GlassFish to Payara Server 4 Migration Guide

The Payara® Platform - Production-Ready,
Cloud Native and Aggressively Compatible.

Migration Guide

Contents

Introduction 1
Why Choose Payara Server Enterprise 2
Overview of Migration Steps 3
Migration From GlassFish 3.X 4
Migrating From GlassFish 4.X 5
Working With Third-Party Libraries 5
How To Deal With Oracle Commercial Features 5
- Coherence Active Cache 6
- Monitoring Scripting Client 6
- Oracle Access Management Integration 7
- Performance Tuner 7
- Load Balancer Configurator Plugin 7
- Domain Administration Server Backup and Recovery 7

Post-Migration 8
- Slow SQL Logging 8
- Health Check Service 8
- Request Tracing Service 8
- Default Role/Group Mapping 9
- Hazelcast 9
- Payara Micro 9
- Other Production Features 9

How is Payara Server Enterprise Better Than GlassFish? 10
Conclusion 12

Introduction

Since Sun Microsystems first developed GlassFish as the reference implementation (RI) of Java EE, it has been used by both developers in its open source form, and by Sun - then, later, by Oracle - in production, fully supported. In a competitive market, GlassFish has fared well and become a popular choice, with the vast majority using the GlassFish Server Open Source Edition. In late 2013, Oracle announced the end of commercial support for GlassFish. By the time of publication of this guide, Premier Support has ended (March 2016) and the clock is ticking on Extended Support, which has less than three years before it ends (March 2019)

Obviously, Oracle's decision has presented a problem: both for users of the open source edition, and licensees of Oracle GlassFish. There have only been three Java EE 7 certified releases of GlassFish Open Source Edition since 2013, and concerns are arising for the future quality of GlassFish, due to the lack of commercial support users raising bugs and funding the fixes for the community, as well as other paying users.

One year after the original announcement of the end of support from Oracle, Steve Millidge - a Java EE expert and the founder of Payara Services Ltd - announced the arrival of Payara Server and, with it, a reintroduction of a 24/7 vendor support option for businesses who had previously chosen GlassFish as a platform. Payara Server proved to be the logical 'drop-in replacement' for GlassFish Server Open Source Edition, helping companies migrate quickly and easily onto a very similar platform with added reassurance of regular releases, bug fixes, patches and enhancements.

As Payara Server has developed, its popularity has grown very quickly among the community due to an open and responsive development team, a regular quarterly release cycle and introduction of some crucial production enhancements which were never available in GlassFish.

Migrating from GlassFish to Payara Server can be a simple and straightforward process, made even simpler with the help of this Guide. With no need for code rewrites or application re-architecting, Payara Server is a credible solution on which to build your Java middleware platform. On the next pages, you will find an overview of the things you will need to consider in your GlassFish to Payara Server migration project; as well as details of Payara Server's tools and features which will make your life (and your application server management) much easier!

Why Choose Payara Server Enterprise

The first decision to make when facing a migration is to consider what platform to migrate to. Staying with a newer version of the same server is the most pain-free path to take. In the case of GlassFish migration, Payara Server is the natural successor, since **the source code is entirely derived from GlassFish** - it is a technology that you and your developers are already familiar with.

