

appNG Platform Administration Guide

Matthias Müller

Version 1.26.3 created on 2023-06-27

Table of Contents

1. Scope	1
2. Setup	1
2.1. Requirements	1
2.2. Configuring the database	1
2.2.1. Configuring the database pool	1
2.2.2. Configuring the database connection	2
2.2.3. Full example	2
2.3. System and environment variables	2
3. Clustering	3
3.1. Using Multicast	3
3.2. Using Hazelcast	3
3.3. Using RabbitMQ	3
3.4. Using Redis	4
4. Caching	4
4.1. Platform configuration	4
4.2. Site configuration	5
5. User Management	7
5.1. User Types	7
5.2. LDAP Connectivity	7
5.3. Password Policies	7
6. Command Line Interface (CLI)	9
7. appNGizer	10
8. Monitoring	11
8.1. Configuration	11
8.2. Services	11
8.2.1. Site health	11
8.2.2. Platform JARs	13
8.2.3. Site JARs	13
8.2.4. System environment	13
8.2.5. System properties	13
8.3. Opentelemetry metrics	13

1. Scope

2. Setup

2.1. Requirements

2.2. Configuring the database

The database connection and pool used by appnG must be configured in `WEB-INF/conf/appnG.properties`.

2.2.1. Configuring the database pool

- `database.type`

The type of the database. Must be one of `mysql`, `mssql`, `hsql` or `mariadb`



If you want to use MariaDB, please choose `mysql` here and add the MariaDB JDBC Driver to the classpath.

Then set `hibernate.connection.url`, `hibernate.connection.driver_class` and `hibernate.dialect` accordingly.

- `database.minConnections`

The minimum number of connections to keep in the connection pool
Default: `3`

- `database.maxConnections`

The maximum number of connections to keep in the connection pool
Default: `10`

- `database.maxLifetime:`

The maximum lifetime in milliseconds of a connection in the pool
Default: `90000`

- `database.validationQuery`

A query used to validate the connection from the pool
No default, in favor of `Connection.isValid()` introduced in JDBC4

- `database.validationPeriod`

The period, in minutes, to execute the validation query.
No default, obsolete if `database.validationQuery` is not set.

- `database.validationTimeout`

The maximum number of milliseconds that the pool will wait for a connection to be validated as alive.
Default: `5000`

- `database.connectionTimeout`

The maximum number of milliseconds that a appnG wait for a connection from the pool.
Default: `5000`

- `database.logPerformance`
Set to `true` to enable performance logging provided by [JDBC Performance Logger](#)
Default: `false`

2.2.2. Configuring the database connection

- `hibernate.connection.url`
The JDBC connection URL
- `hibernate.dialect`
The Hibernate Dialect to use
- `hibernate.connection.driver_class`
The JDBC driver class to use
- `hibernate.connection.username`
The username for the database
- `hibernate.connection.password`
The password for the database

2.2.3. Full example

This example shows how to connect to a mysql database:

```
database.type = mysql
database.minConnections=10
database.maxConnections=20
database.validationQuery =
database.validationPeriod =

hibernate.connection.url = jdbc:mysql://localhost:3306/appng
hibernate.dialect = org.hibernate.dialect.MySQL8Dialect
hibernate.connection.driver_class = com.mysql.jdbc.Driver
hibernate.connection.username = john
hibernate.connection.password = secret
```

2.3. System and environment variables

In `appNG.properties`, you can use the system's environment variables with the syntax `${env.<variable>}`. Additionally, also system properties can be used with the syntax `${sys.<variable>}`.

Check out the following example:

```
database.type = ${sys.DB_TYPE}
hibernate.connection.username = ${env.DB_USER}
hibernate.connection.password = ${env.DB_PASSWORD}
```

3. Clustering

To enable clustering you need to have

- at least two appNG Nodes
- a load balancer that is able to distribute requests between several nodes, for example [HAProxy](#)
- set the [platform property](#) `messagingEnabled` to `true`
- configure the details for the chosen message exchange

There are currently three techniques for appNG to exchange messages between nodes in a cluster:

- [Multicast](#)
- [Hazelcast](#)
- [Redis](#)
- [RabbitMQ](#)

3.1. Using Multicast

The following platform-properties need to be configured when using multicast:

- set `messagingReceiver` to `org.appng.core.controller.messaging.MulticastReceiver` (this is the default)
- set `messagingGroupAddress` to an appropriate value (default: `224.2.2.4`)
- set `messagingGroupPort` to an appropriate value (default: `4000`)

3.2. Using Hazelcast

With [Hazelcast](#), a [Reliable Topic](#) is used to publish and subscribe cluster events.

