INVESTORS EXCHANGE TRANSPORT SPECIFICATION

Version 1.25

Updated: May 12, 2017

Table of Contents

OVERVIEW

IEX's Transport Protocol (IEX-TP) is a transport-layer protocol that allows the transmission of data between systems.

IEX-TP is available via UDP multicast for efficient and scalable transmission of data messages in a "one-to-many" scenario. IEX-TP does not support TCP for transmission (i.e., initial publication).

IEX-TP provides a mechanism for listeners to detect and re-request missed segments via TCP or UDP unicast.

UDP MULTICAST PUBLICATION

When using IEX-TP via UDP multicast, each outbound segment is transmitted only once per sender regardless of the number of listeners. Multiple messages may also be aggregated into a single segment to reduce network traffic.

This document describes the messages sent between an IEX-TP server and its clients. IEX-TP senders transmit segments via UDP multicast to carry the data stream sent to the listeners, and the length of a segment will never exceed the length of the UDP data contents of a packet.

IEX-TP servers will transmit on pre-assigned multicast groups for each messaging protocol on a network via A/B lines and A/B groups. Receivers must subscribe to either or both of the A/B multicast groups to receive data. A/B multicast groups are guaranteed to publish payload messages in the same exact order, however receivers should not assume that the A/B multicast groups will aggregate messages into identical outbound segments.

UNICAST GAP FILL

Depending on the higher-layer protocol requirements, zero or more Gap Fill Servers may be deployed to service any unicast client requests for retransmission of data. When deployed, any data from the start of the current session may be requested for retransmission from the Gap Fill Servers via TCP or UDP unicast. Data from previous sessions may not be requested. See the higher-layer protocol specification for supported gap fill protocols.

The listeners (i.e., receivers) must be configured with specific IP address and port combinations to which they can submit requests for retransmission of data ("Gap Fill Requests").

UDP Gap Fill

To request a gap fill via UDP, the receiver must unicast a Gap Fill Request, as defined in this specification, to the IP and port of a Gap Fill Server specified in the higher-layer protocol.

A receiver may request a limited retransmission via UDP per request. The size of retransmission response via UDP is configured by the higher-layer protocol and identified in its specification ("Maximum UDP Retransmission Response").

The response to a valid UDP Gap Fill Request is one or more standard Outbound Segments unicast back to the source address of the Gap Fill Request. This allows, for example, downstream IEX-TP users to read retransmitted Outbound Segments in their multicast processing socket if the request was made from that port (in other words, the client need

only have one socket open to listen to the multicast publication and to process retransmissions, even though the retransmissions are not multicast).

UDP Gap Fill Test Request

The receiver may send a UDP Gap Fill Request with no request ranges specified to test the availability of a Gap Fill Server ("UDP Gap Fill Test Request"). The response to a valid UDP Gap Fill Test Request is one Outbound Segment with no Payload unicast back to the source address of the UDP Gap Fill Test Request ("Gap Fill Test Response").

TCP Gap Fill

To request a gap fill via TCP, the receiver must first open a TCP/IP connection to the IP and port of a Gap Fill Server specified in the higher-layer protocol. Then, the receiver must send a Gap Fill Request, as defined in this specification, using the existing socket connection. Multiple Gap Fill Requests may be sent on an existing socket connection. Receivers should only establish a connection to a Gap Fill Server if they wish to submit a request. If the receiver does not submit a request immediately upon establishing a connection, their session will be terminated by the Gap Fill Server.

A receiver may request an unlimited retransmission via TCP (in other words, the client may request retransmission of all messages from the start of a given session when using TCP for gap fills). For example, the last message sequence number in a requested range may be larger than the latest sequence number received via multicast to account for new message publication during retransmission.

The response to a valid TCP Gap Fill Request is one or more standard Outbound Segments sent via the existing socket connection. The connection continues until all requested gaps are filled or when retransmission has caught up with the most current publication. Once either condition is met, the Gap Fill Server will close the TCP/IP socket connection.

TERMS

Message

A message is an atomic piece of application information carried by an IEX-TP segment. The contents of a message are defined by the message protocol in use.