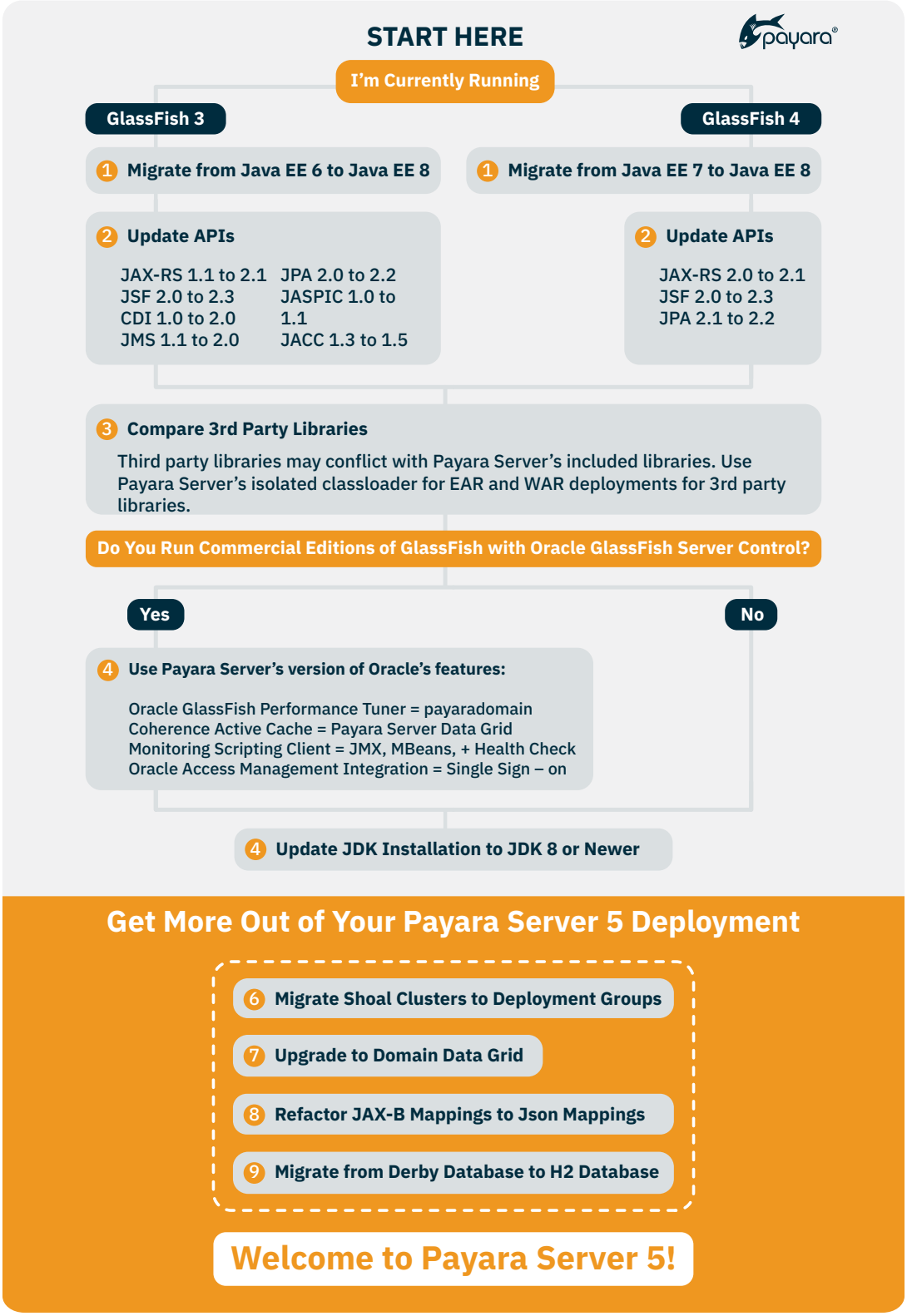
Enterprise quality production and development enhancements (detailed further in this guide), **easy** installation, **powerful** administration interface, **simple to use** tools and features, monthly patch release cycle, backed by emergency hot fixes to continually improve application **security** - these are just a few examples of the many great features of Payara Server that make it an ideal solution for current GlassFish users.

On top of that, Payara Server is entirely open source and production-ready. From the beginning, the ambition of Payara is to be as open and transparent as possible. All the source code and commits are publicly available, and community users are welcome to raise issues against the server (see GitHub) to help increase quality. Since the source code is evolutionary over what is available in GlassFish 4.x, the resulting container is a more natural choice for an upgrade than GlassFish 4.x itself.

Application servers are complex pieces of software and have many areas where edge-case bugs can hide. Payara Server not only releases with new features every three months to the community, but also provides interim “fix-only” builds for Payara Enterprise customers who value stability. This frequency of updates provides strong reassurance that any immediate issues are much more likely to be fixed rapidly. So, unlike the users of GlassFish, Payara Server users do not have to wait 12 (or more!) months for a new release with no guarantee that a fix will be included.

Providing a logical migration path from GlassFish Server Open Source Edition, Payara Server is the platform of choice for your Jakarta EE (Java EE) applications. Our vision is to optimise Payara Server Enterprise to make it the best application server for production Jakarta EE applications; with several support options ranging from Migration & Project Support, 24x7 mission critical support, and 10x5 business hours support delivered by the best middleware engineers in the industry.

Overview of Migration Steps



Migration From GlassFish 3.X

As far as the application server itself is concerned, there were very few major changes between GlassFish 3 (Java EE 6) and 4 (Java EE 7), so the biggest changes to consider are updated APIs which are provided by other projects and incorporated into GlassFish.

Payara Server includes several updated APIs and regularly incorporates updated minor versions of any external projects like, for example, Weld for CDI or Mojarra for JSF. There have been a couple of updated APIs in Java EE 7, which were released through GlassFish, however not all of these are significant changes.