The following platform-properties need to be configured:

- set `messagingReceiver` to `org.appng.core.controller.messaging.HazelCastReceiver`
- `hazelcastTopicName`: The name of the topic (default: `appng-messaging`)

3.3. Using RabbitMQ

With [RabbitMQ](#), appNG uses a queue based publish/subscribe mechanism for cluster communication. The following platform-properties need to be configured:

- set `messagingReceiver` to `org.appng.core.controller.messaging.RabbitMQReceiver`
- `rabbitMQAddresses`: A comma separated list of `<host>:<port>` for RabbitMQ server(s) (default: `localhost:5672`):
- `rabbitMQUser`: Username (default: `guest`)

- `rabbitMQPassword`: Password (default: `guest`)
- `rabbitMQExchange`: Name of the exchange where the receiver binds its messaging queue on. Be aware that this name must be different among different clusters using the same RabbitMQ server (default: `appng-messaging`).
- `rabbitMQAutoDeleteQueue`: If the queue to create should be marked as auto-delete (default: `true`).
- `rabbitMQDurableQueue`: If the queue to create should be marked as durable (default: `false`).
- `rabbitMQExclusiveQueue`: If the queue to create should be marked as exclusive (default: `true`).

3.4. Using Redis

With [Redis](#), cluster communication is based on a publish/subscribe mechanism that works on a messaging channel. The following platform-properties need to be configured:

- set `messagingReceiver` to `org.appng.core.controller.messaging.JedisReceiver`
- `redisMessagingHost`: Host of the Redis server (default: `localhost`).
- `redisMessagingPort`: Port of the Redis server (default: `6379`).
- `redisMessagingPassword`: Password of the Redis server (no default).
- `redisMessagingTimeout`: Timeout is optional. If not defined, Redis default is used (no default).
- `redisMessagingChannel`: Channel where all cluster nodes should publish and subscribe. Be aware that this name must be different among different clusters using the same Redis server (default: `appng-messaging`).

4. Caching

4.1. Platform configuration

Caching is provided by [Hazelcast](#).

A central configuration file is used to configure Hazelcast. The location of this file is specified by the platform property `cacheConfig` (see [Platform properties](#)).

Usually, `cacheConfig` points to `WEB-INF/conf/hazelcast.xml`, which might look as follows for multicast replication:

```

<?xml version="1.0" encoding="UTF-8"?>
<!--suppress XmlDefaultAttributeValue -->
<hazelcast xmlns="http://www.hazelcast.com/schema/config" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.hazelcast.com/schema/config
    http://www.hazelcast.com/schema/config/hazelcast-config-5.0.xsd">

  <properties>
    <property name="hazelcast.logging.type">slf4j</property>
  </properties>
  <cluster-name>appNG</cluster-name>
  <instance-name>dev</instance-name>
  <management-center />
  <network>
    <port auto-increment="true" port-count="100">5701</port>
    <outbound-ports>
      <ports>0</ports>
    </outbound-ports>
    <join>
      <auto-detection enabled="false" />
      <multicast enabled="false">
        <multicast-group>224.2.2.3</multicast-group>
        <multicast-port>54327</multicast-port>
      </multicast>
      <tcp-ip enabled="false" />
      <aws enabled="false" />
      <gcp enabled="false" />
      <azure enabled="false" />
      <kubernetes enabled="false" />
      <eureka enabled="false" />
      <discovery-strategies />
    </join>
    <interfaces enabled="false" />
    <ssl enabled="false" />
    <socket-interceptor enabled="false" />
    <failure-detector>
      <icmp enabled="false" />
    </failure-detector>
  </network>
</hazelcast>

```

For further details about configuring Hazelcast, check out its [Reference Manual](#).

Also check out the section about caching in the [Manager User Manual](#).

4.2. Site configuration

There are several site-scoped properties to configure caching.

- `cacheEnabled` (default: `false`)
Set to `true` to enable caching for the site.
- `cacheExceptions`
Default:

```
/health
/manager
```

URL path prefixes which are never cached, as a multiline value. Contains one prefix per line.

Example:

```
/service/appng/my-application
/service/my-site
```



An entry in `cacheExceptions` beats an entry in `cacheTimeouts`.

- `cacheTimeouts`
The path specific cache timeouts, as a multilined value. The format is `path-prefix = <TTL in seconds>`.
Optionally, with appending `,<client TTL in seconds>`, you can control the `Cache-Control` HTTP header that is sent to the client.
Contains one path-prefix per line.

