System Wide Information Management (SWIM)

Timestamp and Message Sequence Number Policy

Version 1.0

September 29, 2025

SIGNATURE PAGE

SUZANNE N KOPPANEN

Digitally signed by SUZANNE N KOPPANEN
Date: 2025.11.10 11:36:35 -05'00'

Suzanne Koppanen
SWIM Program Manager

SAUGAT JUSTEEN PRASAI

Digitally signed by SAUGAT JUSTEEN PRASAI
Date: 2025.11.10 12:20:12 -05'00'

Saugat Prasai
SWIM Governance Lead

DOCUMENT CHANGE HISTORY

Version	Date	Description of Changes
0.1	**02/23/2024**	**Initial Draft Documentation**
1.0	**05/15/2025**	**Baseline Documentation**

Contents

1. Introduction

1.1. Scope

This document describes the governance and applicability associated with the inclusion of timestamp(s) and message sequence number(s) (MSN) for Java Messaging Service (JMS) messages sent via the System Wide Information Management (SWIM) system.

1.2. Background

SWIM is a Service-Oriented Architecture (SOA) information exchange system. The message exchange pattern associated with this governance policy is exclusive to Publish-Subscribe. The Publish-Subscribe pattern uses a JMS Message Broker to transmit information from the SWIM producer to all SWIM consumers that have registered for this business information.

At the present time, there is no standard naming convention used by SWIM producers for message metadata properties representing timestamps that enable measurement of message latency from a SWIM producer to a SWIM consumer. In addition, it is difficult to determine if a message has been lost in transit from a producer to a consumer. Most published services are delivered at a “best-effort” level of reliability, which stipulates a maximum acceptable message loss percentage. Consumers may have a need to detect any or all of the following conditions in messages in a subscription so that they can take appropriate action: 1) gaps in messages received; 2) duplicate messages received; or 3) messages received out of order.

The FAA makes no warranty, expressed or implied, regarding the accuracy, completeness, or timeliness of any information provided through SWIM services. The FAA makes no assurances or guarantees regarding SWIM message latency. The FAA assumes no responsibility for any loss or damage resulting from the use of this information, including any delays in message transmission.

This governance document provides policies, standards and guidelines for message metadata properties to be added to the SWIM data message: 1) a timestamp field(s); 2) a message key field; 3) an MSN field; 4) a Producer Instance Identification (ID) field; and 5) a Producer Instance Message Sequence Number (PIMSN) field.

For consumers to determine message latencies - the time it takes for a message to get from a source to a destination - it is important that the message includes the timestamps at which various events, such as message receipt and publication, occur.

This timestamp information, described in Section 3.1 of this document, should be part of the message metadata so that consumers do not need to examine the body of the message to determine latency, if indeed a relevant timestamp is available there.

One example of how the data specified in this policy might be used is shown in Appendix B: SWIM Monitor and Dashboard (SWIM MD) Prototype Dashboard In this example the SWIM MD prototype receives SWIM Terminal Data Distribution System (STDDS) messages from the SWIM Cloud Distribution Service (SCDS) that contain similar message keys and timestamps to the ones described in this document. This dashboard was created with the idea that a future version might be used with data from other SWIM producers.

1.3. Applicability

The governance specified in this document is applicable to all current and new SWIM producers and will enable SWIM consumers to use the timestamp and message sequence numbers to calculate latency and

determine whether any messages were dropped in transit. All new SWIM producers and SWIM producers that update their service(s) MUST follow the policies listed in this document. New and current producers may apply for a waiver for exclusion from following these policies (See section 5.1.4 of the SWIM Governance Policies document version 3.1, February 6, 2020, for a description of the wavier procedure).

Current producers should be engaged to ensure that programs understand that certain message sequence numbers and timestamps usage and inclusion into services will be required as a part of any update unless the program has qualified for a waiver (required timestamps are available in Table 2, required message sequence numbers are described in Table 3). SWIM should also engage with producers and consumers before their updates and encourage programs to recognize and utilize message sequence numbers and timestamps even though use is not required by this document.

SWIM consumers have the option to use the timestamp and message sequence number values to calculate latency and determine if any messages were dropped in transit.

