User Guide

Payara Server in Production

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

Contents

Introduction

Once you have developed applications on Payara Server and moved these applications into a production environment, control will pass over to your operations teams. This guide will introduce some features of Payara Server that you may not know about, which are especially useful for the operations teams.

The guide covers the following sections:

- Script your Installation
- Production JVM Settings
- Viewing the Server Logs
- Thread Dump
- Operation Monitoring Services
- Request Tracing
- Payara Health Check
- Slow SQL Logger
- Securing the server

Production Configuration

Script Your Installation

Payara Server comes with a very powerful administration client called **Asadmin CLI**. When moving to production, Asadmin CLI can be used to create scripts to build out Payara Server domains in a repeatable fashion for production, preproduction and test environment. An example asadmin script for building a domain is available from our examples repository; A simple example to create a domain containing a separate deployment group in a Hazelcast cluster and an application deployed to that deployment group:

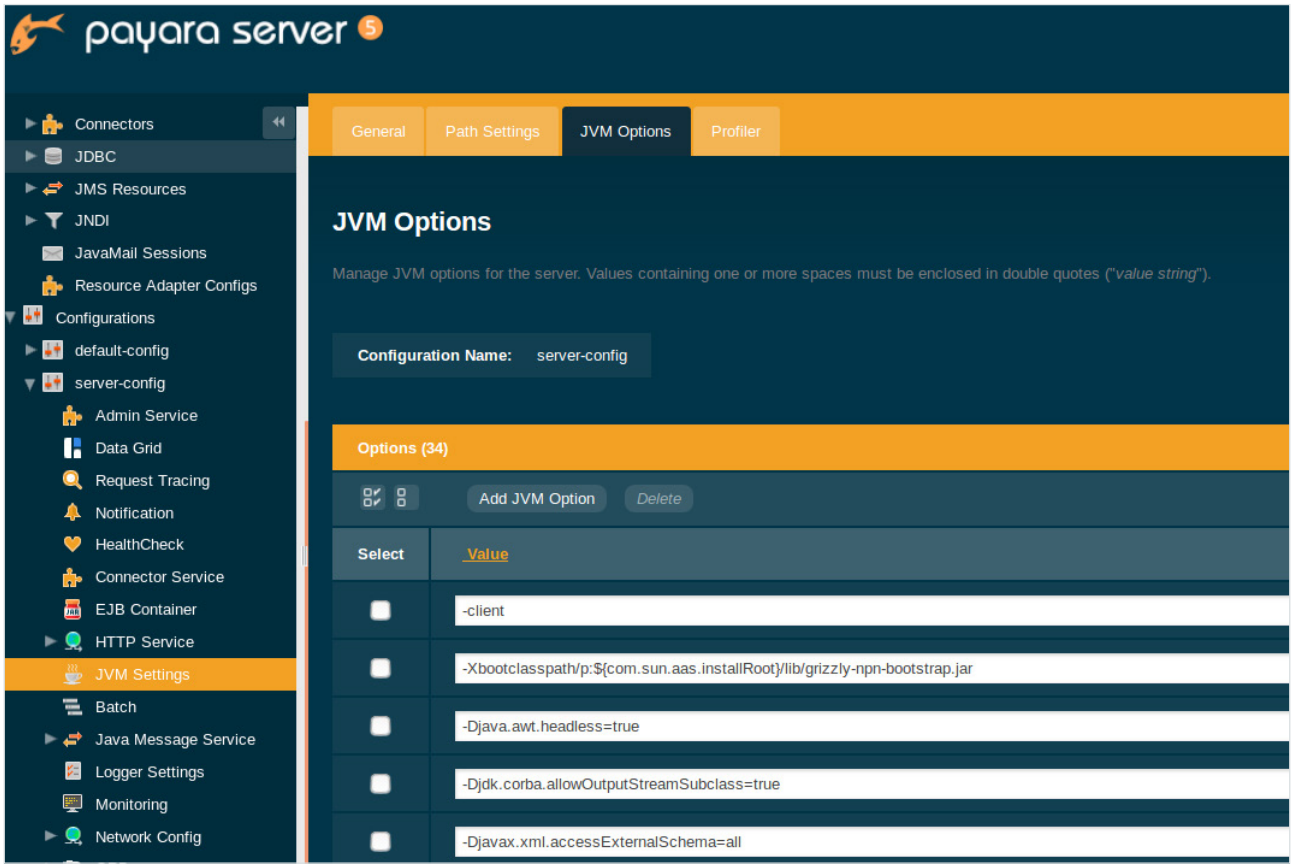
```
asadmin create-domain –nopassword –portbase 10000 example-domain
asadmin start-domain example-domain
asadmin -p 10048 create-deployment-group sample-group
asadmin -p 10048 deploy --name=myapp --contextroot=myapp
    --target=sample-group /path/to/myapp.war
```

Think About Your JVM Settings

Before you put Payara Server into production you should think about what JVM settings you need. First and foremost you should use the latest Java 8 version available to you locally on your operating system. Then you need to set your JVM settings Ideally you are looking for a balance of Heap Size versus GC pause times.

Detailed guidance on a specific heap size depends on the needs of the application but some of the guidelines below should be followed:

- Use the G1 collector for large heaps
- Set your min and max heap sizes to be the same to prevent FullGCs when the heap grows-Xmx2048M -Xms2048M
- Set -XX:+DisableExplicitGC to disable running garbage collector from an application
- Configure garbage collection logging to enable future diagnostics;
  - verbose:gc
  - Xloggc:/path_to_log_file/payara-gc.log
  - XX:+PrintGCDetails
  - XX:+PrintGCDateStamps
- Ensure heap dumps are generated if you run out of memory
  - XX:+HeapDumpOnPutOfMemoryError
  - XX:+HeapDumpPath=/path_to_dump_file/Payara-dump.hprof



JVM Settings can be set in Payara Server from the Admin Console within the JVM Options settings on the server configuration. Or the configuration can be incorporated into your script for building domains using **create–jvm–options** and **delete–jvm–options** asadmin commands:

```
> asadmin delete-jvm-options -Xmx512m:-client
> asadmin create-jvm-options -Xmx2048m:-Xms2048m
```

Consider Using Built-in Production Domain

Payara Server comes with an alternative domain called **production** (or payaradomain in Payara Server 4), which contains configuration much more suitable for production than the default domain called domain1. The default domain is not designed to be run in production, but instead to be a usable example for development and testing purposes and it's not recommended to use it as a template for production settings.

