TEXAS STOCK EXCHANGE

RAKE TCP and UDP

Contents

0.1 Revision History

Date	Version	Notes
June 25, 2025	0.1	Initial version.
September 09, 2025	0.6.draft	Alpha release. Refined message layout, identifiers and enumerations in line with early internal feedback.
October 17, 2025	0.7.draft	Addition of UDP.

Date	Version	Notes
October 28, 2025	0.8.draft	Updated support contact email.

1 Overview

RAKE TCP is a payload agnostic message framing and network session protocol in the tradition of the OSI network layer model. RAKE is message based and guarantees delivery of messages from the Exchange to the Member. RAKE uses TCP/IP sockets for communication and includes a simple authentication mechanism.

Each sequenced message from the Exchange is guaranteed to be delivered in order and exactly once. Un-sequenced messages from the Member have no guarantees for delivery, order, or de-duplication. Every un-sequenced message from the Member will result in one or more corresponding sequenced messages from the Exchange. Members are responsible for tracking responses from the Exchange for their un-sequenced message deliveries. The higher-level protocols such as SEED OrderEntry perform uniqueness guarantees.

Most RAKE server higher-level applications are "Single Message In Flight", meaning the Rake server will queue subsequent messages from the Member until the first message has been processed.

RAKE UDP is a simple message framing protocol for sending messages from the Exchange to Members only. Typically used for market data, although it is also payload agnostic.

1.1 StreamIDs, Session Recovery, and Fail-over

Members should track the sequence number of the messages they have received from the RAKE server. In the event of a network interruption they can logon to the RAKE server and request data to be replayed from a specific sequence number. Both primary and backup RAKE servers maintain the same order of messages.

RAKE sequence numbers start at 1. For TCP clients, logging on with a sequence of 0 skips replay and sends new messages only.

Internal to the Exchange, each matching engine is assigned its own StreamID, and set of symbols to match. RAKE informs the Member which StreamID sourced each message. This is informational only.

1.2 Heartbeats

TCP Heartbeats: Both the Exchange and the Member must send periodic messages to ensure the TCP connection is operating correctly. If network traffic between Exchange and Member is frequent enough that is sufficient. If not, then both sides must send Heartbeat messages. The frequency of these should be no less than once a second. If no traffic or heartbeat message is received after 3 seconds, the TCP connection is assumed to be broken, and the network socket is closed.

No TCP heartbeats should be sent until after a successful LogonResponse from the Exchange.

After a TCP connection, if the Exchange does not receive a `LogonRequest` within 3 seconds, the socket is closed.

UDP Heartbeats: RAKE UDP packets with type HEARTBEAT serve as keepalive indicators. These are sent by the Exchange only and contain no payload data. Members consuming UDP do not send heartbeats back to the Exchange.

2 Types

- This document uses the terms `Byte, Short, Int, Long` to refer to numeric values that are 8,16,32,64 bit long respectively. All are two's complement signed little-endian encoded.

Type	Byte Length
`Byte`	1
`Short`	2
`Int`	4
`Long`	8

- Strings are all `ASCII`, fixed-length. Strings shorter than the fixed-length should be left justified, by right padded with spaces (`ASCII` 0x20) to the full string length
- Enums are encoded as a single `Byte` unless they appear in a bitfield, then they use the number of bits specified.

2.1 LogonResponseCode

Success or failure code from a Logon Request. Additional LogonResponseCode values may be added without notice in the future and should not break Member applications.

Enum Constant	Value	Notes
SUCCESS	0	
INCORRECT_SENDER_COMP	1	
INCORRECT_SESSION	2	
INVALID_NEXT_SEQUENCE	3	The sequence number requested is larger than the largest known sequence, or is less than 0.
INVALID_CONFIGURATION	4	The RAKE server is configured incorrectly for the login request specified. Contact Trading Operations.
INCORRECT_TOKEN	5	

2.2 RakeUdpPacketType

Identifies the RAKE UDP packet variant. Informational only.