1.4. Applicable Related Documents

1.4.1. Government Documents

[1] Federal Aviation Administration (FAA). (2021, October 27). *SWIM Controlled Vocabulary*. FAA.gov. https://www.faa.gov/air_traffic/technology/swim/vocabulary

[2] Bradner, Scott O. (1997, March). *Internet Engineering Task Force Request for Comment (RFC) 2119*. https://datatracker.ietf.org/doc/html/rfc2119

1.4.2. Order Of Precedence

In the event of a conflict between the text in this document and the references cited herein, the text in this document takes precedence. Nothing in this document, however, supersedes applicable laws and regulations unless a specific exemption has been obtained, as noted in the Java Messaging Service Description Document (JMSDD) on the National Airspace System (NAS) Service Registry and Repository (NSRR). JMSDDs can be generated using NSRR and JMSDDs should be consistent with information in the NSRR.

1.5. Terms and Definitions

1.5.1. Key Words

The key words “MUST’, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [1].

These key words are capitalized when used to unambiguously specify requirements. When these words are not capitalized, they are meant in their natural-language sense.

1.5.2. Terms and Definitions

Terms used in this document are defined in the SWIM Controlled Vocabulary [1] along with information about terms’ sources and relationships to other terms unless otherwise noted below.

Table 1: Terminology

Term	Definition
Latency	Message latency can be defined as the time elapsed between any two stages of processing or between the time a message is sent from one point and arrives at another point. The timestamps defined in this document can be used to calculate the latency (elapsed time) between any pair of timestamps and what they represent in the flow of message data.
SWIM Producer	In this document, the term SWIM Producer or Producer designates a SWIM system that creates and publishes JMS messages.
SWIM Producer Instance	The specific producer, entity, or instance that creates and publishes a JMS message.
Broker	A Message Broker can receive messages from multiple destinations, determine the correct destination and route the message to the correct channel. The Broker can also make available the standard JMSTimestamp header field on messages.

2. Policy for Producers

This section contains specific policies for the timestamp, message sequence, and message identification message metadata properties. The fields referenced in the following policies apply to new SWIM producers and to SWIM producers that update their service. See Section 3 and in particular Table 2 and Table 3 for specific details.

a) All SWIM producers MAY include an optional “SWIMDataAvailableTS” Timestamp field in messages they publish as described in Table 2.
b) All SWIM producers MAY include an optional “SWIMDataProcessedTS” Timestamp field in messages they publish as described in Table 2.
c) All SWIM producers MUST include a mandatory “SWIMDataPublishedTS” Timestamp field in messages they publish as described in Table 2.
d) All SWIM producers MAY include a “SWIMMsgKey” Message Key field in messages they publish As described in Table 3.
e) All SWIM producers MAY include a “SWIMMsgSeqNum” MSN field as described in Table 3.
f) All SWIM producers MUST include a “SWIMProducerInstanceID” Producer Instance ID field as described in Table 3.
g) All SWIM producers MUST include a “SWIMProducerInstanceMsgSeqNum” PIMSN field as described in Table 3.
h) No SWIM producers MAY disable the JMSTimestamp header field.

3. SWIM Message Timestamps and Sequence Numbers Standards and User Guidelines

This section describes how SWIM producers will create, and how SWIM consumers can utilize, standardized JMS message properties that contain one or more timestamps, a message key, an MSN for that key, and a PIMSN for all messages published by a distinct SWIM producer instance. These standards and guidelines specify the specific message metadata that SWIM producers are required to provide.

Note that the scope of these standards and guidelines does not encompass details of any upstream data sources used by SWIM producers.

These standards will apply to all messages published by producers. Note, however, that any use of mediation to further transform and publish messages can present an additional special case. Mediation performed by the messaging infrastructure is typically one or more additional publish and subscribe steps that may also follow what is described in this policy. Mediation processes can add their own versions of standard JMS header properties (defined below) by appending "-[*Mediation ID1*]", "-[*Mediation ID2*]", etc. to the standard property name. For example, if the first mediation process re-batched messages, it might add a "SWIMMsgSeqNum-*Rebatched*" property to the new message batch. This would allow the "SWIMMsgSeqNum" property applied by the original producer to be preserved.

