TMX eXtreme Message Transfer (XMT) Protocol

Specification

Document Revision: 1.0
Date of Issue: October 20, 2010

Equities	Derivatives	Fixed Income	Energy	Data
Toronto Stock Exchange	Montréal Exchange	Shorcan	NGX	TMX Datalinx
TSX Venture Exchange	CDCC			PC Bond
Equicom	Montréal Climate Exchange			

Disclaimer and Rights Granted

The information contained in this document and any other information or documentation related to this specification and associated message formats (collectively, along with any upgrades, updates and new versions or releases provided to you by TMX from time to time, referred to as the "Specifications") are provided "as is", and neither TMX nor any of its affiliates makes any representation or warranty, express or implied, as to the contents of the Specifications (or the results obtained by using it) and each such person specifically disclaims any representation, warranty or condition of accuracy, completeness, merchantability, fitness for a particular purpose or that it is error-free. By using the Specifications, you are agreeing o assume the entire risk associated with its use.

TMX and its affiliates shall have no liability for damages of any kind arising in any manner out of or in connection with your use of, or your inability to use, the Specifications, whether direct, indirect, incidental, special or consequential (including, without limitation, loss of data, loss of use, third party claims, lost profits, lost revenues and strict liability), arising under contract or otherwise, whether or not TMX has been advised of, or otherwise might have anticipated the possibility of, such damages.

Please note that TMX will require prior certification of any system designed to connect to its trading and market information as well as your participation from time to time in connection with the testing of changes or upgrades to Specifications or the systems used to connect to TMX.

The information contained in the Specifications is proprietary and confidential information belonging to TMX and its licensors or service providers, as applicable. Copyright and trade-mark rights and any other intellectual property in the Specifications belong to TMX (or its licensors or service providers, as applicable). Your permitted use of the Specifications, in whole or in part, is limited to the non-exclusive, non-transferable, revocable, non-assignable, personal right for you only to build a connection between your systems and TMX's trading and market information system. You are not permitted to use all or any part of the Specifications for order entry other than on Toronto Stock Exchange or TSX Venture Exchange. Any uses not specifically set out herein are prohibited. TMX may revoke this right at any time, on notice to you, and you agree to promptly comply with such notice. Do not reproduce, disclose, distribute, publish, sell, commercialize or make any part of the Specifications available, in whole or in part, to any person without TMX's prior written consent. If you do not agree with, or do not wish to comply with, any of the above, please notify TMX and, in accordance with TMX's instructions, either return or destroy any copies or version of the Specification in your possession.

Contact Information

For questions related to TMX eXtreme Message Transfer Protocol, contact TMX Vendor Services at 416.947.4705 or vendor_services@tmx.com.

Table of Contents

Chapter 1 Overview

The need to create a highly flexible and efficient protocol is becoming more and more important in the microsecond computing world. Flexibility provides fast and low-cost response to the changing demands of new products, variations on existing functionality, and adaption to production. Efficiency ensures the minimum amount of latency in information delivery.

TMX employs the Securities Trading Access Message Protocol (STAMP) in dissemination and recovery of feed messages. While STAMP has certain merits in its bundle management and multi-feed recovery mechanism, it does not meet the demand for low latency. STAMP uses a tag-based approach, which requires extra parsing and formatting to deliver messages. Also, it does not support binary data in the payload, which results in unnecessarily longer messages and more data serialization cycles.

To ensure the minimum amount of latency in preparing and transporting the message, the combination of the shortest message length and the closest form to the source of generation are most important in the design of a protocol. Using a structured payload with no extra tags or delimiters (provided most of the fields are mandatory) and allowing binary data achieves the shortest message length. Minimizing any extra conversion achieves the closest form to the source of generation. Furthermore, allowing parallel streams of data also guarantees linear expandability for future volume growth. Other types of messages are delivered in batch, for example, stock images and order book images for the markets. Therefore, the protocol will be comprehensive if it supports both batch and boxcar delivery.

