqueness of the values they use in the ClOrdID(11) field within a single trading day (Datatec markets do not accept multi-day orders) in order to avoid receiving rejections from DFIX Gateway.

10.2. Order Identification

In OrderCancel/ReplaceRequest(35=G)or OrderCancelRequest(35=F) messages, the order can be identified by either the ClOrdID(11) value of the order being cancelled or amended by sending it in the OrigClOrdID(41) field, or by the OrderID(37) received from Datatec.

10.3. Cross Protocol Order Management

All orders entered by Datatec Frontend Users (i.e. from the Datatec GUI), will be communicated to FIX Clients from the same branch using ExecutionReport(35=8) messages. In these reports the field ClOrdID(11) will not be set but the OrderID(37) field will carry the ID assigned by DFIX Gateway. These orders can be amended, canceled or status requested from FIX Client providing the valid OrderID(37) and the field OrigClOrdID(41) should be omitted.

Orders entered via FIX Clients can be amended or cancelled using the Datatec frontend. FIX Clients must be prepared to receive unsolicited order updates and cancels.

10.4. Workflows

The figures below describe the workflow for different situations.

In the diagrams, **Marketplace** includes all applications on Datatec side, like DFIX Gateway, DataServer, Matching Engine and DMServer.

Figure 3 – Order Entry Workflow

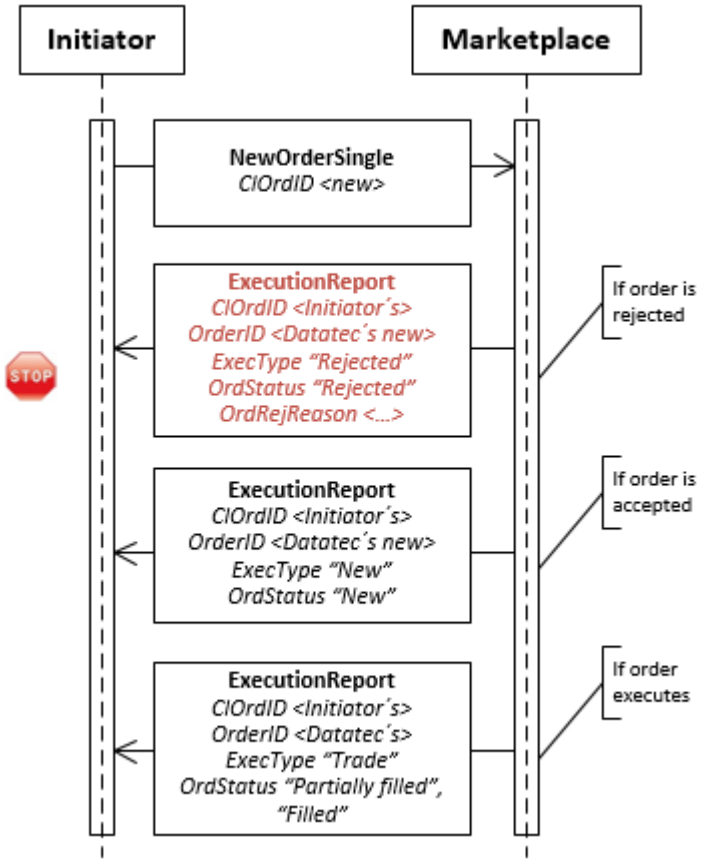


Figure 4 – Order with IOC Entry Workflow

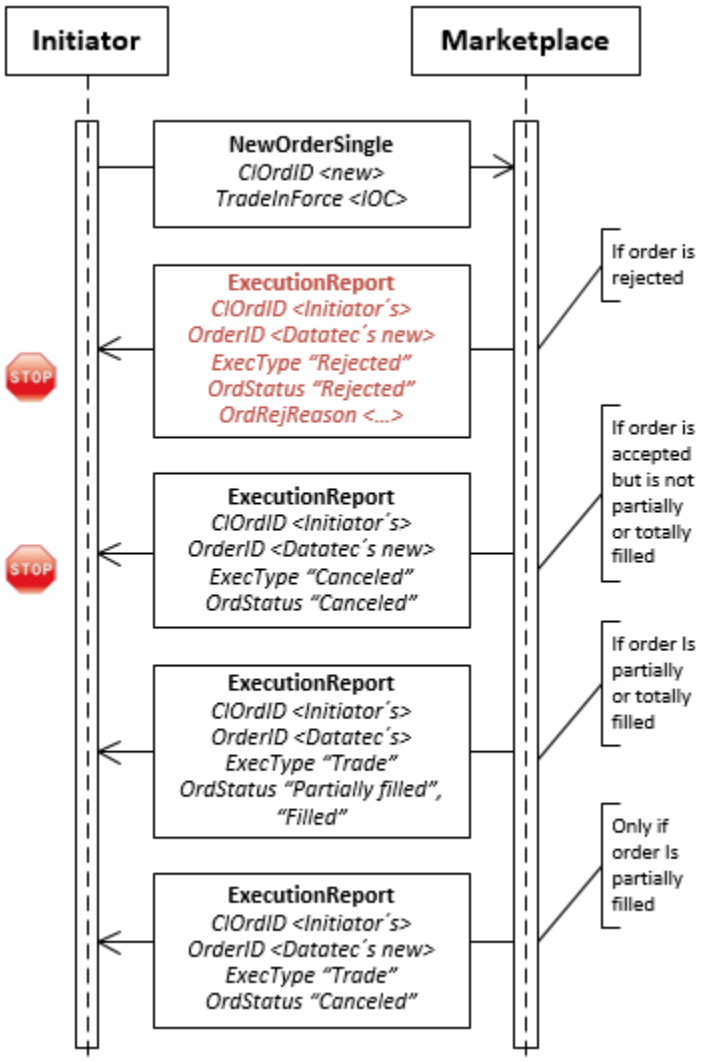


Figure 5 – Order Modification Workflow

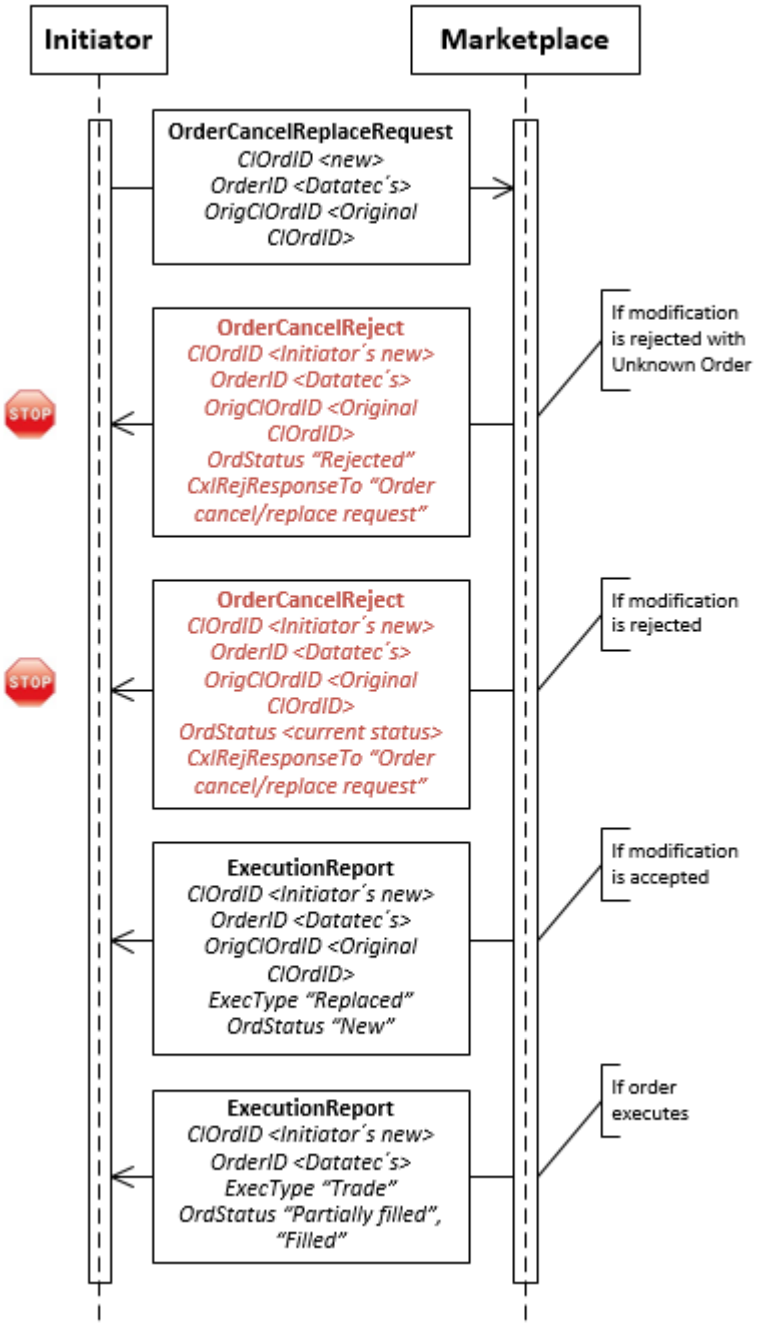


Figure 6 – Order Cancellation Workflow

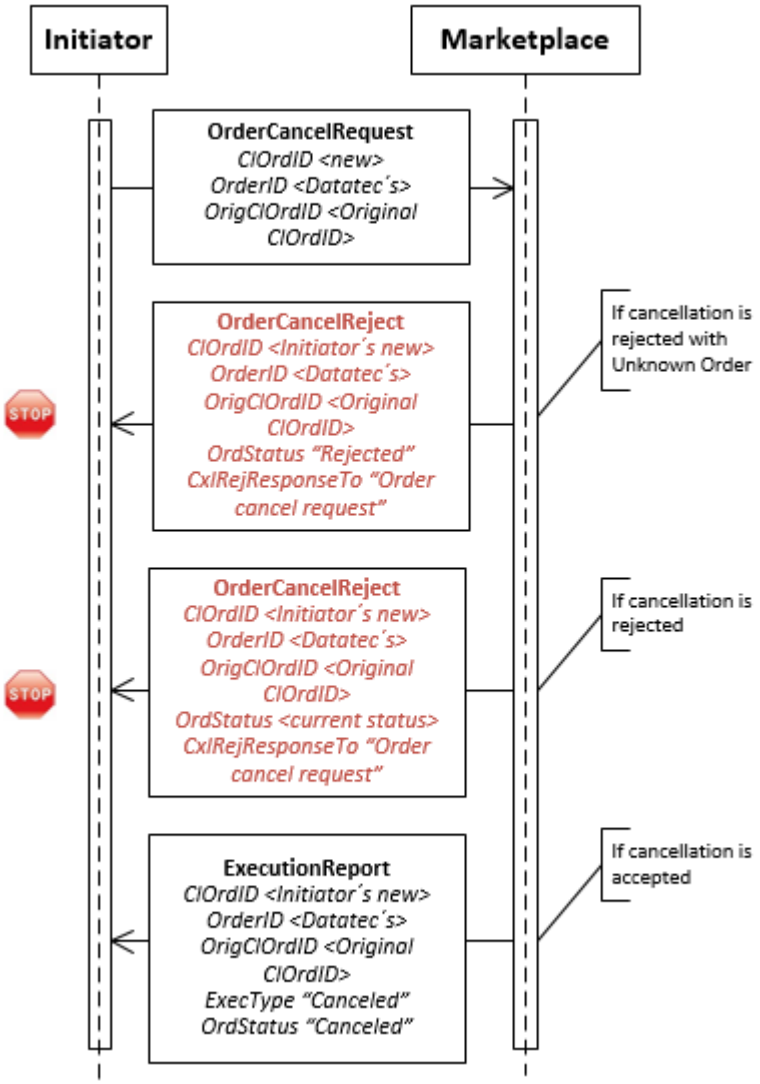


Figure 7 – Order Entered via FIX and Amended via Datatec Frontend (Cross protocol)