In order to use the production domain, just execute the start-domain command with the argument **production**:

```
./asadmin start-domain production
```

To create a new domain based on the production domain, you can execute the following command:

```
asadmin> create-domain –template
    ${PAYARA_HOME}/glassfish/common/templates/gf/
    production-domain.jar myNewProductionDomain
```

To create a new default domain based on the production domain, first, delete the default domain domain1 and then create domain1 with the production-domain template, such as:

```
asadmin> delete-domain domain1
asadmin> create-domain –template
    ${PAYARA_HOME}/glassfish/common/templates/gf/
    production-domain.jar domain1
```

The new domain can be then started without adding the domain name to the start-domain command:

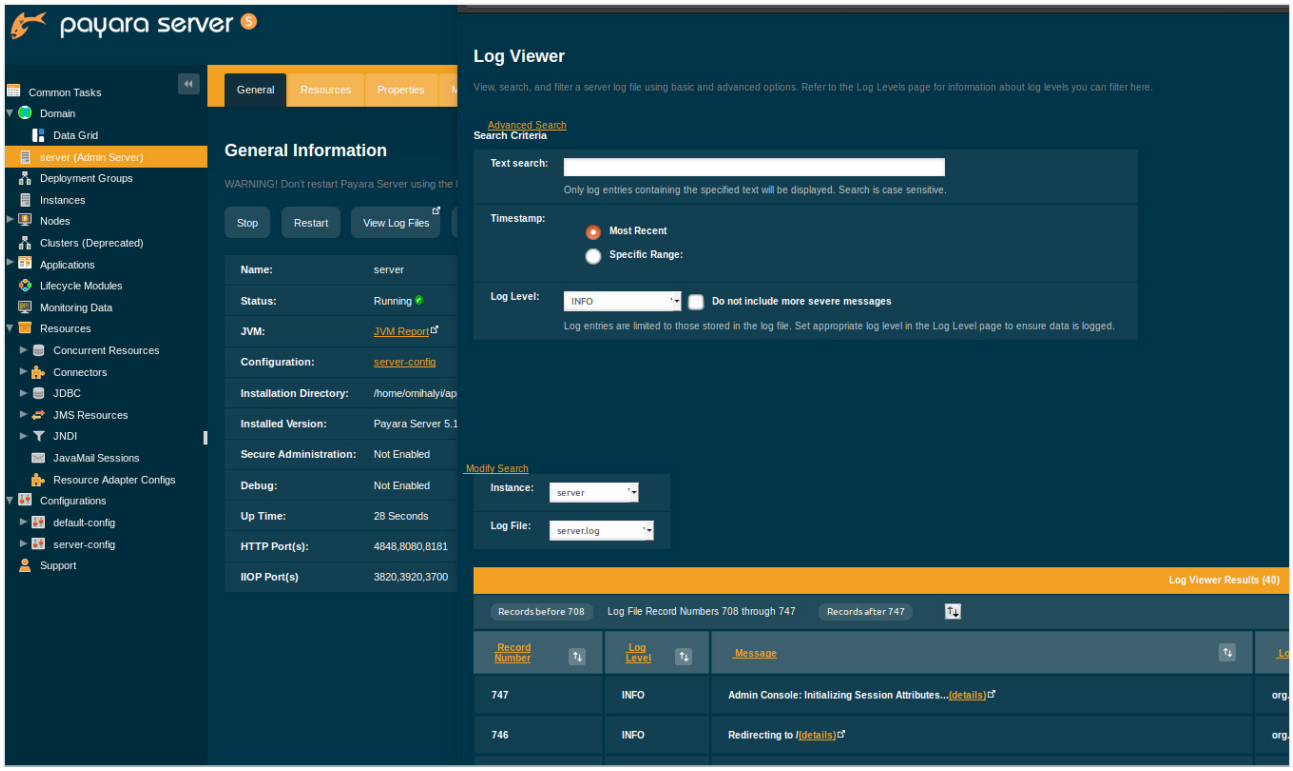
```
./asadmin start-domain
```

More information about the **production** domain in Payara Server Documentation.

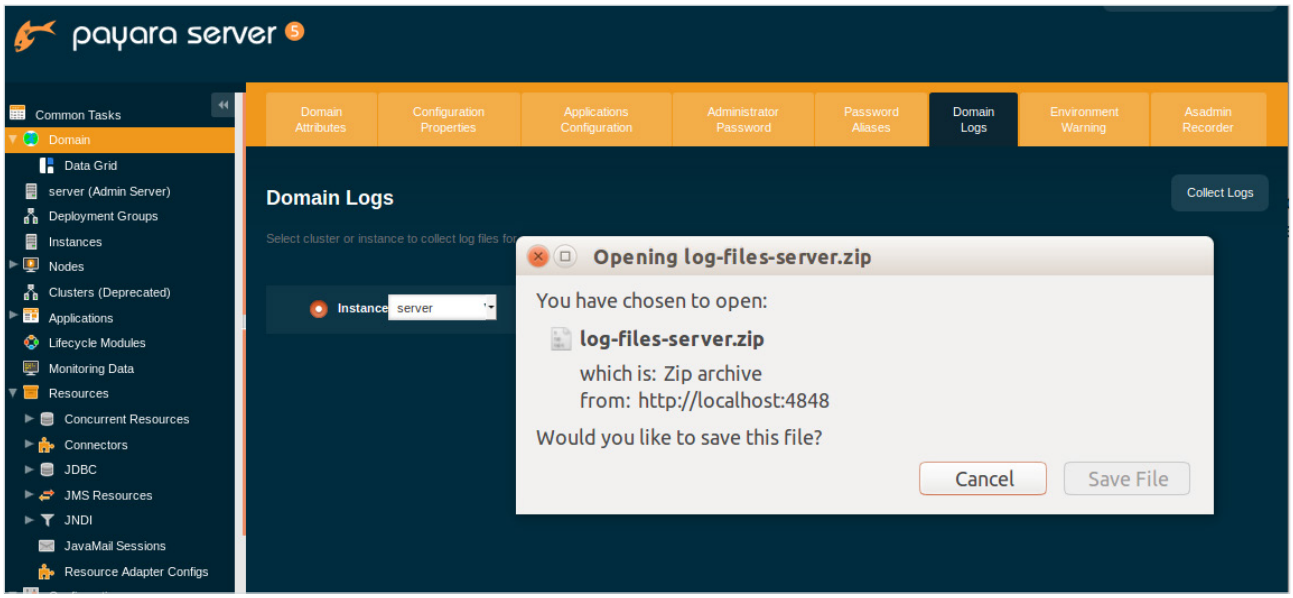
Investigation and Troubleshooting

Viewing the Server Logs

When you are running Payara in production, especially if you are running a large number of servers in a domain across multiple hosts, you do not want to login to multiple machines to view and search the log files. In Payara Server you can see, search and rotate the log files of any instance in the domain from the administration console. Just display the page about the server and press the “View Log Files” button. In the new window, you can select an instance in the domain and its log file you want to examine:



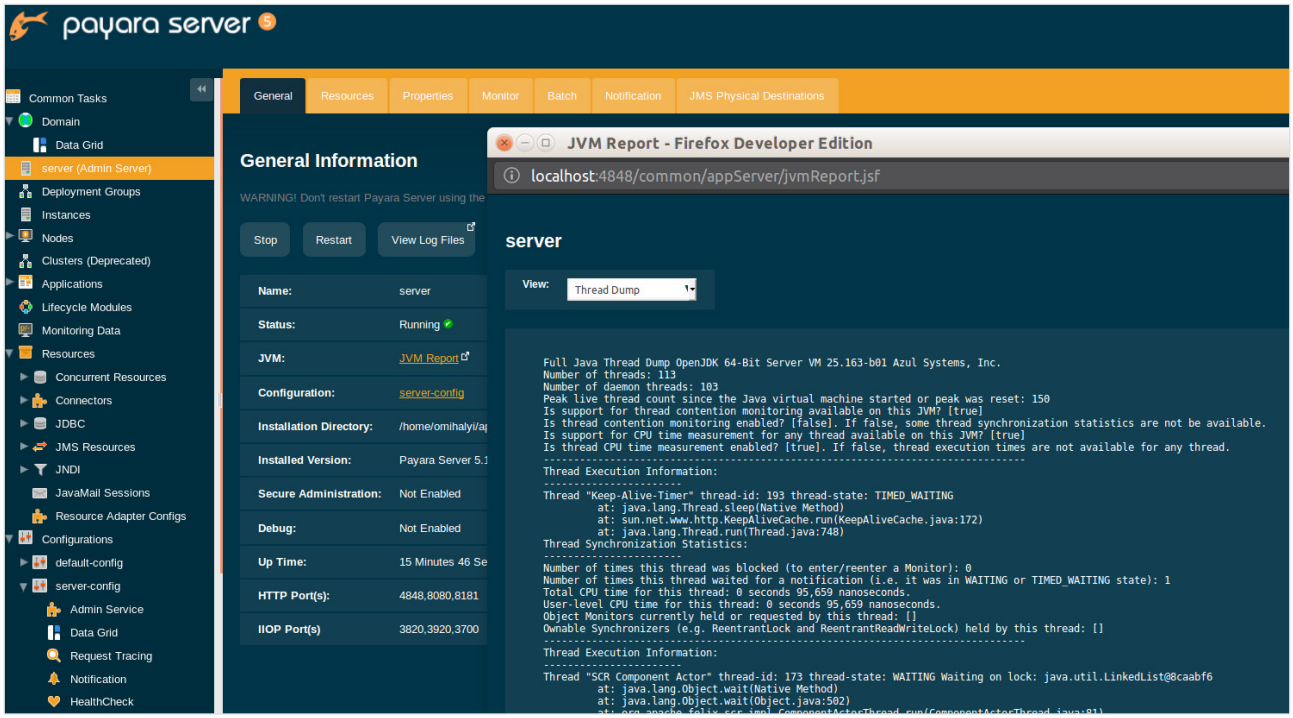
You can also download any log file—select the Domain page, Domain Logs tab, select the instance to download logs for and press “Collect Logs” in the top right corner to download them:



Get a Thread Dump

If a server appears to be running slowly you can get a Thread Dump for any server in the domain from the Admin Console. The thread dump can then be analysed to see why a server is running slowly.

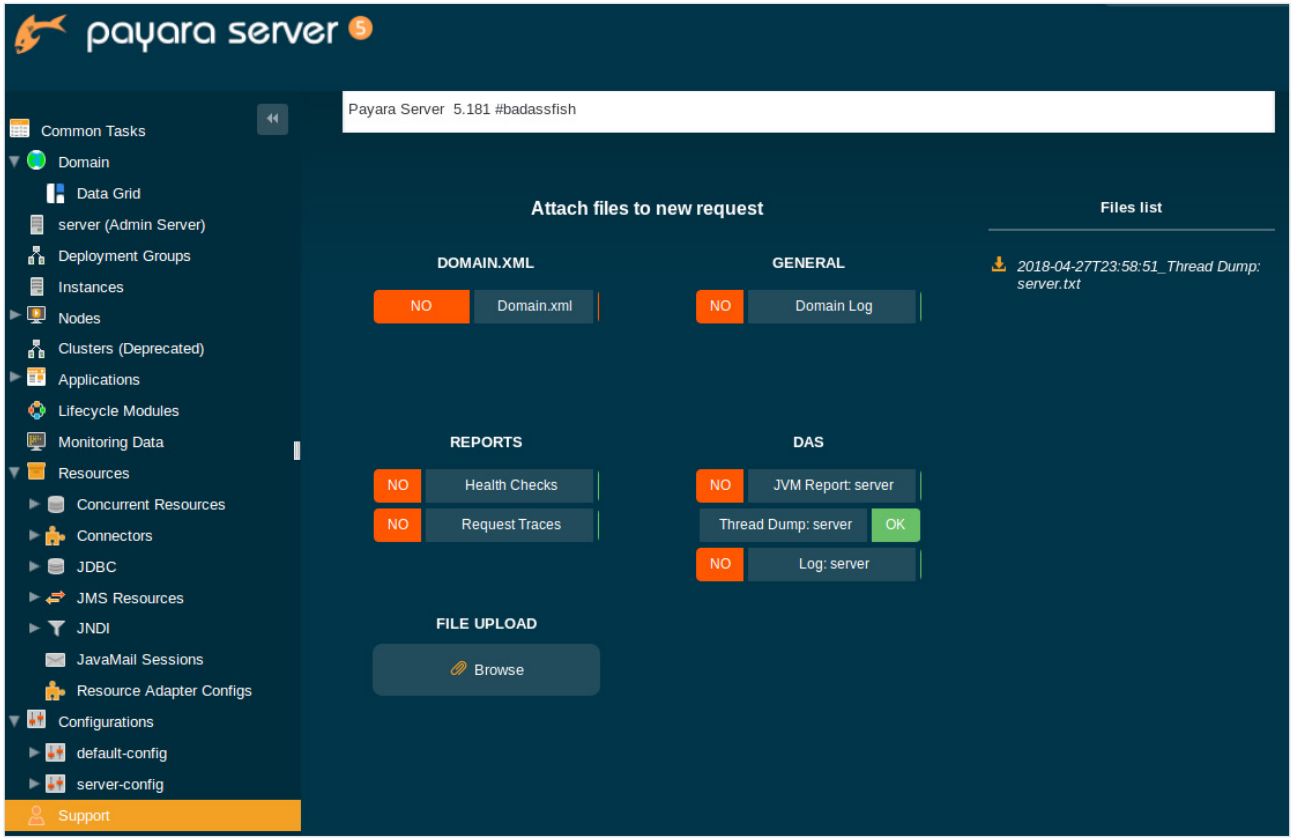
You can get a thread dump from the Admin Console. Go to the server or instance you want to generate a thread dump for and press “JVM Report” in the **General Information** page. Then, in the new window, select **Thread dump** from the “View” select box:



You can also generate a thread dump using the **generate–jvm–report** command:

```
asadmin generate-jvm-report --type=thread
```

A thread dump and a lot of other diagnostic information can also be attached to a support ticket if Payara Server is connected to the Payara Support portal:



Payara Server Operations Services

Payara Server has several services which can be used to monitor the behaviour of Payara Server or applications deployed to it. The first service to understand is the **Notification Service**. This service provides many channels through which to receive monitoring data. There are several services which can feed data through to the configured notifiers: the **JMX Monitoring Logging Service** which can periodically log the values of specific JMX MBeans through the notification service; the Request Tracing Service which can trace each request as it propagates through the application; and the **Payara Health Check Service** which can periodically check the status of various server and host metrics.

In addition to these services, there is also a slow SQL logger which operates independently. By simply specifying a “slow” threshold on a data source, any query made through that data source which takes longer than the threshold will cause both the problematic query and a stacktrace to show where the query originated from to be printed to the server.log file.

The Notification Service

Payara Server comes with a general Notification Service in order to log events which come from other services, such as the JMX Monitoring Logging Service, the Payara Health Check Service or the Request Tracing Service.

The Notification Service provides the ability to disseminate notification events through various channels that are being created by other services. This service is provided with a number of configurable *notifiers*, the default being a log notification mechanism alongside these other notifiers:

- HipChat
- Slack
- NewRelic
- Datadog
- SNMP
- XMPP
- JMS
- Email
- CDI Event Bus

Because other operation monitoring services depend on the Notification Service to output their data, both the Notification Service itself, and at least one notifier, must be enabled for any data to be recorded. If this does not happen, then the data will be discarded.

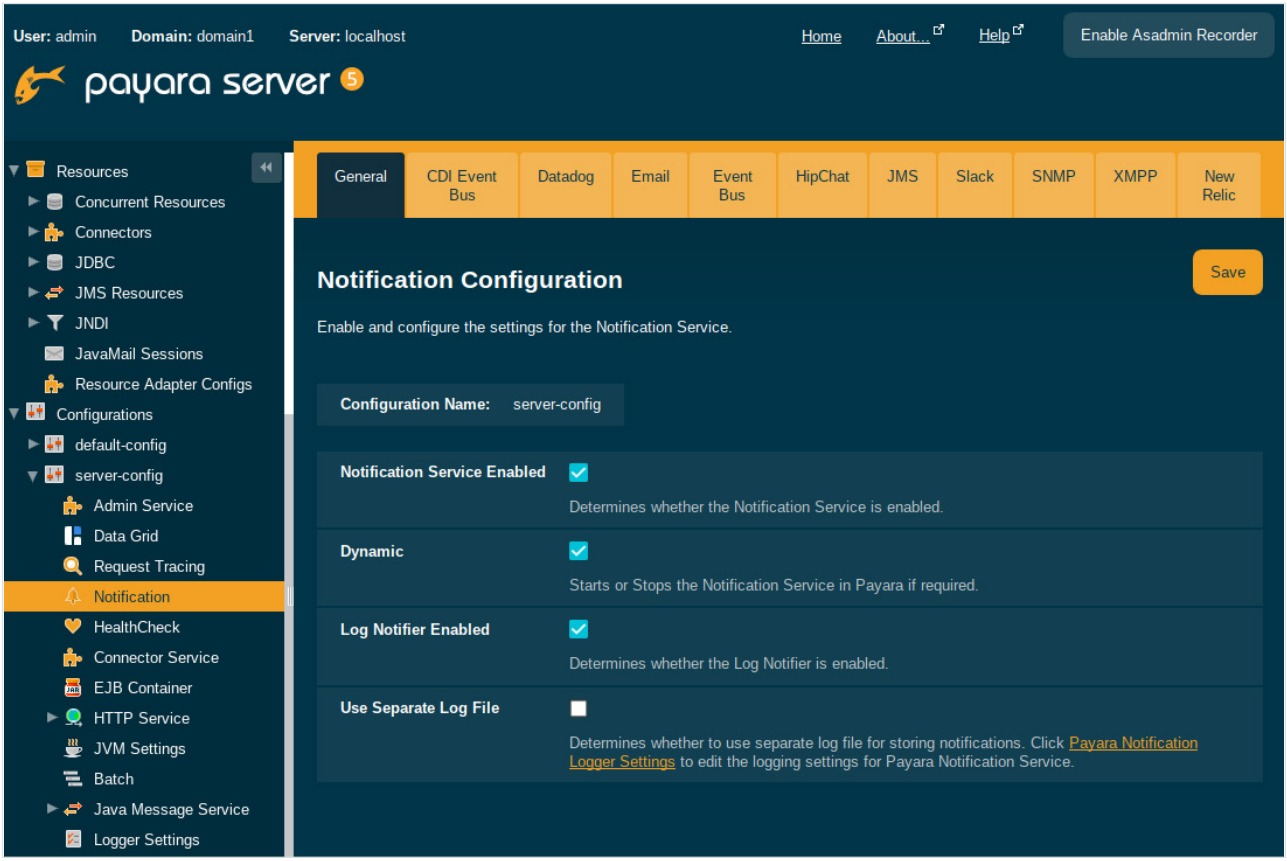
It is for this reason that the Log notifier is the default notifier and is enabled out-of-the-box.

In order to configure a service like the Payara Health Check Service to send notifications to a specific notifier, the following configuration steps are needed:

1. Notification Service needs to be enabled
2. Notifiers in the Notification Service need to be configured to specify how to process the notification
3. After configuring them, notifiers need to be enabled globally in the Notification Service
4. Notifiers also need to be selected (enabled) for each particular service that should send them notifications

Notifiers have to be enabled both globally in the Notification Service and also for each service that sends notifications.

Notifiers are configured on the Notification page of the Administration console:



The Notification Service can be configured with the **`notification-configure`** command.

To enable the Notification Service:

```
notification-configure --enabled=true --dynamic=true
```

The argument `--dynamic` is optional and applies the configuration without restart the server.