APIs with significant updates in Java EE 7 include:

- JAX-RS, from 1.1 to 2.0
- JSF, from 2.0 to 2.2
- CDI, from 1.0 to 1.1
- JMS, from 1.1 to 2.0
- JPA, from 2.0 to 2.1
- JASPIC, from 1.0 to 1.1
- JACC, from 1.3 to 1.5

As always, backwards compatibility is of high importance to Java EE so, for example, a JMS 1.1 MDB should still be able to interact with a modern, JMS 2.0 enabled broker. Any code still using older APIs would be supported fully by the Payara support team in the case of any bugs found in Payara Server.

There are some brand-new APIs, as well as updated existing APIs, released in Java EE 7 in GlassFish which Payara Server has inherited. These include:

- Concurrency
- JBatch
- Websocket
- JSON-P

In addition to APIs which are part of the Java EE 7 specification, Payara Server also implements JCache which, though the specification has yet to be incorporated into Java EE, is a standard implemented by many vendors already. Payara Server provides JCache support by embedding Hazelcast and can therefore make use of dynamic clustering to store session data. These new APIs will be useful to developers, but are of no immediate concern for any migration plan. You may want to keep them in mind for future application development planning.

Migrating From GlassFish 4.X

In the vast majority of cases, **any application which runs on GlassFish 4.x will run on Payara Server**. The only potential hiccup may be where the application to be migrated happened to exploit some incorrect or invalid behaviour in GlassFish, which has subsequently been corrected or fixed in Payara Server.

In these cases, Payara Server Enterprise is fully supported, and the team is available to either assist in changing the app concerned, or providing some other workaround. The main things to consider when migrating from GlassFish 4.x to Payara Server is not what changes might be ***mandated*** - since the number is likely to be zero - the key consideration would rather be what changes are ***possible***, due to the extra features added, such as JCache.

Working With Third-Party Libraries

A significant source of pain for both users of GlassFish 3.x and 4.x is conflicting 3rd party libraries, which may be found in other frameworks like Spring or Grails (based on Spring). Conflicts can arise when you want to use a specific version of a library, but a different version is already included in Payara Server. A common example (until recently) was that Weld shipped an old version of Guava. Since Weld provides the CDI implementation for Payara Server, any application which included Guava would conflict with the server's version and problems would arise.

Payara Server now includes an isolated classloader for both EAR and WAR deployments, so that 3rd party libraries packaged with the application are preferred over those from the server.

There is also **enhanced control over implicit CDI scanning**. By default, as part of the specification, deployments are scanned for any CDI beans which are recognised by having "Bean Defining Annotations". Sometimes, this scanning can cause unwanted effects due to the annotations in 3rd-party libraries triggering the scan, so Payara Server now provides ways to disable implicit CDI scanning from a deployment and to explicitly disable scanning of specific JARs

How To Deal With Oracle Commercial Features

The main advantage of using the commercial edition of GlassFish over the Open Source edition is that you could enjoy the benefits of using the **Oracle GlassFish Server Control**, which is a suite of proprietary features that improves performance, enables fine-grained monitoring and enables

more secure and highly available production deployments. The Oracle GlassFish Server Control is composed of the following six features:

- Load Balancer Configurator Plugin
- Domain Administration Server Backup and Recovery
- Coherence Active Cache
- Monitoring Scripting Client
- Oracle Access Management Integration
- Performance Tuner

Since the Oracle GlassFish Server Control is only available to the commercial version; the source code to these features is licensed separately to the core GlassFish source code so Payara Server doesn't include these features.

If you are considering a migration from the commercial edition of GlassFish and you are using these features intensively, **there are tools and techniques available that can reproduce the functionality of these features and achieve the same objectives**.

Coherence Active Cache

Oracle GlassFish can integrate with Oracle Coherence (an in-memory data grid caching solution) as a replacement for the default in-memory HTTP state replication provided by Shoal.

Payara Server ships with Hazelcast (another in-memory data grid) out-of-the box. It is preconfigured and can be used as the sole clustering method with a simple change in the Admin Console. More information on the versatility of Hazelcast and how to use it can be found in the Payara Server documentation.

Monitoring Scripting Client

With the Monitoring Scripting Client, operations staff can write JavaScript scripts that enable monitoring probes that help track application performance characteristics, troubleshoot functional issues, follow the state of internal components and services and observe the application behavior in general.

Since Payara Server is already instrumented with JMX, a new feature was written to take advantage of the monitoring provided by existing MBeans. Users can now configure MBeans to write values to a log file to be consumed by log-based monitoring solutions, providing a convenient alternative way to consume JMX metrics in businesses which may have firewall restrictions on RMI. Additionally, we have introduced a Health Check service which can report on basic elements of the application server. This is part of a larger push towards a better set of operations features like request tracing and slow SQL logging. More detail on these features can be found below and in our documentation.