1.1 Benefits of a new protocol

The TMX eXtreme Message Transfer (XMT) Protocol has many benefits. It:

- Provides a flexible, readily acceptable, and highly efficient protocol
- Incorporates the benefits of STAMP bundle management and its multi-feed recovery mechanism
- Has a well-defined engine-native structure with binary support
- Allows multi-streaming and logical groups of messages
- Allows batch delivery support with boxcaring capability
- Built-in flexibility for easy certification of the protocol with new users
- Allows n days x 24 hours continuous trading
- Allows parallel versioning
- Avoids any restrictions with other protocols
- Avoids any patent issues between well-known protocols

1.2 Protocol at a glance

The TMX XMT Protocol is a generic protocol that can be used either in synchronous mode (acknowledge) or asynchronous mode (for example, store and forward), at ISO level 5 (Session layer).

It provides simplex/duplex communication between sender and receiver to allow message exchange between applications. It can be used with point-to-point (TCP) connections or connectionless (UDP) configurations with full duplex or simplex setup.

It contains fixed- and variable-length portions in the message, such that it provides quick access to the key information that allows flexible dynamic grouping of messages. To further improve the throughput, it allows the message body to be packed to transmit mass transactions more efficiently.

Unique Message Reference and Sequence Jump mechanisms are employed to speed up the delivery service time and the processing time between persistent message stores (which may reside on different computer systems), while still ensuring re-delivery of messages in failure or disaster situations.

The TMX XMT Protocol specifies what format of the message is exchanged between two service points. It does not specify how to provide message routing and communication, message notifications, transaction-based protection, and process monitoring control. These are provided by either the operating system or a middleware product.

Note: This protocol does not provide specific authentication between the sender and the receiver, because it is intended to be used with trusted systems. If there is any concern with the security aspect of the protocol, the protocol needs to be revised to include the authentication features.

While there is no specific authentication, message integrity is generally guaranteed by the network/transport protocol. In TCP/IP, message integrity is guaranteed by the end-to-end reliability. In UDP, message integrity is guaranteed by the UDP checksum in the UDP header.

To achieve best performance when the protocol is used in connectionless (UDP) mode, fragmentation of packets should be avoided. The size of the total messages should be limited to the minimum MTU size supported along the delivery path.

Chapter 2 Protocol Description

This section provides an overview of the data types, message format, and the session protocol used by XMTP.

Note: In the following sections, the message body is colour coded to represent different types of content: green for **administrative** and **orange** for business.

2.1 Data types

To enable efficient transmission of data, fields use one of the following data types:

- Alphanumeric
- Numeric
- Binary

2.1.1 Alphanumeric (A)

The alphanumeric data type consists of the displayable characters (from hex 0x20 to hex 0x7e) left justified with trailing blanks (hex 0x20). Empty fields are represented by blanks.

2.1.2 Numeric (N)

The numeric data type consists of the displayable digits (from hex 0x30 to hex 0x39) right justified with leading zeroes (hex 0x30). Empty fields are represented by zeroes.

2.1.3 Binary (B)

The binary data type consists of binary data stored in little-endian byte order. The size, in bytes, ranges from 2^0 to 2^3 (1 to 8). Empty fields are represented by nulls (i.e. binary value 0).

2.2 Protocol framing

<Frame> <Protocol> <Version> <Length> <Header> [<Body>]

Field	Subfield	Bytes (Type)	Description / Value
<Frame>	**Start of Frame**	**1 (B)**	0x02
<Protocol>	**Protocol Name**	**1 (A)**	Protocol name. "X"
<Version>	**Protocol Version**	**1 (N)**	Protocol major version. "n" (numeric) is the protocol's version number; for example, 1 for version 1.
<Length>	**Message Length**	**2 (B)**	Length of message from <Header> to [<Body>] inclusive.
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	• "A" – Receiver needs to ack • "D" – Possible duplicates
	Num Body	1 (B)	Number of respective bodies
<Body>	**Optional**	Variable	
	<AdmHdr> <AdmBdy …>	Either	Administration bodies. Only <AdmBdy> repeats Num Body times.
	<BusHdr …> <BusBdy …>	Or	Business bodies. Both <BusHdr> and <BusBdy> repeat Num Body times.