Each notifier provides a separate asadmin command to enable and configure it:

- **notification-log-configure** for the Log notifier
- **notification-email-configure** for the Email notifier
- **notification-slack-configure** for the Slack notifier
- **notification-hipchat-configure** for the HipChat notifier
- **notification-jms-configure** for the JMS notifier
- **notification-newrelic-configure** for the New Relic notifier
- **notification-datadog-configure** for the Datadog notifier
- **notification-snmp-configure** for the SNMP notifier
- **notification-xmpp-configure** for the XMPP notifier

For example, to configure the Log and Email notifiers:

```
notification-log-configure --enabled=true --dynamic=true
notification-email-configure
    --jndiName=mail/exampleEmailNotifier
    --to=notifications@example.com --enabled=true
    --dynamic=true
```

The Request Tracing Service

The Request Tracing Service provides tracing facilities for multiple protocols and process communications done by the components of deployed applications.

The service helps users to detect application slowness and performance degradation by logging requests that exceed a given threshold. The trace data from long-running requests gives insight into solving bottlenecks and other performance issues.

From the 181 release of Payara Server, Request Tracing has been rewritten for compatibility with the OpenTracing specification. This will bring Payara Server's own Request Tracing into line with the upcoming MicroProfile OpenTracing specification. Planned features for Request Tracing include ways to interpret and analyse the data with OpenTracing compatible servers.

Understanding Terms

Since OpenTracing is being developed as a collaboration with multiple vendors, the specification has established a versioned document to track the specification. This specification uses terms in specific ways which may not be familiar to users of Payara Server. These terms are defined below:

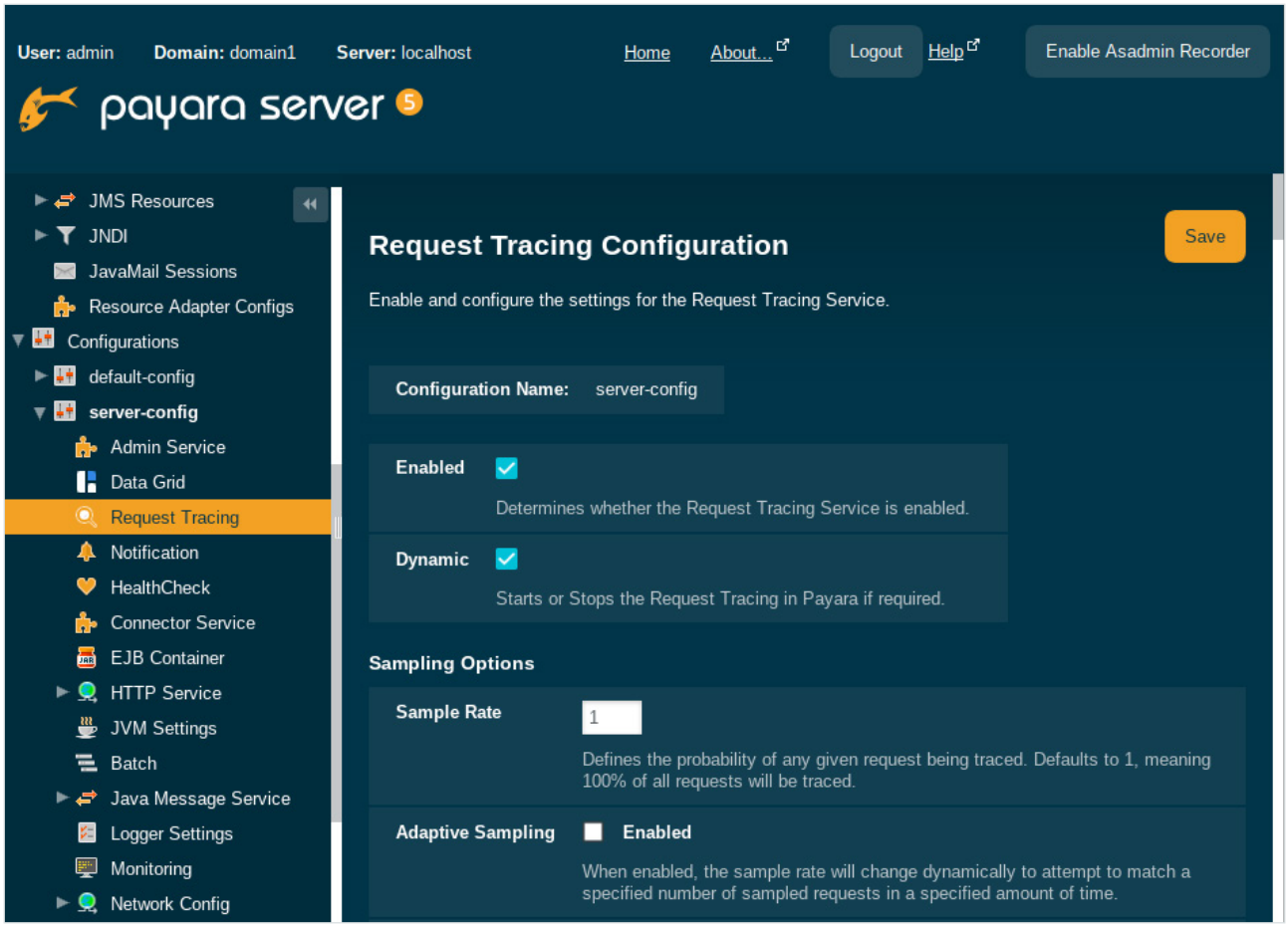
- **Span**
  A "Span" is equivalent to a single operation, or method and records a start time, end time

and a duration. Each span has a unique “spanId” in its spanContext and a human-readable operationName which describes the operation that was completed in the span.

- **Span Context**
  Every span has a “spanContext” property which is used to both identify the span itself as well as the trace that it belongs to.
- **Trace**
  A “Trace” can be thought of as a single request. There is no overarching record for the trace, it is simply a correlation ID which is shared among all the related “spans”. In the spanContext, there is a “traceId” field which is the same among related spans.

