Migrating from Oracle® WebLogic to Payara® Server Enterprise 5 Migration Guide

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

Migration Guide

Contents

Introduction

Migrating from Oracle WebLogic Server to Payara Server Enterprise can be a simple and straight-forward process because both servers rely on the Jakarta EE (Java EE) specifications. This guide gives you an overview of Oracle WebLogic features, also outside the area of Jakarta EE, and what the equivalent feature of Payara Server Enterprise is or how you can achieve the same end result.

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, Oracle GlassFish fared well and became 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.

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 that 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 but with the added reassurance of regular releases, bug fixes, patches, and enhancements.

Payara Server evolved rapidly from just a "supported GlassFish alternative" to a fully Open Source product that can run Jakarta EE applications. It contains several unique features in the areas of monitoring and security, for example, and was the first run-time that integrates Java EE and Eclipse MicroProfile®. Payara Platform Enterprise is the fully supported, commercial version of Payara Platform, consisting of Payara Server and our microservices-ready Payara Micro both products including a choice of Migration & Project Support, 24x7, or 10x5 support options.

Releases

WebLogic server development started in 1995 which was later continued first by BEA Systems and since 2008 by Oracle Corp. It is used by mainly larger companies which use it also in combination with other software from Oracle.

The latest version of Oracle WebLogic Server is 12cR2 which introduced support for Java EE 7. Since the first release of 12cR2, several updates are made available as presented in the following table:

WebLogic Server 12cR2 (12.2.1.4)	September 27, 2019
WebLogic Server 12cR2 (12.2.1.3)	August 30, 2017
WebLogic Server 12cR2 (12.2.1.2)	October 19, 2016
WebLogic Server 12cR2 (12.2.1.1)	June 21, 2016
WebLogic Server 12cR2 (12.2.1.0)	October 23, 2015

This rather slow introduction of the new Java EE versions and long periods between the updates, is in high contrast with the release strategy of Payara Server. Community releases are made frequently available containing new features, fixes and security updates.

Payara Enterprise customers receive monthly releases and have additional benefits that go "beyond the help desk" aspect of support, including a 10-year software lifecycle and access to monthly updates and security fixes to maintain the stability and security of your environment.

Oracle WebLogic Components

As an Oracle Customer, you can choose between various Editions for the Oracle WebLogic Server:

- Oracle WebLogic Standard Edition
- Oracle WebLogic Enterprise Edition
- Oracle WebLogic Suite

These differ in the additional features that are made available to the Server outside of the Java EE world.

And Oracle WebLogic is also part of the Fusion Middleware suite, which also contains integration services, business intelligence, collaboration, and content management.

In this migration guide we will concentrate on the following components:

- Oracle WebLogic Server
- Oracle Coherence

And we will briefly discuss other components found in the Oracle WebLogic Enterprise Edition and what the migration options are after making the migration to the Payara Platform, in case you are using them today in combination with Oracle WebLogic.

WebLogic Core

The Oracle WebLogic Server is built around the Java EE specifications. Since both WebLogic and Payara are largely based on the same implementations of the Java EE specifications, the migrating process for Java EE applications from WebLogic to Payara should be straightforward.

When your applications are not using any implementation specific features and are only using the standard available features of the Java EE Specifications, the application will run without any modifications required. This is the preferred way of working when you are using the Java EE platform. This compatibility between implementations is one of the major strengths of it. Since the majority of implementations follow the specification, and there are compatibility tests performed to verify

this, it allows you to switch implementations and even switch to different servers from other vendors without the need to adjust your application.