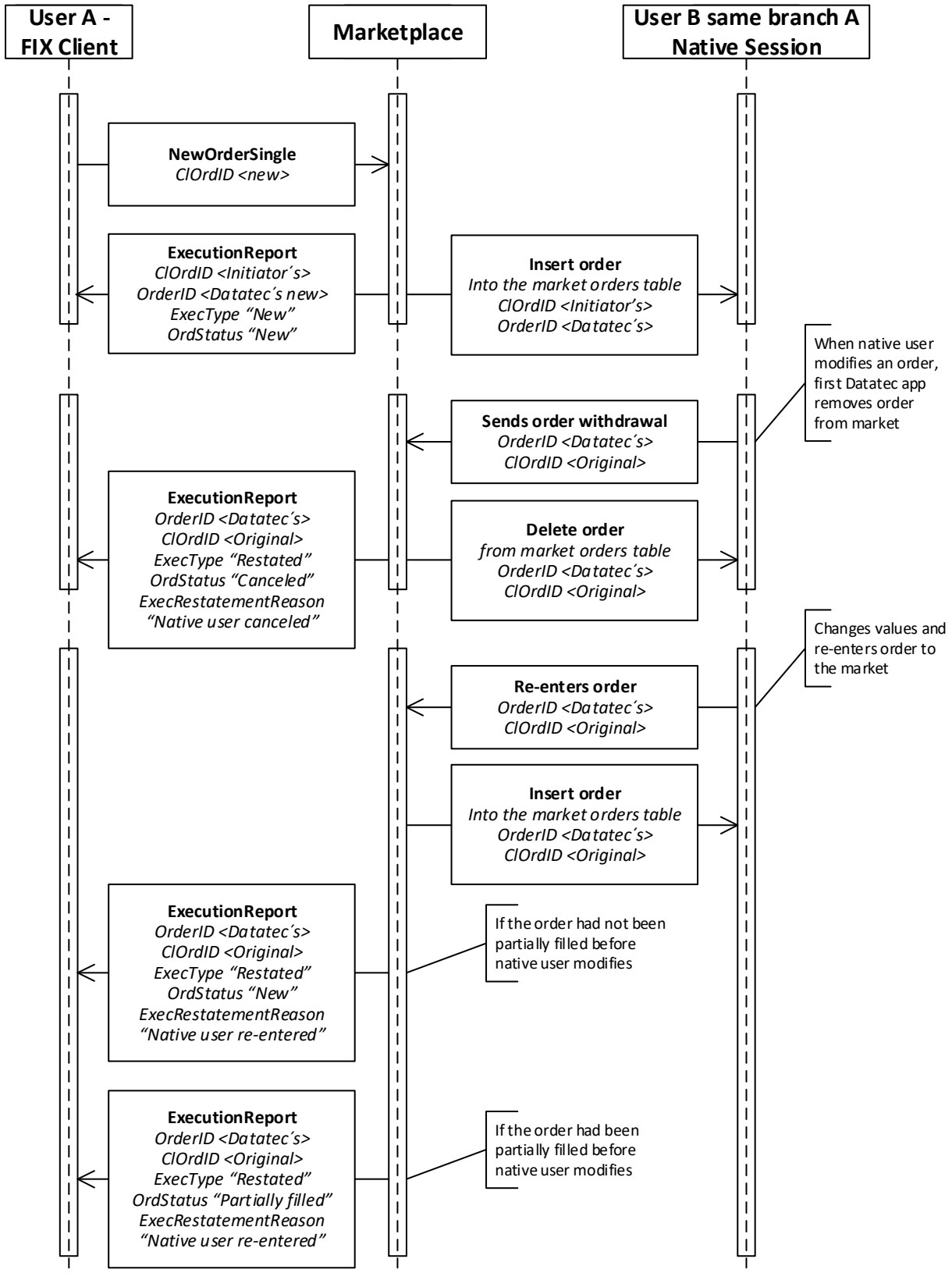


Figure 8 – Order Entered via Datatec Frontend and Amended via FIX (Cross protocol)

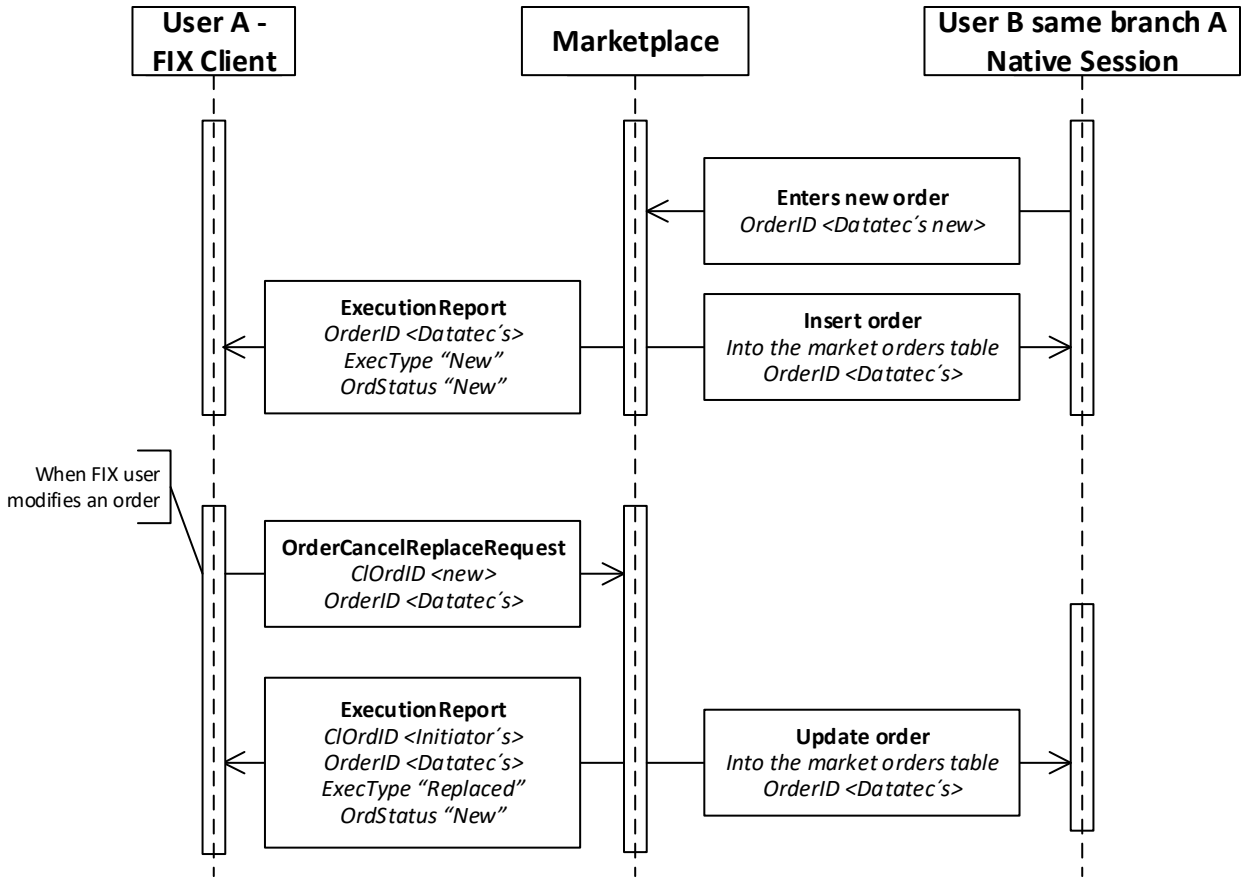
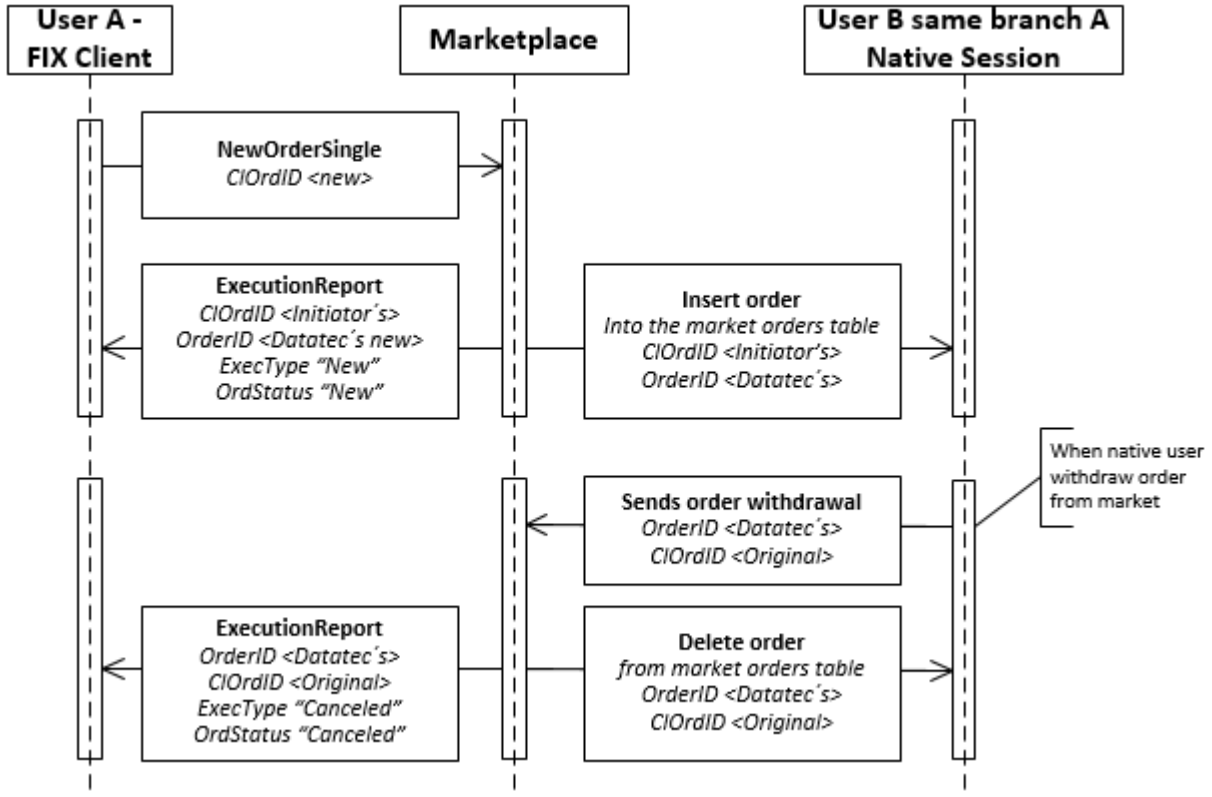


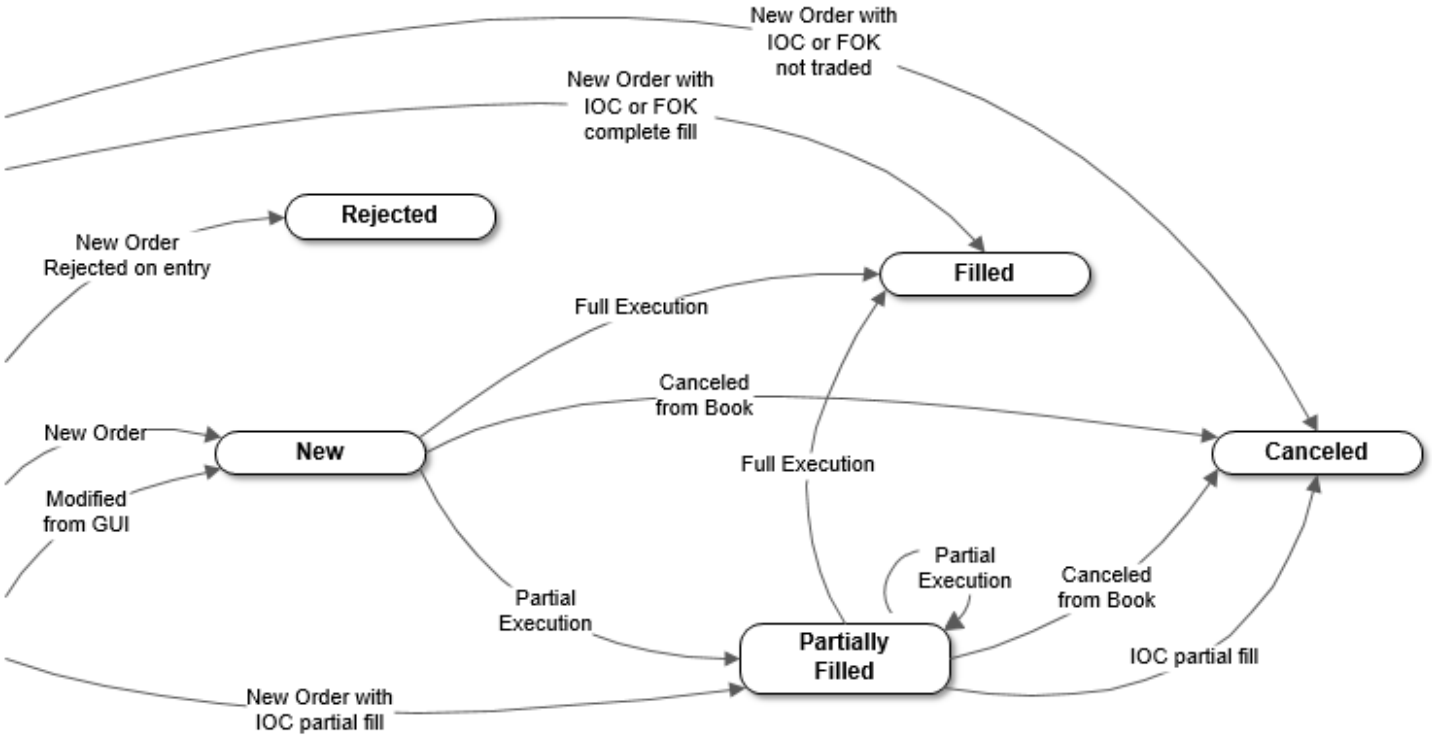
Figure 9 – Order Entered via FIX and Cancelled via Datatec Frontend (Cross protocol)



10.5. Order Status

Order state changes are conveyed in ExecutionReport(35=8) messages. Every state change is communicated in a separate ExecutionReport. The OrdStatus(39) field specifies the state of the order and the ExecType(150) specifies the purpose of the ExecutionReport.

Figure 10 – Order Status States



10.6. Order Submit Request

Orders submitted need to include the side (bid / offer), the instrument (currency pair and type of market); and a unique identifier set by the order submitter for the order, which is sent in the ClOrdID(11) field. The same ID should be referenced in the OrigClOrdID(41) field when modifying or cancelling the order. The DFIX Gateway will send acceptance or rejection messages for the order entry.

Orders should not be submitted for an instrument if the FIX Client has not received a successful Market Definition Subscription response. The DFIX Gateway will validate that this condition is fulfilled before accepting the order; and if it has not, the order submission will be refused.

Orders must be submitted one by one. Batch order submissions are not supported.

