Thread Analysis Report

1. Analysis for Each Problematic Thread
Each PassThroughMessageProcessor thread (“PassThroughMessageProcessor-1”, “-25”, “-50”, “-75”, “-107”) shows a similar stack trace.
Each stack trace begins with a call to Unsafe.park in the native library.
This is immediately followed by calls to LockSupport.park and blocking methods within AbstractQueuedSynchronizer.
Subsequently, each thread is seen calling LinkedBlockingQueue.take to retrieve a new task from the thread pool’s work queue.
Finally, the threads enter the standard ThreadPoolExecutor.runWorker and Thread.run methods.
The pattern indicates that these threads are not actively processing tasks; they are parked and waiting for new work to be scheduled.

2. What Each Thread is Doing and Potential Concerns
Each thread is idling in a waiting state within a thread pool used by the passthrough message processor.
The use of park and subsequent call to take from the LinkedBlockingQueue is a normal pattern for thread pool workers waiting for tasks.
The high number of waiting threads does not necessarily indicate an infinite loop.
However, if many threads remain in this parked state for extended periods, it might imply that no new tasks are being enqueued, or that tasks are taking too long to enter processing.
There is no evidence of resource contention, infinite loops, or deadlocks from these traces.

3. Relation to the System Logs
The system logs show regular informational messages logging “Welcome to HealthcareService” for the GET requests.
An error is logged when a JSON payload conversion fails with an exception: “Illegal character: <d>”.
This error originates from the PayloadFactoryMediator that is attempting to process or transform the payload.
The exception indicates that the input payload contains unexpected characters (“<d>”), likely pointing to an issue where the payload is not valid JSON.
While the waiting state of the threads is normal, the JSON parsing error hints at a mismatch between the expected and the actual payload format.

4. Overall Diagnosis of the Root Cause
The thread dump shows that the PassThroughMessageProcessor threads are in a normal waiting state within the thread pool; they are not hung or deadlocked.
The recurring error in the system logs reveals that the micro integrator is receiving payloads with malformed content (illegal characters for JSON parsing).
Therefore, the root cause appears to be related to payload formatting issues rather than thread locking or resource contention.
It is likely that the endpoint is receiving payload data that does not adhere to the expected JSON format, causing the PayloadFactoryMediator to throw an exception during transformation.

5. Specific Solutions to Address the Identified Issues
Review and validate the payload being sent to the endpoint. Ensure that the payload adheres to proper JSON format without illegal characters.
Implement input validation at the API level to catch and reject malformed JSON before it reaches the mediation logic.
Enhance error handling in the PayloadFactoryMediator to gracefully handle and log descriptive errors when invalid payloads are received.
If the service is expected to handle multiple payload formats, consider configuring a more robust content type detection or transformation strategy.

Monitor and adjust thread pool configurations if necessary; however, the waiting state of the threads appears to be standard behavior when no tasks are available for immediate processing.