You can run Java EE 8 applications on Payara Server Enterprise 5 as it is now a Jakarta EE 8 Full Profile Compatible product, where Oracle WebLogic is only Java EE 7 compatible. In general, all Java EE releases are backward compatible and thus it will not pose any problem to run a Java EE 7 application on Payara Server Enterprise 5. However, small issues, mostly in very complex applications, can cause some minor problems. As a reference, this is the list of the most important updated and newly added Java EE Specifications in Java EE 8:

Updated

- Contexts and Dependency Injection for Java EE (CDI) 2.0
- Java API for RESTful Web Services (JAX-RS) 2.1
- Java Servlet 4.0
- Java API for JSON Processing (JSON-P) 1.1Java API for WebSocket 1.1
- Java Persistence API (JPA) 2.2
- JavaServer Faces (JSF) 2.3
- Bean Validation (BV) 2.0

Newly Added APIs

- Java EE Security API 1.0
- Java API for JSON Binding (JSON-B) 1.0

The only real issue we have encountered today is that the addition of HTTP/2 and the push protocol in Servlet 4.0 is sometimes interfering with the JSF resource loading. But this HTTP/2 feature can easily be turned off in Payara Server 5 so that you still can use HTTP 1.1.

Oracle Coherence

Oracle Coherence is an in-memory data grid and distributed caching solution. Its main purpose is:

- Caching: Store data in a distributed (using different nodes) in-memory cache so that retrieval is faster than from application sources. This is a typical optimization for slowly changing data sets.
- ‘Analytics’: The stored data can be retrieved using sorting, aggregating and searching the data using distributed techniques.
- Transactions: Coherence can be used in a transactional way.
- Events: Events can be distributed between different nodes triggering listener code.

Oracle Coherence can also be used as JCache implementation so that it is an implementation of the specification JSR-107.

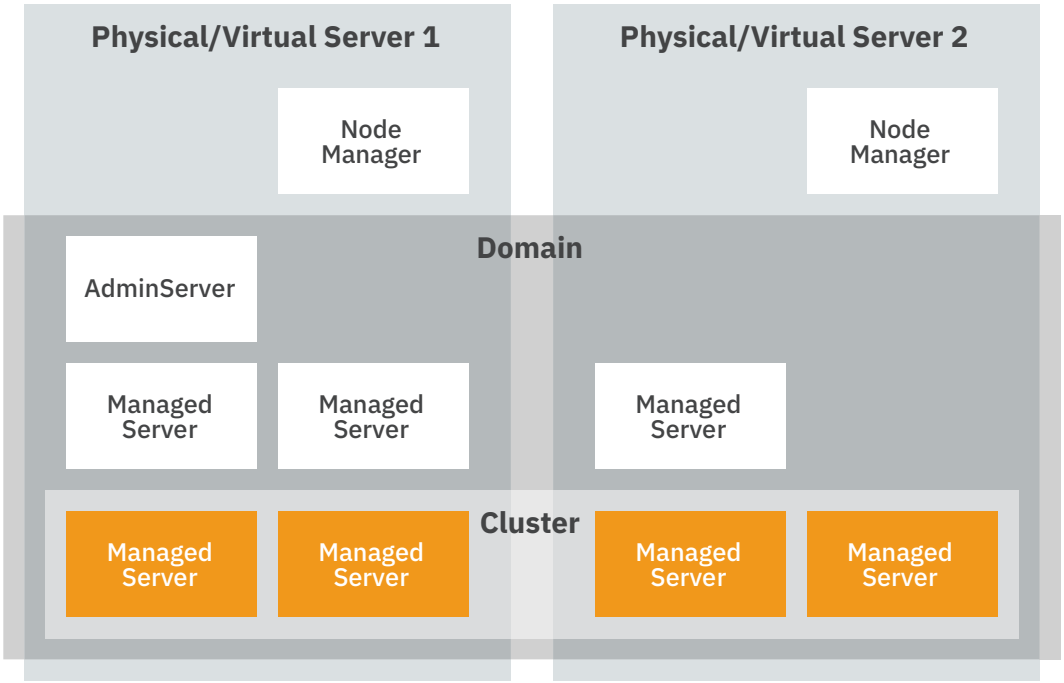
The same functionality is available within Payara Server with the help of the Domain Data Grid in Payara Server implemented using open source Hazelcast:

- Data can be cached using a manual operation (basically, change the statements *CacheFactory.getCache* to *hazelcastInstznce.getMap* for example) or use the JCache option backed by the Hazelcast library.
- Hazelcast has implementations for the collection objects based on the Java Map, List, and Set. So there is no specific import needed to synchronize data between different nodes. You can also make use of a special *ExecutorService* which allows you to run your task within the grid.
- Hazelcast has support for the Java EE Transaction specification. This means you can put data into the grid in a transactional way. It can even join in an XA transaction as a party.
- Just as in the Oracle Coherence case, events can be fired which are triggering listener code on remote nodes.

This means that all use cases for which you are using Oracle Coherence functionality can be mapped in a one to one way to the corresponding Hazelcast functionality which is included by default with Payara Server Enterprise. The default operation mode of Hazelcast is the embedded form (each Payara Server instance participates in the distributed Domain Data Grid) which is in contrast with Oracle Coherence where the recommended setup is a standalone deployment. The latter situation means that more instances need to be managed and monitored. Starting with Payara Platform 5.201, Payara Domain Data Grid contents are now encrypted and protected even at rest, not only during communication between members of the cluster/deployment group.

Clustering in WebLogic VS Payara Server Enterprise

An Oracle WebLogic domain consists of an Administration server and optionally one or more Managed servers which can be clustered in one or more clusters.



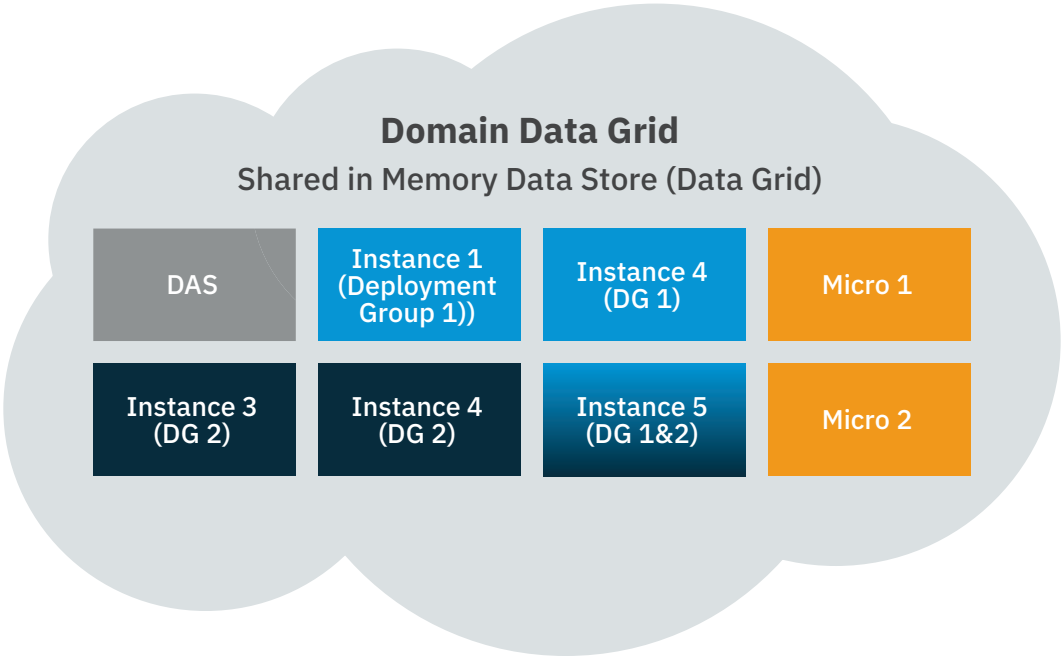
The terminology within the Payara Server is almost identical and there exists a one to one mapping:

- Administration server ↔ DAS (Domain Administration Server)
- Managed Server ↔ Payara Instance
- Cluster ↔ Cluster

But Clusters are deprecated in Payara Server 5 and replaced by a very flexible alternative which is called the Domain Data Grid. Within a cluster, a Managed Server or Payara Instance is dedicated to a cluster instance. They all run the same applications and have the same resource (like database connections) assigned to them. With a Deployment Group, you have very fine-grained control over which applications run on which instances.