The FIX standard **NewOrderSingle**(35=D) message is used for the order submission as detailed below:

Table 25 – NewOrderSingle Message

NewOrderSingle(35=D)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
	Standard Header			Y	MsgType = **D**
11	ClOrdID		String	Y	Unique identifier of the order created by FIX Client. ID should be unique to simplify modifying and cancelling of orders.
<Parties> component				Y	Component used to identify the order submitter (Branch, User and Customer)
453	NoPartyIDs		NumInGroup	Y	Minimum 2, maximum 3.
→	448	PartyID	String	Y	Party identifier, first block should be the 8-character Branch ID, second block should be the Trader short name, and if there is a third block, it should be a beneficiary code.
→	447	PartyIDSource	char	Y	Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier (Datatec mnemonic)
→	452	PartyRole	int	Y	Identifies the role of RequestingPartyID Supported values: 11 = Order Origination Trader 13 = Order Origination Firm 32 = Beneficiary
End **<Parties>** component					
1300	MarketSegmentID		String	Y	Identifies the market segment where the order is being traded, assigned by Datatec Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component				Y	Component used to identify the “Instrument”
55	Symbol		String	Y	Currency pair in CCY/CCY format, from MarketDefinition InstrumentScopeSymbol(1536)
End **<Instrument>** component					
54	Side		char	Y	Side of the order from order submitter's perspective. Supported values: 1 = Buy - Bid 2 = Sell – Offer
<OrderQtyData> component				Y	Component block contains the fields commonly used for indicating the amount or quantity of an order
38	OrderQty		Qty	Y	Quantity of the order, which must be greater or equal to the minimum of the market and less than or equal to the maximum and multiple of the basic fraction
End **<OrderQtyData>** component					

NewOrderSingle(35=D)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
40	OrdType	Char	Y	Order type Supported values: 2 = Limit - mount and price included; will match against orders better or equal to the specified price
44	Price	Price	Y	Price per unit of the quantity
110	MinQty	Qty		Applies only if the market allows, it takes the minimum amount that can be taken / hit in an order
1822	MinQtyMethod	Int		Required if MinQty(110) is present. Supported values: 1001 = Multiple (applies to every execution), if LeavesQty (151) is less than MinQty(110), order will be canceled.
<DisplayInstruction> component				Component block is used to convey instructions on how a reserved order is to be handled in terms of when and how much of the order quantity is to be displayed to the market
1138	DisplayQty	Qty		For Iceberg orders, this is the amount shown to other market participants
1083	DisplayWhen	Char		Required if DisplayQty(1138) is present. Supported values: 1 = Immediate (after each fill) - At the beginning and after each execution
End **<DisplayInstruction>** component				
59	TimeInForce	Char	Y	Supported values: 1 = Good Till Cancel (GTC) 3 = Immediate Or Cancel (IOC)
60	TransactTime	UTCTimestamp	Y	UTC date and time of the order creation
1925	TradeClearingInstruction	int		This field contains one of the clearing conditions allowed in the market segment for orders * FIX 5.0 field added to message Supported values: 0 = Process normally (default, not cleared against central counterparty) 6 = Clear against central counterparty
	Standard Trailer		Y	

10.7. IOC orders

An IOC (Immediate Or Cancel) order is an order that must be filled immediately at the limit price or better. If the order is partially filled or cannot be filled, the unfilled portion will be cancelled.

An IOC order with a value in DisplayQty(1138) (Iceberg IOC order), will be sent to the Matching Engine by the full amount of the order, without taking into account the DisplayQty.

When DFIX Gateway receives the order entry, it creates a unique identifier for the order regardless of whether the order is accepted or rejected, this value will be conveyed in the OrderID(37) field of the ExecutionReport(35=8). Datatec then validates the received information and if the order is not valid, returns an **ExecutionReport**(35=8) message with the reason for rejection. If the order passes the validation tests, the order is sent to market participants and the DFIX Gateway returns an ExecutionReport(35=8) message indicating that the order was accepted and successfully entered into the market.
Subsequent ExecutionReport(35=8) messages will be sent to the FIX Client reporting the status of the order as it changes as a result of activity, such as trading or order amendment.

10.8. Execution Report

The ExecutionReport(35=8) message is used to:

a. Confirm the receipt of an order
b. Reject orders
c. Confirm changes to an existing order (i.e. Cancel or Cancel/Replace requests)
d. Report order status information (e.g. responding to OrderStatusRequest)
e. Report fill information on working orders

When responding to an OrderStatusRequest(35=H), the ExecutionReport(35=8) for an order with an OrdStatus(39) of either "cancelled" or "filled" will only contain the mandatory fields. Non-mandatory fields will not be provided.

Table 26 – ExecutionReport Message

ExecutionReport(35=8)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **8**
37	OrderID	String	Y	Unique identifier for the order created by DFIX Gateway
11	ClOrdID	String		Required when responding to an order submitted by FIX Clients. Unique identifier carried from NewOrderSingle(35=D) When responding to a cancel request this contains the ClOrdID from the OrderCancelRequest(35=F) message. When responding to a modification request this contains the ClOrdID from the OrderCancelReplaceRequest(35=G) message. The value becomes the new ClOrdID for the successfully amended order. When rejecting an OrderStatusRequest(35=H) this contains the OrdStatusReqID from the request.
17	ExecID	String	Y	Unique identifier for the execution created by DFIX Gateway (will be 0 (zero) for ExecType(150)=I (Order Status)).
41	OrigClOrdID	String		Conditionally required when responding to a Cancel request or Cancel/Replace request. It contains the original ClOrdID value of the order being cancelled or modified/amended.

ExecutionReport(35=8)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
150	ExecType	Char	Y	Describes the purpose of or reason for the execution report. Supported values: 0 = New - Accepted 4 = Canceled 5 = Replaced 8 = Rejected D = Restated F = Trade (partial fill or fill) I = Order Status
39	OrdStatus	Char	Y	Identifies the current status of the order Supported values: 0 = New - Accepted 1 = Partially filled 2 = Filled 4 = Canceled 6 = Pending Cancel (i.e. result of Order Cancel Request) 8 = Rejected A = Pending New E = Pending Replace (i.e. result of Order Cancel/Replace Request)
103	OrdRejReason	Int		Required when ExecType(150)=8 (Rejected) Code to identify the reason for order rejection Supported values: 1 = Unknown symbol 2 = Exchange closed 3 = Order exceeds limit 4 = Too late to enter 5 = Unknown order 6 = Duplicated ClOrdID 11 = Unsupported order characteristic 13 = Incorrect quantity 16 = Price exceeds current price band 18 = Invalid price 25 = Insufficient credit limit 99 = Other
1328	RejectText	String		Conditionally required when ExecType(150)=8 (Rejected). Additional text describing the reason for rejection
378	ExecRestatementReason	int		Required if ExecType(150) = D (Restated) The reason for restatement 1000 = GUI Native user canceled 1001 = GUI Native user re-entered
1300	MarketSegmentID	String	Y	Identifies the market segment of the order as specified in the order Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot

ExecutionReport(35=8)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
				51 = PE Dollar Spot
<Instrument> component			Y	Component used to identify the “Instrument”
55	Symbol	String	Y	Currency pair
End **<Instrument>** component				
54	Side	char	Y	Side of the order. Supported values: 1 = Buy - Bid 2 = Sell - Offer
<OrderQtyData> component			Y	Component block contains the fields commonly used for indicating the amount or quantity of an order
38	OrderQty	Qty		Conditionally required when ExecType(150) is not I (Order status) and OrdStatus(39) is not 4 (Cancelled) or 2 (Filled). Quantity of the order
End **<OrderQtyData>** component				
40	OrdType	char		Conditionally required when ExecType(150) is not I (Order status) and OrdStatus(39) is not 4 (Cancelled) or 2 (Filled). Order type Supported values: 2 = Limit - Amount and price included; will match against orders better or equal to the specified price
44	Price	Price		Conditionally required when ExecType(150) is not I (Order status) and OrdStatus(39) is not 4 (Cancelled) or 2 (Filled). Price on order
110	MinQty	Qty		Applies only if the market allows. It takes the minimum amount that can be taken / hit in an order
1822	MinQtyMethod	int		Required if MinQty(110) is present. Supported values: 2 = Multiple (applies to every execution) 1001 = Multiple (applies to every execution), if LeavesQty (151) is less than MinQty(110), order will be canceled.
<DisplayInstruction> component				Component block is used to convey instructions on how a reserved order is to be handled in terms of when and how much of the order quantity is to be displayed to the market
1138	DisplayQty	Qty		For Iceberg orders, here is the amount shown on orders windows
1083	DisplayWhen	char		Required if there is DisplayQty(1138) Supported values: 1 = At the beginning and after each execution
End **<DisplayInstruction>** component				
59	TimeInForce	char	Y	Supported values: 1 = Good Till Cancel (GTC) 3 = Immediate Or Cancel (IOC)

ExecutionReport(35=8)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
60	TransactTime		UTCTimestamp	Y	UTC date and time of the ExecutionReport message.
75	TradeDate		LocalMktDate		Indicates date of trade referenced in this message in YYYYMMDD format. Absence of this field indicates current day
63	SettlType		String		Specific trade settlement period Supported values: 0 = Regular / FX Spot settlement (T+1 or T+2 depending on currency) 1 = Cash (TOD / T+0) 2 = Next Day (TOM / T+1) 3 = T+2 4 = T+3
64	SettlDate		LocalMktDate		Specific date of trade settlement in YYYYMMDD format.
14	CumQty		Qty	Y	Total executed quantity
151	LeavesQty		Qty	Y	Quantity open for further execution. If the OrdStatus (39) is Canceled or Rejected (in which case the order is no longer active) then LeavesQty could be "0", otherwise LeavesQty = OrderQty(38) - CumQty(14).
31	LastPx		Price		Price of this (last) fill. Required if ExecType(150) = F (Trade)
32	LastQty		Qty		Quantity on this (last) fill. Required if ExecType(150) = F (Trade)
880	TrdMatchID		String		Identifier assigned to a trade by a matching system. Required if ExecType(150) = F (Trade)
1057	AggressorIndicator		Boolean		Used to identify whether the order initiator is an aggressor or not in the trade. Supported values: Y = Order initiator is aggressor N = Order initiator is passive
<Parties> component					Repeating group used to specify parties information related to an order or related to a trade. For an order, there may be up to 3 parties: a pair branch and trader and one beneficiary. For a trade, there may be up to 6 parties: two pairs branch and trader as market participants, one beneficiary and a clearing house. For clearing house party the 3-character Firm ID will be sent. For beneficiary its compounded code. For other parties involved the 8-character Branch ID and the Trader short name will be sent.
453	NoPartyIDs		NumInGroup		Number of parties related to a trade.
→	448	PartyID	String		Required when NoPartyIDs > 0. Party identifier
→	447	PartyIDSource	char		Required when NoPartyIDs > 0.

ExecutionReport(35=8)						
Tag	**Field Name**			**DataType**	**Reqd**	**Comments**
						Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier -Datatec mnemonic
→	452	PartyRole		int		Required when NoPartyIDs > 0. Identifies the role of PartyID Supported values: 11 = Order originator Trader 13 = Order originator Firm 17 = Contra Firm 21 = Clearing House 32 = Beneficiary 37 = Contra Trader
→	**<PtysSubGrp>** component					Required when NoPartyIDs > 0. Repeating group of Party sub-identifiers.
→	802	NoPartySubIDs		NumInGroup		Number of PartySubIDs
→	→	523	PartySubID	String		Sub-identifier, DFIX Gateway will send the firm or branch name, or trader name, or the city name
→	→	803	PartySubIDType	int		Type of PartySubID(523) value. Supported values: 1 = Firm – Firm full name or Branch full name 2 = Person – Full name 34 = Address City – Related to the branch
→	End **<PtysSubGrp>** component					
End **<Parties>** component						
58	Text			String		Text describing the status of the order
1925	TradeClearingInstruction			int		This field contains one of the clearing conditions allowed in the market segment for orders * FIX 5.0 field added to message Supported values: 0 = Process normally (default, not cleared against central counterparty) 6 = Clear against central counterparty
	Standard Trailer				Y	

As can be seen, most fields for this message are the same as in the order entry message; in fact, they are copied from the entry message.

The Status and Purpose of the response messages (ExecutionReport) is defined by the values in fields OrdStatus(29) and ExecType(150).

10.9. Order Amend

Amending an order is a more efficient method of changing an order instead of cancelling and sending in a new order. The Order Amend message allows Client applications to change both the price and the size, or to change only the price, or to change only the size, of an active Order. Changing the price may change the position of the order in the Order Book. A Reduction of the order size, with no price change, will not affect the position of the order in the Order Book. Increasing of the order size, without a price change, will cause the order to lose the priority it had over other orders at that price.

It should be noted that the Order Amend should not be used to decrease the size of the order to zero, use the Order Interrupt Request for that purpose.

The **OrderCancelReplaceRequest**(35=G) message is used to change the parameters of an existing order. Do not use this message to cancel the remaining quantity of an order, use the **OrderCancelRequest**(35=F) message for this purpose.

Although the FIX protocol allows for virtually all of the Order attributes to be changed, there are limitations as to what the Datatec systems allows. The following attributes are allowed to change:

- OrderQty (38)
- Price (44)
- TradeClearingInstruction(1925)
- MinQty(110)
- DisplayQty (1138)

Note: Any change to the price of an order or increasing quantities will result in the order losing its position in the order book.

