Thread Analysis Report

1. Analysis of Problematic Threads

PassThroughMessageProcessor-1, PassThroughMessageProcessor-50, and PassThroughMessageProcessor-107
These threads share a similar stack trace where they call Unsafe.park and subsequently wait on a LinkedBlockingQueue.
They are waiting for a new task from the thread pool's work queue, which is a normal operation when no tasks are available.
The fact that over 100 instances of these threads exist suggests that either the thread pool is configured with higher limits than necessary or there is potential thread leakage if threads are not recycled properly.
There is no evidence of a lock or infinite loop within the stack trace; rather, the threads are scheduled to remain in a waiting state until a new payload arrives.

endpoint-jmx-stat-collector
This thread is part of a scheduled operation that runs periodically to collect JMX statistics.
The stack trace indicates it is waiting on a ScheduledThreadPoolExecutor's DelayedWorkQueue, which is normal behavior for periodic tasks.
There is no sign of resource contention as it simply remains parked until the scheduled time triggers data collection.

PassThroughHTTPSSender
The stack trace for this thread shows that it is waiting during a polling operation via the KQueue poll mechanism.
It is part of the non-blocking I/O layer for sending HTTPS requests, and it intentionally waits for an I/O event before processing further.
This waiting indicates that the thread is not in a deadlock, but possibly that it is waiting for an external event which might be delayed due to downstream issues.

2. Relationship to System Logs

The system logs record several successful log mediator messages indicating that the HealthcareAPI is responding with a "Welcome to HealthcareService" message.
However, a critical error appears when processing a message payload, specifically related to JSON parsing.
The error "Illegal character: <d>" in the log is thrown during the processing in the PayloadFactoryMediator and JSON-to-XML conversion.
This error suggests that the endpoint received a payload containing unexpected characters – likely the payload's format does not match the expected JSON format.
As the JSON payload is malformed, the parser fails and throws an exception, leading to the observed error in the logs while the threads downstream simply return to a waiting state.

3. Overall Diagnosis of the Root Cause

The root cause appears to combine an application-level payload formatting issue and a potential misconfiguration in the thread pool management.
The PayloadFactoryMediator is designed to convert incoming JSON into XML.
The exception indicating an illegal character implies the payload contains unexpected content, potentially due to clients inadvertently sending malformed JSON or even XML instead of JSON.
Simultaneously, the large number of PassThroughMessageProcessor threads waiting may point to an overloaded or misconfigured thread pool that fails to efficiently recycle or limit idle threads, even though they are performing as designed.
The network thread (PassThroughHTTPSSender) is waiting for external I/O events, possibly indirectly impacted by the downstream error processing the malformed payload.

4. Suggested Solutions

Review and validate incoming payloads before they reach the PayloadFactoryMediator.
Implement input validation at the API gateway or an earlier mediation step to detect and reject payloads that do not adhere to the expected JSON format.
Consider logging the raw payload content in a safe manner to diagnose the source of the format error.
Examine and adjust the configuration for the thread pool managing the PassThroughMessageProcessor threads.
Ensure that the thread pool’s maximum size and idle timeout settings are tuned to prevent an excessive number of waiting threads and to mitigate any potential thread leakage.
Review the behavior of the PayloadFactoryMediator to confirm that it correctly handles error scenarios, potentially implementing fallback behavior when encountering malformed payloads.
Conduct stress testing and simulate malformed payload scenarios to ensure that the system recovers gracefully without continuously piling up idle threads.

Each of these measures addresses the two primary issues observed in the system logs and thread dumps: payload format errors causing processing exceptions and a thread pool configuration that may not be optimal for the incoming load and error conditions.