Instead of assigning a server (or Instance, in Payara terms) to a Cluster, you assign individual deployed applications and resources (like JDBC resources) to a Deployment Group. You also assign Instances

to this Deployment Group so that each instance runs a certain set of applications and resources according to the membership of the Instance to Deployment Groups. With this architecture, you have a very flexible assignment and usage of your Instances:



Dynamic Clusters

Oracle WebLogic has support for what they call *Dynamic Clusters*. The configuration of a Managed Server, which needs to be preinstalled, is then taken from a template and can be added to the cluster to increase the throughput of the applications.

Payara Server has the same functionality with the *Deployment Groups* features. When you define a new Payara Instance, the configuration is always taken from a template. All the Payara software is even installed for you through SSH, there is no need to have this preinstalled. Once the Instance is created within the Domain configuration, you can assign it to one or more Deployment Groups what result in the installation of all the artifacts (Applications, Resources like JDBC connection pools) specified in that Deployment Group on that newly added instance.

Operations

With the WebLogic Scripting Tool (WLST) you can manage your Oracle WebLogic servers from a script. Besides commands for interacting with WebLogic components, it also has Python integrated to make your scripts more versatile. This integration allows you to write intelligent scripts that decide which commands to perform based on some conditions or checks.

Payara Server has also a command Line tool, *asadmin*. You can manage and operate all aspects from your Payara Server installation with this tool. There is no integration with another scripting language but it can be integrated into OS operations like BASH scripts which provide the needed intelligence if required.

Moreover, the Asadmin Recorder is a unique feature of the Payara Platform which can help you write the script for setting up your environment. It records the steps you perform within the Web Console and write this as commands to a file. This file can then be used to setup your server to the same state as you did manually through the Web Console.

Just like Oracle WebLogic, Payara Server Enterprise has a Rest API to perform all the management operations, and it can be used if you need to integrate or control Payara Server from another environment.

Monitoring

The WebLogic Diagnostics Framework (WLDF) is a monitoring and diagnostic framework. It provides WebLogic administrators with statistics for subsystems like number of Threads active and number of Sessions. Information around this monitoring can be written to various notifications channels like Log files, SMTP, SNMP, JMX, and JMS.

Oracle WebLogic has also support for more advanced monitoring features like Self-Health Monitoring. It detects if a Managed server needs to be restarted due to some failures or SQL Leak Connection detection.

Payara Server Enterprise has an even wider variety of monitoring and notifications channels, including a monitoring console introduced in Payara Platform 5.194 (Payara InSight) and updates to alerts and watches, health checks, request tracing, slow SQL alerts in Payara Platform 5.201.

Just as in Oracle WebLogic, various statistics are available in multiple formats and exposes in various ways:

- As JMX Beans
- As Rest endpoints
- In Prometheus ready format

But there are also many notification channels available. On top of the notification channels available within Oracle WebLogic, Payara Server also supports:

- CDI
- Slack
- New Relic
- DataDog

Just as Oracle WebLogic, the Payara Platform has SQL Leak Connection detection, when a connection isn't returned properly to the pool, but it also offers other Health Checks like Stuck Threads, Total Memory and Heap Memory checks, and CPU usage, just to name a few of them.

Request Tracing

With Request Tracing in Payara Server, one can identify which call to a RESTful endpoint takes a lot of time to process. When multiple endpoints are called in a chain to perform the operation, one can identify which of those calls is the bottleneck. With WebLogic, you can only use the provided functionality of the Jersey framework for this purpose. But Payara goes much further when it has implemented this feature.