Example:

```
# same TTL for internal cache and client:
# Cache-Control: max-age=7200
/service/appng/cached-application = 7200
# different TTL for internal cache and client:
# Cache-Control: max-age=14400
/service/thesite = 3600,14400
# disable client caching:
# Cache-Control: no-cache,no-store,max-age=0
# Expires: Thu, 01 Jan 1970 00:00:00 GMT
/en/index.html = 1800,0
```

- `cacheTimeoutsAntStyle` (default: `true`)
When set to `true`, the path-prefixes defined in `cacheTimeouts` can use [Ant-style path matching](#).
- `cacheTimeToLive` (default: `1800`)
The default TTL for a cache entry in seconds, if there's no matching path defined in `cacheTimeouts`.
- `cacheStatistics` (default: `false`)
Set to `true` to enable caching statistics
- `cacheWatchRepository` (default: `false`)
Set to `true` to watch the repository folder for changes and invalidate cache elements, if

necessary.

- `cacheWatcherRuleSourceSuffix` (default: `((\?\S+)?)`)
The suffix to be removed from a `<from>`-rule element when parsing the rules from `urlrewrite.xml` for the repository watchers. See also the [Beautifying URLs](#) section from the developer's guide,
- `cacheClearOnShutdown` (default: `true`)
Set to `true` to clear the cache on a site shutdown/reload.

5. User Management

AppNG offers extensive possibilities for user administration.

5.1. User Types

In general a distinction is made between **local users**, **single LDAP users** and **LDAP groups**.

For local users, you have full control over all the user's details, like if he may or must (not) change his password.

With setting the platform properties

- `forceChangePassword` to `true`
- `passwordMaxValidity` to a value greater zero (unit: days)

a local user can be forced to change his password after a certain time.

By setting the platform property `inactiveLockPeriod` to a value greater zero (unit: days), users can be locked automatically due to inactivity.

5.2. LDAP Connectivity

It's possible to configure LDAP connectivity on a site level.

For details, check out the [appNG Manager User Guide](#).

5.3. Password Policies

Using the multiline platform property `configurablePasswordPolicy`, it is possible to exactly define how a valid password must be built upon.

These are available settings to do so:

- `minLowerCase` (default: `1`)
The minimum number of lowercase letters (a-z).
- `minUppercase` (default: `1`)
The minimum number of uppercase letters (A-Z).

- `minDigits` (default: 1)
The minimum number of digits (0-9).
- `minSpecialChars` (default: 1)
The minimum number of special characters (see `allowedSpecialChars`).
- `allowedSpecialChars` (default: `!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~`)`
The allowed special characters,
- `minLength` (default: 8)
The minimum length of the password.
- `maxLength` (default: 255)
The maximum length of the password.
- `useHistory` (default: `true`)
When changing the password, make sure it differs from the current one.
- `useUsername` (default: `true`)
Make sure the password does not contain the username, also in reverse order and ignoring case.
- `numCharacterGroups` (default: 4)
The minimum number of different character groups that must be used for a password.
- `allowOtherCharacters` (default: `false`)
Whether or not to allow other characters than letters, digits and the defined special characters.
- `allowWhiteSpace` (default: `false`)
Whether or not to allow whitespaces.
- `generateLength` (default: 8)
The length of a generated password.
- `generateLowercase` (default: 3)
The number of lowercase letters (a-z) for a generated password.
- `generateUppercase` (default: 3)
The number of uppercase letters (A-Z) for a generated password.
- `generateDigits` (default: 1)
The number of digits (0-9) for a generated password.
- `generateSpecialChars` (default: 1)
The number of special characters for a generated password.

Consequently, the implicit default configuration looks as follows:

```
minLowerCase = 1
minUppercase = 1
minDigits = 1
minSpecialChars = 1
allowedSpecialChars = !"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~
minLength = 8
maxLength = 255
useHistory = true
useUsername = true
numCharacterGroups = 4
allowOtherCharacters = false
allowWhiteSpace = false
generateLength = 8
generateLowerCase = 3
generateUppercase = 3
generateDigits = 1
generateSpecialChars = 1
```

6. Command Line Interface (CLI)

There is a comprehensive set of CLI Commands that can be used to configure the appNG Platform.

- Sites
 - [list-sites](#)
 - [create-site](#)
 - [check-site](#)
 - [delete-site](#)
 - [site-setactive](#)
 - [reload-site](#)
- Applications
 - [list-applications](#)
 - [install-application](#)
 - [activate-application](#)
 - [deactivate-application](#)
 - [delete-application](#)
- Users
 - [list-subjects](#)
 - [create-subject](#)
 - [delete-subject](#)
- Groups

- [list-groups](#)
- [create-group](#)
- [add-group](#)
- [delete-group](#)
- Roles
 - [list-roles](#)
 - [add-role](#)
- Permissions
 - [list-permissions](#)
 - [add-permission](#)
 - [remove-permission](#)
- Repositories
 - [list-repositories](#)
 - [create-repository](#)
 - [delete-repository](#)
- Templates
 - [install-template](#)
 - [delete-template](#)
- Properties
 - [list-properties](#)
 - [create-property](#)
 - [update-property](#)
 - [delete-property](#)
- Others
 - [batch](#)
 - [heartbeat](#)

7. appNGizer

A REST-style API for configuring the platform is provided by appNGizer.