Including mediations in the set of producers targeted by these policies could enhance the assessment of missed incoming messages and determination of intermediate latencies. Mediations could populate additional relevant timestamps in published message metadata and determine their own MSNs and PIMSNs for inclusion in published message metadata to avoid discontinuities in mediation-published messages resulting from batching or re-batching of incoming messages.

Mediations have the potential to be enhanced to address these needs and should be considered and discussed with the SWIM team as an adjunct to enhancements made to SWIM producers to comply with the policies herein.

3.1. Message Timestamps

One or more timestamps, as defined in the following table, must be added to each published message by the producer to support calculations of latency. At a minimum, SWIMDataPublishedTS, defined below in Table 2, must be provided unless the program has qualified for a waiver as described in section 1.3, Applicability, although the optional timestamps defined below can also be provided and used, if available, to assess latencies occurring prior to a message being published to SWIM. A fourth timestamp can also be recorded by a consumer when a message is received. Based on the available timestamps, one or more latencies can then be calculated by the consumer.

NOTE: The following JMS message property names SHOULD be used by producers and be reflected in the NSRR.

Table 2: JMS Message Property Names – SWIM Data

JMS Message Property Name	Format	Description	Required
SWIMDataAvailableTS	Unix timestamp in milliseconds (e.g. "1672860055005")	Timestamp of when incoming data is received and available for a producer to process, in Coordinated Universal Time (UTC)	No
SWIMDataProcessedTS	Unix timestamp in milliseconds (e.g. "1672860055036")	Timestamp of when a corresponding outbound message was generated by a producer and was ready to be published, in UTC	No
SWIMDataPublishedTS	Unix timestamp in milliseconds (e.g. "1672860055071")	Timestamp of when a corresponding outbound message was published to SWIM, in UTC	Yes

The standard "JMSTimestamp" header field will also be available to consumers from the broker when receiving a message. SWIM producers may not disable timestamps by using setDisableMessageTimestamp or equivalent.

A chart describing the flow of the JMS message property timestamps is seen in Figure 1.

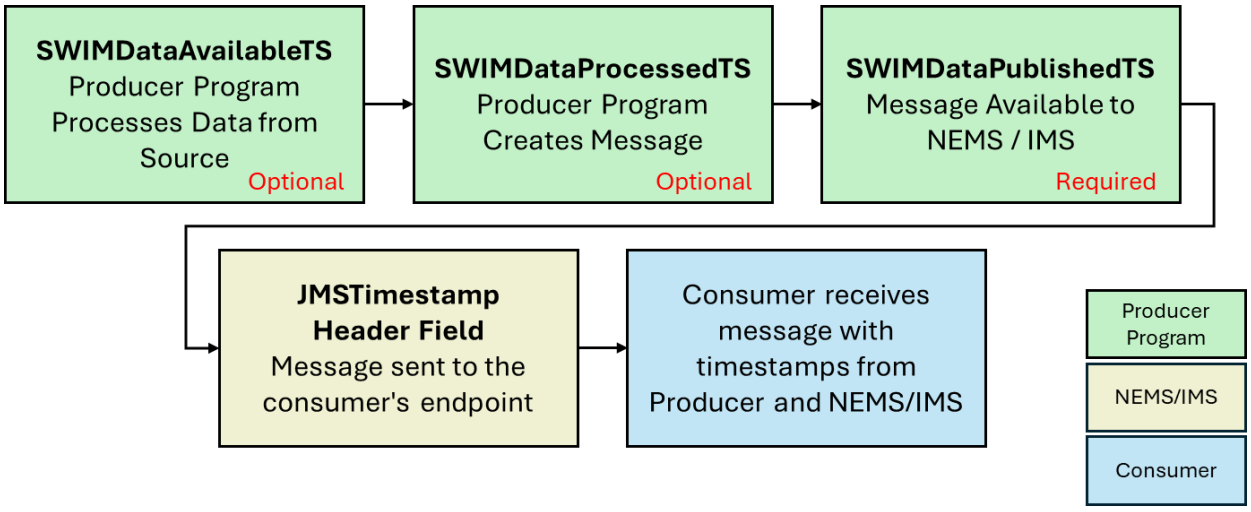

Figure 1: Timestamp Message Properties Flow

Note that both producer and consumer systems should follow the guideline of regularly synchronizing their time with Network Time Protocol (NTP) or equivalent services to support accurate latency calculations. Message producers must not have brokers disable any timestamps.