The Payara Request Tracing does not just trace the call to the REST endpoint, but takes note of the time it spends in EJB bean methods, CDI bean methods and Servlets amongst others when they have the @Traced annotation. All this information is written through the Notification Service so that finding what part of the request is the bottleneck becomes much easier.

Java SE tools

Depending on the Edition of Oracle WebLogic Server you are using, you also have access to some Java SE products from Oracle. The most notable ones are:

- JRockit JVM
- Flight Recorder
- Mission Control

The unique features of the JRockit JVM are migrated into the Oracle JDK and some of them even to the OpenJDK code base. Also, the Flight Recorder and Mission Control tools were recently made open sourceand since Java 11 is available to everyone for free, there is no real need for an Oracle subscription for these items anymore.

There is an added benefit for Payara Enterprise Customers, in that it includes JDK support by Azul when using their JVM. In the case you experience a JVM related issue, you can rely on their support to find a suitable solution or get a fixed build. And as part of our long-term support for a specific version, JDK 8 security updates will also be available from Azul for our customers.

Protocols

Oracle WebLogic uses its own t3 protocol for handling the JNDI remote calls but also for internal communication. It is TCP/IP-based and works over a single connection from client to server, so it has no issue with a Network Address Translation (NAT) component in the network path.

The Payara Platform uses the RMI-IIOP protocol which is defined within Java EE. RMI-IIOP requires a two-way connection in that the client publishes its IP address that the server connects to. Besides being the standard, it also allows you to communicate with any compatible remote system since IIOP is a standard within the industry as a whole not only within Java. Within WebLogic, RMI-IIOP can be activated if needed.

> ***It's important to note that if you are using the remote EJB functionality within a container environment in WebLogic, RMI-IIOP has issues with NAT (Network Address Translation). When NAT translates the addresses, the IP address the client is aware of is not an IP address the server can connect to. As a result, making a connection to a remote EJB deployed within Payara Server in a Container environment like Docker will fail unless you configure the network where both the client and server share the same network.***

Deployment

With Oracle WebLogic, you have the option to change the deployed application or change the descriptors (like *web.xml* or *ejb-jar.xml*) during deployment. According to the documentation of WebLogic, this feature can be used to correct some mistakes after deployment (like the wrong path for a servlet or correcting an EJB name) or change the deployed application according to the environment (test, production).

The equivalent within Payara Server works with variable replacement. Instead of changing values by defining them in an XML file (as the Deployment Plan feature does), we clearly indicate that the

value at run-time will be taken from some configuration. These variables can be placed in many locations like (see the documentation for a full description):

- Descriptor files like web.xml and faces-config.xml
- Annotations like @DataSource and @WebServlet
- JNDI lookup values

The actual value can come from various sources like:

- System properties
- Environment variables
- Aliases like password aliases defined within Payara
- And recently also from MicroProfile sources

Production redeployment is the Oracle WebLogic feature which allows you to deploy the old and new version of the application side by side. The old version still receives the requests from the clients which were active during the installation of the new version. Over time, all clients will be using the new version and a gradual switch, without downtime, is performed.

Payara Server has no built-in feature to perform such an upgrade of your application. However, multiple versions of an application can be deployed side by side, with one of them serving the requests. When the HTTP session information is stored within the Domain Data Grid, a rolling upgrade is possible to switch the active application version, keeping user information. This approach has more restrictions then the WebLogic procedure but achieving blue-green deployments is still possible with some external products like Loadbalancer and Kubernetes if needed.

Cloud

Containerized environments are becoming more important these days. Oracle WebLogic has its own set of Docker images and various examples can be found on how an Orchestration tool like Kubernetes can be used in combination with the WebLogic Server.

Payara Server is no exception in the cloud evolution. We have also official Docker images on DockerHub for the Payara Platform Community Edition and Payara Enterprise specific Docker Images in our Nexus repository (under the repo ‘payara-docker’ for Payara Enterprise customers.

