

## Work Manager and Throttling

LiveCycle ES (and earlier versions) used JMS queues to execute operations asynchronously. In LiveCycle ES2, JMS queues have been replaced by Work Manager. This document provides background information on Work Manager, and provides instructions on configuring Work Manager throttling options.

### About long-lived (asynchronous) operations

In LiveCycle ES2, operations performed by services can be either short-lived (synchronous) or long-lived (asynchronous). Short-lived operations complete synchronously on the same thread from which they were invoked. These operations wait for a response before continuing.

Long-lived operations may span systems or even extend beyond the organization, such as when a client must complete and submit a loan application form as part of a larger solution that integrates multiple automated and human tasks. Such operations must continue while awaiting a response. Long-lived operations perform their underlying work asynchronously, permitting resources to be otherwise engaged while awaiting completion. Unlike a short-lived operation, Work Manager does not consider a long-lived operation complete once it is invoked. An external trigger, such as a system requesting another operation on the same service or a user submitting a form, must occur to complete the operation.

For details on long-lived and short-lived processes, see [Short-lived processes and long-lived processes](#) in the *LiveCycle Workbench ES2 Help*.

### About Work Manager

LiveCycle ES (and earlier versions) used JMS queues to execute operations asynchronously. LiveCycle ES2 uses Work Manager to schedule and execute asynchronous operations via managed threads.

Asynchronous operations are handled in this manner:

1. Work Manager receives a work item for execution.
2. Work Manager stores the work item in a database table and assigns a unique identifier to the work item. The database record contains all of the information required to execute the work item.
3. Work Manager threads pull in work items when the threads become free. Before pulling in the work items, threads can check whether the required services are started, whether there is enough heap size to pull in the next work item, and whether there are enough CPU cycles to process the work item. Work Manager also evaluates attributes of the work item (such as its priority) when scheduling its execution.

LiveCycle Administrators can use Health Monitor to check Work Manager statistics, such as the number of work items in the queue and their statuses. You can also use Health Monitor to pause, resume, retry, or delete work items. (See [View statistics related to Work Manager](#) in the *LiveCycle ES2 Administration Help*.)

## Configuring Work Manager throttling options

You can configure throttling for Work Manager, so that work items are scheduled only when there are enough memory resources available. You configure throttling by setting the following JVM options in your application server.

Property	Description
<code>adobe.work-manager.queue-refill-interval</code>	<p>Specifies the time interval, in milliseconds, that Work Manager uses when checking for new items in its queue.</p> <p>The value for this option is an integer. The default value is 1000 milliseconds (1 second).</p> <p>If the volume of asynchronous invocations is low, you can increase this value. For example, you could increase it to somewhere between 2000 and 5000 (2 to 5 seconds).</p> <p>If the volume of asynchronous invocations is high, the default value should be sufficient, but you can use a lower value if necessary. Decreasing this value too much (for example, below 50, which results in a poll frequency of 20 times per second) causes a substantial overhead on the system.</p>
<code>adobe.workmanager.debug-mode-enabled</code>	<p>Set this option to <code>true</code> to enable debug mode, or to <code>false</code> to disable it.</p> <p>In debug mode, messages regarding Work Manager policy violations and Work Manager pause/resume actions are logged. Set this option to <code>true</code> only when troubleshooting.</p>
<code>adobe.workmanager.memory-control.enabled</code>	<p>Set this option to <code>true</code> to enable throttling based on the memory-control settings described below, or to <code>false</code> to disable throttling.</p>
<code>adobe.workmanager.memory-control.high-limit</code>	<p>Specifies the maximum percentage of memory that can be in use before Work Manager throttles incoming jobs.</p> <p>The default value for this option is 95. This value should be fine for most systems. Increase it only if your system needs to push through to its maximum capacity. But note that as you increase this value, the risk of Out of</p>

	<p>Memory issues also increases.</p> <p>If you are running LiveCycle ES2 in a clustered environment, you may want to set the memory control limit settings differently on different nodes of the cluster. For example, you could have a lower high-limit on nodes A and B, which are programmed in your load balancer for interactive work. And you could have higher high-limits set on nodes C and D, which are not used by the load balancer, but reserved for asynchronous work.</p>
<code>adobe.workmanager.memory-control.low-limit</code>	<p>Specifies the maximum percentage of memory that can be in use before Work Manager stops throttling incoming jobs.</p> <p>The default value for this option is 20. This value should be fine for most systems.</p>

#### To add Java options to JBoss:

1. Stop the JBoss application server
2. Open the `[appserver root]/bin/run.bat` (Windows) or `run.sh` (Linux or UNIX) in an editor and add any of the Java options as required, in the format `-Dproperty=value`.
3. Restart the server.

#### To add Java options to WebLogic:

1. Start the WebLogic Administration Console by typing `http://[host name]:[port]/console` in the URL line of a web browser.
2. Type the user name and password that you created for the WebLogic Server domain and click Log Under Change Center, click Lock & Edit.
3. Under Domain Structure, click Environment > Servers and, in the right pane, click the managed server name.
4. On the next screen, click the Configuration tab > Server Start tab.
5. In the Arguments box, append the arguments you require to the end of the current content. For example, adding `-Dadobe.healthmonitor.enabled=false` disables Health Monitor.
6. Click Save and then click Activate Changes.
7. Restart WebLogic managed server.

#### To add Java options to WebSphere:

1. In the WebSphere Administrative Console navigation tree, do the following for your application server:
  - (WebSphere 6.x) Click Servers > Application servers
  - (WebSphere 7.x) Click Servers > Server Types > WebSphere application servers
2. In the right pane, click the server name.
3. Under Server Infrastructure, click Java and Process Management > Process Definition.
4. Under Additional Properties, click Java Virtual Machine.
5. In the Generic JVM arguments box, type the arguments you require.
6. Click OK or Apply, and then click Save directly to the master configuration.