The following table describes the significance of transitions from each timestamp to the next as well as the primary usage of the transition.

Transition	Significance and Usage
SWIMDataAvailableTS to SWIMDataProcessedTS	How long it took the SWIM producer instance to process the incoming data and prepare a message to be sent. High latency might indicate internal issues with the producer instance.
SWIMDataProcessedTS to SWIMDataPublishedTS	How long it took a SWIM producer instance to publish an outgoing message to the broker. SWIMDataPublishedTS is added to a message by the SWIM producer instance immediately before it attempts to send the message. High latency might indicate internal issues with the producer instance.
SWIMDataPublishedTS to JMSTimestamp Header Field	The latest JMSTimestamp Header Field is populated by the JMS Provider, provides an additional data point for latency calculations, and can be retrieved from the message when the message is received. High latency might indicate issues in the producer instance, such as congestion, temporary unavailability of the NAS Enterprise Messaging System (NEMS) broker, other issues within NEMS, or network issues (this can be narrowed down further depending on what the JMSTimestamp actually represents, which can be determined based on producer instance and consumer connections and path of the message through NEMS).
JMSTimestamp Header Field to Consumer Receives Message with Timestamps	The latest JMSTimestamp Header Field can be retrieved from the message when the message is received, and the transition indicates how long it took a SWIM consumer to receive a message from the JMS Provider. High latency might indicate congestion on the producer instance end, issues within NEMS, issues with the network, or issues with the consumer system that received the message (this can be narrowed down further based on producer instance and consumer connections and path of the message through NEMS).

Note that other groupings of these transitions may be significant.

3.2. Message Keys and Message Sequence Numbers

Whether messages from SWIM are consumed with guaranteed delivery or not, knowing when messages have been missed or duplicated or received out of order may be important to, and actionable by consumers. SWIM will make this discoverable through Message Keys, MSNs, Producer Instance IDs, and PIMSNs provided by producers as JMS message properties, as defined in Table 3 below.

NOTE: The following JMS message property names, as described in this table and below, SHOULD be used by producers and be reflected in the NSRR using the Publish-Subscribe message pattern unless the program has qualified for a waiver as described in section 1.3, Applicability.

Table 3: JMS Message Property Names – Message Key and Sequence Number

JMS Message Property Name	Example	Description	Obligation
SWIMMsgKey	SMES/all/false/ML/KEWR/N90	Message Key for a specific topic. This should be the message topic string or equivalent (or a subset of it, if required) that uniquely defines a stream of messages. The exact format of a Message Key can be determined by the producer. The Message Key should include the sensitivity of the message, if applicable.	Conditional (see **Note 1**)
SWIMMsgSeqNum	7295103	MSN for a specific message corresponding to a given Message Key.	Conditional (see **Note 1**)
SWIMProducerInstanceID (Recommended)	SLC/ATL N90	Identifier of the producer instance. In conjunction with SWIMProducerInstanceMsgSeqNum, this property allows a consumer consuming all messages in a business service with no filtering to determine if any messages are missed.	Conditional (Required only in the case of multiple producer instances)
SWIMProducerInstanceMsgSeqNum	342905938	MSN for all messages of a business service type published by a producer instance (Note: Not per business service, but per producer instance contributing to a business service). If there is more than one producer instance contributing to a service, then the message should contain a JMS property (SWIMProducerInstanceID) whose value indicates the individual producer instance.	Required

An MSN is based upon a corresponding message key or producer instance ID. For example, a STDDS message might have an MSN of “7295103” for a key of “SMES/all/false/ML/KEWR/N90”. If the next MSN received for that same key was “7295105”, the message with MSN “7295104” would likely have been missed (see the discussion below about messages sometimes being duplicated or arriving out of order).

A Message Key will be used in its entirety, and its subfields will not need to be referenced or parsed individually, as those values will also be included separately as application property headers on messages.