Quickstart

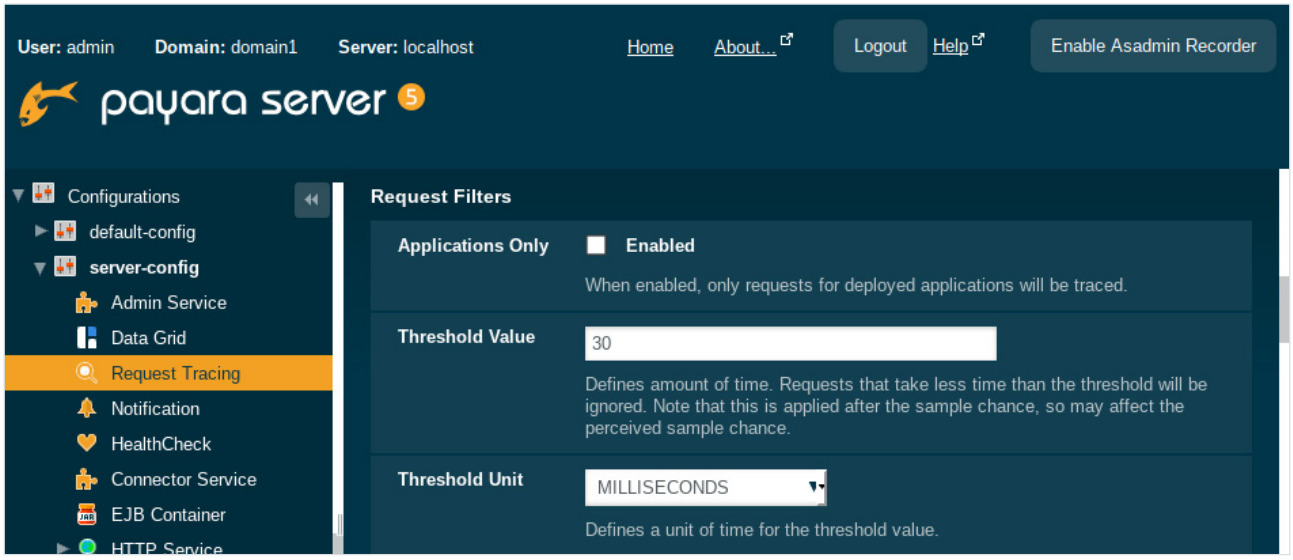
To begin using Request Tracing, it must first be enabled. This can be done dynamically through the admin console, so no restart is required:



As well as simply enabling Request Tracing, a valid notifier needs to be added so that the data can get logged. The log notifier is the most straightforward because it is enabled in the Notification Service by default.

Once these steps are complete, it is still likely that no data is logged, which could be for two reasons:

1. If you have no applications deployed and simply want to trace the activity in the admin console, you must disable the “Applications Only” filter.
2. The default threshold value is set to 30 seconds, which means that any request which does not take that long will not be traced. To trace most things, the threshold can be set to a very low value, such as 30 milliseconds, for example.



The Payara Health Check Service

Problems can occur in production when the Payara Server JVM has memory or garbage collection issues or when the host server has excessive CPU or memory usage. Each Payara Server instance has a built-in Health Check service which is used in production to monitor the health of a number of core subsystems; CPU usage, memory usage, JVM GC and thread CPU usage. Each subsystem can be configured to write to the server log file if a problem is detected. To configure the health check service it first needs to be enabled:

```
asadmin healthcheck-configure --enabled=true --dynamic=true
```

If you want to see the current settings of the healthcheck service use the command:

```
asadmin get-healthcheck-configuration
```

This is an example output from the above command after some Health Check services have been configured:

```
Health Check Service Configuration is enabled?: true
Below are the list of configuration details of each checker listed by its name.
Name  Enabled  Time  Unit
GC    false    10    SECONDS
Name  Enabled  Time  Unit     thresholdPercentage  retryCount
HT    true     10    SECONDS  95                    3
Name  Enabled  Time  Unit     Critical Threshold  Warning Threshold  Good Threshold
CPU   false    10    SECONDS  40                  20                 2
HP    false    8     SECONDS  -                   -                  -
MM    false    7     SECONDS  -                   -                  -
```

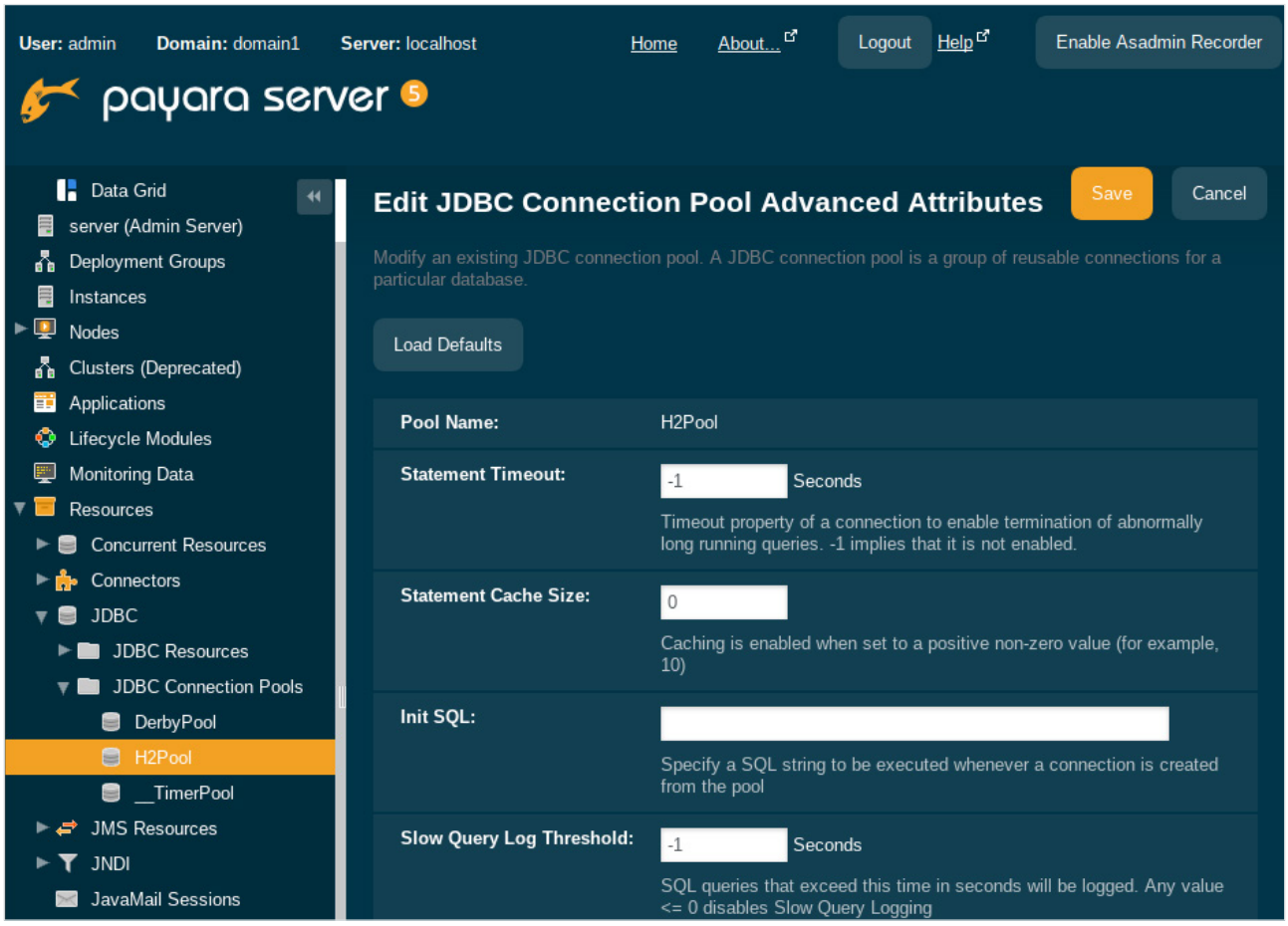
For further information on the Payara Health Check service check out the documentation.