We also created Payara Micro, a specialized packaging of Payara Server which is optimized for cloud and micro-service environments. Payara Server 5.192 introduced Docker nodes and instances to provide better native integration for Payara Server with Docker by mimicking the behaviour of the pre-existing SSH, DCOM, or CONFIG node instances.

With Payara Micro, (see also under the section of innovation), you can easily create a setup which is maintained by orchestration tools like Kubernetes or Docker Swarm, and even makes use of the functionality of these tools like DNS management within Kubernetes.

Due to this correspondence, you can run Payara Server on cloud providers like Azure, Amazon, Jelastic, but also on Oracle Cloud just as you can do with WebLogic Server.

IDE Support

All major IDEs like IntelliJ® IDEA, NetBeans, and Eclipse®, have support for both servers. So, you can use your favorite IDE to develop your application. There exists also a GlassFish plugin for JDeveloper® if that is your IDE of choice. The plugin allows to start, stop and debug a GlassFish domain but since Payara Server has its roots in GlassFish, this plugin also works for Payara Server.

Payara Server also includes a VSCode plugin and the team is currently working toward having dedicated IDE plugins for all major IDEs.

WebLogic Server has support for FastSwap. This allows the server to pick up the changed classes without the need for a redeployment of the application. Payara Platform 5.201 introduced the Hot Deploy feature to help developers run and test applications immediately after making changes to its sources without restarting the Server or manual redeployment to maximize your productivity.

Support

As already mentioned, Payara Server is a complete open source product. You can download the Payara Server Community Edition from the download section of the website and you can find the complete source code on GitHub. However, for companies using Payara Platform in production, you should obtain Payara Enterprise for a secure, stable, and fully-supported software solution, including the following benefits:

- Access to the monthly bug fix and patch releases to keep your server up to date and secure at all times.
- Support for a particular version up to 10 years in case you don’t want to or can’t update every month to the latest version.
- Support for your production and development issues directly from our engineers with a guaranteed response time(depending on the support level, within 1 hour, 24/7).
- Ability to request new features to adapt Payara Server Enterprise to your needs.

Innovation

The team behind Payara Server is also actively working on the future of Java Enterprise applications: Payara Services is an Eclipse Foundation Solutions Member and a Strategic Member of the Jakarta EE working group, shaping the future of Eclipse Jakarta EE™ along with future versions of the platform. Payara Server Enterprise is Jakarta EE certified.

We are also helping related technologies and frameworks. For example, we are also actively involved in the Eclipse MicroProfile specifications. We are not only defining the specifications together with the other involved parties, but Payara Server was one of the first servers which combined the MicroProfile implementations and the ability to run Java EE applications in one run-time. This gives you the ultimate flexibility to choose the right mix of dependencies for your use case and also allows you to implement a strategy of gradual migration from the Java EE platform to a more micro-service alike application structure.

For these microservice applications, Payara Server has a special packaging called Payara Micro. It provides additional specifications like concurrency and the MicroProfile implementations in a single jar file. This Hollow Jar deployment model (where the server is in one single jar and runs your application packaged as a war file) has many advantages over the Fat Jar approach. Each single application code change no longer results in the replacement of the layer in a Docker environment which contains also the server run-time. This reduces Image sizes, bandwidth usages and deployment times.

But enhancements are not only implemented outside the Java EE area. For example, Payara Server Enterprise has many additions in the area of the application security. Using the Security API (added in Java EE 8) there are various authentication and authorization schemes implemented like OAuth2, OpenIdConnect, and JWT tokens.

Why Payara Server Enterprise?

Payara Server is notably better then WebLogic in the following areas:

Production-Ready and Stable With Full Support

When you download Payara Platform Enterprise, you're downloading and adopting a production-ready version of Payara Server or Payara Micro. With a 10-year software lifecycle and monthly release schedule for Payara Enterprise customers including bug fixes and patches, the Payara Platform offers security and stability without the need for upgrading every year or two. Payara Enterprise customers enjoy fast issue resolution directly from our global Engineers for both production and development issues and priority on new features requests.

