MEMX-TCP

V1.1

07/27/20

This specification is being provided to you strictly for informational purposes solely for the purpose of developing or operating systems that interact with the Members Exchange, or MEMX. All proprietary rights and interest in this specification and the information contained herein shall be vested in MEMX and all other rights including, but without limitation, patent, registered design, copyright, trademark, service mark, connected with this publication shall also be vested in MEMX. No part of this specification may be redistributed or reproduced in any form or by any means or used to make any derivative work (such as translation, transformation, or adaptation) without written permission from MEMX. MEMX reserves the right to withdraw, modify, or replace the specification at any time, without notice. No obligation is made by MEMX regarding the level, scope, or timing of MEMX's implementation of the functions or features discussed in this specification.

THE SPECIFICATION IS PROVIDED "AS IS", "WITH ALL FAULTS" AND MEMX MAKES NO WARRANTIES AND DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, OR STATUTORY RELATED TO THE SPECIFICATIONS. MEMX IS NOT LIABLE FOR ANY INCOMPLETENESS OR INACCURACIES IN THE SPECIFICATIONS. MEMX IS NOT LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES RELATING TO THE SPECIFICATIONS OR THEIR USE.

Table of Contents

< THIS PAGE INTENTIONALLY LEFT BLANK>

1 Overview

MEMX-TCP is a TCP-based session-layer protocol for reliable delivery of business messages. Use-cases include filling gaps from a MEMX-UDP or similar unreliable message source, or for streaming data.

All fields in MEMX-TCP are formatted on the wire in network (big endian) byte order. Any numeric fields are unsigned.

Business-layer protocols that use MEMX-TCP will have their messages wrapped with MEMX-TCP headers and messages. Business layer protocols may apply additional restrictions on the MEMX-TCP connection procedure or message flows.

If a malformed or unexpected session-level message is received by the server, the client will be forcibly disconnected via the server sending a TCP `RST` packet. Business level messages may have additional error handling semantics in addition to those applied at the session layer by MEMX-TCP.

1.1 Request Modes

Memx-TCP servers operate in one of two distinct modes: `Replay` or `Stream`. Servers identify the mode they operate in to the client with a `Login Accepted` message.

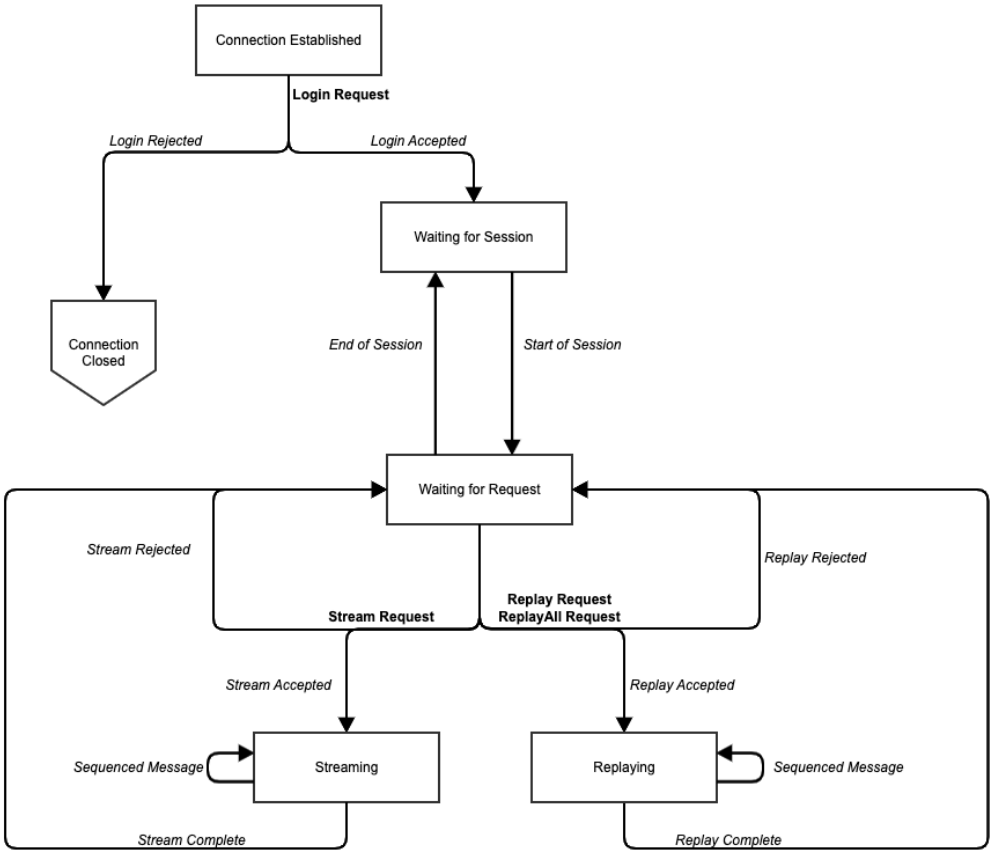
- In `Replay` mode, the server will accept requests for data which starts at a specific sequence number and which specifies a message count. Servers may cap the number of messages which are allowed to be replayed in a given request, session, or client. Any `Stream` request received by a server operating in `Replay` mode will be rejected. Servers may also accept a `ReplayAll` message as an alternative to a `Replay` request, which will replay all available messages from the server, starting at sequence 1, without requiring a starting sequence or count. Replay requests, including the `ReplayAll` request, may be subject to caps on the number of messages returned per request. These caps (if they exist) will be defined per service and as such are not listed in this document.
- In `Stream` mode, the server will accept requests for data which starts at a specific sequence number. If accepted, the server will deliver messages beginning at the requested sequence number to the client as quickly as they are able to process them. Once the client no longer wishes to receive messages, the client should initiate a TCP disconnection. Any `Replay` request received by a server operating in `Stream` mode will be rejected.