Field	Subfield	Bytes (Type)	Description / Value
Administration body			
<AdmHdr>		Variable	
	Msg Length	2 (B)	Length of this message (AdmHdr + AdmBdy) in bytes including this field
	Msg Type	1 (B)	• 0x30 – Heartbeat • 0x31 – Login Request • 0x32 – Login Response • 0x33 – Logout message • 0x34 – Ack message • 0x35 – Replay Request • 0x36 – Sequence Jump • *0x37 – Reserved* • 0x38 – Operation message • 0x39 – Reject
	Admin ID	1 (B)	Admin ID sequence for short window (1 to 255) **Note**: 0 is not allowed as a value.
	<Specific AdmHdr>	Variable	Specific header format for required Msg Type
<AdmBdy …>		Variable	Specific body format for required Msg Type
Business body			
<BusHdr …>		**12**	
	Msg Length	2 (B)	Length of this message (BusHdr + BusBdy) in bytes including this field
	Msg Type	1 (B)	0x41 to 0x7e – Business Messages
	Msg Version	1 (B)	Message version number; for example, 10 for version 1.0.
	Source ID	1 (A)	Source ID. “0” for non-source-related.
	Stream ID	2 (B)	Stream ID per Source ID. 0 for non-stream-related.
	Sequence-0	1 (B)	Reserved for future expansion. Value is 0.
	Sequence-1	4 (B)	4-byte sequence per stream ID. 0 for unsequenced messages.
<BusBdy …>		Variable	Specific format for each MsgType

2.2.1 Msg Type

0x30 – Heartbeat

The Heartbeat is sent by either side during idle periods for the opposite side to detect whether the session is still alive.

0x31 – Login Request

Login Request is used to establish a connection-oriented session for exchanging messages.

0x32 – Login Response

A successful login results in a Login Response to send back to the originator of the request.

0x33 – Logout message

A Logout message is sent to notify the other side that the sending side is leaving the session.

0x34 – Ack message

Ack message is typically used when the implementation of the protocol requires acknowledgements or responses of requests/messages; for example, Replay Request or a synchronous transaction.

0x35 – Replay Request

A request to replay the transmission of previously sent messages.

0x36 – Sequence Jump

Sequence Jump is used to advance the sequence number from the current to the new sequence to indicate that certain messages are to be resent, are no longer available, or are not deterministic in a failure or disaster situation.

0x37 – Reserved

Reserved for future use.

0x38 – Operation message

Operation message is a special form of message to convey a description, to signal the next business messages to be dropped for special situations.

0x39 – Reject

Reject is used in response to the request message, when it fails validation or is not allowed in certain states.

0x41 to 0x7e – Business message

Business message describes the range of business messages.

2.2.2 Ack-Required/Poss Dup

Acknowledgement is required in the receiver side if Ack-Required is “A” in the request or message. This is used only in a transactional-based conversation, where each transaction needs to be acknowledged in the application.

Possible duplicates, indicated by Poss Dup = “D”, could happen in known situations such as a failure, disaster, or special simulation. Whether the duplicate happens in known or unknown situations, the receiver should use a Unique Message Reference or the Admin ID to handle the duplicates and should drop them if true.

Ack-Required and Poss Dup are intentional and different in direction, thus mutually exclusive. That is, Poss Dup does not need the receiver side to acknowledge and Ack-Required will not indicate a possible duplicate.

2.2.3 Session ID

The Session ID identifies the sender and receiver. It also allows multiplexing of data through a single logical connection. Session ID is pre-defined and is constant throughout the business day. Session ID uniquely identifies the logical session point associated with a session. A logical session point is either the sender/responder or receiver/requester of that session.

A logical session allows data multiplexing through different streams (hence different sets of sequence numbers); streams are formed by Source ID, Stream ID and Sequence.

2.2.4 Admin ID

Admin ID is used with administrative messages to provide unique message identification relative to the previous administrative message for a short window. Under normal conditions, administrative messages are sent one or two at a time. Repetition of the same administrative message can be determined by the looking for the same Admin ID. Moreover, the repeated looping of an Admin ID in a short time could indicate the possible malfunction of middleware. The maximum value is 255.

2.2.5 Source ID

Source ID provides unique identification for the different sources that collective systems or applications offer. Subscribers or receivers can use this ID to determine which market they would like to process. A ‘0’ value is used to represent certain non-source-related messages.

