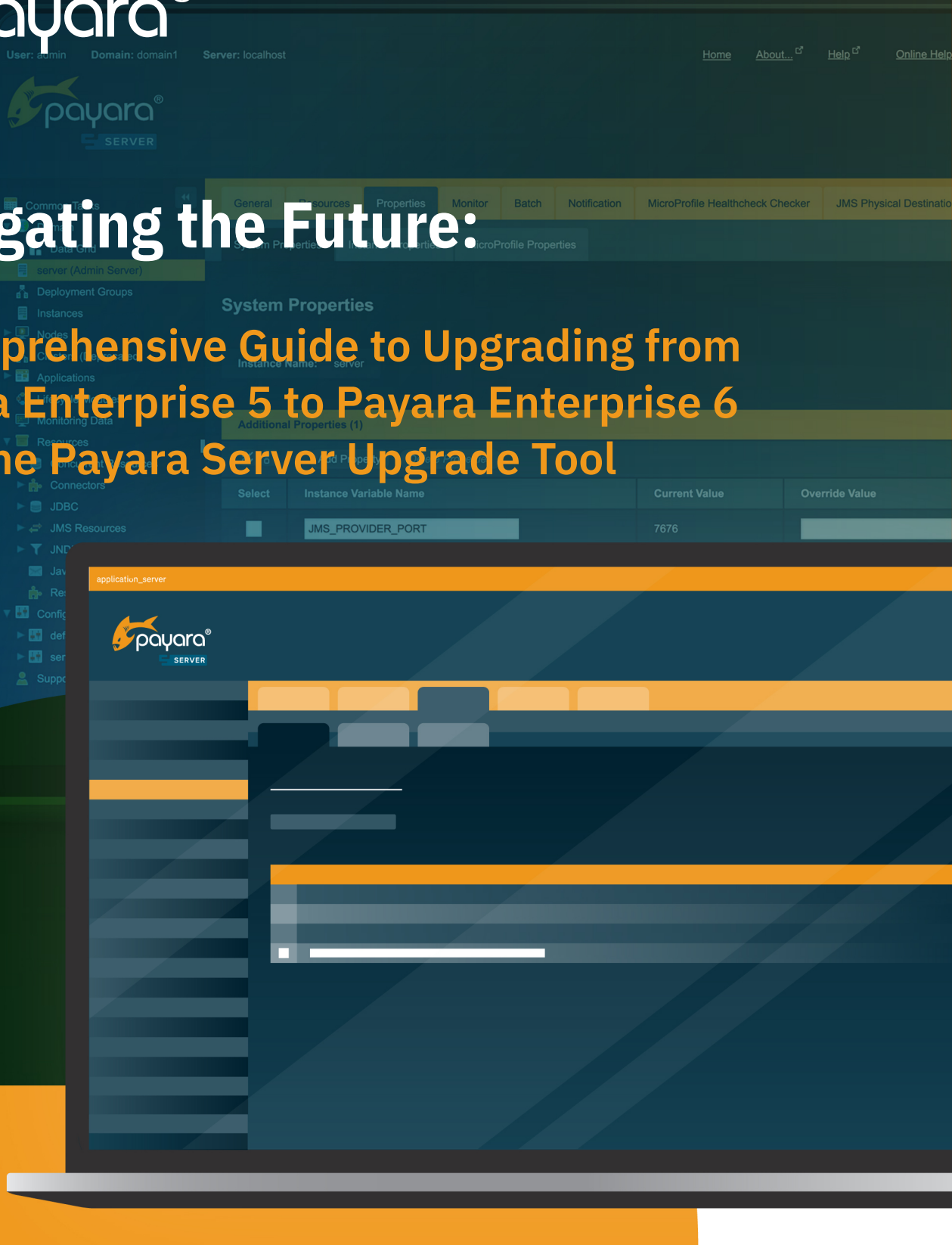




Navigating the Future:

A Comprehensive Guide to Upgrading from Payara Enterprise 5 to Payara Enterprise 6 with the Payara Server Upgrade Tool



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

User Guide

Contents

Guide Updated: **November 2023**

Pre-upgrade Checklist	2
Compatibility Check	2
Dependencies	2
Data Backup	2
Rollback Plan	2
Documentation	2
Downtime Assessment	3
Team Training	3
Testing Environment	3
The Upgrade	3
Staging The Upgrade	3
The upgrade-server command	4
Offline Upgrade	5
Online Upgrade	6
Applying The Upgrade	8
Rolling Back the Staged Upgrade	9
Cleanup	9
Follow Up Upgrade Tasks	10
Distributed Environments	11
Post Upgrade Monitoring	12
Application Performance	12
Error Logs	12
Resource Usage	12
User Experience	13
Database Health	13
Security	13
Network Traffic	13
Failover and Redundancy	13
Data Integrity	13



Custom Scripts and Cron Jobs.....	13
Stakeholder Communication.....	14
Support Channels.....	14
Customer Feedback.....	14
Conclusion.....	14

Guide Updated: **November 2023**

In a world where technology never sleeps, staying ahead is not just an advantage; it's a necessity. This is especially true for organisations running enterprise-level applications on Java EE 8 through Payara Enterprise 5. While this platform has undoubtedly served you well, the release of Jakarta EE 10 and Payara Enterprise 6 marks the beginning of a new era in Enterprise Java development.

One of the biggest benefits of Jakarta EE 10 is that it is more modern and streamlined than previous versions. It is based on the latest Java standards and has been optimized for modern cloud-native applications. This makes it easier for you to build and deploy modern, scalable, and secure Java applications.

Another key benefit of Jakarta EE 10 is that it is more flexible and extensible than previous versions. It is designed to be modular through its various profiles and provides several new APIs. In addition to these general benefits, Jakarta EE 10 also offers several specific benefits for enterprises, such as:

- **Improved performance and scalability:** Jakarta EE 10 has been optimized for performance and scalability, making it ideal for building high-traffic enterprise applications.
- **Enhanced security:** Jakarta EE 10 comes with a number of new security features, such as support for OAuth 2.0 and OpenID Connect. This makes it easier for enterprises to build secure Java applications.
- **Simplified development:** Jakarta EE 10 provides several new features that simplify development, such as support for CDI 4.0. This makes it easier for enterprises to develop maintainable applications.

Overall, Jakarta EE 10 is a significant release that offers several benefits for organisations like yours. It is more modern, streamlined, flexible, extensible, and secure than previous versions.

The task of moving from one production-optimised environment to another can seem daunting. What if something goes wrong? What about downtime? What's the impact on ongoing projects? These are all genuine concerns that must be considered as part of the overall Payara Enterprise 5 upgrade process.

In this guide, we will go through the process of upgrading from Payara Enterprise 5 to Payara Enterprise 6. You'll learn how to leverage the features of the Payara Server Upgrade Tool bundled in the latest release of Payara Enterprise 5 to ensure a seamless transition, without compromising the stability or performance of your enterprise applications. We will walk you through everything from the initial assessment phase to the actual upgrade, and even offer some best practices for post-upgrade monitoring. Let's begin your journey to the next frontier of enterprise Java.

Pre-upgrade Checklist

Before the actual upgrade, it is important to run through the following task list to make the entire upgrade process as painless as possible.

Compatibility Check

Verify that your current applications, libraries, and services are compatible with Payara Enterprise 6 and Jakarta EE 10. If they aren't, you'll need to factor in additional time to upgrade them . To see possible breaking changes in Jakarta EE 10 that impacts the use of Java EE 8 APIs in your application, check out the Payara Upgrade Advisor Tool. You can still deploy your existing Jakarta EE 8 applications to the fully upgraded Payara Enterprise as it ships the deployment transformer tool. This tool will convert your Jakarta EE 8 application to Jakarta EE 10 at deployment. The caveat is that the deployment transformer tool is a best effort tool and is not guaranteed to be available in major future version releases. As such, you should factor in gradually upgrading your existing applications to Jakarta EE 10.