1.2 Connection Procedure

- Once a TCP connection is established, clients should send a login message to authorize the session.
- If the client's credentials are not accepted, a `Login Rejected` message will be sent to the client and the TCP connection will be closed.
- If the login is accepted, the server will send a `Login Accepted` message, which will inform the client of which request mode is allowed on the session. (See above).
- After a `Login Accepted` is received, clients should begin sending `Heartbeat` messages to the server on a periodic basis. If the server fails to receive client heartbeats after some interval, it may close the client's TCP connection.
- At some point after the `Login Accepted` message has been received, clients will receive a `Start Of Session` message which contains the current session ID. If the client is connecting into a server operating with a session that is already in progress, they should expect to receive a `Start Of Session` message immediately following the `Login Accepted` message.

- After receipt of a `Login Accepted` message, clients may send either a `Replay Request`, `ReplayAll Request`, or `Stream Request` message to the server to initiate data transmission.
- If the data transmission request is rejected, clients will receive a `Replay Rejected` or `Stream Rejected` message with a corresponding error reason. Clients may issue a new transmission request without closing and re-opening their connection.
- If the data transmission request is accepted, the server will transmit a `Replay Begin` or `Stream Begin` message containing the next expected sequence number, followed by `Sequenced Message` events containing the requested data.

1.3 State Diagram



Message names in **bold** represent state transitions initiated by client-to-server messages. Message names in *italics* represent state transitions initiated by server-to-client messages.

Note that any state can transition to `Connection Closed` if no heartbeats are received by the server within a time interval, or if the client sends any message not allowed by the specification.

1.4 Miscellaneous Notes

- Any message serves as a heartbeat for the purposes of keeping a session alive.
- Both clients and servers should send heartbeats, if required.

- Any unsequenced messages from the client will be rejected until the login procedure has been completed.
- If the session which is in progress is terminated while a client is streaming or replaying data, every attempt will be made to complete the stream or replay request and deliver the complete set of messages to the client. In the event that the server is unable to completely deliver all requested messages to the client before the session must be closed, clients will be forcibly disconnected before receiving a `Stream Complete` or `Replay Complete` message and should attempt to reconnect to the server.

2 Binary Message Format

2.1 Common Header

Offset	Length	Type	Name	Description
0	1	Byte	Message Type	(see below)
1	2	Numeric	Message Length	Total bytes following the head (does not include this header)

2.2 Message Types

2.2.1 Server-to-Client And Client-To-Server

2.2.1.1 Heartbeat (Message Type = 0)

Heartbeat messages *do not* increment the sequence number of the stream. The Message Length field for a heartbeat is set to 0.

Heartbeats are sent from the client to the server, and the server to the client periodically. If no messages are received by a client or a server for some length of time, the client or server should disconnect.

