MEMX-UDP

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-UDP is a UDP-based session-layer protocol for delivery of business messages. MEMX-UDP does not provide guaranteed delivery of messages. Use-cases include receipt of market data from a single broadcast source.

Note that MEMX-UDP is a broadcast-only protocol, meaning that there are no messages sent from the consumer back to the broadcast source. Missed data is recovered via out-of-band recovery mechanisms, such as MEMX-TCP. These mechanisms vary based on the service and will be described in the specification for the service itself.

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

2 Binary Message Format

2.1 Common Header

Offset	Length	Type	Name	Description
0	1	Byte	Message Type	(see below)
1	1	Numeric	Header Length	Total bytes in this header
2	8	Numeric	Session ID	Session identifier
10	8	Numeric	Sequence Number	Sequence number

2.2 Message Types

2.2.1 Heartbeat (Message Type = 0)

`Heartbeat` datagrams are sent periodically to inform clients that a data session with the provided Session ID is active, that connectivity has not been interrupted, and to notify clients of possible data loss during periods of low message traffic.

`Heartbeat` datagrams do not increment the sequence number of following messages. The Sequence Number in the `Heartbeat` datagram is equal to the highest published `Sequenced Message` sequence number for the provided Session ID. Heartbeats sent before any `Sequenced Messages` are published will contain a Sequence Number of 0. There is no other data following the header of a `Heartbeat` datagram.

2.2.2 Session Shutdown (Message Type = 1)

The `Session Shutdown` datagram will be sent in the place of the `Heartbeat` datagram when no further `Sequenced Messages` will be sent for the provided Session ID. This datagram will be sent more than once to ensure all clients receive this notification.

`Session Shutdown` datagrams do not increment the sequence number of following messages. The Sequence Number in the `Session Shutdown` message is equal to the highest published `Sequenced Message` sequence number for the provided Session ID.

There is no other data following the header of a `Session Shutdown` datagram.

2.2.3 Sequenced Message (Message Type = 2)

Each receipt of a `Sequenced Message` datagram contains one or more messages to be processed by the client. The first message in the `Message List` portion of the datagram has a sequence number equal to the `Sequence Number` field in the Common Header. Each subsequent message in the `Message List` increments this value by one. Therefore, after processing all of the messages contained in the datagram, consumers should expect the first message in the next `Sequenced Message` datagram to to have a sequence number equal to the `Sequence Number` field of the current datagram's Common Header, plus the value of the current datagram's `Message Count` field.

Offset	Length	Type	Name	Description
18	2	Numeric	Message Count	Number of messages in this packet
20	n	Message List	Payload	Variable length repeated group of messages in this packet (see below for details)

2.2.3.1 Message List Element

The `Payload` field above is comprised of exactly `Message Count` elements, each encoded as follows:

Length	Type	Name	Description
2	Numeric	Message Length	Number of bytes (n) in the following message
n	Bytes	Message Payload	Payload of the message

Sequenced Message Datagram Example

A Sequenced Message datagram containing two messages, starting at sequence number 6, with message contents `The Quick brown Fox` (Session Sequence 6) and `Jumped Over the Lazy Dog` (Session Sequence 7) would have a layout as follows:
NOTE: Common Header fields are in grey.

Offset	Length	Type	Name	Value
0	1	Byte	Message Type (Sequenced Message)	`2`
1	1	Numeric	Header Length	`18`
2	8	Numeric	Session ID	Current Session ID
10	8	Numeric	Sequence Number	`6`
18	2	Numeric	Message Count	`2`
20	2	Numeric	Message 1 Length	`19`
22	19	Bytes	Message 1 Content (Session Sequence 6)	`The Quick Brown Fox`
41	2	Numeric	Message 2 Length	`24`
43	24	Bytes	Message 2 Content (Session Sequence 7)	`Jumped Over the Lazy Dog`