The FIX standard **OrderCancelReplaceRequest** (35=G) message is as follow:

Table 27 – OrderCancelReplaceRequest Message

OrderCancelReplaceRequest(35=G)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **G**
11	ClOrdID	String	Y	Unique Identifier for the Amend request, created by the FIX Client (Request ID). This identifier will be used in the ClOrdID tag of the: - Cancel Reject message if the Amend Order request is rejected. - Execution Report in response to the Amend request if accepted, it becomes the client's order ID for the accepted amend order.
37	OrderID	String		Unique Identifier for the original Order that was assigned by DFIX Gateway and sent in the ExecutionReport as a response to original order submission.
41	OrigClOrdID	String		Original ClOrdID of the order that is being requested to be

OrderCancelReplaceRequest(35=G)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					replaced/amended by this message. Conditionally required for orders submitted via FIX.
<Parties> component				Y	Component used to identify the order submitter (Branch, User and Customer)
453	NoPartyIDs		NumInGroup	Y	Minimum 2, maximum 3.
→	448	PartyID	String	Y	Party identifier, first block should be the 8-character Branch ID, second block should be the Trader short name, and if there is a third block, it should be a customer code.
→	447	PartyIDSource	char	Y	Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier (Datatec mnemonic)
→	452	PartyRole	int	Y	Identifies the role of RequestingPartyID Supported values: 11 = Order Origination Trader 13 = Order Origination Firm 32 = Beneficiary
End **<Parties>** component					
1300	MarketSegmentID		String	Y	Identifies the market segment, assigned by Datatec Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component				Y	Component used to identify the “Instrument”
55	Symbol		String	Y	Currency pair in CCY/CCY format, from MarketDefinition InstrumentScopeSymbol(1536)
End **<Instrument>** component					
54	Side		char	Y	Side of the order from order submitter's perspective. This must be the same as the original order. Supported values: 1 = Buy - Bid 2 = Sell – Offer
<OrderQtyData> component				Y	Component block contains the fields commonly used for indicating the amount or quantity of an order
38	OrderQty		Qty	Y	Quantity of the order, which must be greater or equal to the minimum of the market and less than or equal to the maximum and multiple of the basic fraction. OrderQty cannot be set to "0".
End **<OrderQtyData>** component					
40	OrdType		char	Y	Order type Supported values:

OrderCancelReplaceRequest(35=G)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
				2 = Limit -Amount and price included; will match against orders better or equal to the specified price
44	Price	Price	Y	Price per unit of the quantity
110	MinQty	Qty		Applies only if the market allows, it takes the minimum amount that can be taken / hit in an order
1822	MinQtyMethod	int		Required if MinQty(110) is present. Supported values: 1001 = Multiple (applies to every execution), if LeavesQty (151) is less than MinQty(110), order will be canceled.
<DisplayInstruction> component				Component block is used to convey instructions on how a reserved order is to be handled in terms of when and how much of the order quantity is to be displayed to the market
1138	DisplayQty	Qty		For Iceberg orders, here is the amount shown to other market participants
1083	DisplayWhen	char		Required if there is DisplayQty(1138) Supported values: 1 = Immediate (after each fill) - At the beginning and after each execution
End **<DisplayInstruction>** component				
59	TimeInForce	char	Y	Supported values: 1 = Good till cancel(GTC)
60	TransactTime	UTCTimestamp	Y	UTC date and time of the amend/modification request creation
1925	TradeClearingInstruction	int		This field contains one of the clearing conditions allowed in the market segment for orders * FIX 5.0 field added to message Supported values: 0 = Process normally - Default, not cleared against central counterparty 6 = Clear against central counterparty
	Standard Trailer		Y	

10.10. Order Interrupt Request

The **OrderCancelRequest**(35=F) message requests the cancellation of all of the remaining unexecuted quantity of an existing order. The cancel request message should be sent using a new ClOrdID(11) and is treated as a separate entity. The request will only be accepted if the order can successfully be withdrawn from Matching Engine. The ExecutionReport(35=8) message will be relaying the status of cancelled order. Otherwise, OrderCancelReject(35=9) message will be sent.

In the order cancel request message the order can be identified by either its prior ClOrdID(11) in OrigClOrdID(41) field, or by the OrderID(37).

The format of the OrderCancelRequest message is:

Table 28 – OrderCancelRequest Message

OrderCancelRequest(35=F)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
	Standard Header			Y	MsgType = **F**
11	ClOrdID		String	Y	Unique identifier of cancel request created by FIX Client. This identifier will be used in the ClOrdID tag of the: - Cancel Reject message if the order cancel request is rejected. - Execution Report in response to the cancellation request if accepted.
37	OrderID		String		Unique Identifier for the original Order that was assigned by DFIX Gateway and sent in the ExecutionReport as a response of order submission.
41	OrigClOrdID		String		Original ClOrdID of the order that is being requested to be canceled by this message. Conditionally required for orders submitted via FIX.
<Parties> component				Y	Component used to identify the order submitter (Branch, User and Customer)
453	NoPartyIDs		NumInGroup	Y	Minimum 2, maximum 3.
→	448	PartyID	String	Y	Party identifier, first block should be the 8-character Branch ID, second block should be the Trader short name, and if there is a third block, it should be a customer code.
→	447	PartyIDSource	char	Y	Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier (Datatec mnemonic)
→	452	PartyRole	int	Y	Identifies the role of RequestingPartyID Supported values: 11 = Order Origination Trader 13 = Order Origination Firm 32 = Beneficiary
End **<Parties>** component					
1300	MarketSegmentID		String	Y	Identifies the market segment, assigned by Datatec Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component				Y	Component used to identify the “Instrument”

OrderCancelRequest(35=F)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
55	Symbol	String	Y	Currency pair in CCY/CCY format, from MarketDefinition InstrumentScopeSymbol(1536)
End **<Instrument>** component				
54	Side	char	Y	Side of the order from order submitter's perspective. Must be same as the original order being cancelled. Supported values: 1 = Buy - Bid 2 = Sell – Offer
<OrderQtyData> component			Y	Component block contains the fields commonly used for indicating the amount or quantity of an order
38	OrderQty	Qty	Y	Quantity of the order. Required by the FIX Protocol. DFIX Gateway does not use or validate the contents of this tag.
End **<OrderQtyData>** component				
60	TransactTime	UTCTimestamp	Y	UTC date and time of the cancel request creation. Required by the FIX Protocol. DFIX Gateway does not use or validate the contents of this tag.
	Standard Trailer		Y	

10.11. Order Cancel Reject

The OrderCancelReject(35=9) message is issued by the DFIX Gateway upon receipt of an OrderCancelRequest(35=F) message or an OrderCancelReplaceRequest(35=G) message which cannot be honored. Filled orders cannot be modified or cancelled.

The Cancel Reject message should provide the ClOrdID(11) which was specified on the Cancel/Replace Request or Cancel Request message for identification, and the OrigClOrdId should be that of the last accepted order except in the case of CxlRejReason = “Other”. FIX Clients should refer to the RejectText(1328) field for specific information on the reason for the rejection.

The order cancel reject message format is as follows.

Table 29 – OrderCancelReject Message

OrderCancelReject(35=9)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **9**
11	ClOrdID	String	Y	Unique identifier of cancel request created by FIX Client.
37	OrderID	String	Y	Unique Identifier for the original Order that was assigned by DFIX Gateway and sent in the ExecutionReport as a response of order submission. If CxlRejReason="Unknown order", specify "NONE".

OrderCancelReject(35=9)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
41	OrigClOrdID		String		It contains the original ClOrdID(11) of the order which could not be canceled or replaced.
39	OrdStatus		char	Y	Identifies current status of order after this cancel reject is applied. If CxlRejReason = "Unknown Order", this field specify Rejected.
60	TransactTime		UTCTimestamp		UTC date and time of cancelation request rejection assigned by DFIX Gateway.
102	CxlRejReason		int		Code to identify reason for cancel rejection. Supported values: 0 = Too late to cancel 1 = Unknown order 3 = Order already in Pending Cancel or Pending Replace status 6 = Duplicated ClOrdID(11) received 8 = Price exceeds current price band 18 = Invalid price increment 99 = Other – Refer to RejectText(1328) field for exact reason for rejection
434	CxlRejResponseTo		char	Y	Identifies the type of request that a Cancel Reject is in response to. Supported values: 1 = Order cancel request 2 = Order cancel/replace request
1328	RejectText		String		Conditionally required when CxlRejReason(102)=99 (Other) to provide further rejection reasons.
<Parties> component				Y	Component used to identify the order or order cancel/replace submitter (Branch, User and Customer); carried from request
453	NoPartyIDs		NumInGroup	Y	Minimum 2, maximum 3.
→	448	PartyID	String	Y	Party identifier, first block should be the 8-character Branch ID, second block should be the Trader short name, and if there is a third block, it should be a customer code.
→	447	PartyIDSource	char	Y	Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier (Datatec mnemonic)
→	452	PartyRole	Int	Y	Identifies the role of RequestingPartyID Supported values: 11 = Order Origination Trader 13 = Order Origination Firm 32 = Beneficiary
End **<Parties>** component					
	Standard Trailer			Y	

10.12. Interrupt All Orders Request

The OrderMassActionRequest(35=CA) message will be used to request the cancellation of all unexecuted orders matching criteria specified within the request.

Table 30 – OrderMassActionRequest Message

OrderMassActionRequest(35=CA)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
	Standard Header			Y	MsgType = **CA**
11	ClOrdID		String	Y	Unique identifier of order mass action request created by FIX Client.
<Parties> component				Y	Component used to identify the order submitter (Branch and User)
453	NoPartyIDs		NumInGroup	Y	Always 2.
→	448	PartyID	String	Y	Party identifier, first block should be the 8-character Branch ID, second block should be the Trader short name
→	447	PartyIDSource	char	Y	Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier (Datatec mnemonic)
→	452	PartyRole	int	Y	Identifies the role of RequestingPartyID Supported values: 11 = Order Origination Trader 13 = Order Origination Firm
End **<Parties>** component					
1300	MarketSegmentID		String		Identifies the market segment, assigned by Datatec. Required if MassActionScope(1374) = 9 Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component					Component used to identify the “Instrument”
55	Symbol		String		Currency pair in CCY/CCY format, from MarketDefinition InstrumentScopeSymbol(1536)
End **<Instrument>** component					
54	Side		Char		Side of the orders from order submitter's perspective. If it´s specified only the orders from this side will be affected Supported values: 1 = Buy - Bid 2 = Sell – Offer

OrderMassActionRequest(35=CA)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
1373	MassActionType	Int	Y	Specifies the type of action requested. Supported values: 3 = Cancel orders
1374	MassActionScope	Int	Y	Specifies the scope of order mass action request. Supported values: 7 = All orders – All branch orders 9 = All orders for a market segment – For the branch 1000 – All orders for user 1001 – All orders for user for a market segment
60	TransactTime	UTCTimestamp	Y	UTC date and time of the order mass action creation. Required by the FIX Protocol. DFIX Gateway does not use or validate the contents of this tag.
	Standard Trailer		Y	

The OrderMassActionReport(35=BZ) is used to response to an OrderMassActionRequest(35=CA). Note that each affected resting order that is successfully canceled is acknowledged with a separate ExecutionReport(35=8) for each order.

When a FIX Client requests “interrupt all orders”, other FIX Clients in the same firm will receive the ExecutionReport(35=8) messages; only the requester will receive the OrderMassActionReport(35=BZ) message.

Table 31 – OrderMassActionReport Message

OrderMassActionReport(35=BZ)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **BZ**
11	ClOrdID	String	Y	ClOrdID provided on the Order Mass Action Request.
1369	MassActionReportID	String	Y	Unique identifier for the report assigned by DFIX Gateway
<Parties> component			Y	Component used to identify the order submitter of on the Order Mass Action Request (Branch and User)
453	NoPartyIDs	NumInGroup	Y	Always 2.
→	448 PartyID	String	Y	Party identifier, first block should be the 8-character Branch ID, second block should be the Trader short name
→	447 PartyIDSource	char	Y	Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier (Datatec mnemonic)
→	452 PartyRole	int	Y	Required when NoPartyIDs > 0. Identifies the role of RequestingPartyID Supported values:

OrderMassActionReport(35=BZ)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
				11 = Order Origination Trader 13 = Order Origination Firm
End **<Parties>** component				
1300	MarketSegmentID	String		Identifies the market segment, assigned by Datatec. Required if MassActionScope(1374) = 9 Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component				Component used to identify the “Instrument”
55	Symbol	String		Currency pair in CCY/CCY format, from MarketDefinition InstrumentScopeSymbol(1536)
End **<Instrument>** component				
54	Side	Char		Side of the orders from order submitter's perspective. If it´s specified only the orders from this side will be affected Supported values: 1 = Buy - Bid 2 = Sell – Offer
1373	MassActionType	int	Y	Specifies the type of action requested on the Order Mass Action Request Supported values: 3 = Cancel orders
1374	MassActionScope	int	Y	Specifies the scope of order mass action request. Supported values: 7 = All orders 9 = All orders for a market segment 1000 – All orders for user 1001 – All orders for user for a market segment
1375	MassActionResponse	int	Y	Indicates the action taken by the DFIX Gateway as a result of the Action Request Supported values: 0 = Rejected 1 = Accepted
1376	MassActionRejectReason	int		Indicates why Order Mass Action Request was rejected Required if MassActionResponse(1375) = 0 Supported values: 0 = Mass Action Not Supported 6 = Invalid or unknown trading session 8 = Invalid or unknown Market Segment 99 = Other
533	TotalAffectedOrders	Int		Optional field used to indicate the total number of orders affected by the Order Mass Action Request,

OrderMassActionReport(35=BZ)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					Only present in the last message sent.
<AffectedOrdGrp> component					Repeating group used to list of orders affected by the Order Mass Action Request.
534	NoAffectedOrders		NumInGroup		Number of affected orders in this message
→	1824	AffectedOrigClOrdID	String		Required if NoAffectedOrders(534)> 0 and must be the first repeating field in the group. Indicates the client ClOrdID(11) of an order affected by the Order Mass Action Request.
→	535	AffectedOrderID	String		Required if NoAffectedOrders(534)> 0. Indicates the OrderID(37) of an order affected by the Order Mass Action Request.
End <**AffectedOrdGrp**> component					
60	TransactTime		UTCTimestamp		UTC date and time when this report was sent.
58	Text		String		Free format text string
	Standard Trailer			Y	

10.13. Order Status Request

The OrderStatusRequest(35=H) message is used by the FIX Clients to request for the status of an order from the DFIX Gateway. If an OrderStatusRequest(35=H) is issued for an order that is either cancelled, or totally filled, only mandatory fields will be provided in resulting ExecutionReport(35=8) message. Non-mandatory fields will not be provided. If the order is not found or unknown the ExecutionReport(35=8) message will be used to reject the OrderStatusRequest(35=H) message.