The configuration of the Payara Health Check service can be incorporated into your scripts for building your production domain.

Slow SQL Logger

Payara Server includes capabilities to detect and log slow SQL queries executed via a connection pool. The Slow SQL logger monitors all queries executed on the connection pool and if they exceed a configurable execution time in seconds, a warning message is logged into the server log. The warning message (see below) logs the SQL query and the stack trace to the code executing the query. This enables rapid diagnosis, pinpointing the exact lines of code to investigate.

To configure slow SQL logging via the administration console, navigate to the connection pool's Advanced Properties tab and specify the Slow Query Log Threshold time in seconds. A value of -1 disables logging of slow queries.



Similarly, slow SQL logging can be configured using the asadmin **set** command, as with other configuration properties.

For more information about the detection of slow SQL queries, see the Slow SQL logger documentation page.

Securing Payara Server

Payara Server allows securing HTTP protocol and remote access to Payara Server administration interface with encryption and authentication. Payara Server contains self-signed certifications for built-in domains and generates a self-signed certificate for new domains which you can use to encrypt communication. However, it's encouraged to use your own certificate. You may have a self-signed certificate or a certificate signed by a trusted authority. In both cases, it is pretty easy to add them to a Payara Server domain and use them to secure communication channels.

The process of securing your Payara Server with new certificates involves the following steps:

1. add bundled public and private key into the key store used by the Payara Server domain
2. add certificate public key to the trusted certificates in Payara Server domain - optional, but strongly encouraged
3. enable security on HTTP listener, if required
4. enable security on admin interface, if required

Before we can access certificate key stores, we need to make sure that we set Payara Server domain master password. This password is used by the Payara Server to access key stores. You will need to use the same password to access key stores later.

The default master password is **changeit**. You may want to change it by the following command:

```
asadmin change-master-password --savemasterpassword=true mydomain
```

In order to operate with key stores, we will use keytool executable available in the bin directory of your JDK installation. It's recommended that Payara Server is stopped when operating with the key stores.

We need to import `mydomain_certificate` certificate from a PKCS12 bundle (mycert.p12) with our public and private key into the key store file keystore.jks in our Payara Server domain, in the config directory. The command keytool will do the job:

```
keytool -importkeystore -destkeystore keystore.jks
    -srckeystore mycert.p12 -srcstoretype PKCS12
    -alias mydomain_certificate
```

If you don't have a PKCS12 bundle but have the public and private key in another form, you should create the bundle with tools provided by your operating system or third-party software.

The default key store file for trusted certificates can be found at <Payara_install>/glassfish/domains/**mydomain/config/cacerts.jks**.

We need to import our public key file (mycert.crt) into the cacerts.jks keystore with the following command:

```
keytool -importcert -trustcacerts
    -destkeystore cacerts.jks -file mycert.crt
    -alias mydomain_certificate
```

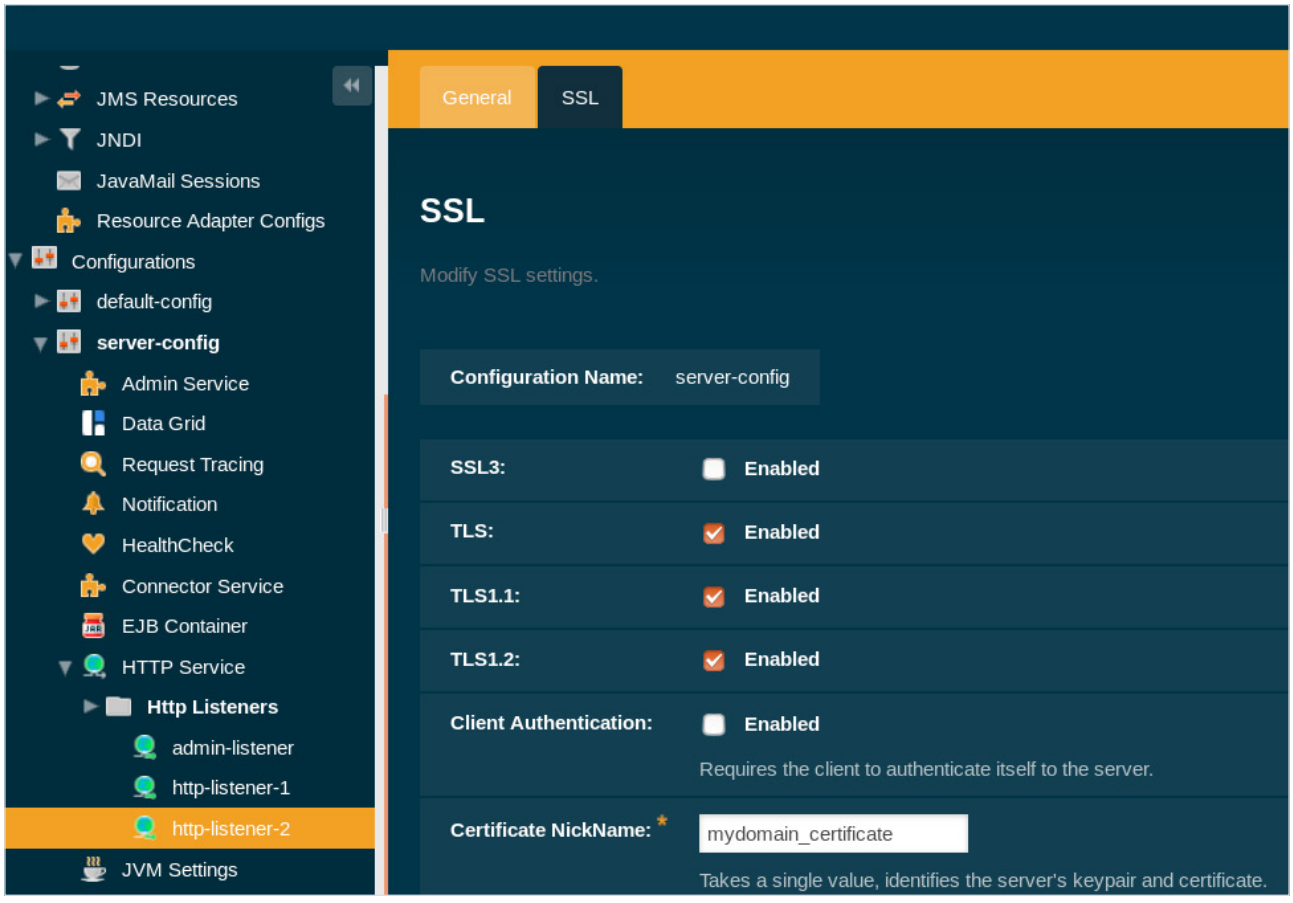
In case of PKCS bundle (mycert.p12), the following command will do the same:

```
keytool -importkeystore -destkeystore cacerts.jks
    -srckeystore mycert.p12 -srcstoretype PKCS12
    -alias mydomain_certificate
```