2.2.6 Stream ID

Stream ID provides unique identification for logical partitions, slices, or groupings, per Source ID. Subscribers or receivers can use the Source ID and the Stream ID to determine what groups under a particular source they would like to process. A ‘0’ value is used to represent certain non-stream-related messages.

2.2.7 Sequence-1

Sequence-1 uniquely identifies each Stream ID under a Source ID. It is a serial, incremental by 1, linear sequence number that allows subscribers or receivers to determine any losses, gaps, or duplicates for the particular group. A ‘0’ value represents certain unsequenced messages. The maximum value is 4,000,000,000.

2.2.8 Sequence-0

Sequence-0 is a future expansion byte for expanding the sequence number for volume growth to trillions and n days x 24 hours continuous trading.

2.2.9 Unique Message Reference (UMR)

The Unique Message Reference (UMR) is a unique identifier to identify a business message across the whole system. It is guaranteed uniqueness for the day can span days (future expansion). It is concatenated by Source ID + Stream ID + Sequence.

Definition

The Unique Message Reference is unique in a trading enterprise. It identifies a specific message. That is, no two messages in the trading enterprise can have the same UMR. Usually the generating sources of the message are responsible for creating the UMR. However, when there are multiple sources of inputs into the trading enterprise, the uniqueness should be coordinated among these sources to ensure the UMR is unique.

Purpose

The Unique Message Reference protects the message from being reprocessed. Under disaster recovery situations, in-flight messages may be resent. To prevent a message from being processed again, it must have a unique identifier such that the back-end system knows that the message has been processed.

Note: UMR is not needed only when the message is safely stored and checkpointed before leaving a service point in any system. However, the service time and the latency of processing will be significantly increased in every service point.

Usage

When a back-end system processes the message, it should check the UMR for uniqueness. The simplest way is to check for every message. An improved method of checking UMR will have further information supplied to assist the checking so that it is only required in uncertain situations raised by the previous service points. This essentially limits the checking to a very small window of messages and hence reduces the impact on the back-end system to almost zero. If the back-end system has already processed the message and finds the same UMR, it ignores the message.

2.3 Protocol synchronization

The XMT Protocol is very flexible and hence does not provide a definitive behaviour in this level of specification. Specific configuration of protocol usage is required per implementation and the specification and the specific implementation configuration should be used together for building the solution. Appendix A, “Broadcast Scenario” on page 24 and Appendix B, “Transaction Scenario” on page 28 depict two typical implementation scenarios.

2.3.1 Heartbeat

The Heartbeat message is sent by either side during idle periods for the opposite side to detect whether the session is still alive. Depending on implementation, it also contains the last sent sequences of the business streams, for detection of possible gaps or lost packets. The Heartbeat interval is either pre-defined for the implementation or negotiable in the Login Request.

Field	Subfield	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	“A” – Receiver needs to ack, if required, using Ack message
	Num Body	1 (B)	Number of respective bodies
<AdmHdr>		**6**	
	Msg Length	2 (B)	Length of this message (AdmHdr + AdmBdy) in bytes including this field
	Msg Type	1 (B)	0x30 – Heartbeat
	Admin ID	1 (B)	Admin ID for short window (1 to 255) **Note**: 0 is not allowed as a value.
	HB Interval	2 (B)	Heartbeat interval in milliseconds (1 to 65536).
<AdmBdy …>		**Variable**	Repeats Num Body times
	Source ID	1 (A)	Source ID. “0” for non-source-related
	Stream ID	2 (B)	Stream ID per Source ID. 0 for non-stream-related.
	Sequence-0	1 (B)	Reserved for future expansion. Value is 0.
	Sequence-1	4 (B)	Sequence per stream ID. 0 for unsequenced messages.

2.3.2 Login Request

The Login Request message is used to establish a connection-oriented session for exchanging messages. It allows certain negotiable parameters to control the characteristics of the session.

Connectionless sessions do not require a Login Request.

To provide the capability of handling multiple concurrent sessions and the many-to-many relationship between the service points, Session ID is used to identify the sender or the originator of the request/message.