Ideally, keys will be specific enough to support all of a SWIM producer's available filters, so that each key and each corresponding stream of data corresponds to the most granular subscription available, and each consumer would nominally be capable of receiving all MSNs in their subscriptions. So, in most cases, the SWIMMsgKey will include all routable fields of a message. A missed MSN for a SWIMMsgKey would indicate a potential missed message, although it's possible one or more apparently missed messages could be received later, out of order.

If a producer has a large number of possible filters and filter values and thus a correspondingly large number of possible keys, that producer may work with the SWIM Program Management Organization (PMO) on a configuration that does not include every possible filter in its message keys (note that message sensitivity cannot be excluded from the keys). The producer could analyze which filters were seldom if ever used by consumers and exclude those filters from the keys, thus providing less granular message keys. However, this could then result in gaps in MSNs for any subscribers who chose to apply a filter not included in the keys.

As an example, an initial assessment of SWIM Flight Data Publication Service (SFDPS) as a producer showed that approximately 244.8 million keys would be required to fully handle all potential combinations of the twelve filters provided by SFDPS, given their possible values. However, after assessing how many filters were actually used by SFDPS subscribers and removing the five of the twelve that were used for filtering by 1.4% of the subscribers or fewer, the revised message key count required was 5050, which is clearly much more manageable for SFDPS. The resulting downside for this small percentage of subscribers is that they would need to avoid using those five filters for their subscriptions to be able to receive meaningful MSNs.

Note 1: If the taxonomy includes a property to indicate sensitive flight data, that property *should at a minimum* be represented in the SWIMMsgKey. In other words, at a minimum, separate message sequences should be established for each potential value that the property indicating sensitive flight data could contain.

For example, messages published by SFDPS contain an FDPS_Sensitive property (which could be set to TRUE or FALSE), and SFDPS should, *at a minimum* ensure that separate sequences are maintained for sensitive messages vs. non-sensitive messages, so that the SWIMMsgSeqNum is incremented separately for messages containing Sensitive Flight Data (SFD), i.e., messages with a value of TRUE for the FDPS_Sensitive property vs. messages not containing SFD, i.e., messages with a value of FALSE for the FDPS_Sensitive property. There is the implication that if this is the only property represented in the SWIMMsgKey by SFDPS, consumers eligible only for non-sensitive data from SFDPS would have to consume all messages they are eligible for (all messages with the FDPS_Sensitive property set to FALSE), and hence not filter messages based on any other criteria, in order to be able to detect missed messages.

A Producer Instance ID and a PIMSN will also be added to every message published for every distinct SWIM producer instance service, which in combination will be unique across all messages published to SWIM. This will support messaging analysis for all messages sent by that producer instance.

Additionally, since the size of MSNs and PIMSNs is finite, MSNs or PIMSNs could theoretically roll over, but this is extremely unlikely to happen due to the 64-bit, Long datatype recommended for MSNs and PIMSNs. Consequently, unless a producer instance happened to persist MSNs and PIMSNs across system reboots, receiving a message with an MSN or PIMSN of 1 would likely be the result of a system being

rebooted and all MSNs and PIMSNs being reset. Note that on a system reset, the lowest MSN or PIMSN published on a message will be 1. Also note that a producer instance may consider intentionally resetting the MSN before it becomes excessively large, if message size is a consideration.

Also note that in cases of failover of the primary producer instance to a backup producer instance, missed or duplicate messages might occur if both are using identical message keys, with possible discrepancies in the published MSNs or PIMSNs resulting due to the failover. This could be avoided by incorporating the Producer Instance ID in the Message Key used so that Message Keys are not duplicated across multiple systems.

Standards for Message Keys, MSNs, and PIMSNs are available in sections 3.2.1, 3.2.2, and 3.2.3.

3.2.1. SWIM Message Keys

A Message Key should be a distinct message metadata property field containing a consistent ordering of subfields separated by slash characters for the sake of readability and to support easy parsing of the subfields if necessary (but not intended to be routable). At a minimum, the subfields should contain the SWIM service name, the SWIM message type code, and the SWIM Producer Instance identification/location, in a well-defined order, if the ordering is defined in the NSRR. Other subfields should also be included as needed, consistent with the details of corresponding subscriptions available to consumers.