Segment

A segment is a collection of messages of a specified messaging protocol, framed in such a way as to support reliable delivery. For UDP a segment is analogous to a packet. For TCP retransmission, a segment may be up to 64KB in length.

Session

A session is a sequence of one or more messages in a given protocol delineated by a unique Session ID. Once a session is terminated, no more messages can be sent or retransmitted on that session.

DATA TYPES

- Long: 8 bytes, signed integer
- Integer: 4 bytes, unsigned integer
- Short: 2 bytes, unsigned integer
- Byte: 1 byte unsigned integer
- Timestamp: 8 bytes, signed integer containing a counter of nanoseconds since POSIX (Epoch) time UTC

All fields are in little endian format.

Note that each byte is represented by two hexadecimal digits in the examples within this specification.

OUTBOUND SEGMENT FORMAT

Outbound Segments are sent by sources to listeners. Each Outbound Segment contains zero or more messages sent to listeners, described by the following IEX-TP Header.

Each Outbound Segment consists of a header and a payload that carries the actual data stream represented as a series of Message Blocks.

IEX-TP Header

Field Name	Offset	Length	Type	Description/Notes
Version	0	1	Byte	1 (0x1) – Version of Transport specification
(Reserved)	1	1		Reserved byte
Message Protocol ID	2	2	Short	Unique identifier of the higher-layer protocol
Channel ID	4	4	Integer	Identifies the stream of bytes/sequenced messages
Session ID	8	4	Integer	Identifies the session
Payload Length	12	2	Short	Byte length of the payload
Message Count	14	2	Short	Number of messages in the payload
Stream Offset	16	8	Long	Byte offset of the data stream
First Message Sequence Number	24	8	Long	Sequence of the first message in the segment
Send Time	32	8	Timestamp	Send time of segment

Total IEX-TP Header length is 40 bytes. See Appendix A for the bitwise representation.

Version

Version of the IEX-TP protocol.

Message Protocol ID

A unique identifier for the higher-layer specification that describes the messages contained within a segment. See the higher-layer protocol specification for the protocol's message identification on IEX-TP.

Channel ID

An identifier for a given stream of bytes/sequenced messages. Messages received from multiple sources which use the same Channel ID are guaranteed to be duplicates by sequence number and/or offset. See the higher-layer protocol specification for the protocol's channel identification on IEX-TP.

Session ID

Session ID uniquely identifies a stream of messages produced by the system. A given message is uniquely identified within a message protocol by its Session ID and Sequence Number.

Stream Offset & Payload Length

Stream Offset is a counter representing the byte offset of the payload in the data stream.

Payload Length is an unsigned binary count representing the number of bytes contained in the segment's payload. Note that the Payload Length field value does not include the length of the IEX-TP Header.

First Message Sequence Number & Message Count

First Message Sequence Number is a counter representing the sequence number of the first message in the segment. If there is more than one message in a segment, all subsequent messages are implicitly numbered sequentially.

Message Count is a count representing the number of Message Blocks in the segment.

Send Time

The time the Outbound Segment was sent as set by the sender.

Message Block

The first field of a Message Block is the two-byte Message Length field. The remainder of the Message Block is the variable length Message Data. The first Message Block field, Message Length, will begin after the last byte of the IEX-TP Header. Subsequent Message Blocks will begin after the last byte of the previous Message Block.

Field Name	Offset	Length	Type	Description/Notes
Message Length	0	2	Short	Length of the message
Message Data	2	Variable		The message data

Message Length

The Message Length is an unsigned binary count representing the number of bytes in a message following the Message Length field. The Message Length field value does not include the two bytes occupied by the Message Length field. The total size of the Message Block is the value of the Message Length field plus two.

Message Data

The Message Data is the actual data of the message being transmitted by Transport. It is variable length and can be zero length. The meaning of the data is application specific.

Heartbeats

Heartbeats are sent by the server if no new messages have been sent in the preceding one second for the given protocol. A Heartbeat will contain the next byte within the stream for the Stream Offset, the sequence number of the next message for First Message Sequence Number, and 0 (zero) for both Payload Length and Message Count.

