Pipeline Manager

The Pipeline Manager is responsible for processing jobs (which contain documents to process) and pipelines.

On this page:

Related pages:

- Configuration
- Configuring Health Checks

Key Concepts

Pipeline managers are essentially passive and wait for jobs to arrive (from feeders). The jobs are put on an internal job queue and are then picked up by execution threads (a thread pool), which processes the job through the pipeline to completion.

Jobs

Pipeline managers process jobs. Jobs can come from either of two sources:

1. Feeders - Receive things to do from outside Aspire and create new jobs which they then send to pipeline managers to execute.
2. Other Pipeline Stages - Sometimes, a job is divided into multiple sub-jobs, for example, a Zip file which may contain many nested documents to process. When this happens, a pipeline stage can create one or more sub-jobs, which are sent to a pipeline manager.

Queues and Threading

Every pipeline manager maintains a queue and a thread pool. New jobs received by the pipeline manager will be first placed on the queue. When a thread becomes available, it will take the next job from the queue and will process that job through the specified pipeline.

Note that the thread carries the job all the way through to completion (see below). This may include branching the job to other pipelines (see the <branches> tag below).

See below for a list of parameters that can control job queueing and threading pools (the size of the queue, the maximum number of threads, etc.). Also note that there is a timeout for idle threads, so that system resources are minimized when not all threads are required.

Multiple Pipeline Managers

If you need multiple queues and multiple thread pools, then just create multiple pipeline managers. This is a useful technique for managing thread pools to ensure that one pool does not get starved.

In general, a pipeline manager should only process a certain type of job. If you have multiple types of jobs, it is best to create multiple pipeline managers. For example, parent jobs and sub-jobs are best handled by multiple pipeline managers to ensure that parent job processing is not starved for threads while the sub-jobs are processing.

Branching or Routing to Other Pipeline Managers

Branching from one pipeline manager to another does not cause the job to be re-enqueued onto the remote pipeline manager's job queue. Instead, the original thread is used to continue processing of the job through the remote pipeline manager's pipeline.

This means that once a thread has accepted a job to process, that thread will process the job all the way through to completion - even if this means calling the process() method of other pipeline managers to process the job.

The same is true when jobs are routed to other Pipeline Managers using routing tables.

Pipeline Managers are also Component Managers

A pipeline manager is a sub-class of component managers. This means that component manager configuration (such as installing bundles and creating components) are also available as part of the pipeline manager configuration.

Job Completion

A job is "completed" in either of two situations:

1. It has been processed through the last pipeline stage and there is no "onComplete" branch defined.
    - If there is an "onComplete" branch defined, then the job will be branched to the specified destination and its processing will continue.
2. Any pipeline stage has returned an exception while processing the job.
    - This will typically cause the job to "complete".
    - However, if the pipeline contains an "onError" branch, then the job may continue processing on some other pipeline (an error handling pipeline, for example).

If either of these two situations occur, then the job is "completed". The pipeline manager will do the following for completed jobs:

1. If the job is a sub-job, the parent job will be notified that this sub-job is complete.
2. If the job is a parent job, then the pipeline manager will block the job until all of its sub-jobs are complete.
3. If there are listeners on the job, they will be notified that the job is completed and each one will be given the opportunity to do further processing on the job.
    - This is typically used for feeders, which may need to do some cleanup and/or bookkeeping about the job once it is completed.
4. Once all feeders are complete, the job is closed.
    - The job's data object (typically an instance of AspireDocument) will be closed.
    - Any objects that are on the "Closeable list" of the AspireDocument (input streams, RDBMS connections, and so on) that are attached to the job object will be closed.
    - All references are set to null to release memory as quickly as possible.

Health Checks

Pipeline managers are also responsible for performing system health checks. Health checks check the overall health of the system for things like:

- Jobs which encounter exception errors
- Jobs which are too slow
- Components which are still initializing after startup

Once configured, the health of the entire server, as well as detailed information on each health check, is available through the admin RESTful interface.