Dependencies

Make a list of all external dependencies your application has . Check if these dependencies have versions that are compatible with Jakarta EE 10 . Also ensure your current Payara Enterprise 5 instance is up to date.

Data Backup

Back up your data before an upgrade. Create a copy of your current server installation as a checkpoint for easy clean slate restore.

Rollback Plan

Have a rollback plan in place. If something doesn't work as expected, you need a quick way to revert to the old version without affecting your users. Again, back up your current installation.

Documentation

Take note of your current configurations, environment variables, and custom scripts. You'll need these for reference, and possibly for reconfiguration after the upgrade.

Downtime Assessment

Estimate the expected downtime during the upgrade and communicate this to stakeholders. Plan the upgrade during off-hours, if possible, to minimise impact on users.

Team Training

Make sure your team is up to date with [Jakarta EE 10 features](#) and Payara Enterprise 6 functionalities. Training sessions on the latest Jakarta EE 10 features and API changes can help your team make the most of the latest release.

Testing Environment

Before upgrading your production environment, perform the upgrade on a testing or staging environment. This will help you identify any issues that you'll need to address in the actual upgrade. This is especially important if you have made heavy customizations to the various server configuration files, for example the domain.xml file . Also, thoroughly test the functionality of your migrated applications to confirm they work as expected.

The Upgrade

Upgrading from Payara Enterprise 5 to 6 can be categorized into three broad parts. The first part entails staging the upgrade, the second part is applying (or rolling back) the staged upgrade and cleaning up after the upgrade. The upgrade command should be run on a fully stopped server instance. You can run the following command to stop the domain.

```
>./asadmin stop-domain production
```

Staging The Upgrade

The Payara Upgrade Tool is bundled with the latest release of Payara Enterprise 5, 5.54.0 at the time of this writing . The tool supports upgrading from the latest Payara Enterprise 5 to the latest Payara Enterprise 6. As such, you will need upgrade your existing installation to the latest Payara Enterprise 5. As of this writing (September 2023), the upgrade tool will upgrade Payara Enterprise 5.54.0 to 6.5.0. The upgrade tool command is run through the asadmin command. The commands shown in the next section should be executed in the ~/payara5/bin folder, where the asadmin utility is.

The upgrade-server command

The full upgrade command to migrate the domain to 6.X is as follows:

```
asadmin [asadmin-utility-options] upgrade-server [--username <username>]
  [--distribution <distribution(default:payara)>] [--version <version>]
  [--stage[=<stage(default:false)>]] [--usedownloaded <usedownloaded>]
  [--domaindir <domaindir>] [-?|--help[=<help(default:false)>]]
```

Option	Type	Description	Default	Mandatory
--username	String	The username for the Payara Nexus server.		Yes, unless --usedownloaded is specified
--version	String	Specifies the version number of the new version of Payara to use.		Yes, unless --usedownloaded is specified
--distribution	Enum	Specifies the distribution of Payara Server to download. This can be payara, payara-web, payara-ml, or payara-web-ml.	payara	No
--stage	Boolean	Determines if the upgrade should be installed in-place, or staged into "x.new" directories. Scripts are provided that can be used to apply and rollback the staged install. This option is automatically enabled when running the command on Windows.	False on Unix, True on Windows	No

Option	Type	Description	Default	Mandatory
<code>--domaindir</code>	String	The directory containing the domains. The domains in this directory are backed up, and their config is used to determine the nodes which will also be upgraded.	<code>\${as-install} /domains</code>	No
<code>--usedownloaded</code>	String	The path to the local Payara Server zip archive to use for upgrading instead of downloading from the Payara Enterprise repository. When this parameter is specified, the <code>--username</code> , <code>--password</code> , and <code>--version</code> parameters are not required.		No

Offline Upgrade

You can upgrade using a local Payara Server 6 ZIP download. In that case, your upgrade command will be as follows.

```
>./asadmin upgrade-server --stage true --usedownloaded /path/to/latest/payara-6.5.0.zip
```

The `--stage` option is `true` by default if this command is run on Windows. However, even if you are executing the upgrade on Windows, I suggest to explicitly set it. This is the option that will hold the upgrade in an intermediate state, literally stage it in different files and folders for easy manual inspection.