Enum Constant	Value	Notes
SEQUENCED_DATA	0	
HEARTBEAT	1	This packet contains no data, and is a heartbeat only. The sequence will be set to the next expected sequence, and the message count will be 0.
START_OF_SESSION	2	The first packet of the session will have this type set. Both the sequence and message count will be set to 0. This packet may repeat multiple times.
END_OF_SESSION	3	The last packet of the session. The sequence number will be the highest known for the session, and the count will be 0. This packet will repeat multiple times at some small interval. No payload data will be sent after the first EOS packet is sent.
GAP_FILL_REQUEST	4	Reserved for future use.
GAP_FILL_RESPONSE	5	Reserved for future use.

3 TCP Messages from Member to Exchange

The following messages are expected to be sent from the Member to the Exchange. The Exchange will never send any of these messages to you.

3.1 LogonRequest

This must be the first message a Member sends to the Exchange after establishing a TCP/IP connection. If this message is sent at any other time, the RAKE server will close the TCP socket.

Field Name	Offset	Size	Type	Notes
length	0	2	`Short`	Length of the message, exclusive of the length field itself.
messageType	2	1	`Byte`	'5'/0x35/53
session	3	8	`Long`	Set to the current trading session if recovering from an outage, or 0 if connecting for the first time.
senderComp	11	8	`Str(8)`	Assigned by the Exchange.
token	19	8	`Str(8)`	Assigned by the Exchange.
nextSequenceNumber	27	8	`Long`	1-based numbering. Use 0 to start at the end (skip currently stored messages). Use 1 to replay all the messages. Send the next sequence number you expect to receive from the Exchange (one greater than your last received sequence) to recover from a disconnection.

3.2 MemberHeartbeat

Send during low traffic. At least once a second.

Field Name	Offset	Size	Type	Notes
length	0	2	`Short`	Length of the message, exclusive of the length field itself.
messageType	2	1	`Byte`	'7'/0x37/55

3.3 TcpUnsequencedMessage

Send message from Member to higher-level protocol at Exchange.

Field Name	Offset	Size	Type	Notes
length	0	2	`Short`	Length of the message, exclusive of the length field itself.
messageType	2	1	`Byte`	'6'/0x36/54
payload	3	type	variable	Variable length payload containing the application message.

4 TCP Messages from Exchange to Member

The following messages will be sent from the Exchange to the Member. It is an error to send any of these messages to the Exchange and will result in the TCP session being closed.

4.1 Debug

A human-readable debug message. As this message usually precedes the server closing the TCP connection, delivery of this message is not guaranteed. The payload of this message contains the ASCII message. It is not null-terminated.

Field Name	Offset	Size	Type	Notes
length	0	2	Short	Length of the message, exclusive of the length field itself.
messageType	2	1	Byte	'0'/0x30/48
payload	3	type	variable	Variable length payload containing the application message.

4.2 EndOfSession

The Exchange trading session has ended. No more un-sequenced messages will be accepted from the Member. No sequenced messages will be sent by the RAKE server after this message.

Field Name	Offset	Size	Type	Notes
length	0	2	Short	Length of the message, exclusive of the length field itself.
messageType	2	1	Byte	'4'/0x34/52

4.3 LogonResponse

Response to a logon request. See responseCode field to determine failure or success.

Field Name	Offset	Size	Type	Notes
length	0	2	Short	Length of the message, exclusive of the length field itself.
messageType	2	1	Byte	'1'/0x31/49
session	3	8	Long	The currently active trading session. This changes daily and will match the trading session fields on other protocols.
nextSequenceNumber	11	8	Long	The sequence number of the next sequenced message the Member will receive
highestKnownSequenceNumber	19	8	Long	The highest sequence number the server knows at the time of logon. Members may use this as a proxy for when they have "caught up" when recovering a RAKE session. However, there is no guarantee that the highest sequence has changed while a Member is rewinding data. Use with caution.
responseCode	27	1	Enum	Enum LogonResponseCode. Success or failure code from a Logon Request. Additional LogonResponseCode values may be added without notice in the future and should not break Member applications.
numberStreamIDs	28	1	Byte	The total number of StreamIDs that will be used in this trading session.

Field Name	Offset	Size	Type	Notes
instance	29	4	Int	Informational only. A unique ID for this RAKE server instance. Will be different between primary and backup. If it changes upon successive connections to the same RAKE server intra-day, the server has been restarted

4.4 ServerHeartbeat

Heartbeats will be sent once a second by the server if no other traffic has been sent

Field Name	Offset	Size	Type	Notes
length	0	2	Short	Length of the message, exclusive of the length field itself.
messageType	2	1	Byte	'3'/0x33/51

4.5 TcpSequencedMessage

A sequenced message from the Exchange. Members should increment their message sequence count upon receipt of this message type.