Check out the following resources:

- [appNGizer User Manual](#)
- [appNGizer Setup Guide](#)
- [appNGizer Platform installation guide](#)

8. Monitoring

The appNG platform offers some built-in monitoring services to check the state of a site and to give some information about the system appNG is running on.

8.1. Configuration

The path to health monitoring can be configured with the platform property `monitoringPath`, using the default value `/health`.

The monitoring path is secured with basic authentication.

The user name is `monitoring`. The password can be defined using the platform property `monitoringPassword`. If not defined, the platform's `sharedSecret` is used.

8.2. Services

Monitoring offers different services, as listed here. All of them use `application/json` as a content-type, making it easy for load balancers like `nginx` or `HAProxy` to make use of them.

8.2.1. Site health

Shows the current site, including its state (one of: `STARTED`, `STARTING`, `STOPPING`, `STOPPED`, `INACTIVE`), applications and properties.

Path: `/health`

Example Response:

```

{
  "name" : "manager",
  "state" : "STARTED",
  "host" : "localhost",
  "domain" : "http://localhost:8080",
  "startupTime" : "2019-12-10T10:46:52.792+01:00",
  "uptimeSeconds" : 2132133465,
  "applications" : {
    "appng-scheduler" : {
      "version" : "1.12.0",
      "description" : "Scheduling using Quartz Scheduler",
      "hidden" : false,
      "privileged" : true,
      "filebased" : true,
      "jars" : [ {
        "name" : "/home/appng/webapps/ROOT/WEB-INF/cache/platform/manager/appng-
scheduler/lib/appng-scheduler-1.12.0.jar",
        "lastModified" : "2019-12-10T10:46:36+01:00"
      } ]
    },
    "appng-manager" : {
      "version" : "1.16.0",
      "description" : "Global appNG administration",
      "hidden" : false,
      "privileged" : true,
      "filebased" : true,
      "jars" : [ {
        "name" : "/home/appng/webapps/ROOT/WEB-INF/cache/platform/manager/appng-
manager/lib/appng-manager-1.17.0-SNAPSHOT.jar",
        "lastModified" : "2019-12-10T10:46:38+01:00"
      } ]
    },
    "appng-authentication" : {
      "version" : "1.12.0",
      "description" : "Authentication Application",
      "hidden" : true,
      "privileged" : true,
      "filebased" : true,
      "jars" : [ {
        "name" : "/home/appng/webapps/ROOT/WEB-INF/cache/platform/manager/appng-
authentication/lib/appng-authentication-1.12.1-SNAPSHOT.jar",
        "lastModified" : "2019-12-10T10:46:36+01:00"
      } ]
    }
  },
  "props" : {
    ...
  }
}

```

8.2.2. Platform JARs

Shows a list of all Jars offered by the platform.

Path: `/health/platform`

8.2.3. Site JARs

Shows a list of all Jars that the site's classloader is built from.

Path: `/health/jars`

8.2.4. System environment

Shows the system's environment as returned by `System.getenv()`.

Path: `/health/environment`

8.2.5. System properties

Shows the system's properties as returned by `System.getProperties()`.

Path: `/health/system`

8.3. Opentelemetry metrics

Offers some request-based [OpenTelemetry](#) metrics that can be consumed by [Prometheus](#) and visualized with [Grafana](#).

Path: `/health/metrics`

The available [histogramm](#) metrics are:

- `<site>::<app>::<type>::act:<event>::<action>`
- `<site>::<app>::<type>::dat:<datasource>`
- `<site>::<app>::<type>`

The possible values for `<type>` are:

- `gui`
Content provided by the appNG GUI
- `jsp`
A JSP file served from a content repository
- `static`
A static file served from a content repository

- `<site>::<app>::<type>::<service>`

The possible values for `<type>` are:

- `webservice`
With `<service>` being an implementation of `org.appng.api.Webservice`

- soap
With `<service>` being an implementation of `org.appng.api.SoapService`
- rest
With `<service>` being an implementation a `org.springframework.web.bind.annotation.Controller`, which includes calls to the [appNG OpenAPI](#)