The `--usedownloaded` takes an absolute path to a zipped Payara Enterprise 6.X installation . As version 6.5.0 is the latest at the time of this writing, I give it an absolute path to the downloaded zip.

Running the offline upgrade command will look as shown below.


```
~/development/apps/servers/payara-upgrade/payara5/bin$ ./asadmin upgrade-server --stage true --usedownloaded /home/seera/development/apps/servers/payara-upgrade/payara-6.5.0.zip
Distribution validated as payara
Do not interrupt the upgrade process, do not shutdown the server or computer.
Extracting zip file /tmp/payara6620763177450943209.zip
Backing up domain configs
Executing command: backup-domain production
Backed up production at Tue Aug 29 18:01:41 GMT 2023.
Executing command: backup-domain domain1
Backed up domain1 at Tue Aug 29 18:01:45 GMT 2023.
Deleting old server backup if present
Deleted old server backup
Deleting staged install if present
Deleted staged install
Moved files to old
Copying extracted files
Extracted files copied
Fixing file permissions
Ignoring NoSuchFileException for glassfish/h2db directory under assumption this is a distribution without this duplicate directory. Continuing fixing of permissions...
File permissions fixed
Upgrade successfully staged, please run the applyStagedUpgrade script to apply the upgrade. It can be found under payara5/glassfish/bin.
Command upgrade-server executed successfully.
```

Online Upgrade

You can omit the `--usedownloaded` version, however. In that case, the upgrade tool will need to be downloaded directly from the official Payara Enterprise Nexus server. For that, the upgrade command becomes:

```
> ./asadmin upgrade-server --stage true --username myPayaraNexusUserName --distribution payara --version 6.5.0
```

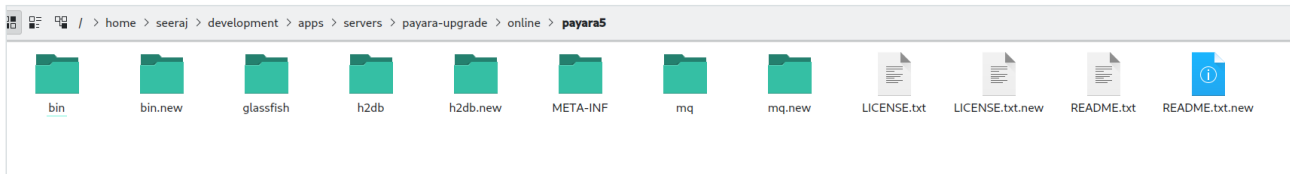
The `--username` takes your Payara Nexus server username. You could omit the `--version` and it would default to the latest Payara Enterprise 6 version at the time you run the upgrade.

When you run the command, the upgrade tool will request you to enter your password, and then proceed to download a copy of the latest Payara Enterprise 6, or the version explicitly specified by the `--version` argument.