Field	Sub-field	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	"A" – Receiver needs to ack by default with Login Response
	Num Body	1 (B)	0
<AdmHdr>		**12**	
	Msg Length	2 (B)	Length of this message (AdmHdr) in bytes including this field
	Msg Type	1 (B)	0x31 – Login Request
	Admin ID	1 (B)	Admin ID for short window (1 to 255) **Note**: 0 is not allowed as a value.
	HB Interval	2 (B)	Negotiable heartbeat interval in milliseconds (1 to 65536) in the request to the receiver
	Replay Win Size	2 (B)	Negotiable maximum number of messages (in thousands) allowed in the window in the request to the receiver (1 to 65000, that is, 65,000,000 messages)
	Replay Win Num	2 (B)	Negotiable maximum number of replay requests allowed in the window in the request to the receiver (1 to 65000)
	Credits	2 (B)	Reserved for future use. Value is 0.

2.3.3 Login Response

A successful login results in a Login Response message being sent back to the originator of the request. Login Response confirms the establishment of a session and confirms all the negotiable parameters that were sent in the Login Request. Any messages/requests sent before receiving the Login Response will be ignored by the receiving side.

Field	Sub-field	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	Not used as this is an acknowledgement
	Num Body	1 (B)	Number of respective bodies
<AdmHdr>		**13**	
	Msg Length	2 (B)	Length of this message (AdmHdr) in bytes including this field
	Msg Type	1 (B)	0x32 – Login Response
	Admin ID	1 (B)	Admin ID of the request
	HB Interval	2 (B)	Negotiable heartbeat interval in milliseconds (1 to 65536) in the request to the receiver.
	Replay Win Size	2 (B)	Confirmed maximum number of messages (in thousands) allowed in the window (1 to 65000, that is, 65,000,000 messages)
	Replay Win Num	2 (B)	Confirmed maximum number of replay requests allowed in the window (1 to 65000)
	Replay Win	1 (B)	Replay window in seconds (1 to 255)
	Credits	2 (B)	Reserved for future use. Value is 0.

2.3.4 Logout

A Logout message is sent to notify the other side that the sending side is leaving the session for the existing phase of the state. A Logout message is accepted only after the session is established successfully. Depending on the implementation, it is either a one-way notification or two-way requiring acknowledgement.

Field	Sub-field	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	"A" – Receiver needs to ack if required
	Num Body	1 (B)	0
<AdmHdr>		**4**	
	Msg Length	2 (B)	Length of this message (AdmHdr) in bytes including this field
	Msg Type	1 (B)	0x33 – Logout Message
	Admin ID	1 (B)	Admin ID for short window (1 to 255) **Note**: 0 is not allowed as a value.

2.3.5 Ack

An Ack message is typically used when the implementation of the protocol requires acknowledgement of requests/messages; for example, synchronous transactions. When required, the receiving side of the request/message responds with an Ack message to confirm the receipt of the request/message or the already sent business messages as a result of a Replay Request.

Field	Sub-field	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	Not used as this is acknowledgement
	Num Body	1 (B)	• 0 for Administration messages ack except Replay Request • n for Replay request ack (or responses) • n for Business messages ack
<AdmHdr>		**4**	
	Msg Length	2 (B)	Length of this message (AdmHdr + AdmBdy) in bytes including this field
	Msg Type	1 (B)	0x34 – Ack Message
	Admin ID	1 (B)	Admin ID of the request/message
<AdmBdy …>		**Variable**	Repeats Num Body times
	Msg Length	2 (B)	Length of this message (BusHdr + BusBdy) in bytes including this field
	Msg Type	1 (B)	0x41 to 0x7e – Business Messages
	Msg Version	1 (B)	Message version number; for example, 10 for version 1.0
	Source ID	1 (A)	Source ID of the Business message.
	Stream ID	2 (B)	Stream ID per Source ID of the Business message
	Sequence-0	1 (B)	Reserved for future expansion. Value is 0.
	Sequence-1	4 (B)	Sequence per stream ID of the Business message
	<BusBdy>	Variable	Business content already sent

2.3.6 Replay Request

The Replay Request message is a request to replay the transmission of previously sent messages. Depending on the implementation, a single replay can specify multiple streams of messages. A successful replay will resend all of the requested messages using Ack-Message with Poss Dup set. The receiving side of the request can choose to resend messages or not, depending on the implementation. If messages are not for retransmission, a Sequence Jump is used to signal a gap of sequenced messages that are no longer available.

