Installation of CzechIdM - Windows

installation, jdbc, quickstart, encryption

We presume that the server is prepared as described in Server preparation - Windows.

This tutorial shows how to install full production-ready version of CzechIdM on standard software setup (java, postgreSQL, Tomcat, Apache httpd). If you are looking for a demo installation please see Getting Started.

Create DB user and database in PostgreSQL

Open a **PSQL** binary from the Start menu (for the OpenSCG PostgreSQL) or fire-up the cmd terminal and run `psql.exe -U postgres` (for the EnterpriseDB PostgreSQL). A windows-cmd-like window should appear with a prompt. Create a db user and a database for CzechIdM.

```
CREATE USER czechidm PASSWORD '*****';

-- Choose appropriate collation and create database.
-- with english collation (we expect the default windows installation with
cp1250/cp1252 and "English_United States" collation).
CREATE DATABASE "czechidm" WITH OWNER 'czechidm' ENCODING 'UTF8' template
'template0';
-- with czech collation
CREATE DATABASE "czechidm" WITH OWNER 'czechidm' ENCODING 'UTF8' LC_COLLATE
= 'czech_czech' LC_CTYPE = 'czech_czech' template 'template0';
```

Use the pgAdmin or PSQL to test the database connection under the `czechidm` user.

JDBC driver installation

Download the newest PostgreSQL JDBC driver(version 42.2.6 and newer) from the this URL and move it to the `C:\Program Files\Apache Software Foundation\Tomcat 9.0\lib\` directory.

Configure environment properties. Select application profile

Run the **Monitor Tomcat** application from the Start menu. Configure following settings:

- Add `C:\CzechIdM\etc;C:\CzechIdM\lib;C:\CzechIdM\lib\*;` to the **beginning of the**CLASSPATH. If you followed the Server preparation - Windows guide, this should already be in place.
- Add `-Dspring.profiles.active=production` and `-Dlog4j2.formatMsgNoLookups=true` to the `Java options`.
- Add `-Djava.security.egd=file:/dev/urandom` to the `Java options`.

Create CzechIdM configuration folders

In CzechIdM, you can store all deployment-specific configuration (i.e. database credentials) outside the war file. This is a configure-once approach which greatly simplifies future deployments.

- The **etc** directory stores configuration files.
- The **lib** directory stores additional jar libraries such as database drivers.
- The **backup** directory stored Groovy scripts backups.
- The **data** directory stores various user-attached files.

Create the directory structure:

```
C:\CzechIdM
C:\CzechIdM\etc
C:\CzechIdM\lib
C:\CzechIdM\backup
C:\CzechIdM\data
```

Create SSL truststore

Open the Git Bash and navigate to the `/c/czechidm/etc`. Then create fake certificate which will be, for this time, the only certificate in the truststore.

```
openssl genrsa -out fakecert.key
# if the following command fails, remove the parameter -subj and supply the values interactively
openssl req -new -key fakecert.key -out fakecert.csr -subj "//C=CZ\ST=Czech Republic\L=Prague\O=BCV\CN=CzechIdM placeholder cert"
openssl x509 -req -in fakecert.csr -signkey fakecert.key -days 1 -sha256 -out fakecert.crt
keytool -importcert -file fakecert.crt -alias placeholder-cert -keystore truststore.jks
    Enter keystore password:  ENTER SOME PASSWORD HERE AND REMEMBER IT FOR LATER
    Re-enter new password:
    ...
    Trust this certificate? [no]:  yes
    Certificate was added to keystore

rm fakecert.key fakecert.csr fakecert.crt
```

Then adjust Tomcat configuration - the `JAVA_OPTS` - as you did before. Add path to the truststore `-Djavax.net.ssl.trustStore=C:/CzechIdM/etc/truststore.jks` and truststore password `-Djavax.net.ssl.trustStorePassword=THE PASSWORD YOU ENTERED WHEN CREATING KEYSTORE`.

Save the configuration and restart the Tomcat for changes to take effect.

Create CzechIdM configuration

Now we will create configuration files the CzechIdM will use.

Code snippets in this chapter can be **mostly** copy-pasted or (but please read through whole chapter to be aware of setting you have to adjust). Configuring the CzechIdM is about altering four or five lines altogether.

- The **C:\CzechIdM\etc\secret.key** is a file with confidential storage secret key. This key has to have 128 bit (= 16 bytes). Creation of the **secret.key** is a bit tricky (because Windows). Open the Git Bash, run the **vim** editor and type the key into the file. Then check its format.

```
cd /c/czechidm/etc
# start the vim editor
vim secret.key
# press "i" to switch to input mode
# type the 16 characters of the secret key
# press ESC to switch to command mode
# type :wq
# press ENTER
# now you should see that secret.key file has been created, check its
contents
# the file should be EXACTLY 17 BYTES LONG, 16 bytes for your key and the
last byte "0a"
xxd -p secret.key
... hex dump here ... text dump here ...
... 0a ...
```

- The **C:\CzechIdM\etc\quartz-production.properties** file stores, the Quartz scheduler configuration. You can use the following snippet as a production-safe configuration file.