Running the online upgrade command will look as shown below.

```
~/development/apps/servers/payara-upgrade/online/payara5/bin$ ./asadmin upgrade-server --stage true --username lsaeed --distribution payara --version 6.5.0
Enter AS_ADMIN_NEXUSPASSWORD password>
Distribution validated as payara
Do not interrupt the upgrade process, do not shutdown the server or computer.
Downloading new Payara version...
Downloading from https://nexus.payara.fish/repository/payara-enterprise-downloadable-artifacts/fish/payara/distributions/payara/6.5.0/payara-6.5.0.zip
Extracting zip file /tmp/payara12382473858361039372.zip
Backing up domain configs
Executing command: backup-domain production
Backed up production at Thu Aug 31 15:17:06 GMT 2023.
Executing command: backup-domain domain1
Backed up domain1 at Thu Aug 31 15:17:06 GMT 2023.
Deleting old server backup if present
Deleted old server backup
Deleting staged install if present
Deleted staged install
Moved files to old
Copying extracted files
Extracted files copied
Fixing file permissions
Ignoring NoSuchFileException for glassfish/h2db directory under assumption this is a distribution without this duplicate directory. Continuing fixing of permissions...
File permissions fixed
Upgrade successfully staged, please run the applyStagedUpgrade script to apply the upgrade. It can be found under payara5/glassfish/bin.
Command upgrade-server executed successfully.
```

Whether you run the upgrade the command on or offline, the result is that the upgrade gets staged. All the staged files end with the `.new` suffix. For example, the `payara5` server folder, after the staging looks as follows.



Where all the .new files are the upgrade tool staged files. For example, the /payara5/glassfish/config/branding/glassfish-version.properties file looks as follows:

```
product_name=Payara Server Enterprise
brief_product_name=Payara Server
abbrev_product_name=Payara
major_version=5
minor_version=54
update_version=0
build_id=850
version_prefix=
version_suffix=#badassfish
default_domain_template=appserver-domain.jar
admin_client_command_name=asadmin
distribution=payara
```

While that of /payara5/glassfish/config/branding.new/glassfish-version.properties looks like this:

```
product_name=Payara Server Enterprise
brief_product_name=Payara Server
abbrev_product_name=Payara
major_version=6
minor_version=5
update_version=0
build_id=11
version_prefix=
version_suffix=#badassfish
default_domain_template=appserver-domain.jar
admin_client_command_name=asadmin
distribution=payara
```

With the staging done, we are now ready to apply the upgrade.

Applying The Upgrade

You can apply the staged upgrade by running the `applyStagedUpgrade` command. This tool is a standalone utility that is found in the `/payara5/glassfish/bin` folder. Navigate to that folder and run the `applyStagedUpgrade` as follows:

```
> ./applyStagedUpgrade
```

This command will apply the staged upgrade and make a backup of your domains. Because we didn't specify a `-domainidir` argument to the upgrade tool, the `applyStagedUpgrade` will back up the domains in the default domains folder `/payara5/glassfish/domains`.

Running the command results in the following.

```
~/dev/a/se/payara-/payara5/glassfish/bin ./applyStagedUpgrade ✓
Server installation path: ..
Moving common to old
Moving staged common to expected location
Moving config/branding to old
Moving staged config/branding to expected location
Moving config/osgi.properties to old
Moving staged config/osgi.properties to expected location
Moving h2db to old
Moving staged h2db to expected location
mv: cannot stat './h2db.new': No such file or directory
Ignoring error moving missing staged h2db directory to old, assuming this is a distribution without the duplicate di
rectory
Moving ../h2db to old
Moving staged ../h2db to expected location
Moving legal to old
Moving staged legal to expected location
Moving modules to old
Moving staged modules to expected location
Moving osgi to old
Moving staged osgi to expected location
Moving lib to old
Moving staged lib to expected location
Moving ../README.txt to old
Moving staged ../README.txt to expected location
Moving ../LICENSE.txt to old
Moving staged ../LICENSE.txt to expected location
Moving ../mq to old
Moving staged ../mq to expected location
Moving bin to old
Moving staged bin to expected location
Moving ../bin to old
Moving staged ../bin to expected location
Moving domains/production/osgi-cache to old
Moving staged domains/production/osgi-cache to expected location
Distribution validated as payara
Reinstalling nodes for domain production
Node directory /home/seeraj/development/apps/servers/payara-upgrade/payara5/glassfish/nodes/localhost-production not
found, skipping.
Reinstalling nodes for domain domain1
Node directory /home/seeraj/development/apps/servers/payara-upgrade/payara5/glassfish/nodes/localhost-domain1 not fo
und, skipping.
Command reinstall-nodes executed successfully.
~/development/apps/servers/payara-upgrade/payara5/glassfish/bin □
```

The staged upgrade is now applied, and the domains in the server have been backed up to a folder called backups in each respective domain folder. For example, the production domain backup can be found in `/payara5/glassfish/domains/production/backups/production_2023_08_29_v00001.zip`. The zip file contains the state of the domain before applying the staged upgrade.