Cloud-Native and Aggressively Compatible

Payara Server Enterprise is optimized for production environments in any environment: cloud, on-premises, or hybrid. As Payara Services Ltd is a Solutions Member of the Eclipse Foundation and Strategic Member of the Jakarta EE working group, you'll find the Payara Platform is positioned for future compliance with Jakarta EE, and is among the first application services to earn Jakarta EE certification. In addition, the Payara Platform is container-friendly, including Docker and Kubernetes, and compatible with services you're already using such as Microsoft Azure™, Amazon AWS, and MicroProfile. Our subscription costs are cloud-user friendly and stay the same whether you're running your environment on-premise or on a public cloud.

Open Source Software with a Future You Help Define

The Payara Platform is derived from GlassFish Open Source the Java EE reference implementation and available for immediate download. Payara Enterprise customers are invited to customer advisory calls to help drive evolution of the Payara Platform, as new features and enhancements are developed to meet the needs of customers. Being open source, the Payara User Community allows you to submit your ideas, feedback, and collaboration to ensure Payara Server Enterprise is the best option for production Jakarta EE applications and Payara Micro Enterprise is the best option for running containerized Jakarta EE applications in a modern virtualized infrastructure.

Feature	Oracle® WebLogic 12R2 (enterprise edition)	Payara® Server 5
License	Proprietary	Open Source
Release Frequency	Irregular	Quarterly for the Community, Monthly for Customers
Production Support	Yes	Yes
Component Upgrades	Irregular	Quarterly
Supported IDEs	Eclipse® / NetBeans® / IntelliJ® IDE / JDeveloper®	Eclipse® / NetBeans® / IntelliJ® IDE / JDeveloper® / VSCode
Support for Java EE Applications	Java EE 7	Java EE 8
Caching Tools	Oracle Coherence / JCache	Hazelcast / JCache
Clustering	Every node runs same apps	Every nodes runs same app or fine grained assignment
Scripting tool	WSLT	asadmin

Feature	Oracle® WebLogic 12R2 (enterprise edition)	Payara® Server 5
Scripting extensions	Python scripts	OS / BASH scripts
Asadmin command recorder	No	Yes
Deployment Flexibility	Deployment plan (XML based)	Variables (System, environment, MicroProfile config values)
Notification channels Monitoring	• JMX Beans • Log files • SMTP • SNMP • JMS	• JMX Beans • Log files • SMTP • SNMP • JMS • CDI • Slack • New Relic • Datadog
Slow SQL Logging	No	Yes
Health Check Service	Yes	Yes
Request Tracing	JAX-RS (Jersey) only	JAX-RS, CDI, EJB, Servlet, ...
OAuth2/OpenId Connect support	Requires Fusion Middleware or custom dev	Natively supported (through Security API)
Micro Services Distribution	No	Payara® Micro
MicroProfile support	No	Yes
Uber-jar and Hollow Jar	No	Yes
Docker Support	Yes	Yes
Cloud Providers Support	Oracle Cloud, Amazon, Microsoft Azure™, Jelastic	Microsoft Azure, Jelastic, Amazon, Oracle Cloud
Zero Deployment downtime	Parallel versions with gradual switch	Parralel versions with one time switch
Upgrade tool	Yes	No (Coming soon)
JDK Support	Yes	Yes (through Azul® JVM)

Where to Get Migration Help

Get Payara Platform Enterprise

If you're running the Payara Platform in production but are not yet using Payara Enterprise, obtaining the proper licensing for your enterprise environment provides several benefits. Included with a Payara Enterprise subscription is the choice of support options, including Migration & Project Support to access Payara Platform experts and accelerate your project delivery while reducing risks and costs and helping you deliver your project on time and within budget, or 24x7 or 10x5 support options.

Learn More About Payara Enterprise.

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