The sender sends a message with Admin ID to the receiver, and the receiver acknowledges the request by sending multiple Ack messages containing the previously sent business messages.

Admin ID provides a coordinated identifier for retransmission requests from the requester. A requester issuing multiple requests can use this ID to process requests accordingly.

Field	Sub-field	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	"A" – Receiver needs to ack by default
	Num Body	1 (B)	Number of respective bodies
<AdmHdr>		**8**	
	Msg Length	2 (B)	Length of this message (AdmHdr + AdmBdy) in bytes including this field
	Msg Type	1 (B)	0x35 – Replay Request
	Admin ID	1 (B)	Admin ID for short window (1 to 255) **Note**: 0 is not allowed as a value.
	Session ID	4 (B)	Session ID of the source to replay the message
<AdmBdy …>		**Variable**	Repeats Num Body times
	Source ID	1 (A)	Source ID
	Stream ID	2 (B)	Stream ID per Source ID
	Sequence-0	1 (B)	Reserved for future expansion. Value is 0.
	Sequence-1 Start	4 (B)	Sequence per stream ID. Starting sequence.
	Sequence-1 End	4 (B)	Sequence per stream ID. Ending sequence.

2.3.7 Sequence Jump

The Sequence Jump message is used to advance the sequence number from the current to the new sequence to indicate that certain messages are not to be resent, no longer available, or not deterministic in a failure or disaster situation. The receiving side should skip to the new sequence and continue from there. In a failure or disaster situation, a Unique Message Reference is required to determine duplicates.

Field	Sub-field	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	"A" – Receiver needs to ack if necessary
	Num Body	1 (B)	Number of respective bodies
<AdmHdr>		**5**	
	Msg Length	2 (B)	Length of this message (AdmHdr + AdmBdy) in bytes including this field
	Msg Type	1 (B)	0x36 – Sequence Jump
	Admin ID	1 (B)	• Admin ID for short window (1 to 255) or • Admin ID of the request/message
	Reason Code	1 (B)	Reason code for the jump: • 0x01 – Do not resend • 0x02 – No longer available • 0x03 – Disaster
<AdmBdy …>		**Variable**	Repeats Num Body times.
	Source ID	1 (A)	Source ID
	Stream ID	2 (B)	Stream ID per Source ID
	Sequence-0	1 (B)	Reserved for future expansion. Value is 0.
	Sequence-1 Current	4 (B)	Sequence per stream ID. Current next sequence.
	Sequence-1 New	4 (B)	Sequence per stream ID. New next sequence.

2.3.8 Operation

The Operation message is a special form of message to convey a description, to signal the next business messages to be dropped for a special situation, and in future the connection/replay/reject credits information.

Field	Sub-field	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	"A" – Receiver needs to ack if necessary
	Num Body	1 (B)	0.
<AdmHdr>		**105**	
	Msg Length	2 (B)	Length of this message (AdmHdr) in bytes including this field
	Msg Type	1 (B)	0x38 – Operation Message
	Admin ID	1 (B)	Admin ID for short window (1 to 255) **Note**: 0 is not allowed as a value.
	Operation Code	1 (B)	Code for the message: • 0x00 – Information • 0x01 – Warning • 0x02 – Alert • 0x03 – Next business message to be dropped
	Message	100 (A)	Alphanumeric descriptive message.

2.3.9 Reject

The Reject message is used in response to the Request message, when it fails validation or is not allowed in certain states. Fatal conditions in a rejection will cause immediate dissolution of the connection/session.