Gap Fill Test Response

Gap Fill Test Responses are sent by the server in response to a valid UDP Gap Fill Test Request. A Gap Fill Test Response contains the next byte within the stream for the Stream Offset, the sequence number of the next message for First Message Sequence Number, and 0 (zero) for both Payload Length and Message Count. Said differently, a Gap Fill Test Response resembles a Heartbeat unicast back to the source address of the UDP Gap Fill Test Request sender.

GAP FILL REQUEST

The Gap Fill Request is sent to request the retransmission of a particular message, a group of messages, or a portion of the bytestream. The Gap Fill Request is sent to a Gap Fill Server by a receiver. A receiver may need to send this request when it detects a sequence number or stream offset gap in the received segment. A receiver may send one Gap Fill Request with multiple Request Range Blocks specified.

Each Gap Fill Server only responds to Gap Fill Requests of one Request Type. The implementation of higher-layer protocols using IEX-TP may not support all Request Types. See the higher-layer protocol specification for supported retransmission Request Type(s) and other Gap Fill Server information. All Request Range Blocks in a Gap Fill Request must use the same Request Type, not overlap, and monotonically increase. These same restrictions apply over multiple TCP Gap Fill Requests sent using the same TCP/IP socket connection. Invalid Gap Fill Requests are ignored by Gap Fill Servers and, in the case of a request sent via TCP, the session will be terminated by the Gap Fill Server.

Gap Fill Request Header

Field Name	Offset	Length	Type	Description/Notes
Version	0	1	Byte	1 – Version of Transport specification
Request Type	1	1	Byte	1 – Sequenced Messages 2 – Bytestream
Message Protocol ID	2	2	Short	Unique identifier of the higher-layer protocol
Channel ID	4	4	Integer	Identifies the stream of bytes/sequenced messages
Session ID	8	4	Integer	Identifies the session
Request Range Count	12	4	Integer	Number of ranges requested

Total Gap Fill Request Header length is 16 bytes. See Appendix A for the bitwise representation.

Version

Version of IEX-TP protocol in use.

Request Type

Identifies how the Request Range Block of this request is specified: either Sequenced Messages or Bytestream. The Request Range Block contents must match the Request Type identified.

Message Protocol ID

Indicates the Message Protocol ID to which this request belongs. See the higher-layer protocol specification for the protocol's message identification on IEX-TP ("Message Protocol ID").

Channel ID

Indicates the Channel ID to which this request belongs. See the higher-layer protocol specification for the protocol's channel identification on IEX-TP ("Channel ID").

Session ID

Indicates the Session ID to which this request belongs. Typically, when initially requesting a gap fill, a receiver will inspect the first packet via UDP multicast to determine the current active session, and apply such Session ID to the Gap Fill Request Header. This process will prevent a receiver from accidentally requesting data from a previous session.

Request Range Count

Request Range Count is a count representing the number of Request Range Blocks in this request. When sending a UDP Gap Fill Test Request, the Request Range Count should be zero (0).

Request Range Block (Sequenced Messages)

Field Name	Offset	Length	Type	Description/Notes
First Message Sequence Number	0	8	Long	Sequence of the first message requested
Last Message Sequence Number	8	8	Long	Sequence of the last message requested

Total Gap Fill Request Block length is 16 bytes. See Appendix A for the bitwise representation.

First Message Sequence Number & Last Message Sequence Number

First Message Sequence Number is a counter representing the sequence number of the first message in the segment requested. The Last Message Sequence Number is a counter representing the sequence number of the last message in the segment requested. The request range is inclusive of the First Message Sequence Number and the Last Message Sequence Number.

Request Range Block (Bytestream)

Field Name	Offset	Length	Type	Description/Notes
Starting Stream Offset	0	8	Long	Offset of the first byte requested
Ending Stream Offset	8	8	Long	Offset of the last byte requested

Total Gap Fill Request Block length is 16 bytes. See Appendix A for the bitwise representation.

Starting Stream Offset & Ending Stream Offset

Starting Stream Offset is a counter representing the byte offset of the start of the payload in the data stream requested. Ending Stream Offset is a counter representing the byte offset of the end of the payload in the data stream requested. The request range is inclusive of the Starting Stream Offset and the Ending Stream Offset.

