1

Message Threading Architecture in Oracle BPEL Process Manager 10.1.3.4

This document describes the message threading architecture in Oracle BPEL Process Manager beginning with release 10.1.3.4.

Message Threading Architecture

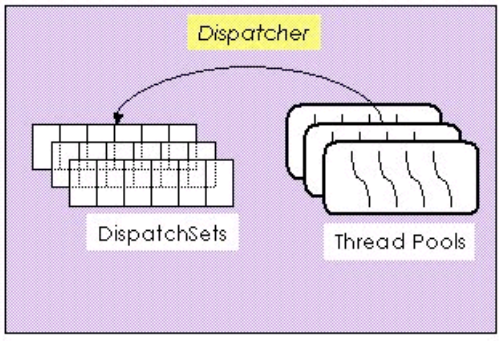
Messages received by clients meant to instantiate a process or internal messages generated during processing of an asynchronous process are managed by the dispatcher module. This module contains three type of dispatch sets:

- System
- Invoke
- Engine

The internal messages (also called dispatch messages) each belong to one of the three dispatch sets.

When the dispatcher needs to schedule a dispatch message for execution, it enqueues the message into a thread pool. Each dispatch set contains a thread pool (`java.util.concurrent.ThreadPoolExecutor`). The thread pool implementation notifies its threads when a message has been enqueued and also ensures that the appropriate number of threads are instantiated in the pool. Figure 1–1 provides details.

Figure 1–1 Thread Pool Implementation



Three configuration properties are provided for you to tune dispatch set threads. Table 1–1 provides details.

Table 1–1 Properties

Property	Description	Values
dspSystemThreads	The total number of threads allocated to process system dispatcher messages. System dispatch messages are general clean-up tasks that are typically processed quickly by the server (for example, releasing stateful message beans back to the pool). Typically, only a small number of threads are required to handle the number of system dispatch messages generated during run time.	The default value is `2` threads. Any value less than `1` thread is changed to the default.
dspInvokeThreads	The total number of threads allocated to process invocation dispatcher messages. Invocation dispatch messages are generated for each payload received by Oracle BPEL Server and are meant to instantiate a new instance. If the majority of requests processed by the engine are instance invocations (as opposed to instance callbacks), greater performance may be achieved by increasing the number of invocation threads. Higher thread counts may cause greater CPU utilization due to higher context switching costs.	The default value is `40` threads. Any value less than `1` thread is changed to the default.
dspEngineThreads	The total number of threads allocated to process engine dispatcher messages. Engine dispatch messages are generated whenever an activity must be processed asynchronously by Oracle BPEL Server. If the majority of processes deployed on Oracle BPEL Server are durable with a large number of dehydration points (midprocess receive, onMessage, onAlarm, and wait activities), greater performance may be achieved by increasing the number of engine threads. Note that higher thread counts can cause greater CPU utilization due to higher context switching costs.	The default value is `60` threads. Any value less than `1` thread is changed to the default.

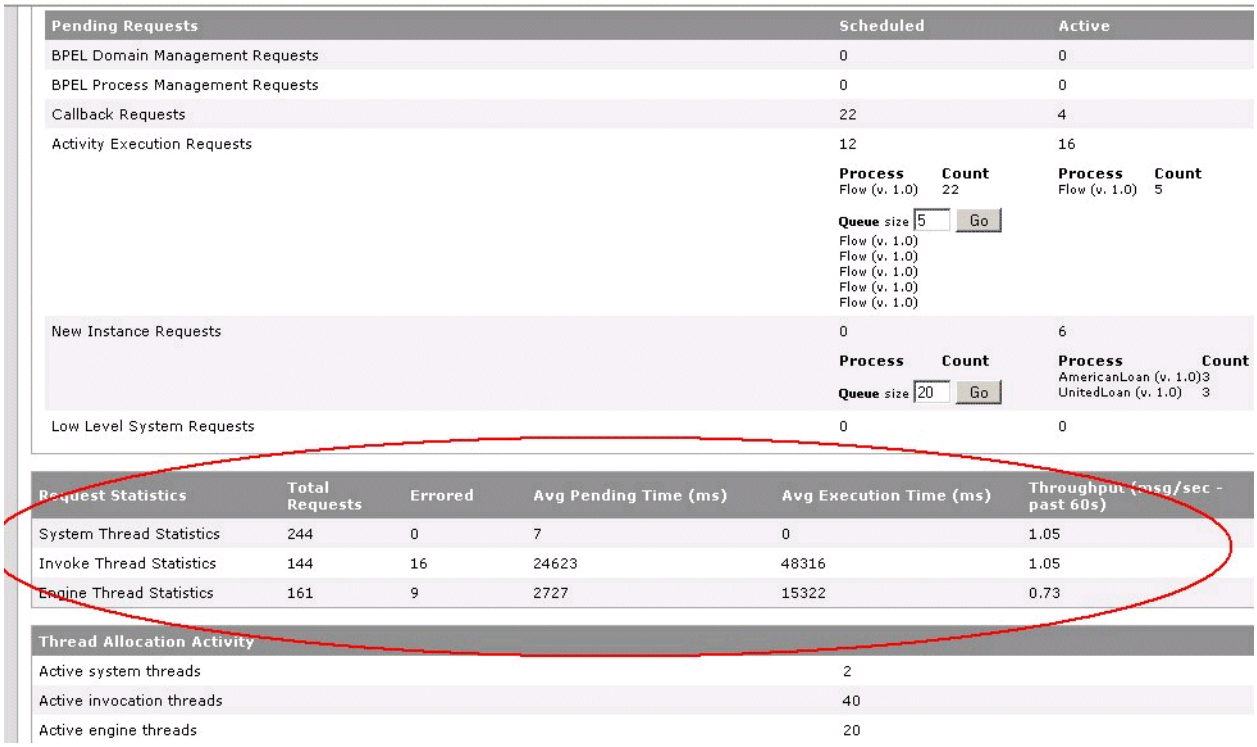
These properties are also described in the 10.1.3.4 Oracle BPEL Process Manager documentation:

`http://download.oracle.com/docs/cd/E12524_01/relnotes.1013/e12523/bpelrn.htm#sthref16`

The minimum number of threads for each thread pool is one. You cannot set any of these properties to `0` or to a negative number.

Message statistics for each individual dispatch set are provided on the Threads page in Oracle BPEL Control. Figure 1–2 provides details.

Figure 1–2 Threads Page in Oracle BPEL Control



Pending Requests	Scheduled	Active
BPEL Domain Management Requests	0	0
BPEL Process Management Requests	0	0
Callback Requests	22	4
Activity Execution Requests	12 Process Count Flow (v. 1.0) 22 Queue size 5 Go Flow (v. 1.0) Flow (v. 1.0) Flow (v. 1.0) Flow (v. 1.0) Flow (v. 1.0)	16 Process Count Flow (v. 1.0) 5
New Instance Requests	0 Process Count Queue size 20 Go	6 Process Count AmericanLoan (v. 1.0) 3 UnitedLoan (v. 1.0) 3
Low Level System Requests	0	0

Request Statistics	Total Requests	Errored	Avg Pending Time (ms)	Avg Execution Time (ms)	Throughput (msg/sec - past 60s)
System Thread Statistics	244	0	7	0	1.05
Invoke Thread Statistics	144	16	24623	48316	1.05
Engine Thread Statistics	161	9	2727	15322	0.73

Thread Allocation Activity	
Active system threads	2
Active invocation threads	40
Active engine threads	20

The **Request Statistics** table provides thread statistics. This table consists of the following fields.

> **Note:** There is no difference in how the statistics are calculated for each type of thread.

- **Request Statistics**:

  Displays the three types of dispatch set threads.

  - **System Thread Statistics** can be tuned with the **dspSystemThreads** property.
  - **Invoke Thread Statistics** can be tuned with the **dspInvokeThreads** property.
  - **Engine Thread Statistics** can be tuned with the **dspEngineThreads** property.

- **Total Requests:**

  Displays the total number of requests for each type of thread. This represents the total number since the last server startup or since statistics were last cleared in Oracle BPEL Control.

- **Errored**:

  Displays the total number of processing errors for each type of thread.

  The error count is only incremented when an exception is thrown from inside the EJB and is caught at the dispatcher layer. Errors that are caught inside the engine and handled (that is, dehydrated during an asynchronous audit) are not surfaced to the dispatcher as a processing error. You receive an exception in the log similar to `"Dispatch error, failed to handle message ..."`.

  Examples of possible errors are as follows:

- System messages: The only system dispatch set error currently processed is if there is an error with the release of stateful EJBs back to the pool.
- Invoke messages: Invoke dispatch set errors can be anything. The invoke message results in an instance being created and saved. Therefore, any system errors encountered during processing such as transaction timeouts, database access errors, out-of-memory errors, EJB problems, and so on, can all cause an error here.
- Engine messages: The engine dispatch set processes a number of message types, such as callback resolutions, activity expiration callbacks, activity executions (if `bpelx:exec` issues a checkpoint), nonblocking invoke callbacks, and asynchronous audit writes. Any errors encountered here result in the engine dispatch set error count being increased.

Note: The error condition can be the same for the invoke and engine dispatch sets (for example, transaction timeouts, out-of-memory errors, database access errors, and so on). However, what matters is which thread encounters the error during processing. There is no correlation with the error type and the error count for these two dispatch sets.

- **Avg Pending Time**:

  Displays the average time in milliseconds that a message stays in the queue before it is picked up by a thread and executed (this does not include the execution time). This is calculated as the total sum of pending times divided by the number of messages.

- **Avg Execution Time**:

  Displays the average type in milliseconds it takes to process a message for each type of thread. The execution time measurement starts when a message is dequeued and stops when the action associated with the message is complete. This is calculated as the total sum of execution times divided by the number of messages.

- **Throughput (msg/sec Past 60s)**:

  Displays the number of messages processed per second for each thread. This is calculated as the number of messages executed, with the execution time for each placed into a circular array list and checked against the times in the last minute.