JSR 282 Public Review
Briefing for the Java Community Process Executive Committee
San Francisco CA, 13th of September 2019
Agenda

• Goals
• Business, Market, Ecosystem Justification
• Background
• History
• Expert Group
• Technical Scope and Features
• Deliverables: Specification, RI, TCK, IP, Other
• Schedule
• Collaboration, Participation, and Transparency
• Questions, discussion, next steps
Goals

• The original goal was to address some of the simpler enhancements that have been requested in the Real-Time Specification for Java (RTSJ) of which 21 were listed explicitly.

• This has lead to a re-evaluation of the specification to clarify ill defined parts of the specification and complete partially defined features such as user defined clocks and happenings.

• Providing better integration with current conventional Java implementations has also become important.
Business, Market, Ecosystem Justification

- The RTSJ extends Java ecosystem into deeply embedded systems, especially where realtime response is required.
- The RTSJ 1.0 was good starting point for using Java for realtime and embedded applications.
- JSR 282 updates the RTSJ to the current state of the art by clarifying its semantics and filling in major gaps.
- This is not a new standard, but a refinement of an existing one based on field experience.
- Required to make further inroads in replacing C and C++ in embedded systems, thus broadening the Java ecosystem.
Background

- Update to JSR-1: Real-Time Specification for Java (RTSJ)
- RTSJ refines Java semantics and adds APIs for realtime
  - no changes to Javac necessary
  - fully compatible with conventional Java Language
- Targets all platforms (was J2ME)
- This is a single JSR platform
- Necessary for extending Java ecosystem into realtime and embedded systems
History

• The RTSJ was completed in December 1998
• JSR 282 was approved in August 2005
• Early Draft Review Q2 2009
• Peter Dibble left TimeSys in May 2010
• aicas became specification lead in October 2012
• IP transfer from TimeSys in August 2014
• Second Early Draft Review Q4 2014
• Third Early Draft Review Q1 2017
• Fourth Early Draft Review Q2 2018
The Expert Group

• The EG consists of the following members:
  - Industrial: aicas, IBM, Atego, Ethan Blanton
  - Academic: Andy Wellings (realtime system expert)
  - Other Communities: Ben Brosgol (Ade Industrial)
• The EG meets weekly by teleconference
• The EG communicates internally with Webex, e-mail, Daft postings, and an SVN repository
Inhibitors of Realtime Response

- Garbage Collection
  - Convention GC can interrupt high priority tasks
- Just-In-time Compilation
  - Only fast after compilation, maybe
- Scheduling
  - Undefined; de facto fair; no real task prioritization; just niceness
- Synchronization
  - Low priority task can block a higher priority task (priority inversion)
Realtime Refinements and APIs

- Garbage Collection
  - Deterministic GC
  - Memory Areas
- Static Compilation
  - Fast every time, but changes deployment model
- Scheduling
  - FIFO
  - Round Robin Scheduling
- Synchronization
  - Priority inheritance
  - Priority ceiling emulation
RTSJ 2.0 Features

- **Base Module**
  - Realtime Threads
  - Events & Handlers
    - Timers
  - Priority Inversion Avoidance
  - Monolithic and World Clocks
  - Base Memory Areas
  - Schedulers
  - CPU Affinity

- **Device Access**
  - Happenings & ISR
  - Raw Memory
  - DMA Support

- **Alternate Memory Areas**

- **Resource Enforcement**

- **POSIX**

- **CONTROL**
  - ATC & Abort
Examples

- **Industrial Applications**
  - Spül — mattress fabrication
  - Schlafhost — dynamic thread repair in weaving

- **Automotive**
  - Perrone Robotics — autonomous driving
  - FCA — remote monitoring

- **Frameworks**
  - Resource enforcement in OSGi
Intellectual Property

- License fulfills JCP requirements
  - Nondiscriminatory License for RI and TCK
  - checked by JCP Legal
- Contributor Agreement similar to that of OpenJDK
- The collaboration tools are free to use as EG member
  - Webex guest
  - open source tools
- All IP belongs to Spec Lead
RI and TCK

- The TCK is an extension to the RTSJ TCK and is being developed by the EG
- TimeSys had published an RI
- aicas is developing a new RI
Other deliverables

- The Specification is more than just the JavaDocs:
  - Semantics and
  - Rationale (including some examples)

- EG will work to provide freely available
  - user’s guide,
  - compatibility library (for development)
  - sample code, and
  - FAQ
## Roadmap

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Review Start</td>
<td>September 2019</td>
</tr>
<tr>
<td>Public Review End</td>
<td>October 2019</td>
</tr>
<tr>
<td>RI Core Modules</td>
<td>February 2020</td>
</tr>
<tr>
<td>Complete RI and TCK</td>
<td>August 2020</td>
</tr>
<tr>
<td>Final Release</td>
<td>September 2020</td>
</tr>
</tbody>
</table>
Collaboration with other community groups

• Collaborating with JSR-302
  - Ensure maximal compatibility
  - Issues and changes worked with JSR-302
  - Small changes to support JSR-302 on RTSJ 2.0
  - Two EG members are also JSR-302 members

• OSGi Alliance
  - Realtime version of OSGi
  - Resource Enforcement
Mailing List and Forums

- Mailing list: jsr282-feedback@aicas.com
- Twitter: @realtimejava #RTSJ
- Discussion: https://www.linkedin.com/groups/8147216/
Questions, discussion, next steps