EXAMPLE MESSAGES

Note that in the following examples, each byte is represented by two hexadecimal digits and the messages are from DEEP v1.0.

Multicast Publication

IEX-TP Header

Version	01	// Version: 1
(Reserved)	00	// (Reserved)
Message Protocol ID	04 80	// DEEP v1.0
Channel ID	01 00 00 00	// Channel: 1
Session ID	00 00 87 42	// Today's current Session ID
Payload Length	48 00	// 72 bytes
Message Count	02 00	// 2 messages
Stream Offset	8c a6 21 00 00 00 00 00	// 2,205,324 bytes
First Sequence Number	ca c3 00 00 00 00 00 00	// Sequence: 50122
Send Time	ec 45 c2 20 96 86 6d 14	// 2016-08-23 15:30:32.572839404

Segment Payload

Message Block 1

Message Length	26 00	// 38 bytes
Message Type	54	// T = Trade Report
Sale Condition Flags	00	// Non-ISO, Regular Market Session, Round or Mixed Lot, Trade is subject to Rule 611, execution during continuous trading
Timestamp	ac 63 c0 20 96 86 6d 14	// 2016-08-23 15:30:32.572715948
Symbol	5a 49 45 58 54 20 20 20	// ZIEXT
Size	64 00 00 00	// 100 shares
Price	24 1d 0f 00 00 00 00 00	// $99.05
Trade ID	96 8f 06 00 00 00 00 00	// 429974

Message Block 2

```
Message Length          1e 00                   // 30 bytes
Message Type            38                      // 8 = Price Level Update on the Buy Side
Event Flags             01                      // Event processing complete
Timestamp               ac 63 c0 20 96 86 6d 14 // 2016-08-23 15:30:32.572715948
Symbol                  5a 49 45 58 54 20 20 20 // ZIEXT
Size                    e4 25 00 00             // 9,700 shares
Price                   24 1d 0f 00 00 00 00 00 // $99.05
```

Gap Fill Request

Gap Fill Request Header

```
Version                 01                      // Version: 1
Request Type            01                      // 1 = Sequenced Messages
Message Protocol ID     04 80                   // DEEP v1.0
Channel ID              01 00 00 00             // Channel: 1
Session ID              00 00 87 42             // Today's current Session ID
Request Range Count     02 00 00 00             // 2 request range blocks
```

Request Range Block 1 (Sequenced Messages)

```
First Message Sequence Number   34 30 00 00 00 00 00 00   // Sequence: 12340
Last Message Sequence Number    3e 30 00 00 00 00 00 00   // Sequence: 12350
```

Request Range Block 2 (Sequenced Messages)

```
First Message Sequence Number   d5 dd 00 00 00 00 00 00   // Sequence: 56789
Last Message Sequence Number    df dd 00 00 00 00 00 00   // Sequence: 56799
```