Table 32 – OrderStatusRequest Message

OrderStatusRequest(35=H)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **H**
37	OrderID	String		The OrderID(37) of the order whose status is being requested. Conditionally required if ClOrdID(11) is not provided. Either OrderID or ClOrdID must be provided.
11	ClOrdID	String		The ClOrdID(11) of the order whose status is being requested. Conditionally required if OrderID(37) is not provided.
790	OrdStatusReqID	String		Optional, Unique identifier of order status request, created by FIX Client. If provided it will be used on Execution Report when this request is rejected. If it is not provided, the rejection will use the ClOrdID.
1300	MarketSegmentID	String	Y	Identifies the market segment where the order belongs Supported values:

OrderStatusRequest(35=H)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
				71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component			Y	Component used to identify the “Instrument”
55	Symbol	String	Y	Currency pair in CCY/CCY format.
End **<Instrument>** component				
54	Side	char	Y	Side of order Supported values: 1 = Buy - Bid 2 = Sell – Offer
	Standard Trailer		Y	

10.14. Execution Report Returned Tags

The following table shows tags in ExecutionReport(35=8) message that are returned based on various scenarios where this message is sent.

Table 33 – Execution Report Returned Tags

Tag	New Order Accepted	New Order Rejected	Order Cancel/ Replace Accepted	Order Cancel Accepted	Unsolicited Cancel	Order Partially Filled	Order Totally Filled	Order Status
OrderID(37)	R	R	R	R	R	R	R	R
ClOrdID(11)	RF	RF	RF	RF	RF	RF	RF	RF
ExecID(17)	R	R	R	R	R	R	R	R
OrigClOrdID(41)			R	R				
ExecType(150)	R	R	R	R	R	R	R	R
OrdStatus(39)	R	R	R	R	R	R	R	R
OrdRejReason(103)		R						
RejectText(1328)		R						
ExecRestatementReason(378)	C		C	C				
MarketSegmentID(1300)	R	R	R	R	R	R	R	R
Symbol(55)	R	R	R	R	R	R	R	R
Side(54)	R	R	R	R	R	R	R	R
OrderQty(38)	R	C	R	R	R	R	R	R
OrdType(40)	R	C	R	R	R	R	R	R
Price(44)	C	C	C	C	C	C	C	C
TradeClearingInstruction(1925)	C	C	C	C	C	C	C	C
MinQty(110)	C	C	C	C	C	C	C	C
MinQtyMethod(1822)	C	C	C	C	C	C	C	C

Tag	New Order Accepted	New Order Rejected	Order Cancel/ Replace Accepted	Order Cancel Accepted	Unsolicited Cancel	Order Partially Filled	Order Totally Filled	Order Status
DisplayQty(1138)	C	C	C	C	C	C	C	C
DisplayWhen(1083)	C	C	C	C	C	C	C	C
TimeInForce(59)	C	C	C	C	C	C	C	C
TransactTime(60)	R	R	R	R	R	R	R	R
TradeDate(75)						R	R	
SettlType(63)						R	R	
SettlDate(64)						R	R	
CumQty(14)	R	R	R	R	R	R	R	R
LeavesQty(151)	R	R	R	R	R	R	R	R
LastPx(31)						R	R	
LastQty(32)						R	R	
TrdMatchID(880)						R	R	
AggressorIndicator(1057)						R	R	
Parties	R	R	R	R	R	R	R	R
Text(58)	C	C	C	C	C	C	C	C

C	Conditionally required – Based on Input, Transaction, Query or Error condition
R	Required as part of ExecutionReport message
RF	Required as part of ExecutionReport for an order entered via FIX

11.Trade Capture Reporting

The TradeCaptureReport(35=AE) message is used for a number of reasons, including:

- Relaying Confirmed Trades to various parties not involved in the execution, such as clearing houses, regulatory bodies, the exchange's back office systems.

- Relaying Confirmed Trades to counterparties of the trade.

In both cases, those messages are outbound from the marketplace as a reply to a TradeCaptureReportRequest(35=AD) message.

Trades occur when the Matching Engine identifies matching conditions between buy and sell orders. On Datatec platforms Trades generated by the Matching Engine do not need confirmation from any counterparty, so the trades are sent as confirmed.

Trade Capture Reporting involves the following messages:

- TradeCaptureReportRequest(35=AD)
- TradeCaptureReport(35=AE)
- TradeCaptureReportRequestAck(35=AQ)

DFIX Gateway will support two ways of sending trade reports:

- Real-time trade reports are provided via the TradeCaptueReport(35=AE) message. FIX Clients that have sent a subscription request will receive trade details of each eligible trade immediately after it is executed.

- Query-based trade reports also are provided via the TradeCaptureReport(35=AE) message, but the FIX Client will receive trade reports based on the criteria provided in the request. The request can specify a date, date and time or a pair of date and time. When a single date is provided the client will receive all trades for that date. When date and time is provided the client will receive trade as of that date and time onwards. When a pair of date and time is provided, the client will receive trades between those dates and times.

  Queries involving more than one trading day can only be requested outside of market hours, if these queries are received during market hours, the DFIX Gateway will reject them.

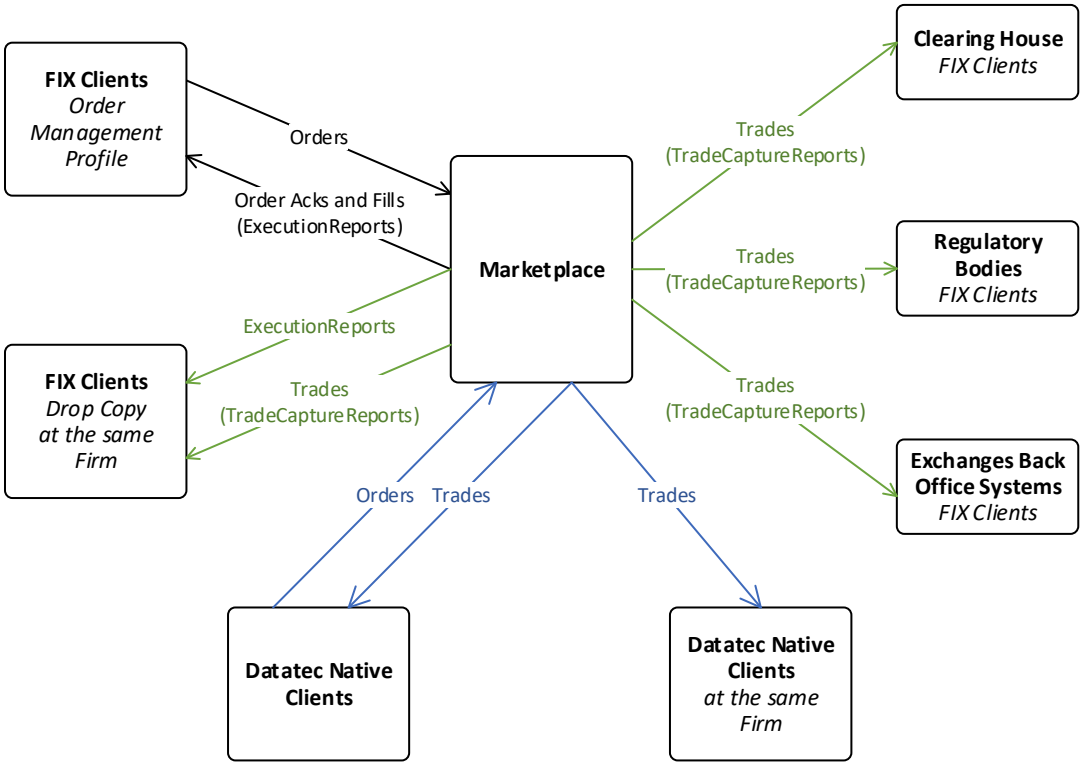
"Eligible trades" mean that DFIX Gateway will send only the trades that client is authorized to receive. For example, Clearing Houses, Regulatory, Exchange Back Office Systems have the right to receive all market trades; however, participant firms will only receive trades where the firm is one of the counterparties involved in the trade.

11.1. Workflows

The following workflows illustrate when Trade Capture Reports are sent to multiple parties and how they ask for and receive Trade Capture Reports.

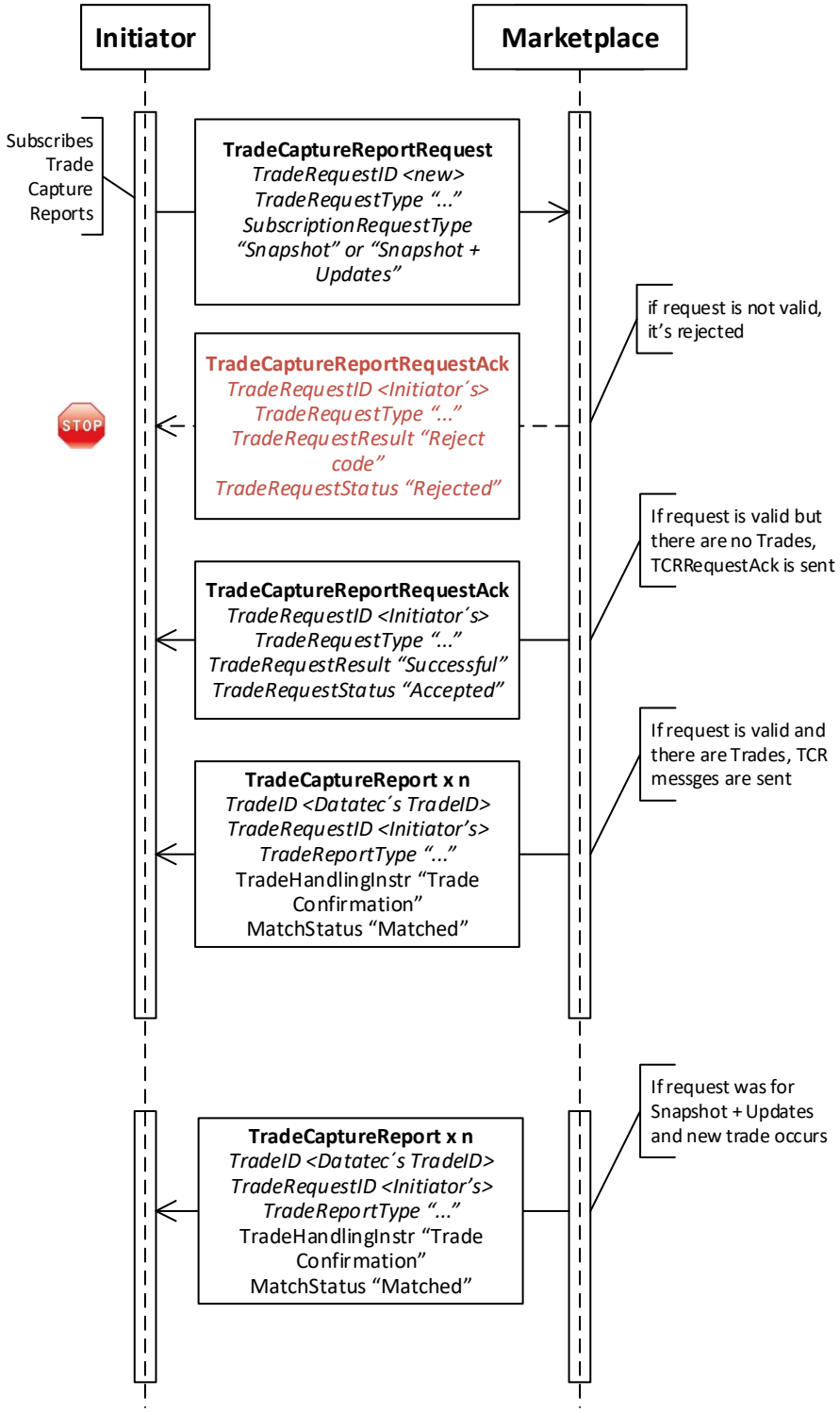
11.1.1. Workflow for multiple parties

Figure 11 – Trade Capture Reporting for Multiple Parties



11.1.2. Trade Capture Reporting Workflow

Figure 12 – Trade Capture Reporting Workflow



11.2. Trade Capture Report Request

A FIX Client may use the TradeCaptureReportRequest(35=AD) message to request the details of all eligible trades or those that meet certain criteria. The server will respond as follows:

- If the request is accepted and there are trades that match the request, the DFIX Gateway will transmit trades using TradeCaptureReport(35=AE) messages immediately without sending TradeCaptureReportRequestAck(35-AQ). Each TradeCaptureReport(35-AE) will include the TradeRequestID(568) of the request it is sent in response to. The last TradeCaptureReport(35=AE) will set value Y (Last Message) in LastRptRequested(912) of the last trade.

- If the request is accepted but there are no trades to send, the DFIX Gateway will transmit a TradeCaptureReportRequestAck(35=AQ) message with TradeRequestResult(749)=0 (Successful) and TradeRequestStatus(750)=0 (Accepted).

- If the request is rejected, TradeCaptureReportRequestAck(35=AQ) message will be sent with TradeRequestStatus(750)=2 (Rejected) and the reason will be specified in the field TradeRequestResult(749).

The TradeCaptureReportRequest(35=AD) is used to:

1. Request all of a firm's trades,
2. Request all market trades. Only for certain type of clients such as the clearing house, regulator or exchange back office,
3. Request all trades that a FIX user is authorized to see that occurred after a specified date/time (one date/time supplied in TrdCapDtGrp component),
4. Request all trades that a FIX user is authorized to see that occurred after the first specified date/time and before the second specified date/time (two date/times supplied in TrdCapDtGrp component),
5. Unsubscribe to trade capture reports.

