IEX OPTIONS
BINARY SESSION PROTOCOL SPECIFICATION

Version 1.00

Updated: February 5th, 2026

OVERVIEW

IEX Options LLC ("IEX Options") is a wholly owned subsidiary of IEX Group, Inc. that operates an options exchange trading system (the "System"). The System is designed to offer an electronic limit order book for transacting listed options.

This IEX Options Binary Session Protocol document (Session Protocol) specifies the requirements for clients wishing to connect to IEX Options to conduct options trading operations. The business messages that clients use to conduct their operations and other business level requirements are contained in the IEX Binary Options Protocol Specification.

This document is offered for informational purposes at this time, and the contents are subject to change. As of February 5, 2026, IEX Options LLC is not yet an operating trading venue. Some of the functionality described may not have received regulatory approval.

DEFINITIONS

Client: A user that conducts options trading on IEX Options. The *client* initiates a TCP/IP connection request to a *gateway*. It uses information obtained from a provisioning process that is defined between the client firm and IEX Options as part of onboarding to initiate the connection. After the connection request is accepted by the *gateway* and authenticated, a *client session* is established that is associated with the *client*.

Client Session / Session: The component within the *gateway* that represents the presence of a *client* on IEX Options.

Gateway / Client Gateway: The component of the IEX Options information processing facility that hosts *client session*s.

PRIMITIVE DATA TYPES

The following primitive data types are used throughout this document. All integer types are little-endian.

Type	Min	Max	Size (bytes)
int8	-128	127	1
uint8	0	255	1
uint16	0	65'535	2
uint32	0	4'294'967'295	4
uint64	0	18'446'744'073'709'551'615	8
bool	false	true	1

The STRING(n) data type is a fixed size array of characters right padded with the NULL character, 0x00. The size of the array is specified by the value within the parenthesis.

Type	Backing primitive	Size (bytes)
STRING(n)	Array of char of length n.	n

The Timestamp data type is a 64-bit unsigned little-endian integer that represents Unix Time in epoch nanoseconds..

Type	Backing primitive	Size (bytes)
Timestamp	uint64	8

SBE includes a header for each message. The SBE header is followed by the SBE body for the message.

Field	Offset	Length	Type	Notes
Block Length	0	2	uint16	The number of bytes in the message body (does not include the header bytes)
Template ID	2	2	uint16	Identifier of the message template (i.e., the message type)
Schema ID	4	2	uint16	The identifier of a message schema.
Version	6	2	uint16	The version number of the message schema that was used to encode the message.

TRANSPORT PROTOCOL

TCP/IP using IPv4 is the transport protocol used to communicate between the *client* and the *gateway*.

Each *client* has an assigned destination IP address and port and source IP address. This combination uniquely identifies the *client* for purposes of establishing a TCP/IP connection with the *gateway*.

The *gateway* listens for TCP connection requests during IEX Options' operating hours between System Open time and System Closed time.

FRAMING PROTOCOL

The *client* and *gateway* communicate by exchanging a series of framed packets.

Each framed packet has:

A. A two-byte, little-endian PacketLength that indicates the length of rest of the packet.
B. A payload of SBE-encoded messages starting with an SBE message header (blockLength, templateId, schemaId, and version).

Field	Offset	Length	Type	Notes
PacketLength	0	2	uint16	Number of bytes after this field until the next packet
Payload	2	variable	uint8	Payload of the packet

Framed packets received with an invalid SBE message header or templateId, an excessively large packet length, or a length inappropriate for the packet type as defined by the templateId are rejected by closing the connection.

SESSION LEVEL PROTOCOL

Session Login

A *client session* begins when a *client* connects to the *gateway*, sends a **Login Request**, and receives a **Login Response** with **Success** status.

The *gateway* responds to the **Login Request** by sending a **Login Response** with **Success** status if the request contains the provisioned LogonId and Token, as defined in the Session Protocol Messages section below. Otherwise, the login is rejected, a protocol error is registered, and the *gateway* sends a **Login Response** packet with **Invalid LogonId or Invalid Token** status.

The *gateway* registers a protocol error and closes the connection immediately if the *client* sends any packet type other than a **Login Request** packet before the *gateway* accepts the login.

The *gateway* registers a protocol error and closes the connection if the *client* fails to send a **Login Request** packet within 30 seconds of the *gateway* accepting the client's TCP connection.

Once the **Login Request** has been accepted, the *client* and *gateway* may exchange session protocol message packets.