UDP Gap Fill Test Request and Response

UDP Gap Fill Test Request

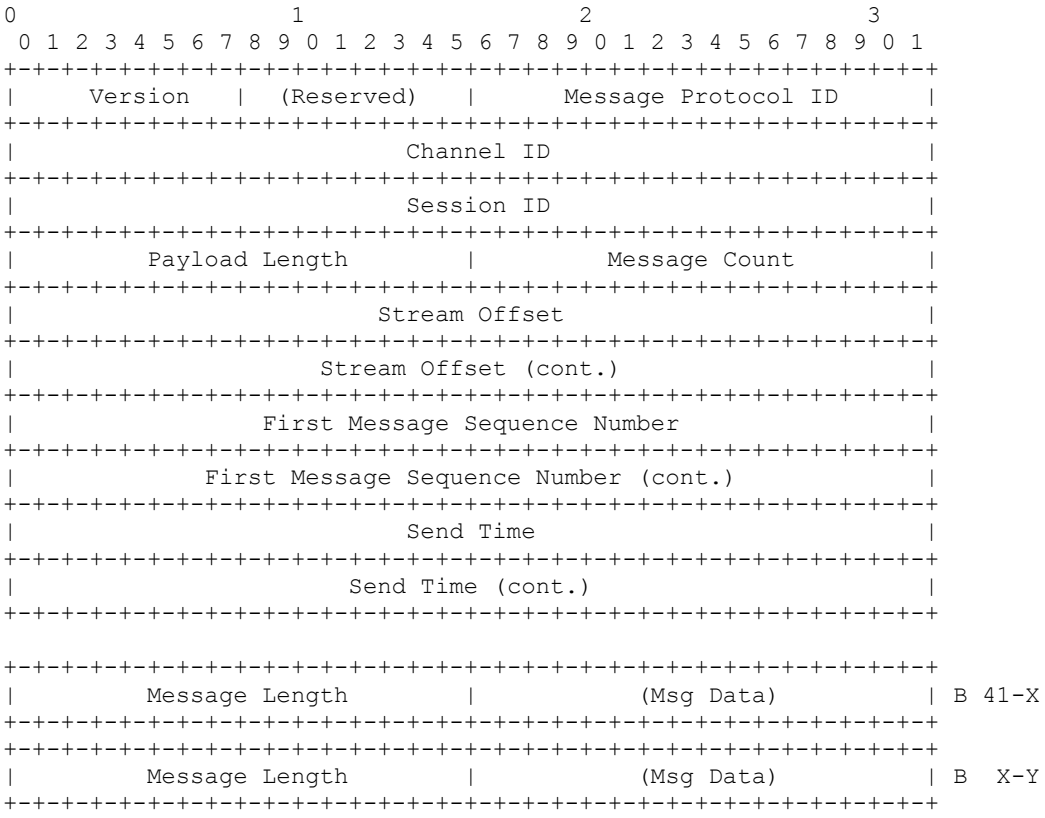
```
Version                 01                      // Version: 1
Request Type            01                      // 1 = Sequenced Messages
Message Protocol ID     04 80                   // DEEP v1.0
Channel ID              01 00 00 00             // Channel: 1
Session ID              00 00 87 42             // Today's current Session ID
Request Range Count     00 00 00 00             // Must have 0 request range blocks
```

Gap Fill Test Response

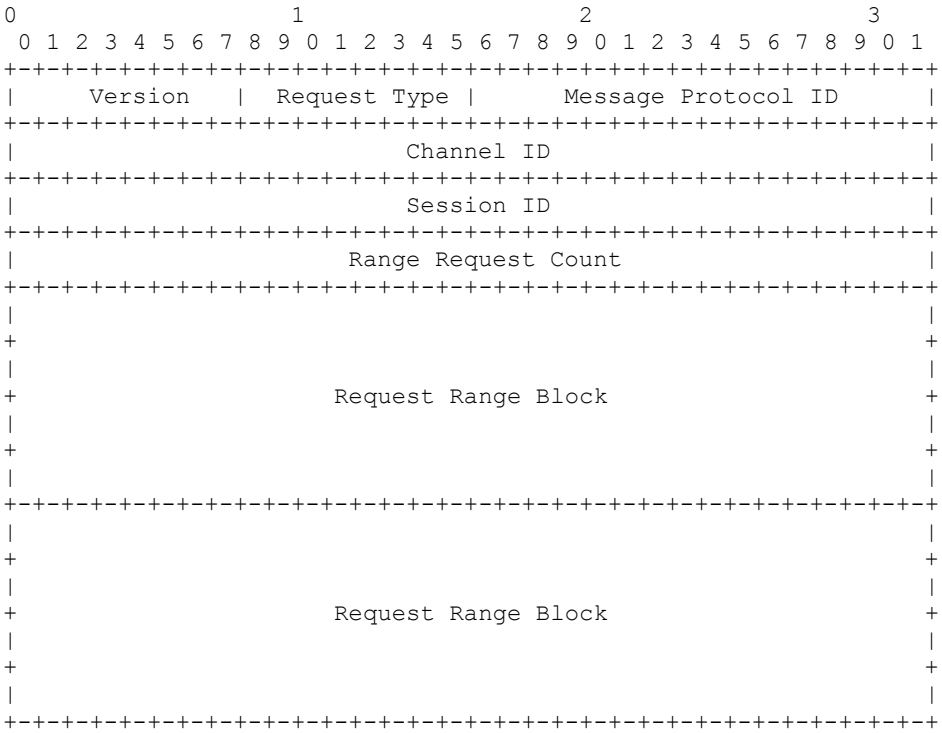
Version	01	// Version: 1
(Reserved)	00	// (Reserved)
Message Protocol ID	02 80	// TOPS v1.5
Channel ID	01 00 00 00	// Channel: 1
Session ID	00 00 87 42	// Today’s current Session ID
Payload Length	00 00	// 0 bytes
Message Count	00 00	// 0 messages
Stream Offset	8c a6 21 00 00 00 00 00	// 2,205,324 bytes
First Sequence Number	ca c3 00 00 00 00 00 00	// Sequence: 50122
Send Time	ec 45 c2 20 96 86 6d 14	// 2016-08-23 15:30:32.572839404