2.2.2 Client-to-Server

2.2.2.1 Login Request (Message Type = 100)

Offset	Length	Type	Name	Description
3	1	Byte	Token Type	Token type
4	*n (max:255)*	Bytes	Token	Login token

Token Types:

Token Type	Description
P	Static Password

When using the Token Type = Static Password approach in the Login Request, the ‘Token’ field should be formatted as ‘user:password’. Where user and password are those supplied by MEMX.

2.2.2.2 Replay Request (Message Type = 101)

Offset	Length	Type	Name	Description
3	8	Numeric	Session ID	The identifier for the session for which data is desired
11	8	Numeric	Next Sequence Number	The first requested sequence number.
19	4	Numeric	Count	Total number of messages to include in the replay.

Replay requests may be sent on a supported server connection after a client completes the login process and receives a `Session Started` message.

Clients must specify their desired session ID copied from a `Session Started` message.

The sequence number must be less than the current maximum sequence minus the requested count. If the request is accepted, clients will receive up to `Count` available messages, starting from the message with a sequence number which matches the requested sequence number.

If any replay request is rejected due to validation or capability mismatches, a `Replay Rejected` message will be sent from the server, and the client will be able to try their request again.

2.2.2.3 ReplayAll Request (Message Type = 102)

Offset	Length	Type	Name	Description
3	8	Numeric	Session ID	The identifier for the session for which data is desired

ReplayAll requests may be sent on a supported server connection after a client completes the login process and receives a `Session Started` message.

Clients must specify their desired session ID copied from a `Session Started` message.

If accepted, the client will receive a `Replay Accepted` message, followed by all messages available on the server at the time of request wrapped in `Sequenced Message` messages, followed by a `Replay Complete` message.

If the request is rejected due to validation or capability mismatches, a `Replay Rejected` message will be sent from the server, and the client will be able to try their request again.

2.2.2.4 Stream Request (Message Type = 103)

Offset	Length	Type	Name	Description
3	8	Numeric	Session ID	The identifier of the session for which data is desired
11	8	Numeric	Next Sequence Number	The first requested sequence number. A value of `0` indicates the current maximum sequence number of the session.

Stream requests may be sent on a supported server connection after a client completes the login process and receives a `Session Started` message.

Clients must specify their desired session ID copied from the `Session Started` message as a means of removing state race conditions between the server's session state and the client.

The sequence number must be less than the current maximum sequence of the requested session. If the request is accepted, clients will receive continuous streaming messages until the end of stream event is received, starting from the requested sequence number.

If any stream request is rejected due to validation or capabilities mismatches, a `Stream Rejected` message will be sent from the server, and the client will be able to try their request again.

2.2.2.5 Unsequenced Message (Message Type = 104)

Offset	Length	Type	Name	Description
3	n	Bytes	Payload	Payload of the message

Unsequenced messages allow clients to send business level messages to the server while streaming data. The messages may be sent on a supported server connection after the user completes the login process and successfully negotiates a `Stream` session.

The unsequenced message is inferred to apply to the session ID of the in-progress `Stream` session. If the `End of Session` message is received, no further unsequenced messages will be accepted on this connection.

The format of the payload of the message is defined by the protocol which is wrapped by this session.

If an unsequenced message is malformed or invalid, the client will be forcibly disconnected.

2.2.3 Server-to-Client

2.2.3.1 Login Accepted (Message Type = 1)

Offset	Length	Type	Name	Description
3	1	Byte	Supported Request Mode	The request mode that this connection supports. (See Below)

Supported Request Modes:

Mode ID	Description
S	Stream Mode
R	Replay Mode

Login Accepted messages *do not* increment the sequence number of the stream.

2.2.3.2 Login Rejected (Message Type = 2)

Offset	Length	Type	Name	Description
3	1	Byte	Reject Code	The code for the rejection type. (See below)

Reject codes:

Reject Code	Retryable?	Description
T	No	Malformed Token
U	No	Token type unsupported (by this server)
V	No	Token Type invalid (on any server)
A	No	Authorization failed

Login Rejected messages *do not* increment the sequence number of the stream.

2.2.3.3 Start of Session (Message Type = 3)

Offset	Length	Type	Name	Description
3	8	Numeric	Session ID	A session identifier for the current session on this connection.