Protocol Errors

Protocol errors occur when the Session Level Protocol requirements outlined in this section are violated. A running count of protocol errors is kept per session. If the count exceeds 100 the Denial-of-Service mode is entered. When this happens, the *session* will:

- Reset the protocol error counter,
- Cancel orders based on customer 'Cancel on Disconnect' settings for the *session*,
- If the *client* has successfully logged in, send a **Terminate** with Reason "Protocol Error",
- Disconnect the *session,* and
- Refuse connection attempts for 60 seconds.

Violation	Action
Connection request from source IP address not matching the assigned destination address and port	Immediately close connection
Framed packets received with an invalid templateId, an excessively large packet length, or a packet length inappropriate for the packet type as defined by the templateId	Immediately close connection
Any packet type other than **Login Request** immediately after connecting	Immediately close connection
Failure to send **Login Request** within 30 seconds of connecting	Immediately close connection
Invalid logonId	**LoginResponse**; Invalid logonId
Invalid token	**LoginResponse**; Invalid token
Writing any packet to a client-to-gateway sub-session that has not been joined	**SubsessionLeave**; Subsession not joined
Writing any packet to a gateway-to-client or reference-data sub-session	**SubsessionLeaveResponse**; Invalid message
Writing business message that is not appropriate for the provisioned session type	**SubsessionLeaveResponse**; Invalid message
Failure to send a **ClientHeartbeat** or **SequencedMessage** within 5 seconds of the	**Terminate**; Heartbeat timeout. Immediately close connection

gateway sending a **GatewayHeartbeat** with KeepAlive true	
Out of sequence **SequencedMessage**	**SubsessionLeaveResponse**; Message out of sequence
Incorrect SubsessionId in **SequencedMessage**	**SubsessionLeaveResponse**; Invalid SubsessionId
Joining the client-to-gateway subsession with the same SubsessionId on the backup *session* while joined on the primary *session* and vice versa	**SubsessionLeaveResponse**; **JoinRevoked** on the session with the existing client-to-gateway subsession join
Client closes connection without having sent a **LogoutRequest**	No response possible but counts as a protocol error

Sub-sessions

Once a *session* is established the *client* may elect to participate in zero or more sub-sessions. There are three types of sub-sessions:

- Client-to-gateway: write only, used to send trading business messages to the *gateway*
- Gateway-to-client: read only, used to receive trading business messages, except reference data packets, from the *gateway*
- Reference-data: read-only, used to receive reference data messages from the *gateway*

The *gateway* advertises the availability of sub-sessions by sending a **Gateway Heartbeat** packet once per second after a successful Login Response. Each SubsessionInfo in the Gateway Heartbeat includes the subsession type, a subsessionId, whether the client is currently joined and the next sequence number. The *gateway* may randomly delay sending the initial **Gateway Heartbeat** packet by up to one second after sending the **Login Response** packet.

To participate in a sub-session the *client* sends a **Subsession Join** message specifying a subsessionId advertised in a Gateway Heartbeat message. The *gateway* responds by sending a **Subsession Join Response** packet with **Success** status if the request contains an available sub-session. Otherwise, the join is rejected, and the *gateway* sends a **Subsession Join Response** packet with status containing the reason for the rejection.

The *client* may leave a sub-session by sending a **Subsession Leave** message. The *gateway* responds by sending a **Subsession Leave Response** message with appropriate status and closes the sub-session.

Client-to-gateway sub-session

To send trading business messages to the *gateway,* the *client* must join the client-to-gateway sub-session. The *client* must specify the SubsessionId for the client-to-gateway sub-session from the **Gateway Heartbeat** message. The StartSequence should be the NextSeqNo also from the **Gateway Heartbeat** message. This will be the sequence number of the first **Sequenced Message** that the *client* sends.

The *gateway* sends an unsolicited **Subsession Leave Response** with status **JoinRevoked** and closes the sub-session if the *client* joins the client-to-gateway sub-session with the same SubsessionId on the backup *session* while joined on the primary *session* and vice versa.

Gateway-to-client sub-session

A *client* joins the gateway-to-client sub-session to receive trading business messages, typically responses to messages sent on the client-to-gateway sub-session, and unsolicited business messages such as execution reports. These messages are sequenced messages. The gateway-to-client sub-session may be used as a binary “drop copy” session.

The *client* specifies the range of sequenced messages to receive using the StartSequence and EndSequence fields of the **Subsession Join** message. The *gateway* sends an unsolicited **Subsession Leave Response** message and closes the sub-session when the requested range is satisfied. The *client* may use a very large EndSequence, for example 2^32 – 1, to receive messages until explicitly leaving the sub-session or until the *gateway* closes it at the end of the trading day.