Field	Sub-field	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	Not used
	Num Body	1 (B)	• 0 for Administration messages rejection • n for Business messages rejection
<AdmHdr>		**36**	
	Msg Length	2 (B)	Length of this message (AdmHdr + AdmBdy) in bytes including this field
	Msg Type	1 (B)	0x39 – Reject Message
	Admin ID	1 (B)	Admin ID of the request/message
	Reject Code	1 (B)	Code for rejection. • 0x00 – Information • 0x01 – Warning • 0x02 – Critical • 0x03 – Fatal
	Reject Subcode	1 (B)	• 0x01 – Invalid syntax • 0x02 – Invalid session states/conditions • 0x03 – Invalid values • 0x04 – Function not implemented • 0x05 – Function not allowed • 0x06 – Function temporarily not available, but retryable • 0x07 – Message temporarily not available, but retryable • 0x08 – Duplicate request/message • 0x09 – Others, see Message
	Message	30 (A)	Alphanumeric descriptive message

Field	Sub-field	Bytes	Description / Value
<AdmBdy ...>		**Variable**	Repeats Num Body times
	Msg Length	2 (B)	Length of this message (BusHdr) in bytes including this field
	Msg Type	1 (B)	0x41 to 0x7e – Business Messages
	Msg Version	1 (B)	Message version number; for example, 10 for version 1.0
	Source ID	1 (A)	Source ID of the Business message.
	Stream ID	2 (B)	Stream ID per Source ID of the Business message.
	Sequence-0	1 (B)	Reserved for future expansion. Value is 0.
	Sequence-1	4 (B)	Starting sequence per stream ID of the Business message that results in the rejection

2.3.10 Business

The Business message contains Business Content messages. It can also be used as a repeat of the last sent business message (with Poss Dup) in special situations.

Field	Sub-field	Bytes	Description / Value
<Header>		**6**	
	Session ID	4 (B)	Constant session ID
	Ack-Required/ Poss Dup	1 (A)	• "A" – Ack required if set for transaction mode • "D" – Possible duplicate if set, otherwise new message
	Num Body	1 (B)	Number of BusHdr and BusBdy
<BusHdr …>		**12**	
	Msg Length	2 (B)	Length of this message (BusHdr + BusBdy) in bytes including this field
	Msg Type	1 (B)	0x41 to 0x7e – Business Messages
	Msg Version	1 (B)	Message version number; for example, 10 for version 1.0
	Source ID	1 (A)	Source ID of the Business message.
	Stream ID	2 (B)	Stream ID per Source ID of the Business message.
	Sequence-0	1 (B)	Reserved for future expansion. Value is 0.
	Sequence-1	4 (B)	Sequence per Stream ID of the Business message.
<BusBdy …>		**Variable**	Specific format for each MsgType.

Chapter 3 Business Content Messages

The format of Business Content messages, which are processed by the business application in back-end systems, will be chosen and defined according to the specific implementation of this protocol between applications.

Content is supported in alphanumeric, numeric, and binary data types, as specified in section 2.1, “Data types”, on page 6.

Appendix A Broadcast Scenario

Legend :

…..> UDP broadcast from feed session

-----> TCP request/response in recovery session

===== Connection established

~~~~~ Connection dissolved

Scenario	Sender message / response	Receiver request	Description
1	**Broadcast from Sender**		
1.1	• Msg Type – Heartbeat • Session ID – 08080103 • Num Body – 2 • Admin ID – 1 • HB Interval – 1000 • Sequence-1 – 0 • Sequence-1 – 0	……>	Heartbeat from sender containing last sent sequence. Very first sequence is unsequenced yet, which is 0. There are 2 parallel streams in the session from sender. Heartbeat interval is pre-defined at 1 second.
1.2	• Num Body - 2 • Msg Type – 0x41 • Source ID – Q • Stream ID – 101 • Sequence-1 – 1 • Msg Type – 0x42 • Source ID – Q • Stream ID – 102 • Sequence-1 – 1	……>	Business messages from sender. Different Msg Types are boxed in the same message without violating the priority sequence. Receiver checks the Sequence-1 for gaps or duplicates for all streams.
2	**Recovery session establishment** (optional, only needed when receiver requires to recovery gaps)		
2.1	===== <-----	• Msg Type – Login Request • Session ID – 09070013 • Admin ID – 101	Receiver connects to IP/Port. Receiver logs in to establish a point-to-point session for recovery of messages.
~~~~~