quartz-production.properties

```
org.quartz.scheduler.instanceName=idm-scheduler-instance
org.quartz.scheduler.instanceId=AUTO
org.quartz.scheduler.skipUpdateCheck=true
org.quartz.threadPool.class=org.quartz.simpl.SimpleThreadPool
org.quartz.threadPool.threadCount=10
org.quartz.threadPool.threadPriority=4
org.quartz.jobStore.class=org.quartz.impl.jdbcjobstore.JobStoreTX
org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.Po
stgreSQLDelegate
org.quartz.jobStore.useProperties=false
org.quartz.jobStore.misfireThreshold=60000
org.quartz.jobStore.tablePrefix=qrtz_
```

- The **C:\CzechIdM\etc\logback-spring.xml** specifies logging configuration. This is the default logging configuration that you can use out of the box.

logback-spring.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- https://springframework.guru/using-logback-spring-boot/ -->
<!-- http://logback.qos.ch/manual/appenders.html -->
<configuration>
        <!-- !!!BEWARE!!! The specification of the LOG PATTERNS overrides
the default configuration and increases the maximum length of the
%logger{<size>} attribute.
        It is neccessary for correct function of the AUDIT logging feature
(redmine ticket #2717). If AUDIT logger key is longer then the set limit it
gets shortened
        and SIEM software is not able to parse logs properly. -->
        <property name="CONSOLE_LOG_PATTERN" value="%d{yyyy-MM-dd
HH:mm:ss.SSS} %5level %relative --- [%thread] %logger{60}.%M : %msg%n"/>
        <property name="FILE_LOG_PATTERN" value="%d{yyyy-MM-dd HH:mm:ss.SSS}
%5level %relative --- [%thread] %logger{60}.%M : %msg%n"/>

        <springProperty name="spring.profiles.active"
source="spring.profiles.active"/>
        <include resource="org/springframework/boot/logging/logback/file-
appender.xml"/>
        <include
resource="org/springframework/boot/logging/logback/defaults.xml"/>
        <springProfile name="production">

                <springProperty name="spring.datasource.driver-class-name"
source="spring.datasource.driver-class-name"/>
                <springProperty name="spring.datasource.url"
source="spring.datasource.url"/>
                <springProperty name="spring.datasource.username"
source="spring.datasource.username"/>
                <springProperty name="spring.datasource.password"
source="spring.datasource.password"/>

                <appender name="DB"
class="eu.bcvsolutions.idm.core.exception.IdmDbAppender">
                    <connectionSource
class="ch.qos.logback.core.db.DriverManagerConnectionSource">
                      <driverClass>${spring.datasource.driver-class-
name}</driverClass>
                      <url>${spring.datasource.url}</url>
                      <user>${spring.datasource.username}</user>
                      <password>${spring.datasource.password}</password>
                    </connectionSource>
                </appender>

                <appender name="DB_ASYNC"
class="ch.qos.logback.classic.AsyncAppender">
                <appender-ref ref="DB" />
                     <includeCallerData>true</includeCallerData>
```

```
                </appender>

                <logger name="eu.bcvsolutions" level="INFO">
                     <appender-ref ref="DB_ASYNC" />
                </logger>

                <logger name="org.springframework" level="INFO"/>
                <logger name="org.springframework.web" level="INFO"/>
                <logger name="org.hibernate.SQL" level="INFO"/>
                <logger name="org.hibernate.type.descriptor.sql.BasicBinder"
level="INFO"/>
                <logger name="AUDIT" level="INFO"/>
                <appender name="idm"
class="ch.qos.logback.core.rolling.RollingFileAppender">
                        <encoder>
                                <pattern>
                                        %d{yyyy-MM-dd HH:mm:ss.SSS} %5level
%relative --- [%thread] %logger{36}.%M : %msg%n
                                </pattern>
                        </encoder>
                        <file>
                                logs/catalina.log
                        </file>
                        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
                                <fileNamePattern>logs/catalina.%d{yyyy-MM-
dd}.log</fileNamePattern>
                                <maxHistory>90</maxHistory>
                        </rollingPolicy>
                </appender>
                <root level="INFO">
                        <appender-ref ref="idm"/>
                </root>
        </springProfile>
</configuration>
```

- The most important file is **C:\CzechIdM\etc\application-production.properties** (application-PROFILE.properties, where the PROFILE is the profile you run the IdM under). You can use most of the file as-is, there is a bit of configuration needed though. This is a template file:

application-production.properties

```
# Doc: https://wiki.czechidm.com/devel/dev/configuration/backend

idm.pub.app.instanceId=idm-primary
idm.pub.app.stage=production

spring.datasource.url=jdbc:postgresql://localhost:5432/czechidm
spring.datasource.username=czechidm
spring.datasource.password=********** TODO *********
spring.datasource.driver-class-name=org.postgresql.Driver
```