At this point, the certificate is available for use in your domain. You may use it to secure an HTTP listener or replace the default certificate for the listener http-listener-2, which is secured by default.

In Admin Console, go to Configurations -> server-config -> HTTP Service -> HTTP Listeners > my-http-listener, where my-http-listener is the HTTP listener you want to configure. By default, there already exists http-listener-2, which is already secured by TLS protocol, using default certificate with alias s1as and listens on port 8181. If you are adding security to an insecure listener, make sure that security is **enabled**.

On the SSL tab, we also want to apply our new certificate - **Certificate NickName** should contain the alias of our certificate. In our case it is **mydomain_certificate**:



If you are not sure about the alias of your certificate, use the following command to find out its alias from the keystore.jks file in the domain config directory:

```
keytool -list -keystore keystore.jks
```

This command should list **glassfish-instance** and **s1as** stock certificates, as well as any new certificates you add.

HTTP listeners can also be secured using asadmin commands. The following commands will enable security and all available TLS protocols:

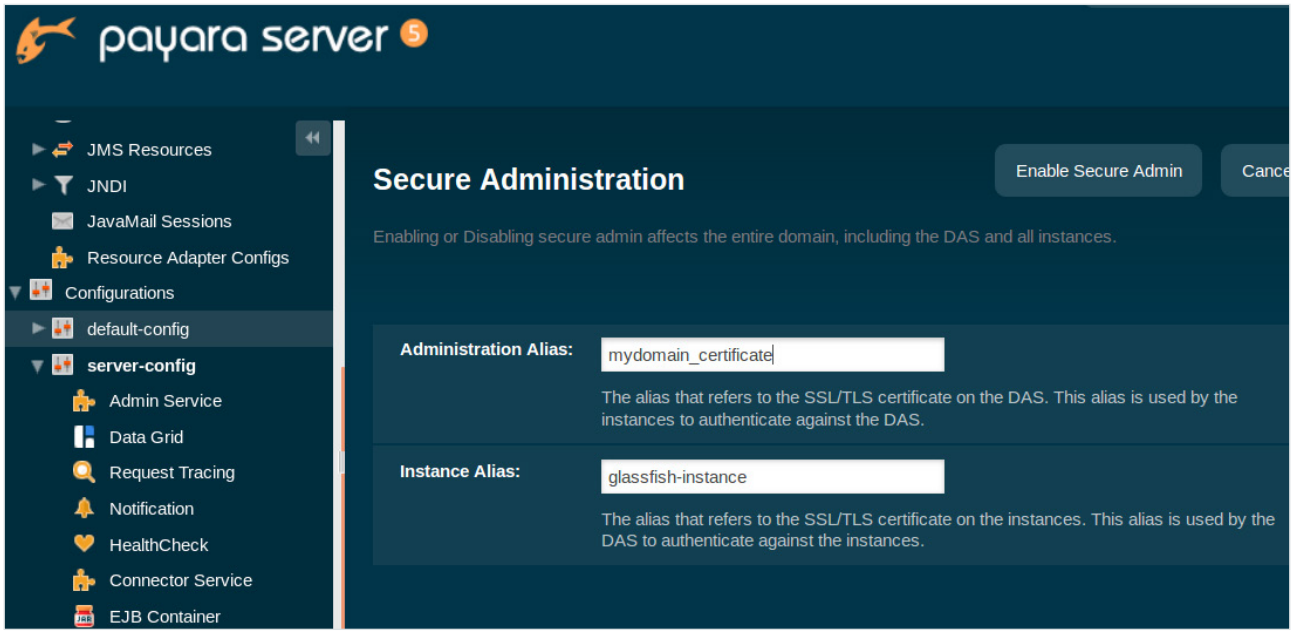
```
asadmin set configs.config.server-config.network-config.protocols.protocol.
http-listener-2.security-enabled=true
asadmin set configs.config.server-config.network-config.protocols.protocol.
http-listener-2.ssl.tls-enabled=true
asadmin set configs.config.server-config.network-config.protocols.protocol.
http-listener-2.ssl.tls11-enabled=true
asadmin set configs.config.server-config.network-config.protocols.protocol.
http-listener-2.ssl.tls12-enabled=true
```

And the following will apply the new certificate instead of the default s1as:

```
asadmin set configs.config.server-config.network-config.protocols.protocol.
http-listener-2.ssl.cert-nickname=mydomain_certificate
```

Securing the admin listener is a bit different. It requires that all users with admin access have a non-empty password.

In Admin Console, securing administration has a dedicated page in order to secure both the admin HTTP listener and communication between admin server and instances in the domain in a single place. It's accessible from the **server (Admin Server)** link in the sidebar, which contains the "Secure Administration" button, which opens the following page:



The button “Enable Secure Admin” enables encryption for both HTTP admin listener and communication between instances.

Secure administration can be enabled also by an asadmin command:

```
asadmin enable-secure-admin --adminalias=mydomain_certificate
```

After enabling secure administration you need to restart the server, as well as other instances in the domain.

Payara Server Enterprise

If you're using Payara Server in production you need Payara Server Enterprise. Payara Server Enterprise is a cloud-native middleware application platform supporting mission critical production systems with reliable and secure deployments of Jakarta EE (Java EE) applications on premise, in the cloud, or hybrid environments. A stable platform with monthly releases, bug fixes, and a 10-year software lifecycle, Payara Server is aggressively compatible with the ecosystem components you're already using, provides broad integration with cloud vendors, and support for Docker and Kubernetes. Development in collaboration with an industry-leading DevOps team and the global Payara community ensures Payara Server is the best option for production Jakarta EE (Java EE) applications today and for the future.

Learn more about Payara Enterprise: https://www.payara.fish/enterprise/

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