Reference-data sub-session

A *client* joins the reference-data sub-session to receive reference data messages containing information identifying symbols, options series, and other reference data. As with the gateway-to-client sub-session, the *client* specifies the range of sequenced messages to receive using the StartSequence and EndSequence fields of the **Subsession Join** message. The *gateway* sends an unsolicited **Subsession Leave Response** message and closes the sub-session when the requested range is satisfied. The *client* may use a very large EndSequence, for example 2^32 – 1, to receive reference data messages until explicitly leaving the sub-session or until the *gateway* closes it at the end of the trading day.

Heartbeats

Once a *session* is established the *client* must send a **Client Heartbeat** packet once per five seconds if no other **SequencedMessage** messages have been sent.

If there has been no such *client* messaging activity on the *session* for five seconds, the gateway will warn the *client* by setting the KeepAlive field in the next **Gateway Heartbeat** packet. If there is no *client* messaging activity on the *session* for one additional five second period, the *gateway* sends a **Terminate** message with **HeartbeatTimeout** status, registers a protocol error, then closes the connection.

Session Termination

Either the *client* or the *gateway* may terminate the session.

The *client* terminates the session by sending a **Logout Request** message. The *gateway* generally responds by sending a **Terminate** message with the **Success** status then closes the connection.

The *gateway* may unilaterally terminate a session for various reasons. In this event, the *gateway* sends an unsolicited **Terminate** message with the termination reason then closes the connection. The *client* should close their connection immediately after having received the **Terminate** message.

After successful login, the *gateway* will not unilaterally close the session's TCP/IP connection without sending a **Terminate** message under typical circumstances. The *client* is discouraged from closing their end of TCP/IP connection before the *gateway* closes the connection. Doing so counts as a protocol error.

Sequenced Message Packets

Business level messages are exchanged using **Sequenced Message** messages.

Either the *client* or the *gateway* may send a **Sequenced Message** to convey a business message. The SequenceNumber in the first **Sequenced Message** packet the *client* sends after joining the client-to-gateway sub-session must be the value from the NextSeqNo field for the sub-session in the previous **Gateway Heartbeat** message.

Subsequent **Sequenced Message** must carry a SequenceNumber that is incremented by 1 from the previous packet.

If the *client* leaves the sub-session and later rejoins it, the NextSeqNo field in the SubsessionInfo field of the **Gateway Heartbeat** message will contain a SequenceNumber that is one more than the last **Sequenced Message** that the *client* sent before leaving the sub-session.

If the *client* sends an out-of-sequence **Sequenced Message**, the *gateway* registers a protocol error and closes the sub-session by sending an unsolicited **Subsession Leave Response** message with an appropriate Reason.

Session Protocol Types

These session protocol-specific types, in addition to the Primitive Data Types above, are used in session protocol message definitions.

Status

The Status enumeration (int8) conveys the result of some request.

Value	Meaning	Notes
0	Success	
1	SessionLevelReject	
2	LogoutRequested	
3	Unknown	
4	InvalidLogonId	
5	InvalidToken	
6	AlreadyLoggedIn	
7	LoginTimeout	No **LoginRequest** packet received within 30 secs of connecting
8	InvalidPacketLength	
9	InvalidMessage	
10	JoinRevoked	Client-to-gateway **SubsessionJoin** on another session in same group
11	HeartbeatTimeout	
12	MessageOutOfSequence	
13	InvalidSubsessionId	
14	SubsessionNotJoined	
15	DenialOfService	
16	InvalidArgument	
17	AlreadyJoined	

Subsession

The Subsession enumeration (int8) names sub-sessions.

Value	Meaning
0	ClientToGateway
1	GatewayToClient
2	ReferenceData

SubsessionId

SubsessionId provides a unique identifier for sub-sessions within a Trading Environment. **SubsessionId** has a total order such that a sub-session created before another sub-session on the same trading session is ordered before the newer sub-session. This property allows the client to select the latest sub-session when there are more than one with the same **Subsession** value.

Type	Size	Notes
uint64	8	Unique identifier for the sub-session

SubsessionInfo Repeating Group

The **SubsessionInfo** group provides information on the sub-sessions that the *client* may join.