Oracle Access Management Integration

Oracle GlassFish Server includes a special security provider, implemented as a custom JASPIC (JSR-196, Java Authentication Service Provider Interface for Containers) module that allows enterprise applications to authenticate and take advantage of the Single Sign On functionality provided by its integration with Oracle Access Manager.

Since Oracle Access Manager is a proprietary product, there is no special integration provided for it, however Payara Server itself contains a simple Single Sign On solution that can be easily configured using standard JAAS mechanisms. When SSO is enabled, all web applications deployed on the same virtual server will share authentication state, so if a user logs in to a web application he will be implicitly logged for all other remaining applications that require the same authentication.

Performance Tuner

The Oracle GlassFish Performance Tuner is a special tool that can be used to automatically tune in the settings of a standalone instance or cluster configuration group by answering a series of questions. GlassFish will take your answers and suggest settings to you and give you the option of either applying all changes directly to the configuration or using the provided instructions to manually apply these changes.

There is, at the time of writing, no replacement feature for the performance tuner plugin however, since it is designed to be used in a one-off way to tune your domain for production, we have provided a new bundled domain and domain template called payaradomain which has some sensible defaults already configured with production in mind, and these go beyond performance tuning.

Load Balancer Configurator Plugin

The Load Balancer plugin is a utility which integrates with a web server (Oracle iPlanet Web Server, Oracle HTTP Server, Apache Web Server and Microsoft IIS) with GlassFish such that configuration can be managed from GlassFish rather than from the web server itself.

There is no replacement feature for this in Payara Server at the time of writing. We recommend to manually setup a load balancer using Apache Web Server with the ***mod_jk*** Tomcat connector or configure an *Nginx* web server with sticky sessions using its available plugins.

Domain Administration Server Backup and Recovery

Both the commercial and open source editions of GlassFish have the ability to backup and restore domains. The added value of the Oracle GlassFish plug-in is that it allows you to schedule when backups take place natively within GlassFish. A further benefit is that these scheduled backups do not necessitate stopping the domain while the backup takes place.

Since we saw the value of having the ability to schedule backups, we have taken the decision to create a new generic scheduler service which will be able to run asadmin commands at a specified time. This feature is currently still in development, and will be announced on our blog when it is finished.

Post-Migration

There are lots of new features added in Payara Server and not available in GlassFish which can help take your application into the future after you have successfully migrated.

Slow SQL Logging

A crucial production feature which allows you to easily detect when a query to the database exceeds a specific time. This enables you to drill down to the actual line of code impacting production performance enabling rapid triage and fix of production performance issues in the database or inefficient SQL code in your Java EE applications. Slow SQL logging is enabled for a specific datasource in the administration console in the advanced properties. When a query exceeds the configured threshold, a WARNING is output into the server log along with a full stack trace of the code that invoked the SQL, allowing rapid identification of the offending code.

Health Check Service

Another powerful tool that makes it easier for the Operation Teams to run Payara Server in production by periodically checking Host CPU Usage; Host Memory Usage; Payara Server's JVM Garbage Collections; Payara Server's JVM Heap Usage; CPU Usage of individual threads. If there is a problem with any of these metrics and they exceed a configurable threshold then a Warning, Error or Critical message is logged to the server's log file, enabling operations teams to rapidly detect problems or work out what happened after problems have occurred.

Request Tracing Service

Ideal tool for developers, it helps you to identify performance issues and their causes to successfully solve them. It allows you to trace requests through the server. All the following request types are traced when the service is enabled: REST (JAX-RS endpoints); Servlet (handling HTTP requests); SOAP Web Service Requests; WebSocket; execution of EJB timers; inbound JMS message handled by a message-driven bean; JBatch job is created; a new task is executed in a managed executor. Request Tracing comes with full asadmin and administration console integration, so you shouldn't have to go hacking around in the domain.xml . You can read more about the Request Tracing Service in our documentation.

Default Role/Group Mapping

GlassFish already has the ability to set default group to role mapping in the security configuration for the server, but there is no portable option that can be set in the deployment descriptors. Payara Server introduces an additional setting for deployment descriptors to explicitly enable or disable the default role mapping. This will mean that more configuration necessary for your application can be maintained within the deployment itself, rather than in the application server.

Hazelcast

Payara Server ships with Hazelcast but it is not enabled by default. Merely turning Hazelcast on will unlock a huge amount of power, but there are even more features that are available with a little further configuration, such as “Lite” nodes - a storage-disabled cluster member which can still access cache data without paying the JVM heap size cost. You can find out more about using Hazelcast with Payara Server in our documentation.

