ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH1003

# jddd MIGRATION TO OpenJDK11+: BENEFITS AND PITFALLS

E. Sombrowski, K. Rehlich, G. Schlesselmann, DESY, Hamburg, Germany

*Abstract*

The Java Doocs Data Display (jddd) is a Java-based tool for creating and running graphical user interfaces for accelerator control systems. It is the standard graphical user interface for operating the European XFEL accelerator. Since Java 8 Oracle introduced a number of major changes in the Java ecosystem's legal and technical contexts that significantly impact Java developers and users. The most impactful changes for our software were the removal of Java Web Start, Oracles new licensing model and shorter release cycles. To keep jddd up to date, the source code had to be refactored and new distribution concepts for the different operating systems had to be developed. In this paper the benefits and pitfalls of the jddd migration from Oracle Java8 to OpenJDK11+ will be described.

## INTRODUCTION

jddd [1, 2] is a common tool for designing and running control system windows (also called panels) at DESY Hamburg and DESY Zeuthen. It has a graphical editor with a rich set of ready-made widgets for control panel design. Synoptical displays can easily be created without any programming knowledge. So far, more than 15000 panels have been designed in Hamburg and Zeuthen.

At DESY Hamburg statistical data about the jddd usage are collected. The average and maximum number of panels started per day, users per day and sessions started per day are displayed in Table 1. In recent years usage has been steadily increasing.

Table 1: jddd Usage Statistics 2018 at DESY Hamburg

| | #started panels per day | #started diff. panels per day | #started jddd sessions per day |
|---|---|---|---|
| av | 2982 | 558 | 169 |
| max | 7221 | 1089 | 434 |

## JAVA UPGRADE

Since Java 8 Oracle made significant modifications in the Java ecosystem [3]. To keep jddd up to date, new solutions had to be found for the following changes:

### *Oracles New Licensing Model*

Starting with Java 11 Oracle offers two distinct Java releases with different license models:

- Oracle JDK under commercial OTN License Agreement for Java SE [4]: This release offers a long term support, but is only free of charge for development and tests. For commercial use high costs incur.
- Oracle OpenJDK under the open source GNU General Public License v2 with Classpath Exception (GPLv2+CPE): This release is free of charge, but does not offer long term support.

As an alternative to Oracle a zoo of long term support OpenJDK distributions from different communities and vendors is available, like AdoptOpenJDK, Corretto by Amazon or Read Hat OpenJDK.

Oracle has been and will stay the reference implementation for Java. We decided to use Oracle OpenJDK, because the license model fits our needs and environment best.

### *Dealing with Shorter Release Cycles*

Some months ago Oracle introduced a new release cycle for Java. It changed from a feature-based to a time-based release cycle. There are now two types of releases:

- Major releases for Oracle JDK and OpenJDK every 6 months, which are only supported until the next release.
- Commercial Oracle JDK Long Term Support (LTS) releases every 3 years: The most recent LTS release is Java 11, which came out in September 2018 and will be supported until 2026. The next will be Java 17 in September 2021.

The advantage of short release cycles is that new Java features will be usable much faster.

It also makes migration easier. The changes are smaller and more incremental, so each upgrade is easier and less of change.

We started jddd migration from Java 8 to OpenJDK11+ in September 2018 with OpenJDK 11, which turned out to be quite easy. No major code changes had to be done. Since OpenJDK11 jddd application are always compiled with the latest OpenJDK release without any problems.

### *Replacement of Java Web Start*

During the past years Java Web Start and Java Network Launching Protocol (JNLP) were used for the internal distribution of jddd to the users. To access the application the users had to install a current Java that contained support for JNLP. To start jddd, they would click on a JNLP link and the Java Web Start program would download the JNLP file, interpret it, download the current version of the application, and run it in a security sandbox.

Java Web Start provided an easy and uncomplicated way to distribute a completely configured application to everyone's desktop independent of the operating system. As Java Web Start is no longer part of Java, a substitution for JNLP had to be found.

Java 8 is the last version providing a separate Java Runtime Environment (JRE). Since Java 9 there is only the Java Development Kit (JDK) available, which is basically the JRE plus a compiler, a debugger and several other tools. Instead of a separate JRE Oracle introduced the module system Jigsaw. The idea of Jigsaw is to modularize the Java

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH1003



application and to use jlink for the generation of a custom JRE that contains only the platform modules required for a given application.