```
spring.datasource.validationQuery=SELECT 1
spring.datasource.test-on-borrow=true
spring.jpa.generate-ddl=false
spring.jpa.hibernate.ddl-auto=none
flyway.enabled=true

scheduler.properties.location=quartz-production.properties

logging.config=c:/czechidm/etc/logback-spring.xml

idm.sec.core.demo.data.enabled=false

# attachments will be stored under this path.
# new directories for attachment will be created in this folder
(permissions has to be added)
# System.getProperty("user.home")/idm_data will be used if no path is
given
idm.sec.core.attachment.storagePath=c:/czechidm/data
# configuration property for default backup
idm.sec.core.backups.default.folder.path=c:/czechidm/backup

idm.pub.security.allowed-origins=http://localhost
# Generate JWT token security string as "cat /dev/urandom | tr -dc 'a-
z0-9' | head -c VALUE" where VALUE can be from 1 to 255.
# We recommend the VALUE to be at least 25.
idm.sec.security.jwt.secret.token=********** TODO *********
idm.sec.security.jwt.expirationTimeout=36000000

# Cipher secret key for crypt values in confidential storage
# for crypt values is used secretKey or secretKey defined by file -
secretKeyPath
#cipher.crypt.secret.key=XXXXXXXXXXXXXXX
cipher.crypt.secret.keyPath=c:/czechidm/etc/secret.key

# Defaults for: emailer.*
# test.enabled=true means mail WILL NOT be sent
idm.sec.core.emailer.test.enabled=true
# http://camel.apache.org/mail.html
idm.sec.core.emailer.protocol=smtp
idm.sec.core.emailer.host=something.tld
idm.sec.core.emailer.port=25
# idm.sec.core.emailer.username=czechidm@domain.tld
# idm.sec.core.emailer.password=password
idm.sec.core.emailer.from=czechidm@localhost

# Default user role will be added automatically, after an identity is
logged in
# could contains default authorities and authority policies
configuration
# for adding autocomplete or all record read permission etc.
idm.sec.core.role.default=userRole
```

```
# Admin user role
idm.sec.core.role.admin=superAdminRole

# Max file size of uploaded file. Values can use the suffixed "MB" or
"KB" to indicate a Megabyte or Kilobyte size.
spring.servlet.multipart.max-file-size=100MB
spring.servlet.multipart.max-request-size=100MB
```

Adjust database configuration

If you followed this howto, the only thing you should need to adjust is a **spring.datasource.password** propetry. Set it to the password for czechidm user in PostgreSQL. If necessary, adjust other database connection properties...

```
spring.datasource.url=jdbc:postgresql://localhost:5432/czechidm
spring.datasource.username=czechidm
spring.datasource.password=********** TODO *********
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.validationQuery=SELECT 1
spring.datasource.test-on-borrow=true
```

Generate JWT token

Set value of the **idm.sec.security.jwt.secret.token** property as is described in the template file:

```
# Generate JWT token security string as "cat /dev/urandom | tr -dc 'a-z0-9'
| head -c VALUE" where VALUE can be from 1 to 255.
# We recommend the VALUE to be at least 25.
idm.sec.security.jwt.secret.token=********** TODO *********
```

Local confidential storage

Local confidential storage is encrypted by AES algoritm. Read more. Confidential storage is encrypted by a key found in **secret.key** file you already created.

There are two properties in application-production.properties that influence the confidential storage:

- You can set the 128bit (16byte) or 256bit (32byte) key directly in the property file using **cipher.crypt.secret.key** property or
- you can create separate file (in our case **secret.key**) containing a random string. Then you reference this file with **cipher.crypt.secret.keyPath** property.

On Windows, you have to use separate file **secret.key**.

CzechIdM doesn't contain any default key for crypt confidential storage. Please define it before you start using the IdM.

Confidential storage uses AES/CBC/PKCS5Padding (more info) algorithm which operates with 128bit or 256bit key.

Attachment store

In CzechIdM, users can sometimes add attachments (say, attach *.jpeg photo to their employee card request). Those files are stored in the attachment store. With the following property, you can configure, where the store is. If you used sample property file, the store is by-default located under `C:\CzechIdM\data`.

```
# attachments will be stored under this path.
# new directories for attachment will be created in this folder (permissions has to be added)
# System.getProperty("user.home")/idm_data will be used if no path is given
idm.sec.core.attachment.storagePath=c:/czechidm/data
```

Environment

If you install CzechIdM in multiple environments (typically test and production), you can display a label in the navigation bar which will tell the users in which environment they work. The default value "production" doesn't display any label. Set the value to `test` to display the label marking the Test environment.

```
# Application stage (development, test, production (default))
idm.pub.app.stage=production
```

Deploy the CzechIdM

CzechIdM is deployed as a WAR archive.

- Download the latest CzechIdM WAR archive.
- Stop the Tomcat service.
- Renamed it to `idm.war` and deploy it to the stopped Tomcat server (to the `webapps` folder).
- Start the Tomcat container and it will deploy the CzechIdM application. CzechIdM will load its configuration from the `C:\CzechIdM\etc` directory automatically.

Change default admin password

In the fresh CzechIdM installation, there is one user identity - **admin** with password **admin**. Right after you install the application, go to https://yourserver.tld/idm and change the default password.

Configure IdM

Follow some final configuration steps: Installation of CzechIdM - Final steps.