Field Name	Offset	Size	Type	Notes
length	0	2	Short	Length of the message, exclusive of the length field itself.
messageType	2	1	Byte	'2'/0x32/50
streamId	3	1	Byte	The internal Exchange StreamID that created this message. see Session Recovery.
payload	4	type	variable	Variable length payload containing the application message.

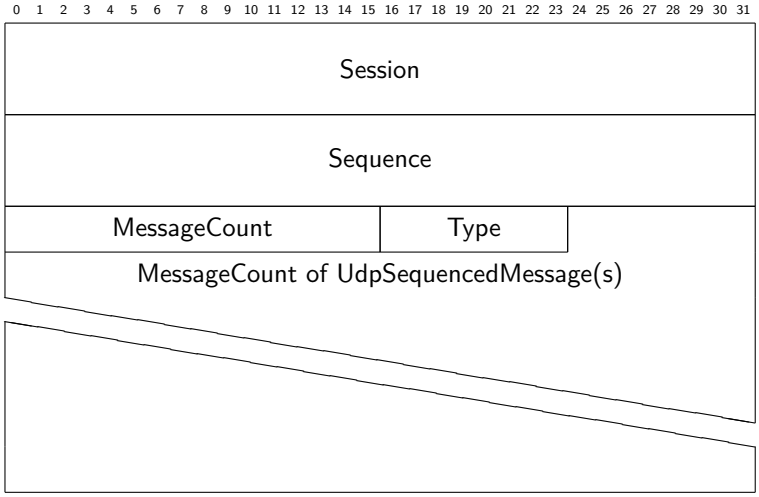
5 UDP Messages from Exchange to Member

The following describes the UDP framing for messages to the Member. The sequence of messages will be identical between RAKE TCP and RAKE UDP. Members consuming RAKE UDP may connect to RAKE TCP to request gap fill or rewind services by logging on with the appropriate sequence number. The requested messages will be delivered via the RAKE TCP session.

Each UDP packet contains the header as described below. Messages immediately follow the header. Each starts with a 16-bit little endian length. The length is exclusive of the length field itself. The number of messages in a packet is given in the header. A message will never span a packet boundary. Packets will never be larger than an Ethernet MTU.

5.1 UdpHeader

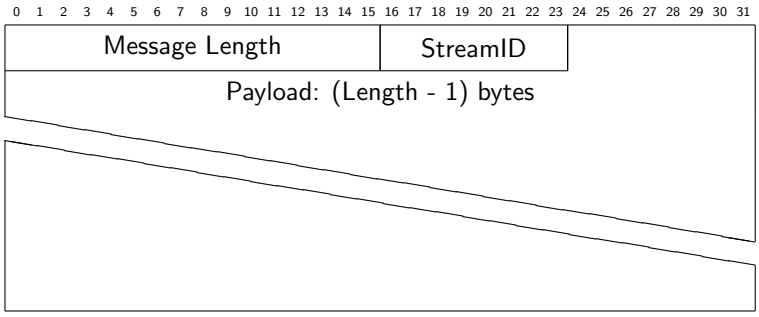
The following diagrams show the layout of a RAKE UDP packet. They do not include the IP/UDP header itself.



Field Name	Offset	Size	Type	Notes
session	0	8	`Long`	The currently active trading session. This changes daily and will match the trading session fields on other protocols.
sequence	8	8	`Long`	The message sequence number of the first message in the payload. Members should increment their message sequence based on the number of messages in the payload.
messageCount	16	2	`Short`	The number of messages in the payload.
type	18	1	`Enum`	Enum RakeUdpPacketType. Identifies the RAKE UDP packet variant. Informational only.

5.2 UdpSequencedMessage

Each individual message will have a length, a streamID, and an application message.



Field Name	Offset	Size	Type	Notes
length	0	2	`Short`	Length of the message, exclusive of the length field itself. note: inclusive of the `streamID` field.
streamId	2	1	`Byte`	The internal Exchange StreamID that created this message. see Session Recovery.
payload	3	type	variable	Variable length payload containing the application message.