Name	Type	Offset	Length	Notes
Group: subSessions (dimensions: blockLength=14, numInGroup=uint8, numInGroupType=uint8, min=0, max=20)				
subsessionType	Subsession	0	1	The sub-session type
subsessionId	SubsessionId	1	8	Uniquely identifies the sub-session
joined	bool	9	1	Whether client has joined sub-session
nextSeqNo	SequenceNumber	10	4	Sequence number starting with 1

Session Protocol Messages

Login Request

The *client* sends a **Login Request** (client-to-gateway) packet immediately upon establishing a new TCP/IP connection with the *gateway*. *Client* and *gateway* must have mutually agreed upon the LogonId and Token fields. These provide simple authentication to prevent a *client* from inadvertently connecting to the wrong *gateway*.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 56
SBE header	2	8		*SchemaId: 20000, TemplateId: 1, BlockLength: 48, Version: 0*
LogonId	10	16	STRING(16)	
Token	26	32	STRING(32)	

Login Response

The *gateway* sends a **Login Response** (gateway-to-client) message in response to having received a **Login Request** message. This message will always be the first message sent by the *gateway* after receiving a **Login Request** whether the request is accepted or not.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 25
SBE header	2	8		*SchemaId: 20000, TemplateId: 2, BlockLength: 17, Version: 0*
LogonId	10	16	STRING(16)	
Status	26	1	Status	

Gateway Heartbeat

The *gateway* starts sending **Gateway Heartbeat** (gateway-to-client) messages after sending a **Login Response** with **Success** status and once per second thereafter.

The **Gateway Heartbeat** message serves two purposes. It advertises the availability of sub-sessions that the *client* may join and acts as a heartbeat for those sub-sessions.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 9 + 2 (blockLength + numInGroupType) + count * 14

SBE header	2	8		*SchemaId: 20000, TemplateId: 3, BlockLength: 1, Version: 0*
keepAlive	10	1	bool	
Group: SubsessionInfo (dimensions: blockLength=uint8 (14), numInGroupType=uint8 (0-20))				
subsessionInfo		14	SubsessionInfo	

Client Heartbeat

The *client* must send a **Client Heartbeat** (client-to-gateway) message once every five seconds if no other **Client Heartbeat** or **Sequenced Message** messages have `been sent.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 8
SBE header	2	8		*SchemaId: 20000, TemplateId: 4, BlockLength: 0, Version: 0*

Subsession Join

After login a *client* may send a **Subsession Join** (client-to-gateway) message to begin participating in a sub-session. The SubsessionId names the sub-session to join and can be obtained from the **Gateway Heartbeat** message that references the desired sub-session.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 24
SBE header	2	8		*SchemaId: 20000, TemplateId: 8, BlockLength: 16, Version: 0*
SubsessionId	10	8	SubsessionId	
StartSequence	18	4	SequenceNumber	Start sequence number; must be greater than or equal to 1
EndSequence	22	4	SequenceNumber	End sequence number for read sub-sessions

Subsession Join Response

Sent by the gateway in response to a **Subsession Join** (client-to-gateway) message.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 17
SBE header	2	8		*SchemaId: 20000, TemplateId: 9, BlockLength: 9, Version: 0*
SubsessionId	10	8	SubsessionId	
Status	18	1	Status	

Subsession Leave

The **Subsession Leave** (client-to-gateway) message is sent by the *client* to leave a previously joined sub-session.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 16
SBE header	2	8		*SchemaId: 20000, TemplateId: 10, BlockLength:8, Version: 0*
SubsessionId	10	8	SubsessionId	

Subsession Leave Response

Sent by the *gateway* in response to a **Subsession Leave** (client-to-gateway) message and to various sub-session protocol errors such as an out-of-sequence **Sequenced Message** message.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 17
SBE header	2	8		*SchemaId: 20000, TemplateId: 11, BlockLength: 9, Version: 0*
SubsessionId	10	8	SubsessionId	
Reason	18	1	Status	

Logout Request

The *client* may send a **Logout Request** (client-to-gateway) message to request the connection be terminated. Upon receiving a **Logout Request** message, the *gateway* sends a **Terminate** message and closes the connection.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 8
SBE header	2	8		*SchemaId: 20000, TemplateId: 5, BlockLength: 0, Version: 0*

Terminate

The *gateway* sends a **Terminate** (gateway-to-client) message to inform the *client* that the session is terminated and then closes the connection.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 9
SBE header	2	8		*SchemaId: 20000, TemplateId: 6, BlockLength: 1, Version: 0*
Reason	10	1	Status	

Sequenced Message

Either the client or the gateway may send a **Sequenced Message** (gateway-to-client or client-to-gateway) message to convey a business message. The SequenceNumber must be incremented by one for each message sent.

Field	Offset	Length	Type	Notes
Packet length	0	2	uint16	Length is 28 + Payload length
SBE header	2	8		*SchemaId: 20000, TemplateId: 7, BlockLength: 20, Version: 0*
SubsessionId	10	8	uint64	Subsession ID of contained message
Sequence	18	4	uint32	
Timestamp	22	8	Timestamp	UNIX Epoch in nanoseconds
Payload		variable	uint8	Defined by the business message being sent

REVISION HISTORY

Version	Date	Change
1.00	February 5, 2026	Initial publication of document.