Rolling Back the Staged Upgrade

You can roll back the applied upgrade by using the `rollbackUpgrade` tool found in the same folder as the `applyStagedUpgrade` tool. It moves the `.old` installation back into the expected "current" location, and the applied upgrade back into `.new` directories. It will then rollback the nodes of the domains in the default domain dir, or the domains in the directory provided using `--domaindir`

Cleanup

With the staged upgrade applied, you can cleanup the whole process by running the `cleanupUpgrade` command. When you applied the staged upgrade, several `.old` folders were created. These folders contain the files of the "old" server, in this case Payara Enterprise 5 before the application of the staged upgrade. This script cleans up by removing those folders. **It is important to note that running this script will prevent you from rolling back or applying a staged upgrade.** Navigate to the `/payara5/glassfish/bin` folder and run

```
> ./cleanupUpgrade
```

Running the command will result in the following.



```
~/development/apps/servers/payara-upgrade/payara5/glassfish/bin ./cleanupUpgrade
Deleting old common
Deleting old config/branding
Deleting old config/osgi.properties
Deleting old h2db
Deleting old ../h2db
Deleting old legal
Deleting old modules
Deleting old osgi
Deleting old lib
Deleting old ../README.txt
Deleting old ../LICENSE.txt
Deleting old ../mq
Deleting old bin
Deleting old ../bin
Deleting old domains/production/osgi-cache
Deleting upgrade-tool property files
```

This is the end of the automatic upgrade tool's work. At this point you have a fully upgraded Payara Enterprise 6 installation. The next step is minor manual task that should be performed to fully complete the upgrade.

Follow Up Upgrade Tasks

The upgrade tool does not amend the default-web.xml file. You will have to do this manually. Open the file `/payara5/glassfish/domains/<your-domain>/default-web.xml` file in your text editor. Search for and replace:

```
org.apache.jasper.servlet.JspServlet
```

With

```
org.glassfish.wasp.servlet.JspServlet
```

Also replace the entire `<system-jar-includes> <init-param>` block with

```
<init-param>
  <param-name>system-jar-includes</param-name>
  <param-value>
    /lib/
    \lib\
    expressly.jar
    jakarta.servlet-api.jar
    jakarta.servlet.jsp-api.jar
    jakarta.servlet.jsp.jstl-api.jar
    jakarta.servlet.jsp.jstl.jar
    jakarta.jms-api.jar
    jakarta.faces.jar
    wasp.jar
    jspcaching-connector.jar
    web-glue.jar
    hibernate-validator.jar
    jakarta.ejb-api.jar
    jakarta.enterprise.deploy-api.jar
    jakarta.activation-api.jar
```

```
    angus-activation.jar
    jakarta.mail-api.jar
    angus-mail.jar
    jakarta.persistence.jar
    jakarta.resource-api.jar
    jakarta.security.auth.message-api.jar
    jakarta.security.jacc-api.jar
    jakarta.transaction-api.jar
    webservices-osgi.jar
    weld-osgi-bundle.jar
    jersey-mvc-jsp.jar
  </param-value>
</init-param>
```

And with that, you now have a fully upgrade Payara Enterprise 5 to Payara Enterprise 6, capable of running Jakarta EE 10 applications. The [documentation page](#) of the Payara Server Upgrade Tool has more detail about configuring the log level and other parameters of the tool for debugging purposes if required.

