appNG Platform Administration Guide

Matthias Müller

Version 1.23.1 created on 2021-11-03

Table of Contents

1. Scope . . . 1
2. Setup . . . 1
   2.1. Requirements . . . 1
   2.2. Separating ${appngData} . . . 1
3. Clustering . . . 1
   3.1. Using Multicast . . . 1
   3.2. Using Hazelcast . . . 1
   3.3. Using RabbitMQ . . . 2
   3.4. Using Redis . . . 2
4. Caching . . . 2
   4.1. Platform configuration . . . 2
   4.2. Site configuration . . . 3
5. User Management . . . 4
   5.1. User Types . . . 4
   5.2. LDAP Connectivity . . . 5
   5.3. Password Policies . . . 5
6. Command Line Interface (CLI) . . . 6
7. appNGizer . . . 8
8. Monitoring . . . 8
   8.1. Configuration . . . 8
   8.2. Services . . . 8
      8.2.1. Site health . . . 8
      8.2.2. Platform JARs . . . 10
      8.2.3. Site JARs . . . 10
      8.2.4. System environment . . . 10
   8.3. System properties . . . 10

1. Scope

2. Setup

2.1. Requirements

2.2. Separating ${appngData}

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`
- `rabbitMQAdresses`: 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 it's 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`
  Set to `true` to enable caching for the site.
- `cacheExceptions`
  URL path prefixes which are never cached, as a multiline value. Contains one prefix per line.
  Example:

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

- `cacheTimeouts`
  The path specific cache timeouts, as a multiline value. The format is `path-prefix = TTL in seconds`. Contains one path-prefix per line.
  Example:

  ```
  /service/appng/cached-application = 7200
  /service/thesite = 3600
  ```

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

8.3. System properties

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

Path: `/health/system`