Packaging the application with this reduced JRE would be the most lightweight and performant solution for application deployment. Unfortunately jddd depends on many old non-modularised libraries, which makes it impossible to use jlink.

For this reason we decided to compile and bundle all jddd jars together with the complete OpenJDK. Always the latest OpenJDK version is used to ensure the best possible safety.

Four different packages are needed at DESY (see screenshots in Fig. 1). One for each:

- The jddd editor
- XFEL MainTaskbar
- FLASH MainTaskbar
- SINBAD MainTaskbar

On Windows these packages are available via a centrally managed software repository. On Linux packages are provided for Debian, CentOS and RedHat. On Mac the jddd packages are available as Mac applications.

## STARTING OLD JAVA APPLICATIONS VIA A jddd BUTTON

Figure 1 shows the jddd based control windows for XFEL, FLASH and SINBAD (called MainTaskbar), which are used as entry point to start all needed graphical user interfaces (GUIs). These are mainly jddd panels, but can also be Matlab, Python or other Java applications.

Because it would be much effort to upgrade all old Java applications, we found a simple solution to start these GUIs without any code and JNLP start script modification: We added jaws, a Java Web Start replacement by Cosylab [5], to our jddd packages. The jddd application now internally replaces all javaws calls by jaws. Because the Java Virtual Machine is backward compatible and can run older bytecode, most old Java applications (except JavaFX applications) can be started with the current OpenJDK from the jddd package.

Using jaws is a suitable intermediate solution until the transition from Java 8 to OpenJDK11+ will be finished for all applications.

## UPGRADE OF THE MESSAGING SYSTEM - FROM JMS TO MQTT

The jddd software utilizes a messaging system for collecting runtime statistics (as shown in table 1), evaluating failures/exceptions and sending panel update notifications within running jddd applications at DESY.

With the Java upgrade we decided to switch the messaging system from JMS to the more popular MQTT protocol. The Eclipse Paho Java Client library [6] provides an open-source implementation of the MQTT messaging protocol. It is well documented and fits our needs perfectly.

As broker we decided to use the open-source message broker Eclipse Mosquitto [7] because it is lightweight and available as Debian package.

## UPGRADE OF THE jddd WEB INTERFACE

For remote monitoring and expert assistance an HTML5 version of jddd has been developed [8]. At server side jddd is started in a Tomcat application server. A buffered image of each panel is created with an update rate of 0.5 Hz. For client/server communication the WebSocket protocol is used. Images are sent from the server to the client, mouse events are sent in the opposite direction.

The web interface has been successfully tested with OpenJDK11+ and Tomcat 9. The upgrade is scheduled for the end of this year.

## CONCLUSION

For the Java update of our jddd software, the following decisions were made:

- We use Oracle OpenJDK11+.
- We stick to Oracles short release cycles and compile and distribute jddd always with the current OpenJDK version.
- We deploy jddd packages including all jars, the complete JDK and the jaws software for starting old Java applications via jddd.

The migration from Java 8 to OpenJDK11+ was successful. No major problems were observed, no matter on which operating system. Unfortunately it has not been possible to switch completely to the new Java module system, due to many legacy libraries. The plan is to replace these libraries to be able to use jlink in future.

The concept of jddd packages containing the current JDK has proven itself. Even though the packaging and deployment procedure is more complicated now compared to Java Web Start before, the benefit is that jddd no longer depends on the appropriate preinstalled Java version.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WESH1003