APPENDIX A: BITWISE REPRESENTATION

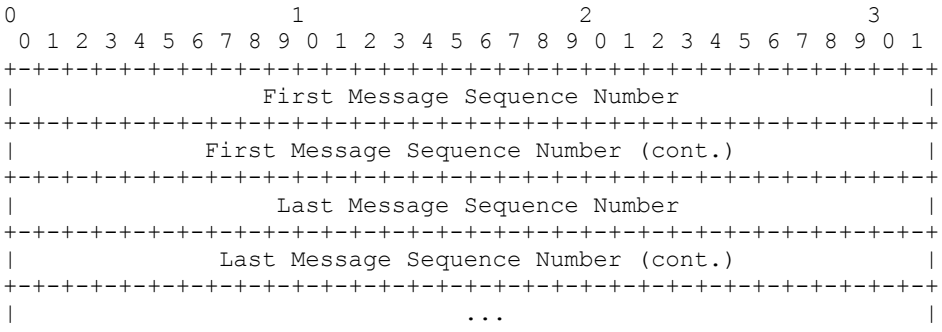
Segment with Two Message Blocks



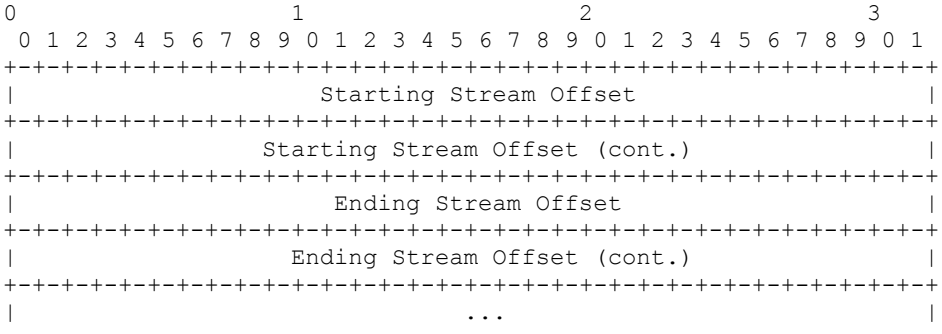
Gap Fill Request with Two Request Range Blocks



Request Range Block where Request Type is Sequenced Stream



Request Range Block where Request Type is Bytestream



REVISION HISTORY

Version	Date	Change
1.00	August 20, 2015	Initial document
1.10	February 23, 2016	Fixed typos
1.20	June 28, 2016	Added retransmission functionality
1.21	August 8, 2016	Clarified Request Header's Request Type field values
1.22	August 17, 2016	Provided details regarding usage of Version, Channel ID, and Session ID Clarified timeout for TCP retransmission after a session is established
1.23	August 23, 2016	Added Example Messages section
1.24	October 26, 2016	Added support to test Gap Fill Requests via UDP, even before any data is published on the multicast feed (UDP Gap Fill Test Request and Gap Fill Test Response)
1.25	May 12, 2017	Clarified expectations regarding outbound segment construction across A/B multicast groups Updated examples to reflect DEEP v1.0