For example, the STDDS “SMES/all/false/ML/KEWR/N90” key indicates the STDDS service is “SMES”, the message is suitable for delivery to “all” consumers (i.e. not sensitive), the Limiting Aircraft Data Displayed (LADD) value is “false” to indicate the data does not need to be blocked at the FAA, the STDDS message type of “ML” indicates a Multilateration message, the source of the data was “KEWR” indicating the Newark airport, and the New York Terminal Radar Approach Control Facility (TRACON) (“N90”) published the message. These fields together constitute a Message Key that is directly related to an MSN that a consumer can use to assess any missed messages.

3.2.2. SWIM Message Sequence Numbers

Each message published by a producer (or coordinating producer systems) will need to include a message metadata property field containing the next sequential, unused MSN corresponding to the specified Message Key. MSNs should be 64-bit, Long datatype numbers starting at one.

3.2.3. SWIM Producer Instance Message Sequence Numbers

Each message published by a distinct SWIM producer system will need to include a JMS message property field containing the next sequential, unused PIMSN for that system. PMSNs should be 64-bit, Long datatype numbers starting at one. The SWIM ProducerInstanceID in conjunction with the SWIM ProducerInstanceMsgSeqNum helps identify a stream of messages from each producer instance that contributes to a business service, and enables the detection of missed messages by a consumer that consumes all messages without filtering.

3.2.4. SWIM Producer Instance ID

Identifies the specific SWIM producer system or location that published the message. For example, for STDDS this would be a TRACON identifier like “D10”, which would designate the STDDS instance running at D10 in Dallas, TX.

4. Using the Data

The steps outlined below describe how SWIM consumers may use the timestamps in a JMS message to determine latency on SWIM and between their clients and SWIM. See Figure 1 for the various timestamps which may be available, in addition to a timestamp which can be recorded by the consumer when a message is received. Note that various latencies between these timestamps can be calculated according to a consumer's needs. See the following steps for some sample latency calculations.

4.1. Using SWIM MSN and PIMSN

Consumers will need to save the most recent previous MSN for each key received to be able to detect gaps in messages received. (It's also possible for consumers to retain a list of message gaps and then compare new MSNs which have a lower MSN than the saved MSN against that list to identify duplicate messages or messages which fill in those gaps.

For example, assume that the Message Key for a particular stream of messages is "SMES/all/false/ML/KEWR/N90". For this specific Message Key, each message published should contain the next greater number from the MSN sequence.

Assume that the consumer just received a message with an MSN of 746 and saves this MSN to compare to the MSN of the next message. If the next message received for the same Message Key has an MSN of 749, that indicates that two messages, 747 and 748, did not arrive as expected.

Note that missed messages can typically only be discovered once a subsequent message is received showing a gap in prior received MSNs or PMSNs. When a consumer system first starts up, there may be no previous MSNs or PMSNs to compare to, so some initial missed messages might not be detected in this case.

4.2. Capturing Timestamps

Step 1: Capture the "SWIMDataPublishedTS" timestamp added by the producer

New producers following the updated JMS timestamp policy should add the SWIMDataPublishedTS application property to their messages. This should be a timestamp added by the producer right before a message is sent. Otherwise, the producer may identify an alternative JMS property to serve as a timestamp added prior to publishing their message if the timestamp property is identified in NSRR. If the client is logging the message and its properties in a flat file or database, the consumer can look to the JMS property identified in NSRR. If the consumer wants to measure latency to the client application, it will need to capture the JMS property there.

JMS property: SWIMDataPublishedTS	Long Producer Timestamp:
SWIMDataPublishedTS=17060805888531	long Producer_Timestamp = theMessage.getLongProperty("SWIMDataPublishedTS");

The SWIMDataPublishedTS should be measured in epoch time in milliseconds. For other producer timestamps, the NSRR should describe the expected property format.

Step 2: Capture the "JMSTimestamp" Header Field added on the broker

The value of the JMSTimestamp header property is a long value added by the JMS Provider. It is populated by the JMS Provider when a message is sent to the JMS broker and may subsequently be updated while in transit between the producer and consumer. What exactly is contained in the JMSTimestamp value depends on the path between the producer and consumer. If the client is logging

the message and its properties in a flat file or database, the consumer can look for the JMSTimestamp header field. Alternatively, if the consumer wants to measure latency to the client application, it will need to capture the JMS header field there. Message producers must not attempt to disable the JMSTimestamp.