Payara Micro

Payara Micro is our microservice platform and works a bit differently to the traditional application server. It enables you to run war files from the command line without any application server installation. Payara Micro is small, <70 MB in size and incredibly simple to use. With its automatic and elastic clustering, Payara Micro is designed for running Jakarta EE (Java EE) applications in a modern containerized/ virtualized infrastructure, using automated provisioning tools like Chef, Ansible or Puppet. Using the Hazelcast integration each Payara Micro process will automagically cluster with other Payara Micro processes on the network, giving web session resilience and a fully distributed data cache using Payara’s JCache support. Payara Micro also comes with a Java API so it can be embedded and launched from your own Java applications - see how.

Other Production Features

Payara Server has **a full web based administration console** which is is fully featured and provides a single view of all clustered and standalone Payara servers. It also has a **fully scriptable Command Line Interface** for the administration of a Payara domain and a full REST based management console. Payara Server provides **a full set of monitoring JMX MBeans** enabling simple and rapid integration to any JMX based monitoring program to provide historical metrics and alerts. And on top of that, Payara Server **supports rolling upgrades of Java EE applications** and, if you choose the 24x7 support option with your Payara Enterprise subscription, can be **fully supported in production** and development around the clock with 1 hour response time for priority one issues.

How is Payara Server Enterprise Better Than GlassFish?

- **Monthly releases** – bug fixes, and patches to ensure security and stability of your production environment.
- **Migration & Project Support, 24x7, or 10x5 support options included** – with unlimited tickets, on-boarding support, and fully supported production binaries and ecosystem components.
- **10-year software lifecycle**
- **Enhanced reporting** and admin functions to create custom consoles or integrate with other JMX-based monitoring tools.
- **Optimised for mission critical production systems** in any environment: cloud, on-premises, or hybrid.
- **Security, monitoring and DevOps features** added, including Slow SQL Logging, Healthcheck Service, Asadmin Recorder and more.
- Support for modern **cluster topologies** through the use of Hazelcast.
- **Docker support** for rapid deployment of virtualised Java EE applications.
- **Payara Micro** for containerised application deployment.
- **Vibrant community** and high activity on the Payara GitHub Profile.

Feature	GlassFish 5.x	Payara Server Enterprise
License	Open Source	Open Source
Release frequency	Irregular	Monthly
Releases in 2019	1	22: 4 community stream, 12 stability stream, 8 feature stream
Security fixes	Infrequent	• Instant emergency & backported fixes for support customers • As soon as possible for community
Production support	⊗	✓
Migration & Project Support	⊗	✓
Component Upgrades (e.g. Tyrus, Mojarra)	Irregular	As needed
Supported IDEs	• Eclipse • Netbeans • IntelliJ IDEA	• Eclipse • Netbeans • IntelliJ IDEA • Visual Studio Code

Feature	GlassFish 5.x	Payara Server Enterprise
Caching tools		JCache, Domain Data Grid, Payara Scales (additional cost)
Automatic Clustering	✗	✓ via Hazelcast
Asadmin command recorder	✗	✓
Slow SQL logging	✗	✓
Healthcheck service	✗	✓
Request tracing	✗	✓
Monitoring logging	✗	✓
Microservices distribution	✗	✓ Payara Micro
MicroProfile support		Compatible with MicroProfile
Docker support	Community provided	Official images
HTTP & HTTPS port auto-binding	✗	✓ (Payara Micro only)
Generate Uber JAR	✗	✓ (Payara Micro only)
Production-tuned domain template	✗	✓
Upgrade tool	✗	✗ Coming soon
Jakarta EE Compatible	✓	✓
Runs on JDK 11	✗	✓

Conclusion

All things considered, **migrating from GlassFish to Payara Server should be an easy and painless process**. There may be minor hiccups in certain edge-cases, but the majority of the work that may need to be done will be reserved for making use of new features, provided by either Java EE 7 APIs, Payara Server itself, or even Java 8 language features.

If you're still at the early stages, however, the fact that Payara Server is still, operationally, largely similar to GlassFish means that **you can "try-before-you-buy" and see how easy it is to swap over simply by using Payara Server to start up your existing domain**. It is a simple and easy thing to do and can give you good early insight into just how easy it will be to get started with Payara Server.

For more complex migrations, the Payara Support team is available to assist in migrations, all included with a Payara Server Enterprise subscription.

sales@payara.fish **+44 207 754 0481** **www.payara.fish**