Scenario	Sender message / response	Receiver request	Description
2.2	• Msg Type – Login Response • Session ID – 00000032 • Admin ID – 101 • HB Interval – 1000 • Replay Win Size – 1000 • Replay Win Num – 90 • Replay Win – 30	----->	Sender accepts login and replies with pre-defined parameters of the session. A recovery session is now established.
2.3	• Msg Type – Heartbeat • Session ID – 00000032 • Num Body – 0 • Admin ID – 1 • HB Interval – 1000	----->	Heartbeat on recovery session to indicate session is alive.
3	**Gap recovery from Receiver** (optional, only needed when receiver requires to recover gaps)		
3.1	• Msg Type – Heartbeat • Session ID – 08080103 • Num Body – 1 • Admin ID – 100 • HB Interval – 1000 • Sequence-1 – 233	…..>	Receiver's expected sequence from this stream is 231. A gap of 3 messages is detected from the heartbeat.
3.2	<-----	• Msg Type – Replay Request • Session ID – 09070013 • Admin ID – 102 • Session ID – 08080103 • Num Body – 1 • Sequence-1 Start – 231 • Sequence-2 End – 233	Receiver requests the resending of 3 messages from the broadcast session for the stream.

Scenario	Sender message / response	Receiver request	Description
3.3	• Msg Type – Ack Message • Session ID – 00000032 • Poss Dup – 'D' • Admin ID – 102 • Num Body – 3	----->	Sender responds with the requested business messages, again in one write. **Note**: Admin ID acknowledges the requested one.
	• Sequence-1 – 231 • Sequence-1 – 232 • Sequence-1 – 233		
4	**Requested gaps no longer available** (continuation from 3.2)		
4.3	• Msg Type – Sequence Jump • Session ID – 00000032 • Admin ID – 102 • Num Body – 1 • Reason Code – 0x02 • Sequence-1 Current – 231 • Sequence-1 New – 233	----->	Sequence 231 and 232 for the stream are no longer available. Sender expects Receiver to skip requesting from those again.
4.4	• Msg Type – Ack Message • Session ID – 00000032 • Poss Dup – 'D' • Admin ID – 102 • Num Body – 1	----->	Sequence 233 is resent again.
	Sequence-1 – 233		
5	**Invalid Login request from Receiver**		
5.1	===== <-----	• Msg Type – Login Request • Session ID – 09070013 • Admin ID – 101 • Replay Win Size – 10000	Receiver connects first, then logs in to establish a point-to-point session for the recovery of messages and to specify the replay window size outside the allowable range.

Scenario	Sender message / response	Receiver request	Description
5.2	• Msg Type – Reject • Session ID – 00000032 • Admin ID – 101 • Num Body – 0 • Reject Code – 0x01 • Reject sub code – 0x05	----->	Sender rejects the request for the login with “Warning: Function not allowed”. Connection is still alive.
5.3	<-----	• Msg Type – Login Request • Session ID – 01111113 • Admin ID – 101	Receiver logs in to establish a point-to-point session for the recovery of messages and specifies incorrectly the session ID.
5.4	• Msg Type – Reject • Session ID – 00000032 • Admin ID – 101 • Num Body – 0 • Reject Code – 0x02 • Reject sub code – 0x09 • Message – “Please contact Vendor Services”	-----> ~~~~~	Sender rejects the request for the login. Connection is dissolved as this is fatal.
6	**Logout from Receiver** (optional)		
6.1	<----- ~~~~~	• Msg Type – Logout • Session ID – 09070013 • Admin ID – 101	Receiver logs out without needing sender to acknowledge.

Appendix B Transaction Scenario

To be provided in the next version of the protocol.

Appendix C Credits

TMX XMT Protocol Committee
Derek Hwong, TSX-FIX and TSX-STAMP technical contact, FIX Inter-Party Latency Working Group, FIX Global and Canadian Exchanges and Markets Committees, TSX Express Protocol Committee.

Appendix D Acronyms

The following acronyms are used in this document:

- **FIX**: Financial Information Exchange
- **IP**: Internet Protocol
- **ISO**: International Standards Organization
- **MTU**: Maximum Transmission Unit
- **STAMP**: Securities Trading Access Message Protocol
- **TCP**: Transfer Control Protocol
- **UDP**: User Datagram Protocol
- **XMT**: eXtreme Message Transfer
- **XMTP**: eXtreme Message Transfer Protocol

TMX Vendor Services
The Exchange Tower
130 King St. W.
Toronto, ON M5X 1J2

phone: 416.947.4705
e-mail: vendor_services@tmx.com