Example on how to retrieve the JMSTimestamp header field in Java.

JMS property: JMSTimestamp	Long JMS Timestamp:
JMSTimestamp=17060805888693	long JMS_Timestamp = theMessage.getJMSTimestamp();

Per the JMS 1.1 Specification, the JMSTimestamp will be in epoch time in milliseconds. The client can convert this to a human readable time format for logging purposes.

4.3. Measuring Latency

The following steps may be used by a consumer to calculate latency, including optional steps for consumers:

Step 3: Subtract the JMS Timestamp from the timestamp added by the producer

To calculate latency from a producer to the broker, a consumer will need to convert the producer timestamp to epoch time if not already in epoch time. Then to calculate latency in milliseconds, a consumer will simply subtract the producer instance timestamp from the JMSTimestamp. This latency could represent the latency within the producer instance between when the producer instance applied its timestamp and when the JMS Provider applied the timestamp just before the actual sending of the message to the broker to which the producer connects, or the latency between the producer instance and a JMS broker (within NEMS) that applied the timestamp again (which one it is depends on the type of broker within NEMS to which the producer instance connects and whether there are multiple JMS brokers on the path between producer instance and consumer).

Example of calculating producer latency in Java after converting producer timestamp to epoch time	long producerToBrokerLatencyInMilliseconds = JMS_Timestamp – Producer_Timestamp;

Step 4: Analyze the difference between the timestamps to determine if there is very high latency or possible NTP issues

The latency calculated should be in milliseconds. The larger the message, the higher the probability of higher latency. Latency for a 500-byte message should be less than or equal to one second. High latency between the producer instance timestamp and the JMSTimestamp may indicate an issue on the publisher or SWIM. In the case where the calculated latency is a negative number, it could be indicative of an NTP issue on the producer instance or SWIM.

Step 5: (Optional) Capture a timestamp when the message is received by the consumer client

If the client is logging the message and its properties in a flat file or database, the consumer can log the time consumed. Alternatively, if the consumer wants to measure latency to the client application, it will need to capture the time consumed. This would be done after the receive function if using a MessageConsumer or after the onMessage function if using a MessageListener.

Example of capturing time consumed	long Consumer_Timestamp = System. currentTimeMillis();

Step 6: (Optional) Subtract the timestamp when the message is consumed from the JMS Timestamp header field

To calculate the latency in milliseconds from when the JMS Provider received the message to when the consumer received the message, a consumer will simply subtract the JMSTimestamp from the consumed timestamp.

Example of calculating consumer latency in Java	long consumerLatencyInMilliseconds = Consumer_Timestamp – JMS_Timestamp;

Step 7: (Optional) Analyze the difference between the timestamps to determine if there is very high latency or possible NTP issues

The latency calculated should be in milliseconds. The larger the message, the higher the probability of higher latency. There are no requirements for latency between SWIM and the consumer. If a consumer is consuming from a queue and has been disconnected but then reconnects, the latency calculated from SWIM to the consumer could be high. If the consumer is not pulling the messages fast enough from SWIM, the consumer latency could be high. In the case where the calculated latency is a negative number, it could be indicative of an NTP issue on the consumer or SWIM.

Appendix A: Acronym List

Acronym	Definition
FAA	Federal Aviation Administration
GMSN	Global Message Sequence Number
ID	Identification
IMS	Information Management Service
JMS	Java Message Service
JMSDD	Java Messaging Service Description Document
LADD	Limiting Aircraft Data Displayed
MD	Monitor and Dashboard
MSN	Message Sequence Number
NAS	National Airspace System
NEMS	NAS Enterprise Messaging System
NSRR	NAS Service Registry and Repository
NTP	Network Time Protocol
PIMSN	Producer Instance Message Sequence Number
PMO	Program Management Organization
RFC	Request for Comment
SCDS	SWIM Cloud Distribution Service
SFD	Sensitive Flight Data
SFDPS	SWIM Flight Data Publication Service
SOA	Service-Oriented Architecture
STDDS	SWIM Terminal Data Distribution System
SWIM	System Wide Information Management
TRACON	Terminal Radar Approach Control Facilities
UTC	Coordinated Universal Time

