JAKARTA EE

Jakarta Batch

Jakarta Batch Team, https://projects.eclipse.org/projects/ee4j.batch

1.0, August 02, 2019

Table of Contents

```
Specification: Jakarta Batch

Version: 1.0

Status: Final Release

Release: August 02, 2019
```

Copyright (c) 2019 Eclipse Foundation.

Eclipse Foundation Specification License

By using and/or copying this document, or the Eclipse Foundation document from which this statement is linked, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to copy, and distribute the contents of this document, or the Eclipse Foundation document from which this statement is linked, in any medium for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the document, or portions thereof, that you use:

- link or URL to the original Eclipse Foundation document.
- All existing copyright notices, or if one does not exist, a notice (hypertext is preferred, but a textual representation is permitted) of the form: "Copyright (c) [$date-of-document] Eclipse Foundation, Inc. [url to this license]"

Inclusion of the full text of this NOTICE must be provided. We request that authorship attribution be provided in any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of Eclipse Foundation documents is granted pursuant to this license, except anyone may prepare and distribute derivative works and portions of this document in software that implements the specification, in supporting materials accompanying such software, and in documentation of such software, PROVIDED that all such works include the notice below. HOWEVER, the publication of derivative works of this document for use as a technical specification is expressly prohibited.

The notice is:

"Copyright (c) 2018 Eclipse Foundation. This software or document includes material copied from or derived from [title and URI of the Eclipse Foundation specification document]."

Disclaimers

THIS DOCUMENT IS PROVIDED "AS IS," AND THE COPYRIGHT HOLDERS AND THE ECLIPSE FOUNDATION MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

THE COPYRIGHT HOLDERS AND THE ECLIPSE FOUNDATION WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of the copyright holders or the Eclipse Foundation may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

Chapter 1. Introduction

The Jakarta Batch project describes the XML-based job specification language (JSL), Java programming model, and runtime environment for batch applications for the Java™ platform.

The specification ties together the Java API and the JSL (XML) allowing a job designer to compose a job in XML from Java application artifacts and conveniently parameterize them with values for an individual job. This structure promotes application reuse of artifacts across different jobs.

Some key features:

- **checkpoint / restart** - The application read-process-write loop is performed under a global transaction, one "batch" or "chunk" of data at a time, with the batch implementation atomically storing a "checkpoint" at the end. This checkpoint provides an index into the data stream which allows you to restart a job after an earlier execution hits a failure (or is stopped), such that picks up where you left off (at the checkpointed value).

- **steps** - jobs can be composed of steps to allow reuse of step logic and definitions within multiple jobs, as well as to faciliate restart (at the step the job left off at).

- **XML configuration** - Configuration is externalized from Java code into XML and parameterized through a variety of "job property" substitutions. As one example, this allows database lock tuning (for locks held during the duration of the chunk transaction) to be tuned without touching Java code.

- **partitions** - The read-process-write loop can be broken up into multiple units running in parallel against different segments of the input data.

The specification allows the flexibility for batch jobs to be scheduled or orchestrated in any number of ways, and stops short of defining any APIs or constructs regarding scheduling or orchestration of multiple or repeated jobs.