Distributed Environments

The above upgrade steps assume a single running installation. However, you may have a distributed setup for running your workload. All the pre upgrade steps discussed above still apply. You should use the above upgrade steps for each installation. However, the following steps add additional pointers to how you can carry out the upgrade for all the installations with minimal impact.

1. **Disable Auto-Scaling:** If you're using cloud-native features like auto-scaling, disable them temporarily.
2. **Rolling Restart:**
 - 2.1. If possible, perform a rolling restart to upgrade instances one at a time to minimize downtime.
 - 2.2. Redirect traffic away from the node you are about to upgrade.
3. **Update One Node:**
 - 3.1. Stop the Payara Server instance.
 - 3.2. Replace the Payara Server binaries with the new version.
 - 3.3. Start the instance and test to make sure it works as expected.

4. **Traffic Test:**

- 4.1. Enable a small percentage of traffic to the newly upgraded node and monitor.
- 4.2. If everything works as expected, proceed to upgrade other nodes.

5. **Upgrade Remaining Nodes:**

- 5.1. Use the same process as in step 3 for all other nodes in your cluster.

6. **Enable Auto-Scaling:** If you disabled auto-scaling, re-enable it.

After following through with these tips, you should be able to upgrade all the installations with minimal downtime and disruption. Please note however, that these are high-level pointers, your environment might require other considerations and steps. Ensure you make a thorough assessment of your setup before embarking on the upgrade.

Post Upgrade Monitoring

After successfully upgrading, especially in a complex environment like a Payara Enterprise server, post-upgrade monitoring is essential to ensure that everything is working as expected. The following are some pointers to focus on after the upgrade.

Application Performance

Use performance monitoring tools to compare the application's performance metrics (response times, error rates, etc.) pre-and-post upgrade. Investigate if there are any significant deviations. Of course, this presupposes that you have also upgraded your Java EE 8 applications to Jakarta EE 10 as Payara Enterprise 6 only supports Jakarta EE 10.

Error Logs

Continuously monitor server logs for any error messages or warnings. Unexpected entries could indicate compatibility issues that went unnoticed during testing. This should help in proactively identifying problems and fixing them.

Resource Usage

Keep an eye on CPU, memory, and storage usage. An increase in resource consumption could indicate a configuration issue or a leak. Resource usage, at worst, should stay as it was. Any abnormal spike should be investigated and remedied.

User Experience

Check that all user-facing functionalities are working correctly. Use real-user monitoring tools to identify any disruptions or degradation in user experience.

Database Health

Monitor database query times and error rates. Ensure that the database is interacting correctly with the upgraded application and server.

Security

Run vulnerability assessments to ensure that the new setup has not opened any security loopholes. Keep an eye on access and error logs for any suspicious activity.

Network Traffic

Monitor incoming and outgoing network traffic for irregular patterns, which could indicate misconfiguration or security issues post upgrade.

Failover and Redundancy

Test failover mechanisms to ensure they still operate correctly post-upgrade. Verify that backup and redundancy systems are fully functional.

Data Integrity

Make sure that data flow between different components (e.g., application layers, databases, third-party services) remains intact. Run data integrity checks where necessary.

Custom Scripts and Cron Jobs

If you have any custom automation scripts or scheduled jobs, validate that they are still working as intended.

Stakeholder Communication

Keep internal stakeholders updated about the status of the post-upgrade monitoring. Swift communication can help in quick resolution of any issues that may arise.

Support Channels

Keep an eye on the Payara support channel and email communication channels for any patches or updates that might be released post your upgrade, which could be critical for your set-up.

Customer Feedback

Sometimes, users may notice issues that your internal tests did not catch. Monitor customer feedback channels for any reports of issues or changes in behaviour.

Post-upgrade monitoring is not just about identifying issues; it's also an opportunity to understand the impact of the new features and optimizations that the upgrade has brought in. Use this period to gather valuable insights that can help in future upgrade planning and execution.