For the first three cases, the FIX Client may request "Snapshot + Updates" or just a "Snapshot". For the last request case, only one snapshot will be provided of the market data.

In all cases, the request must specify the market segment and instrument that trades belong to.

Table 34 – TradeCaptureReportRequest Message

TradeCaptureReportRequest(35=AD)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **AD**
568	TradeRequestID	String	Y	Unique identifier for the trade request created by FIX Client. For SubscriptionRequestType(263)=2 (Unsubscribe) this field must specify the same request ID of the subscription request being disabled.
569	TradeRequestType	int	Y	Type of Trade Capture Report Request Supported values:

TradeCaptureReportRequest(35=AD)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					0 = All Trades
263	SubscriptionRequestType		Char		Subscription Request Type If the field is absent, SubscriptionRequestType(263)=0 (Snapshot) will be the default. Supported values: 0 = Snapshot 1 = Snapshot + Updates (Subscribe) 2 = Disable previous Snapshot + Update Request (Unsubscribe)
1300	MarketSegmentID		String	Y	Identifies the market segment that trades belong to * FIX 5.0 field added to message Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component				Y	Component used to identify the “Instrument”
55	Symbol		String	Y	Currency pair in CCY/CCY format, from MarketDefinition if FIX client has requested MarketDefinitions. Supported values: USD/COP USD/CLP USD/PEN
End **<Instrument>** component					
<TrdCapDtGrp> component					Repeating group used to request for trades after a specified date/time (1 instance) or between two dates/times (2 instances).
580	NoDates		NumInGroup		Number of dates provided (must be 1 or 2 if specified).
→	75	TradeDate	LocalMktDate		Conditionally required if NoDates(580)> 0 Indicates date of trading day.
→	60	TransactTime	UTCTimestamp		To request trades for a specific time.
End **<TrdCapDtGrp>** component					
	Standard Trailer			Y	

11.3. Trade Capture Report Request Ack

The TradeCaptureReportRequestAck(35=AQ) message is used indicate:

- That the Trade Capture Report Request was valid but no trades were found that matched the selection criteria specified in the request,
- That the Trade Capture Report Request was invalid for some business reason, such as request is not authorized, invalid or unknown instrument, etc.

Table 35 – TradeCaptureReportRequestAck Message

TradeCaptureReportRequestAck(35=AQ)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **AQ**
568	TradeRequestID	String	Y	Unique identifier carried from the request
569	TradeRequestType	int	Y	Type of TCR Request carried from the request
263	SubscriptionRequestType	char		Subscription Request Type carried from the request Supported values: 0 = Snapshot 1 = Snapshot + Updates (Subscribe) 2 = Disable previous Snapshot + Update Request (Unsubscribe)
1300	MarketSegmentID	String	Y	Identifies the market segment that trades belong to, carried from the request * FIX 5.0 field added to message Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component			Y	Component used to identify the “Instrument”
55	Symbol	String	Y	Currency pair in CCY/CCY format
End **<Instrument>** component				
749	TradeRequestResult	int	Y	Result of Trade Capture Report Request Supported values: 0 = Successful (default) 1 = Invalid or unknown instrument 8 = TradeRequestType not supported 9 = Not authorized 99 = Other
750	TradeRequestStatus	int	Y	Status of Trade Capture Report Request Supported values: 0 = Accepted 2 = Rejected
58	Text	String		Free format text string
	Standard Trailer		Y	

11.4. Trade Capture Report

The TradeCaptureReport(35=AE) message can be:

- Sent as reply to a TradeCaptureReportRequest(35=AD) message,
- Used to report trades to the subscribed users
- Used to report privately negotiated trades between counterparties (Trade Registration)

The table below includes the structure of the Trade Capture Report message used in all use cases mentioned.

Table 36 – TradeCaptueReport Message

TradeCaptureReport(35=AE)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **AE**
1003	TradeID	String		Unique trade identifier generated by Datatec. Conditionally required in all TCR messages sent from DFIX Gateway.
1041	FirmTradeID	String		Unique ID assigned to a trade by the Firm to track a trade within the Firm system. This ID can be assigned either before or after submission to the DFIX Gateway. Conditionally required in all post trade workflow TCR messages.
856	TradeReportType	int		Type of Trade Report Supported values: 0 = Submit 2 = Accept 3 = Decline 5 = No/Was 7 = Trade Break 11 = Alleged New 13 = Alleged No/Was 15 = Alleged Trade Break
939	TrdRptStatus	int		Status of the trade report Supported values: 0 = Accepted 1 = Rejected 2 = Cancelled 4 = Pending New 5 = Pending Cancel 6 = Pending Replace
568	TradeRequestID	String		Request ID if the Trade Capture Report is in response to a TCRRequest message. Echoes the value from the TCRRequest.
828	TrdType	int		Type of trade Supported values: 0 = Regular trade (Default) 22 = Privately negotiated trade – Registered trade

TradeCaptureReport(35=AE)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
829	TrdSubType		int		Contain additional trade type qualifiers. Supported values: 36 = SWAP converted 1000 = Next Day converted 1001 = FIX converted The following values applies for MarketSegmentID = 71 on Trade Registration 1002 = Swap O/N Contado 1003 = Swap Forward 1004 = Swap Future
1123	TradeHandlingInstr		char		Specifies how the TCR should be handled Supported values: 0 = Trade Confirmation 3 = One-party report for pass through
912	LastRptRequested		Boolean		Indicates whether this message is the last report message in response to a request message. The absence of this field should be understood as not the last message Supported values: N = Not last message - Default if not specified Y = Last Message
263	SubscriptionRequestType		char		Subscription Request Type carried from the request if there was any
17	ExecID		String		Identifier assigned by DFIX Gateway for the ExecutionReport(35=8) when trade occurs
423	PriceType		int		Defines the type of price used in the market Supported values: 20 = Normal rate representation
<RootParties> component					Repeating group used for acting parties that applies to the whole message. When Broker registered a trade, their data will be sent here.
1116	NoRootPartyIDs		NumInGroup		Number of parties, maximum 2.
→	1117	RootPartyID	String		Required when NoRootPartyIDs > 0. Party identifier – 8-character Branch ID or Trader short name
→	1118	RootPartyIDSource	char		Required when NoRootPartyIDs > 0. Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier - Datatec mnemonic
→	1119	RootPartyRole	int		Required when NoRootPartyIDs > 0. Identifies the role of RequestingPartyID Supported values: 7 = Entering Firm 36 = Entering Trader - Broker

TradeCaptureReport(35=AE)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
End **<RootParties>** component				
1300	MarketSegmentID	String	Y	Identifies the market segment that trades belong to Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component			Y	Component used to identify the “Instrument”
55	Symbol	String	Y	Currency pair in CCY/CCY format
End **<Instrument>** component				
31	LastPx	Price		Trade Price
32	LastQty	Qty	Y	Trade Quantity
15	Currency	Currency	Y	Primary currency of the specified currency pair. Used to qualify LastQty(32); i.e. what currency the LastQty is denominated. Supported values: USD = US Dollar
60	TransactTime	UTCTimestamp	Y	UTC date and time of the trade represented by this TCR
75	TradeDate	LocalMktDate		Indicates date of trade referenced in this message in YYYYMMDD format. Absence of this field indicates current day, but is useful when reporting other than current day trades.
63	SettlType	String		Specific trade settlement period Supported values: 0 = Regular / FX Spot settlement (T+1 or T+2 depending on currency) 1 = Cash (TOD / T+0) 2 = Next Day (TOM / T+1) 3 = T+2 4 = T+3
64	SettlDate	LocalMktDate		Specific date of trade settlement in YYYYMMDD format.
573	MatchStatus	char		The status of this trade with respect to matching or comparison. Supported values: 0 = Compared, matched or affirmed 1 = Uncompared, unmatchedor unaffirmed
<TrdCapRptSideGrp> component			Y	Repeating group for trade side instances
552	NoSides	NumInGroup	Y	Always 2
→	54 Side	char	Y	Side for this block of fields in the repeating group. Must be the first field in this repeating group.

TradeCaptureReport(35=AE)							
Tag	**Field Name**				**DataType**	**Reqd**	**Comments**
							Supported values: 1 = Buy 2 = Sell
→	**<Parties>** component						Repeating group, is used to identify and convey information of the entities in this side of the trade
→	453	NoPartyIDs					Number of parties involved in the side of the trade For Trade Registration Request this field should be a minimum of 2 party instances and a maximum of 3 party instances. For other uses a minimum of 4 party instances and a maximum of 5 party instances.
→	→	448	PartyID		String		Required if NoPartyIDs(453) > 0 Identification of the party. 8-character Branch code, or Trader short name, or the Beneficiary code.
→	→	447	PartyIDSource		char		Required if NoPartyIDs(453) > 0 Used to identify class source of PartyID value Supported values: C = Generally accepted market participant identifier (Datatec mnemonic)
→	→	452	PartyRole		int		Required if NoPartyIDs(453) > 0 Used to identify role of PartyID in the side of the trade 11 = Order originator Trader 13 = Order originator Firm 17 = Contra Firm 32 = Beneficiary 37 = Contra Trader
→	→	**<PtysSubGrp>** component					Required when NoPartyIDs(453)> 0 and the message is been sent by DFIX Gateway. Not required when this message is sent by FIX Clients, Repeating group of Party sub-identifiers.
→	→	802	NoPartySubIDs		NumInGroup		Number of PartySubIDs
→	→	→	523	PartySubID	String		Sub-identifier, DFIX Gateway will send the branch name, or trader name, or city name.
→	→	→	803	PartySubIDType	int		Type of PartySubID(523) value. Supported values: 1 = Firm – Firm full name or Branch full name 2 = Person – Full name 34 = Address City – Related to the branch
→	→	End **<PtysSubGrp>** component					
→	End **<Parties>** component						
→	**<TradeReportOrderDetail>** component						Order details for the order associated with this side of the trade.
→	37	OrderID			String		ID of the order associated with this side of the trade.

TradeCaptureReport(35=AE)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
→	11	ClOrdID	String		Client order ID of the order associated with this side of the trade. Required if the order belongs to FIX Client and is not a registered trade.
→	End **<TradeReportOrderDetail>** component				
→	1057	AggressorIndicator	Boolean		Used to identify whether the side's order initiator is an aggressor or not in the trade. Supported values: Y = Order initiator is aggressor N = Order initiator is passive
End **<TrdCapRptSideGrp>** component					
751	TradeReportRejectReason		int		Only used for Trade Registration and Post Trade operations, Indicates the reason of Trade Capture Report rejected Supported values: 0 = Successful (default) 101 = Expired 99 = Other
1328	RejectText		String		Text explaining the reason for rejection
1925	TradeClearingInstruction		int		This field contains one of the clearing conditions allowed in the market segment for orders and trades Supported values: 0 = Process normally (default, not cleared against central counterparty) 6 = Clear against central counterparty
	Standard Trailer			Y	

11.5. Trade Capture Report Submitted and Returned Tags

The following table shows tags in TradeCaptureReport(35=AE) message that are submitted or returned based on various scenarios where this message is sent.

Table 37 – Trade Capture Report Submitted and Returned Tags

Tag	Reporting a Matched Trade	Trade Registration							
		Submit New Trade	Send Alleged Trade	Send Expired Request	Counterparty Declines	Send Decline to initiator	Counterparty Accepts	Reject by Credit Limits	Send Confirmed Trade
TradeID(1003)	R		R	R	R	R	R	R	R
FirmTradeID(1041)		R		C		R	R	R	R
TradeReportType(856)		R	R	R	R	R	R	R	R
TrdRptStatus(939)				R				R	R
TradeRequestID(568)	C								
TrdType(828)		R	R	R	R	R	R	R	R
TrdSubType(829)	C	C	C	C	C	C	C	C	C
TradeHandlingInstr(1123)	R								R
LastRptRequested(912)	C								
SubscriptionRequestType(263)	C								
ExecID(17)	R								
PriceType(423)	R	R	R	R	R	R	R	R	R
RootParties		C	C	C	C	C	C	C	C
MarketSegmentID(1300)	R	R	R	R	R	R	R	R	R
Symbol(55)	R	R	R	R	R	R	R	R	R
LastPx(31)	C	C	C	C	C	C	C	C	C
LastQty(32)	R	R	R	R	R	R	R	R	R
Currency(15)	R	R	R	R	R	R	R	R	R
TransactTime(60)	R	R	R	R	R	R	R	R	R
TradeDate(75)	C	C	C	C	C	C	C	C	C
SettlType(63)	C	C	C	C	C	C	C	C	C
SettlDate(64)	C	C	C	C	C	C	C	C	C
MatchStatus(573)	R								R
TrdCapRptSideGrp	R	R	R	R	R	R	R	R	R
TradeReportRejectReason(751)				R				R	
RejectText(1328)				R				R	
TradeClearingInstruction(1925)	C	C	C	C	C	C	C	C	C

C	Conditionally required– Based on Input, Transaction, Query or Error condition
R	Required as part of TradeCaptureReport message

12. Trade Registration

When a trade is agreed between two counterparties outside the electronic exchange system, this trade can be entered into the system. Trade registration allows two counterparties to formally agree that a trade took place outside the electronic markets on the platform, usually to comply with reporting requirements. This section details the message flows and specific messaging considerations for implementation of a privately negotiated two party trade confirmation.

The implementation will use what FIX calls the "entity-based model" of identifying the trade. What this means is each side should supply their own trade identifier and DFIX Gateway will also supply a centrally assigned trade identifier. The identifiers will not change during the life of the trade.