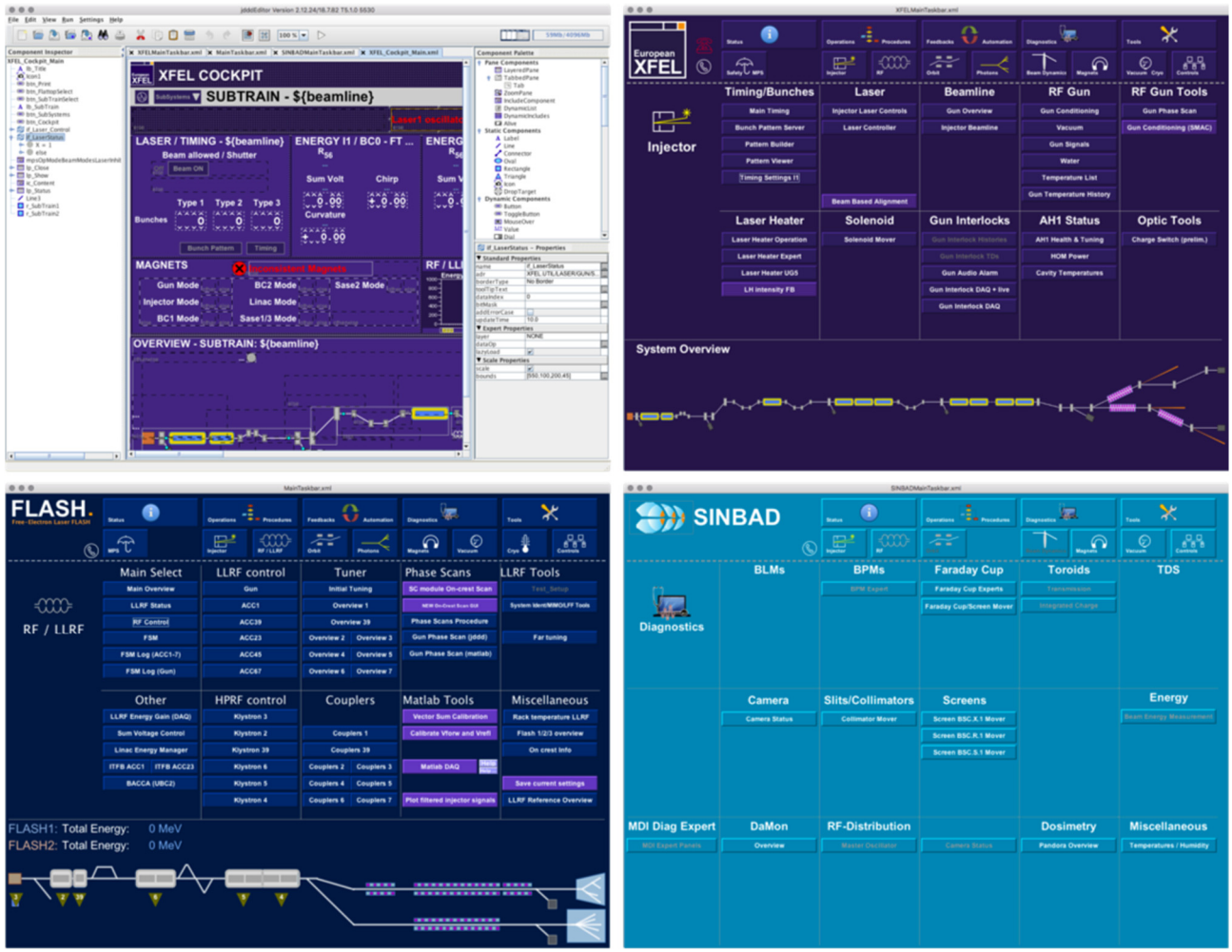


Figure 1: Screenshots of the jddd Editor (top left), XFEL MainTaskbar (top right), FLASH MainTaskbar (bottom left), SINBAD MainTaskbar (bottom right).

## REFERENCES


[1] doocs/jddd, http://doocs-web.desy.de

[2] E. Sombrowski, A. Petrosyan, K. Rehlich, and W. Schütte, "jddd: A Tool for Operators and Experts to Design Control System Panels," in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper TUMIB09, pp. 544-546.

[3] Oracle Java Client Roadmap Update, https://www.oracle.com/technetwork/java/javase/javaclientroadmapupdate2018mar-4414431.pdf

[4] Oracle Technology Network License Agreement for Oracle Java SE, https://www.oracle.com/downloads/licenses/javase-license1.html

[5] COSYLAB, https://www.cosylab.com

[6] Eclipse Paho MQTT Java library at GitHub, https://github.com/eclipse/paho.mqtt.java

[7] Eclipse Mosquitto MQTT broker homepage, https://mosquitto.org

[8] E. Sombrowski, R. Kammering, and K. R. Rehlich, "A HTML5 Web Interface for JAVA DOOCS Data Display," in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 1056-1058. doi:10.18429/JACoW-ICALEPCS2015-WEPGF150


.