Conclusion

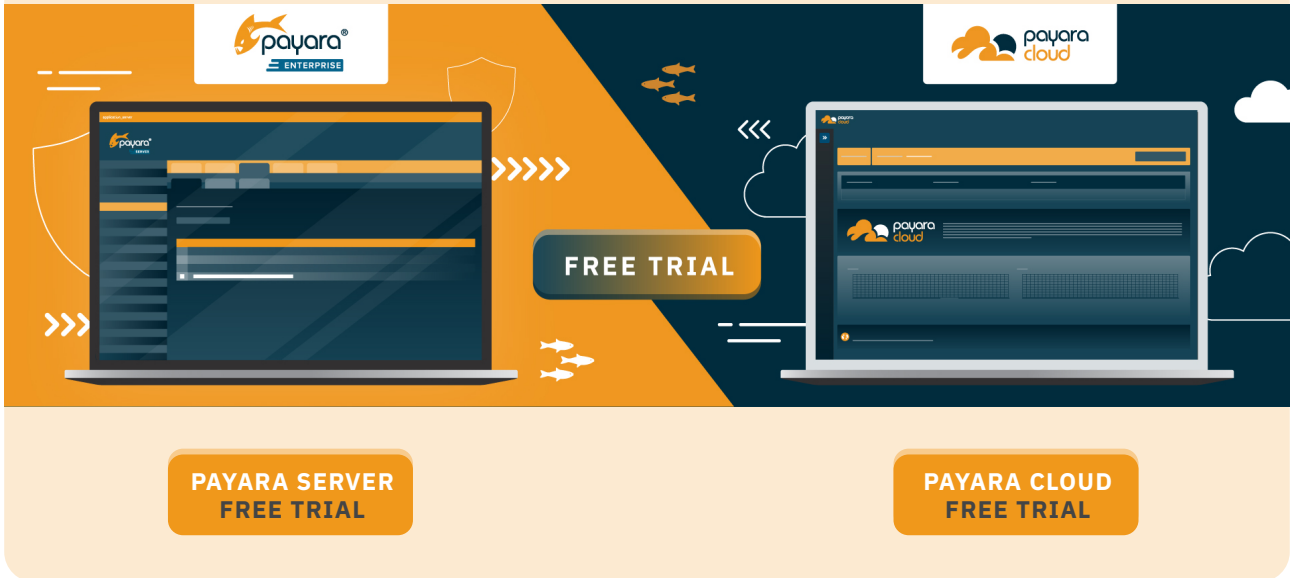
The process of transitioning from Payara Enterprise 5 to Payara Enterprise 6 is one that entails numerous considerations, from initial assessment and planning to implementation and post-upgrade monitoring. This guide aimed to streamline that transition for you through the Payara Server Upgrade Tool, mitigating the risks and complexities typically associated with such an upgrade.

We trust that the strategies and best practices outlined in this guide have provided you with the confidence and technical understanding required to execute a smooth and successful upgrade. Now that you have transitioned to Payara Enterprise 6, you are well-positioned to leverage the robust capabilities and enhancements that come with Jakarta EE 10 and MicroProfile 6, thereby improving the performance, scalability, and security of your enterprise applications.

In summary, the move to Payara Enterprise 6 represents not just an upgrade in your enterprise Java capabilities but also a strategic advancement in how you develop, deploy, and manage your applications. Thank you for your time and engagement with this guide. Should you have further inquiries or require additional support, please do not hesitate to reach out to the relevant support channels. We are committed to your success and are here to assist you in any way we can.

We wish you the very best in your continued journey with Payara and Jakarta EE. Happy coding and onward to new horizons.

Interested in Payara? Try Before You Buy



sales@payara.fish



UK: +44 800 538 5490
Intl: +1 888 239 8941



www.payara.fish

Payara Services Ltd 2023 All Rights Reserved. Registered in England and Wales; Registration Number 09998946
Registered Office: Malvern Hills Science Park, Geraldine Road, Malvern, United Kingdom, WR14 3SZ