Appendix B: SWIM Monitor and Dashboard (SWIM MD) Prototype Dashboard

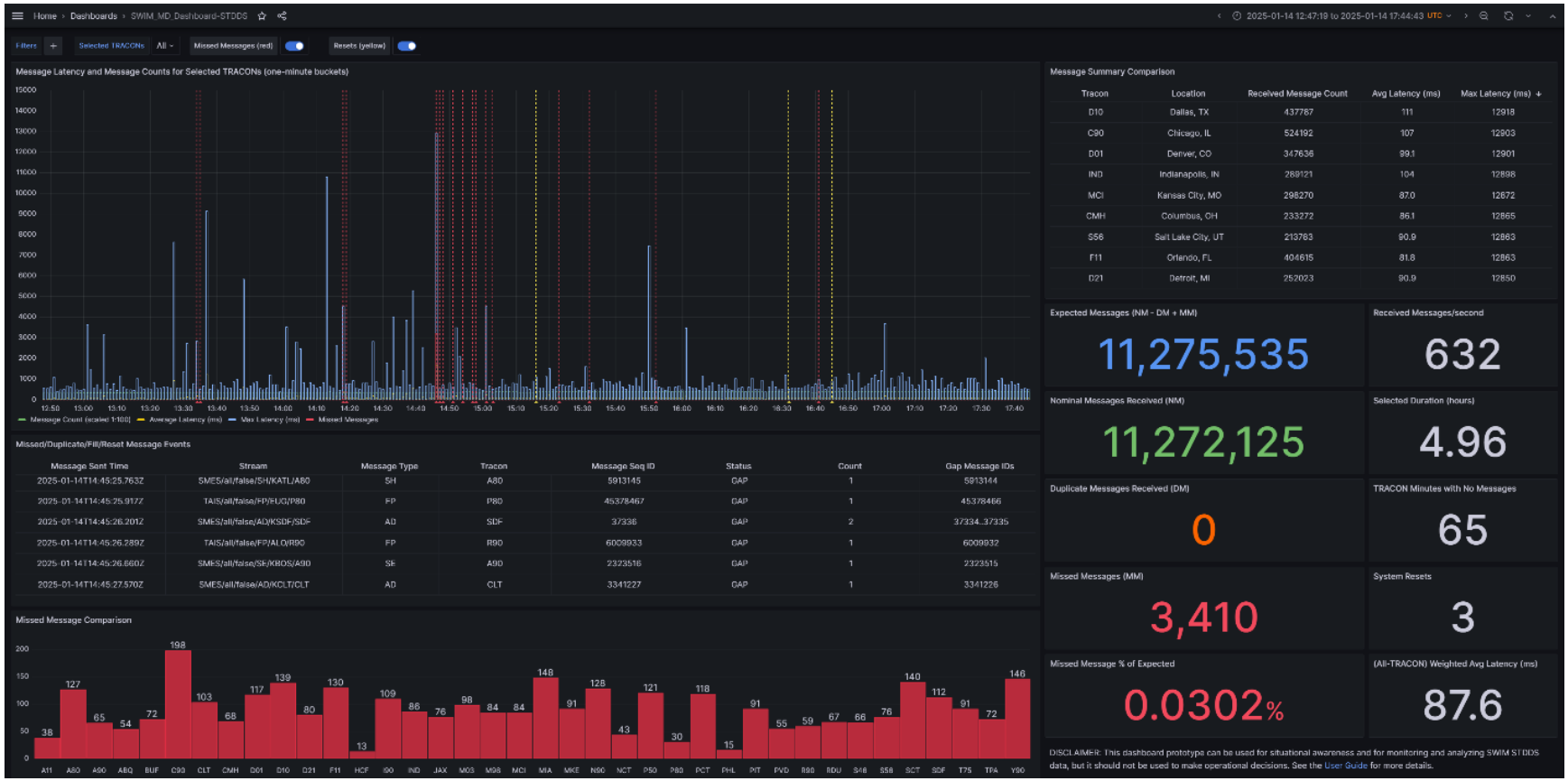

Figure 2: SWIM Monitor and Dashboard (SWIM MD) Prototype Dashboard