A start of session message will be received when a session becomes available on the connection. In the case where the client is connected before the session has been opened on the server, the start of session message may be received some time after the login process has been completed. In the event where the session is already in progress before the client connects, the start of session message will be received immediately following the `Login Accepted` message.

Start of session messages *do not* increment the sequence number of the stream.

2.2.3.4 End of Session (Message Type = 4)

No additional messages will be received on the previously active session once this message has been received.

2.2.3.5 Replay Begin (Message Type = 5)

Offset	Length	Type	Name	Description
3	8	Numeric	Next Sequence Number	The sequence number of the next message in the stream
11	4	Numeric	Pending Message Count	The number of messages to be delivered in this replay.

Replay begin messages *do not* increment the sequence number of the stream.

The last replay request received by the server was accepted. The current state of the connection has been changed, and the sequence number of the next sequenced message will be the one specified in the `Next Sequence Number` field of this message. A Replay Begin message *may* return a pending message count less than the requested count, based on the configured maximum replay size on the server. If this occurs, the client should issue another request for the remainder of the desired message range after receiving the Replay Complete message for the current request.

If the current session is terminated during a replay and the server cannot complete the replay before it begins to shut down, the server will send a `Replay Complete` message (with a count less than the `Pending Message Count` field in the `Replay Accepted` message) followed immediately by an `End of Session` message.

2.2.3.6 Replay Rejected (Message Type = 6)

Offset	Length	Type	Name	Description
3	1	Byte	Reject Code	The code for the rejection type. (See below)

Reject codes:

Reject Code	Retryable?	Description
R	No	Replay requests are not allowed by this server. Must use stream requests to receive data.
A	No	Replay all requests are not allowed by this server. Must use ranged replay requests to receive data.
P	No	The session ID on the request is not the active session
S	Yes	The start sequence number is out of range

The last replay request received was rejected by the server. The current mode of the connection (sequence number, &c) remains unchanged, and the client should submit a corrected replay request to attempt to change the state of the connection.

Replay rejected messages *do not* increment the sequence number of the connection.

2.2.3.7 Replay Complete (Message Type = 7)

Offset	Length	Type	Name	Description
3	4	Numeric	Message Count	The number of messages which were sent in the replay.

Replay complete messages *do not* increment the sequence number of the stream.

The replay which was in progress has been completed. Clients may now issue a new replay request to receive more data, or disconnect.

2.2.3.8 Stream Begin (Message Type = 8)

Offset	Length	Type	Name	Description
3	8	Numeric	Next Sequence Number	The sequence number of the next message in the stream
11	8	Numeric	Max Sequence Number	The maximum sequence number currently published on this stream

Stream begin messages *do not* increment the sequence number of the stream.

The last stream request received by the server was accepted. The current state of the connection has been changed, and the sequence number of the next sequenced message will be the one specified in the `Next Sequence Number` field of this message.

The `Max Sequence Number` field provides the maximum sequence number published for the requested Session ID at the time this request was made. This number may be used to determine the minimum number of messages that need to be processed to be considered “caught up”, but it should be noted that this number may still increase while the client is replaying previously published messages.

2.2.3.9 Stream Rejected (Message Type = 9)

Offset	Length	Type	Name	Description
3	1	Byte	Reject Code	The code for the rejection type (See below)

Reject codes:

Reject Code	Retryable?	Description
R	No	Stream requests are not allowed by this server. Must use replay requests to receive data.
P	No	The session ID on the request is not the active session
S	Yes	Start sequence number out of range

The last stream request received was rejected by the server. The current mode of the connection (sequence number, &c) remains unchanged, and the client should submit a corrected stream request to attempt to change the state of the connection.

Stream rejected messages *do not* increment the sequence number of the connection.

2.2.3.10 Stream Complete (Message Type = 10)

Offset	Length	Type	Name	Description
3	8	Numeric	Total Sequence Count	The count of messages that were sent on this stream.

The stream which was in progress has been completed. No more sequenced messages will be published on this Session.
Usually, this message will be followed by an `End of Session` message.

Stream complete messages *do not* increment the sequence number of the connection.

2.2.3.11 Sequenced Message (Message Type = 11)

Offset	Length	Type	Name	Description
3	`n`	Bytes	Payload	Payload of the message

Sequenced messages *do* increment the sequence number of the stream.

A sequenced message sent from the server to client. The format of the payload of the message is defined by the protocol which is wrapped by this session.