The entire workflow uses the TradeCaptureReport(35=AE) and TradeCaptureReportAck(35=AR) messages. The way the flow works is: Initiator sends in a TradeCaptureReport(35=AE) with all the trade details to the DFIX Gateway indicating they did this trade with the identified counterparty. DFIX will send an "alleged trade" TradeCaptureReport(35=AE) message to the counterparty telling them the initiator alleged the trade against them. The counterparty has to accept or reject the alleged trade. The counterparty responds with an answer sent to DFIX Gateway. Finally DFIX will send the confirmed trade to both parties if the counterparty accepts the alleged trade. For details see the workflow diagrams in the sub-sections below.

The following table describes some of the key FIX fields used in two party trade confirmations:

Table 38 – Key FIX fields in Two Party Trade Confirmation

Field	Description
TradeID(1003)	Unique trade identifier generated by Datatec. Trade IDs are unique for a given business date. Once assigned, this ID is used to identify the trade in post trade operations.
FirmTradeID(1041)	Unique ID assigned to a trade by the Firm to track a trade within the Firm system. The initiator, on submission of the trade registration will provide its own trade identifier in this field. The counterparty will also provide its own trade identifier when responding to DFIX. FirmTradeIDs of the respective parties in the trade are not revealed to each other. The ID must be unique for the given business date.
TradeReportType(856)	Type of Trade Report
TrdRptStatus(939)	Status of the trade report (Status of the trade itself)
TradeHandlingInstr(1123)	Specifies how the TCR should be handled
TradeClearingInstruction(1925)	Contains one of the clearing conditions allowed in the market segment for orders
MatchStatus(573)	The status of this trade with respect to matching or comparison.
TrdAckStatus(1523)	Field in TradeCaptureReportAck(35=AR) used to indicate the status of trade submission

12.1. Workflows

The whole process of trade Registration has been divided in two workflow diagrams; the first one includes the submission of the trade from the initiator and how it is conveyed to the counterparty. The second one includes the rejection or acceptance sent by counterparty and how the process is finished.

Figure 13 – Trade Registration Workflow Part 1

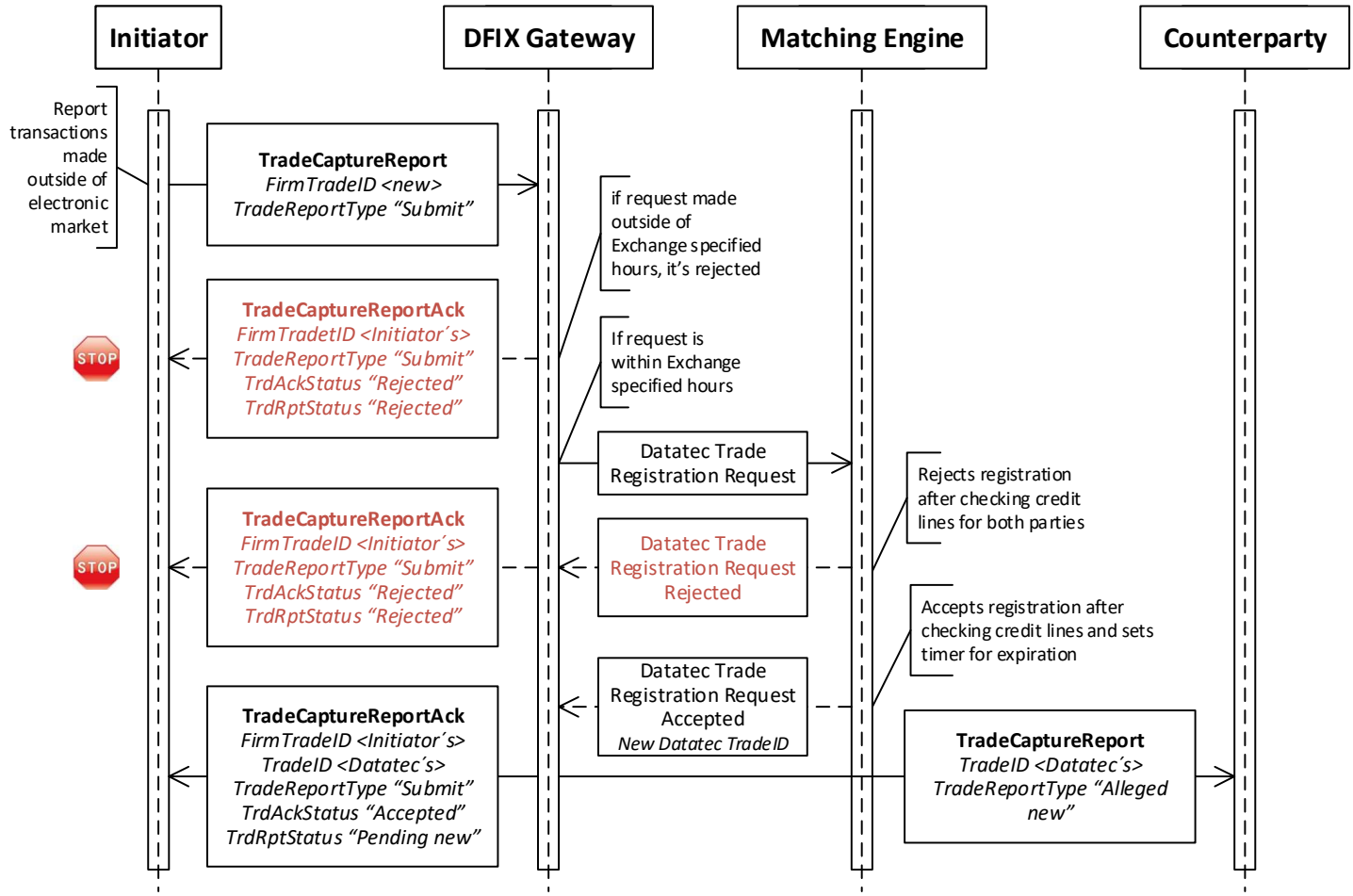
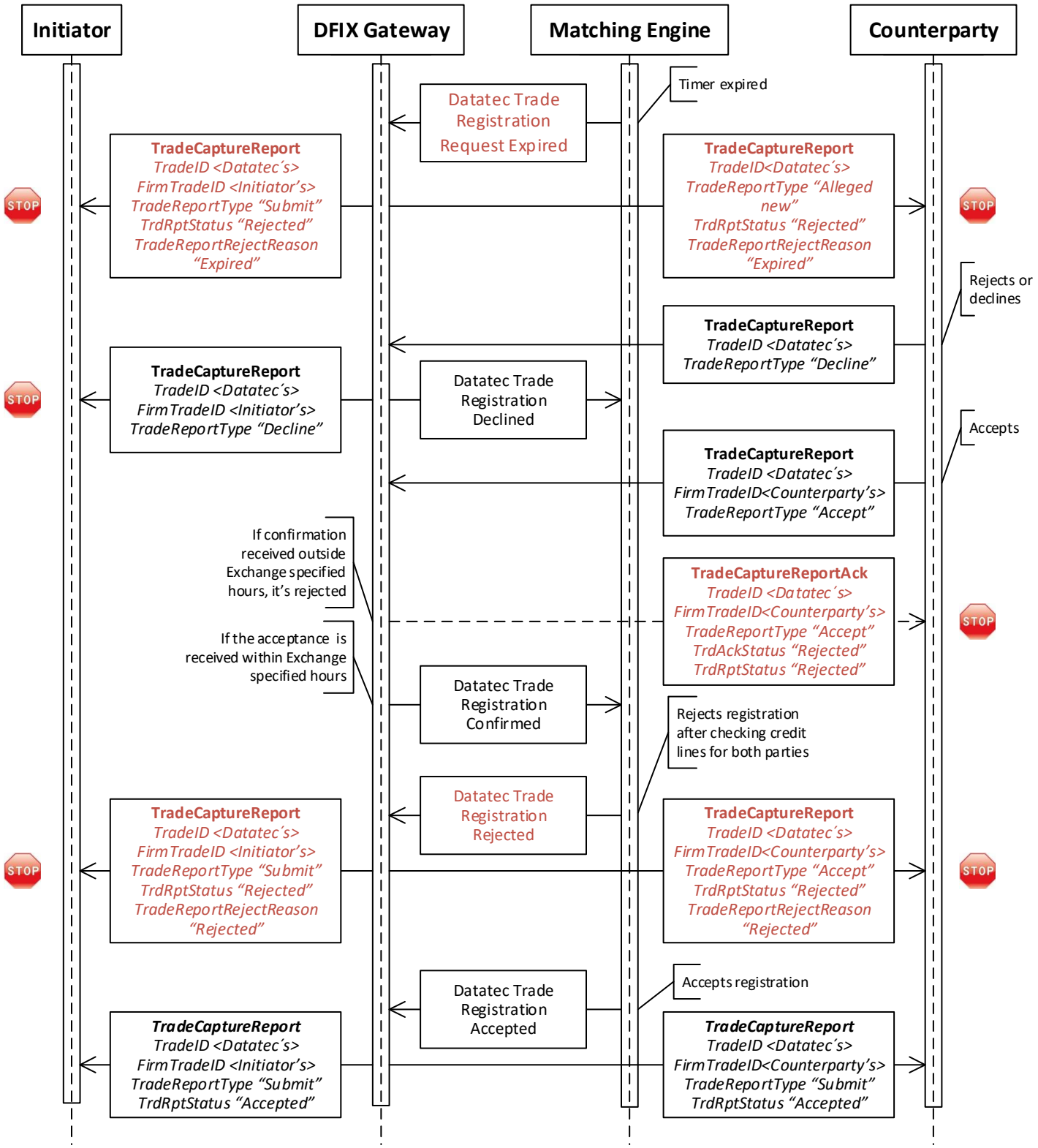


Figure 14 – Trade Registration Workflow Part 2



12.2. Trade Capture Report Ack

The TradeCaptureReportAck(35=AR) message can be:

- Used by DFIX to acknowledge Trade Capture Reports received
- Used by DFIX to reject a Trade Capture Repot received

These use cases of Trade Capture Report Ack message are included in the Trade Registration workflows.

Most of the fields in TCRAck message are carried over from Trade Capture Report being acknowledged or rejected; only fields TrdRptStatus(939), TrdAckStatus(1523), TradeReportRejectReason(751) and RejectText(1328) are set by the DFIX Gateway before sending.

Table 39 – TradeCaptureReportAck Message

TradeCaptureReportAck(35=AR)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
	Standard Header		Y	MsgType = **AR**
1003	TradeID	String		Unique trade identifier assigned by Datatec
1041	FirmTradeID	String	Y	Unique ID assigned to a trade by the Firm
856	TradeReportType	int	Y	Type of Trade Report, carried from TCR
939	TrdRptStatus	int	Y	Status of the trade report Supported values: 0 = Accepted 1 = Rejected 4 = Pending New 5 = Pending Cancel 6 = Pending Replace
1523	TrdAckStatus	int	Y	Status of the trade submission (not the trade report) Supported values: 0 = Accepted 1 = Rejected
828	TrdType	int		Type of trade Supported values: 0 = Regular trade (Default) 22 = Privately negotiated trade – Registered trade
829	TrdSubType	int		If MDEntryType is 2 (Trade), this may contain additional trade type qualifiers. Used in conjunction with TrdType(828). Supported values: 36 = SWAP converted 1000 = Next Day converted 1001 = FIX converted The following values applies for MarketSegmentID = 71 on Trade Registration 1002 = Swap O/N Contado 1003 = Swap Forward

TradeCaptureReportAck(35=AR)					
Tag	**Field Name**		**DataType**	**Reqd**	**Comments**
					1004 = Swap Future
1123	TradeHandlingInstr		char		Specifies how the TCR should be handled Supported values: 0 = Trade Confirmation 3 = One-party report for pass through
17	ExecID		String		Identifier assigned by DFIX Gateway for the ExecutionReport(35=8) when trade occurs
423	PriceType		int		Defines the type of price used in the market Supported values: 20 = Normal rate representation
<RootParties> component					Repeating group used for acting parties that applies to the whole message. When Broker registered a trade, their data will be sent here.
1116	NoRootPartyIDs		NumInGroup		Number of parties, maximum 2
→	1117	RootPartyID	String		Required when NoRootPartyIDs > 0. Party identifier – 8-character Branch ID or Trader short name
→	1118	RootPartyIDSource	char		Required when NoRootPartyIDs > 0. Identifies the source of PartyID value. Supported values: C = Generally accepted market participant identifier - Datatec mnemonic
→	1119	RootPartyRole	int		Required when NoRootPartyIDs > 0. Identifies the role of RequestingPartyID Supported values: 7 = Entering Firm 36 = Entering Trader - Broker
End **<RootParties>** component					
1300	MarketSegmentID		String	Y	Identifies the market segment where trades belong to Supported values: 71 = CO Dollar Spot 76 = CO Dollar Next Day 39 = CL Dollar Spot 51 = PE Dollar Spot
<Instrument> component				Y	Component used to identify the “Instrument”
55	Symbol		String	Y	Currency pair in CCY/CCY format, from MarketDefinition
End **<Instrument>** component					
31	LastPx		Price		Trade Price
32	LastQty		Qty		Trade Quantity
15	Currency		Currency	Y	Primary currency of the specified currency pair. Used to qualify LastQty(32); i.e. what currency the LastQty is

