Reactive Streams

Reactive Streams is an initiative to provide a standard for asynchronous stream processing with non-blocking back pressure. This encompasses efforts aimed at runtime environments (JVM and JavaScript) as well as network protocols.

JDK9 `java.util.concurrent.Flow`

The interfaces available in JDK9's java.util.concurrent.Flow, are 1:1 semantically equivalent to their respective Reactive Streams counterparts. This means that there will be a migratory period, while libraries move to adopt the new types in the JDK, however this period is expected to be short - due to the full semantic equivalence of the libraries, as well as the Reactive Streams <-> Flow adapter library as well as a TCK compatible directly with the JDK Flow types.

Read this if you are interested in learning more about `Reactive Streams` for the JVM.

The Problem

Handling streams of data—especially "live" data whose volume is not predetermined—requires special care in an asynchronous system. The most

prominent issue is that resource consumption needs to be controlled such that a fast data source does not overwhelm the stream destination. Asynchrony is needed in order to enable the parallel use of computing resources, on collaborating network hosts or multiple CPU cores within a single machine.

The main goal of Reactive Streams is to govern the exchange of stream data across an asynchronous boundary—think passing elements on to another thread or thread-pool—while ensuring that the receiving side is not forced to buffer arbitrary amounts of data. In other words, back pressure is an integral part of this model in order to allow the queues which mediate between threads to be bounded. The benefits of asynchronous processing would be negated if the communication of back pressure were synchronous (see also the Reactive Manifesto), therefore care has to be taken to mandate fully non-blocking and asynchronous behavior of all aspects of a Reactive Streams implementation.

It is the intention of this specification to allow the creation of many conforming implementations, which by virtue of abiding by the rules will be able to interoperate smoothly, preserving the aforementioned benefits and characteristics across the whole processing graph of a stream application.

Scope

The scope of Reactive Streams is to find a minimal set of interfaces, methods and protocols that will describe the necessary operations and entities to achieve the goal—asynchronous streams of data with non-blocking back pressure.

End-user DSLs or protocol binding APIs have purposefully been left out of the scope to encourage and enable different implementations that potentially use different programming languages to stay as true as possible to the idioms of their platform.

We anticipate that acceptance of this Reactive Streams specification and experience with its implementations will together lead to wide integration, for example including Java platform support in future JDK releases or network protocol support in future web browsers.

Working Groups

Basic Semantics

The basic semantics define how the transmission of stream elements is regulated through back-pressure. How elements are transmitted, their representation during transfer, or how back-pressure is signaled is not part of this specification.

JVM Interfaces (Completed)

This working group applies the basic semantics to a set of programming interfaces whose main purpose is to allow the interoperation of different conforming implementations and language bindings for passing streams between objects and threads within the JVM, using the shared memory heap.

As of *August 23rd, 2019* we have released version 1.0.3 of Reactive Streams for the JVM, including Java API, a textual Specification, a TCK and implementation examples.

New in 1.0.3 is that the JDK9 [adapter library] is included in the main jar.

Corresponding code artifacts are available on Maven Central:

```
<dependency>
  <groupId>org.reactivestreams</groupId>
  <artifactId>reactive-streams</artifactId>
  <version>1.0.3</version>
</dependency>
<dependency>
  <groupId>org.reactivestreams</groupId>
  <artifactId>reactive-streams-tck</artifactId>
  <version>1.0.3</version>
</dependency>
<dependency>
  <groupId>org.reactivestreams</groupId>
  <artifactId>reactive-streams-tck-flow</artifactId>
  <version>1.0.3</version>
</dependency>
<dependency>
  <groupId>org.reactivestreams</groupId>
  <artifactId>reactive-streams-examples</artifactId>
  <version>1.0.3</version>
</dependency>
```

The source code for these is available on github. Please use github issues for providing feedback.

All artifacts and specifications are released under Creative Commons Zero into the Public Domain.

Read more about `Reactive Streams 1.0.3` for the JVM here.

A Note for Implementors

To get started implementing the final specification, it is recommended to start by reading the README and the Java API documentation, then taking a look

at the Specification then taking a look at the TCK and the example implementations. If you have an issue with any of the above, please take a look at closed issues and then open a new issue if it has not already been answered.

This work was performed in the reactive-streams-jvm repository.

JavaScript Interfaces

This working group defines a minimal set of object properties for observing a stream of elements within a JavaScript runtime environment. The goal is to provide a testable specification that allows different implementations to interoperate within that same runtime environment.

This work is performed in the reactive-streams-js repository.

Network Protocols

This working group defines network protocols for passing reactive streams over various transport media that involve serialization and deserialization of the data elements. Examples of such transports are TCP, UDP, HTTP and WebSockets.

This work is performed in the reactive-streams-io repository.