Japannext JNX

MoldUDP64 Specification (All Markets)

Version 1.01

Last updated: January 16, 2026

1 Introduction

This document explains access to the **trading services** of **Japannext PTS** via the **MoldUDP64** protocol. It provides an overview of the protocol and describes the packet types.

For further information and inquiries regarding trading services, and for questions concerning connectivity, contact Japannext Technical Support at ito@japannext.co.jp.

2 Overview

MoldUDP64 is a lightweight networking protocol that enables scalable **one-to-many message transmission**. In addition, MoldUDP64 enables clients to **detect and re-request missed packets**.

2.1 Network Stack

MoldUDP64 serves as the point-to-point transport layer for higher-level protocols such as **ITCH**. In turn, MoldUDP64 uses **UDP** as its transport protocol (**Figure 1**).

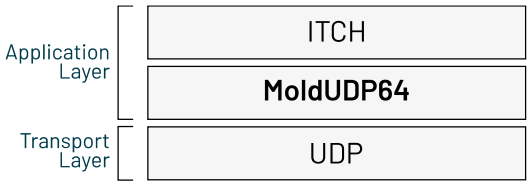

Figure 1 - MoldUDP64 stack

2.2 Core Features

Core design features of MoldUDP64 are as follows:

- Single transmission of outbound packets, regardless of the number of clients.
- Aggregation of multiple messages into a single network packet.
- Detection and re-requesting of missed packets.

2.3 Terminology

- **Transmitter**—MoldUDP64 server.
- **Listener**—MoldUDP64 client.
- **Message**—atomic piece of information transported by MoldUDP64.
  - Messages of up to 64 KB are supported, but size should be minimized to ensure efficient transmission.
  - Message content is defined by higher-level applications.
- **Session**—sequence of one or more logically grouped messages.
  - Sessions remain active until terminated.

2.4 Data Types

This specification uses the following data types:

Data type	Description
NUM	Unsigned integer [1]
ANUM	User-defined data (binary or string)

[1] All number fields in MoldUDP64 messages are **binary big-endian** (most significant byte first). This does not apply to data in the **Message Data** field of a **Message Block**.

2.5 Packet Types

MoldUDP64 handles the following packet types:

Packet type	Description
Downstream	Sent by transmitter to listener. Encapsulates messages.
Request	Sent by listener to re-request server. Used to request missed messages.

2.6 General Operation

2.6.1 UDP multicast

For data streaming, such as real-time ITCH market data dissemination, transmitters send data stream packets to listeners via **UDP multicast**. Each type of downstream MoldUDP64 data stream is configured for a specific multicast group to which listeners must subscribe.

2.6.2 UDP unicast

If a listener fails to receive a packet, the listener may request, via **UDP unicast**, retransmission of that packet, to which the transmitter responds also via **UDP unicast**. Note that this response is directed to the socket used for the retransmission request. This feature requires at least one re-request server to be deployed and listeners to be configured with the relevant sockets for request submission. In addition, if the listener is restarted, it requires the session and sequence number of the next expected message.

2.6.3 Receiver operation example

Typical MoldUDP64 listener operation is as follows:

1. Open a UDP socket and join the desired multicast group.
2. Examine the first received packet to determine current session. If the received session does not match the expected session, abort and report.
3. Examine the sequence number of the first received packet. If the sequence number does not match the next expected sequence number, send a **Request Packet** to the Request Server with expected packet number and wait for a new packet. Repeat step as necessary.
4. Process each message in the packet. If a **Downstream Packet** with Message Count set to End of Session is received, handle the End of Session event.
5. Wait for a new packet and return to step 3.

3 Downstream Packet

A MoldUDP64 transmitter sends **Downstream Packets** to MoldUDP64 listeners. Each MoldUDP64 packet contains the following:

- A **Header**
- One or more **Message Blocks**

Message Blocks carry the actual data stream messages. Each MoldUDP64 packet contains a payload of 0 or more whole messages.

3.1 Header

All MoldUDP64 packets start with a fixed-size 20-byte header.

Name	Offset	Length	Value	Notes
Session	0	10	ANUM	Session to which this packet belongs.
Sequence Number	10	8	NUM	Sequence number of first message in packet. Additional messages are implicitly numbered sequentially.
Message Count	18	2	NUM	Number of messages in packet. The maximum payload size of a Downstream Packet is determined by sender. Value = '0' denotes Heartbeat. Value = '0xFFFF' (decimal 65535) denotes End of Session.

3.2 Message Block

Each Message Block contains one message. The first Message Block field starts immediately after the Header (20 bytes from the beginning of the packet). Subsequent Message Blocks start immediately after the last byte of the previous Message Block.

Name	Offset	Length	Value	Notes
Message Length	*	2	NUM	Length in bytes of Message Data.
Message Data	*	*	ANUM	Application-specific message data. Can be zero length.

* = Variable value

3.3 Heartbeats

Transmitters periodically send Heartbeat packets to enable listeners to detect link failures. Heartbeat packets consist of a **Downstream Packet Header** specifying the next expected **Sequence Number** and **Message Count** = **0**.

Note: The transmitter sends a Heartbeat packet when more than **1 second** has elapsed since the last data transmission.

3.4 End of Session

Transmitters send End of Session packets in place of Heartbeat packets to indicate that the current session is ending. End of Session packets consist of a **Downstream Packet Header** specifying the next expected **Sequence Number** and **Message Count** = **0xFFFF** (decimal 65535).

A listener may make re-requests for packets in the current session as long as the End of Session packets are being transmitted.

Note: The sending interval for End of Session packets is **1 second**.

4 Request Packet

Listeners send a **Request Packet** to a re-request server to request **retransmission** of a **particular message** or **group of messages**. **Request Packets** enable listeners to obtain previously missed messages after detecting a gap in sequence numbers in received messages. The re-request server responds by returning a standard **Downstream Packet** via **UDP unicast**.

Field name	Offset	Length	Value	Notes
Session	0	10	ANUM	Session to which packet belongs.
Sequence Number	10	8	NUM	Sequence number of first requested message.
Requested Message Count	18	2	NUM	Number of messages requested for retransmission.

Caution: The total size of the requested messages must not exceed the maximum payload size of a UDP/IP packet. Otherwise, only the messages that fit in a single UDP/IP packet will be returned, and additional retransmission requests will need to be made.

Note: Unicast **Downstream Packet** responses are receivable via the listener's multicast socket, provided that the corresponding **Request Packet** is sent from the same socket. Accordingly, the listener requires only a single socket for both multicast and unicast reception.

Tip: Colocating caching re-request servers can reduce latency and bandwidth.

5 Revision History

Date	Version	Description
2026-01-16	1.01	Minor format update. No changes to content.
2025-09-08	1.00	Initial version.