TradeCaptureReportAck(35=AR)						
Tag	**Field Name**			**DataType**	**Reqd**	**Comments**
						denominated. Supported values: USD = US Dollar
60	TransactTime			UTCTimestamp	Y	UTC date and time of the trade represented by this TCR
75	TradeDate			LocalMktDate		Indicates date of trade referenced in this message in YYYYMMDD format.
63	SettlType			String		Specific trade settlement period Supported values: 0 = Regular / FX Spot settlement (T+1 or T+2 depending on currency) 1 = Cash (TOD / T+0) 2 = Next Day (TOM / T+1) 3 = T+2 4 = T+3
64	SettlDate			LocalMktDate		Specific date of trade settlement in YYYYMMDD format.
573	MatchStatus			char		The status of this trade with respect to matching or comparison. 0 = Compared, matched or affirmed 1 = Uncompared, unmatched or unaffirmed
<TrdCapRptAckSideGrp> component					Y	Repeating group for side instances carried from TCR
552	NoSides			NumInGroup	Y	Always 2
→	54	Side		char	Y	Side for this block of fields in the repeating group. Must be the first field in this repeating group. Supported values: 1 = Buy 2 = Sell
→	**<Parties>** component					Repeating group, is used to identify and convey information of the entities in this side of the trade; carried from TCR being acknowledged.
→	453	NoPartyIDs		NumInGroup		Number of parties involved in the side of the trade
→	→	448	PartyID	String		Required if NoPartyIDs(453) > 0 Identification of the party.
→	→	447	PartyIDSource	char		Required if NoPartyIDs(453) > 0 Used to identify class source of PartyID value Supported values: C = Generally accepted market participant identifier (Datatec mnemonic)
→	→	452	PartyRole	int		Required if NoPartyIDs(453) > 0 Used to identify role of PartyID in the side of the trade 11 = Order originator Trader 13 = Order originator Firm

TradeCaptureReportAck(35=AR)							
Tag	**Field Name**				**DataType**	**Reqd**	**Comments**
							17 = Contra Firm 32 = Beneficiary 37 = Contra Trader
→	→	**<PtySubGrp>** component					Required when NoPartyIDs(453)> 0. Repeating group of Party sub-identifiers.
→	→	802	NoPartySubIDs		NumInGroup		Number of PartySubIDs
→	→	→	523	PartySubID	String		Sub-identifier, DFIX Gateway will send the branch name, or trader name, or city name.
→	→	→	803	PartySubIDType	int		Type of PartySubID(523) value. Supported values: 1 = Firm – Firm full name or Branch full name 2 = Person – Full name 34 = Address City – Related to the branch
→	→	End **<PtySubGrp>** component					
→	End **<Parties>** component						
→	**<TradeReportOrderDetail>** component						Order details for the order associated with this side of the trade.
→	37	OrderID			String		ID of the order associated with this side of the trade.
→	11	ClOrdID			String		Client order ID of the order associated with this side of the trade. Required if the order belongs to FIX Client and is not a registered trade.
→	End **<TradeReportOrderDetail>** component						
→	1057	AggressorIndicator			Boolean		Used to identify whether the side order initiator is an aggressor or not in the trade. Supported values: Y = Order initiator is aggressor N = Order initiator is passive
End **<TrdCapRptAckSideGrp>** component							
751	TradeReportRejectReason				int		Only used for Trade Registration and Post Trade operations, Indicates the reason of Trade Capture Report rejected Supported values: 0 = Successful (default) 1 = Invalid Party information 2 = Unknown instrument 3 = Unauthorized to report trades 4 = Invalid trade type 5 = Price exceeds current price band 101 = Expired 99 = Other
1328	RejectText				int		Text explaining the reason or rejection
1925	TradeClearingInstruction				int		This field contains one of the clearing conditions

TradeCaptureReportAck(35=AR)				
Tag	**Field Name**	**DataType**	**Reqd**	**Comments**
				allowed in the market segment for orders and trades * FIX 5.0 field added to message Supported values: 0 = Process normally (default, not cleared against central counterparty) 6 = Clear against central counterparty
	Standard Trailer		Y	

Appendix A: FIX Data Types

The following table details the standard FIX Data Type definitions used by the standard. This is provided here for convenience as the FIX Message Tables references the standard data types. Any user defined field (a.k.a. custom fields) will also use standard FIX data types as indicated in the message table where the user defined field is used.

When an enumerated field requires clarity of the data type of the corresponding value field (e.g. PartySubIDType and PartySubID) for a given enumeration, the standard FIX data type will also be used and indicated in the elaboration for the enumeration.

This table is extracted from the FIX Protocol standard data dictionary.

Table 40 – FIX Data Type Descriptions

Type Name	Base Type	Description
int		Sequence of digits without commas or decimals and optional sign character (ASCII characters "-" and "0" - "9"). The sign character utilizes one byte (i.e. positive int is "99999" while negative int is "-99999"). Note that int values may contain leading zeros (e.g. "00023" = "23"). Examples: 723 in field 21 would be mapped int as |21=723|. -723 in field 12 would be mapped int as |12=-723| The following data types are based on int.
Length	int	int field representing the length in bytes. Value must be positive.
TagNum	int	int field representing a field's tag number when using FIX "Tag=Value" syntax. Value must be positive and may not contain leading zeros.
SeqNum	int	int field representing a message sequence number. Value must be positive.
NumInGroup	int	int field representing the number of entries in a repeating group. Value must be positive.
DayOfMonth	int	int field representing a day during a particular month (values 1 to 31).
float		Sequence of digits with optional decimal point and sign character (ASCII characters "-", "0" - "9" and "."); the absence of the decimal point within the string will be interpreted as the float representation of an integer value. All float fields must accommodate up to fifteen significant digits. The number of decimal places used should be a factor of business/market needs and mutual agreement between counterparties. Note that float values may contain leading zeros (e.g. "00023.23" = "23.23") and may contain or omit trailing zeros after the decimal point (e.g. "23.0" = "23.0000" = "23" = "23."). Note that fields which are derived from float may contain negative values unless explicitly specified otherwise. The following data types are based on float.
Qty	float	float field capable of storing either a whole number (no decimal places) of "shares" (securities denominated in whole units) or a decimal value containing decimal places for non-share quantity asset classes (securities denominated in fractional units).
Price	float	float field representing a price. Note the number of decimal places may vary. For certain asset classes prices may be negative values. For example, prices for options strategies can be negative under certain market conditions. Refer to Volume 7: FIX Usage by Product for asset classes that support negative price values.

Type Name	Base Type	Description
		Example(s): Strk="47.50"
PriceOffset	float	float field representing a price offset, which can be mathematically added to a "Price". Note the number of decimal places may vary and some fields such as LastForwardPoints may be negative.
Amt	float	Float field typically representing a Price times a Qty Example(s): Amt="6847.00"
Percentage	float	Float field representing a percentage (e.g. 0.05 represents 5% and 0.9525 represents 95.25%). Note the number of decimal places may vary.
char		Single character value, can include any alphanumeric character or punctuation except the delimiter. All char fields are case sensitive (i.e. m != M). The following fields are based on char.
Boolean	char	char field containing one of two values: 'Y' = True/Yes 'N' = False/No
String		Alpha-numeric free format string can include any character or punctuation except the delimiter. All String fields are case sensitive (i.e. morstatt != Morstatt).
MultipleCharValue	String	String field containing one or more space delimited single character values (e.g. |18=2 A F|).
MultipleStringValue	String	String field containing one or more space delimited multiple character values (e.g. |277=AV AN A|).
Country	String	String field representing a country using ISO 3166 Country code (2-character) values.
Currency	String	String field representing a currency type using ISO 4217 Currency code (3-character) values. Example(s): StrkCcy="USD"
Exchange	String	String field representing a market or exchange using ISO 10383 Market Identifier Code (MIC) values.
MonthYear	String	String field representing month of a year. An optional day of the month can be appended or an optional week code. Valid formats: YYYYMM YYYYMMDD YYYYMMWW Valid values: YYYY = 0000-9999 MM = 01-12 DD = 01-31 WW = w1, w2, w3, w4, w5.

Type Name	Base Type	Description
		Example(s): MonthYear="200303" MonthYear="20030320" MonthYear="200303w2"
UTCTimestamp	String	String field representing time/date combination represented in UTC (Universal Time Coordinated, also known as "GMT") in either YYYYMMDD-HH:MM:SS (whole seconds) or YYYYMMDD-HH:MM:SS.sss (milliseconds) format, colons, dash, and period required. Valid values: YYYY = 0000-9999 MM = 01-12 DD = 01-31 HH = 00-23 MM = 00-59 SS = 00-60 (60 only if UTC leap second) sss=000-999 (indicating milliseconds). Leap Seconds: Note that UTC includes corrections for leap seconds, which are inserted to account for slowing of the rotation of the earth. Leap second insertion is declared by the International Earth Rotation Service (IERS) and has, since 1972, only occurred on the night of Dec. 31 or Jun 30. The IERS considers March 31 and September 30 as secondary dates for leap second insertion, but has never utilized these dates. During a leap second insertion, a UTCTimestamp field may read "19981231-23:59:59", "19981231-23:59:60", "19990101-00:00:00". (see http://tycho.usno.navy.mil/leapsec.html) Example(s): TransactTime="20011217-09:30:47"
UTCTimeOnly	String	String field representing time-only represented in UTC (Universal Time Coordinated, also known as "GMT") in either HH:MM:SS (whole seconds) or HH:MM:SS.sss (milliseconds) format, colons, and period required. This special-purpose field is paired with UTCDateOnly to form a proper UTCTimestamp for bandwidth-sensitive messages. Valid values: HH = 00-23 MM = 00-59 SS = 00-60 (60 only if UTC leap second) sss=000-999 (indicating milliseconds). Example(s): MDEntryTime="13:20:00.000"
UTCDateOnly	String	String field representing date-only represented in UTC (Universal Time Coordinated, also known as "GMT") in YYYYMMDD format. This special-purpose field is paired with UTCTimeOnly to form a proper UTCTimestamp for bandwidth-sensitive messages. Valid values: YYYY = 0000-9999 MM = 01-12 DD = 01-31.

Type Name	Base Type	Description
		Example(s): MDEntryDate="20030910"
LocalMktDate	String	String field representing a Date of Local Market (as oppose to UTC) in YYYYMMDD format. This is the "normal" date field used by the FIX Protocol. Valid values: YYYY = 0000-9999 MM = 01-12 DD = 01-31. Example(s): BizDate="2003-09-10"
TZTimeOnly	String	String field representing the time represented based on ISO 8601. This is the time with a UTC offset to allow identification of local time and timezone of that time. Format is HH:MM[:SS][Z|[+|-hh[:mm]]] where HH = 00-23 hours MM = 00-59 minutes SS = 00-59 seconds hh = 01-12 offset hours mm = 00-59 offset minutes. Example: 07:39Z is 07:39 UTC Example: 02:39-05 is five hours behind UTC, thus Eastern Time Example: 15:39+08 is eight hours ahead of UTC, Hong Kong/Singapore time Example: 13:09+05:30 is 5.5 hours ahead of UTC, India time
TZTimestamp	String	String field representing a time/date combination representing local time with an offset to UTC to allow identification of local time and time zone offset of that time. The representation is based on ISO 8601. Format is YYYYMMDD-HH:MM:SS[Z|[+|-hh[:mm]]] where YYYY = 0000 to 9999 MM = 01-12 DD = 01-31 HH = 00-23 hours MM = 00-59 minutes SS = 00-59 seconds hh = 01-12 offset hours mm = 00-59 offset minutes Example: 20060901-07:39Z is 07:39 UTC on 1st of September 2006 Example: 20060901-02:39-05 is five hours behind UTC, thus Eastern Time on 1st of September 2006 Example: 20060901-15:39+08 is eight hours ahead of UTC, Hong Kong/Singapore time on 1st of September 2006 Example: 20060901-13:09+05:30 is 5.5 hours ahead of UTC, India time on 1st of September 2006
data	String	String field containing raw data with no format or content restrictions. Data fields are always

Type Name	Base Type	Description
		immediately preceded by a length field. The length field should specify the number of bytes of the value of the data field (up to but not including the terminating SOH). Caution: the value of one of these fields may contain the delimiter (SOH) character. Note that the value specified for this field should be followed by the delimiter (SOH) character as all fields are terminated with an "SOH".
Pattern		Used to build on and provide some restrictions on what is allowed as valid values in fields that uses a base FIX data type and a pattern data type. The universe of allowable valid values for the field would then be the union of the base set of valid values and what is defined by the pattern data type. The pattern data type used by the field will retain its base FIX data type (e.g. String, int, char).
Tenor	Pattern	Used to allow the expression of FX standard tenors in addition to the base valid enumerations defined for the field that uses this pattern data type. This pattern data type is defined as follows: Dx = tenor expression for "days", e.g. "D5", where "x" is any integer > 0 Mx = tenor expression for "months", e.g. "M3", where "x" is any integer > 0 Wx = tenor expression for "weeks", e.g. "W13", where "x" is any integer > 0 Yx = tenor expression for "years", e.g. "Y1", where "x" is any integer > 0
Reserved100Plus	Pattern	Values "100" and above are reserved for bilaterally agreed upon user defined enumerations.
Reserved1000Plus	Pattern	Values "1000" and above are reserved for bilaterally agreed upon user defined enumerations.
Reserved4000Plus	Pattern	Values "4000" and above are reserved for bilaterally agreed upon user defined enumerations.
XMLData	String	Contains an XML document raw data with no format or content restrictions. XMLData fields are always immediately preceded by a length field. The length field should specify the number of bytes of the value of the data field (up to but not including the terminating SOH).
Language	String	Identifier for a national language - uses ISO 639-1 standard Example(s): en = English es = Spanish
LocalMktTime	String	String field representing the time local to a particular market center. Used where offset to UTC varies throughout the year and the defining market center is identified in a corresponding field. Format is HH:MM:SS where HH = 00-23 hours MM = 00-59 minutes SS = 00-59 seconds In general only the hour token is non-zero. Example: 07:00:00
XID	String	The purpose of the XID datatype is to define a unique identifier that is global to a FIX message. An identifier defined using this datatype uniquely identifies its containing element, whatever its type and name is. The constraint added by this datatype is that the values of all the fields that have an ID datatype in a FIX message must be unique.
XIDREF	String	The XIDREF datatype defines a reference to an identifier defined by the XID datatype.