Configuration - backend

configuration, final, property, properties, config, setup

The application uses a Spring boot configuration in the `application.properties` files. All the configuration items which are used solely for idm begin with `idm.` prefix. The configuration items from the file can be overloaded through a setting agenda in the gui ⇒ a server restart isn't needed for changing the configuration with `idm.` prefix, which was one of the main goals. The configuration is saved in the database. Use `ConfigurationService` for reading and saving configuration items.

Naming conventions of the configuration items in idm:

- `idm.` - configuration items for the needs of idm
- `idm.pub.` - public configuration items - published on a public rest endpoint (e.g. version)
- `idm.sec.` - system configuration items - published on a secure rest endpoint and available for configuration by the application administrator. They are used for backend configuration. If configuration item is confidential, then value is stored in confidential storage and value is not send to frontend, application logs etc. Items with key `password`, `token`, `secret` are automatically set as confidential - use it for configuration items defined in property file only.
- `idm.sec.<module>.` or `idm.pub.<module>.` - configuration items of the given module. Use `ModuleDescriptor#getId()` as <module>.
- if the name of a configuration item contains the `password` or `token` chain, the value of the configuration item is hidden in the rest interface listing (or rather replaced with substitute characters).
- It is better to use constants for keys, e.g. `ConfigurationService.IDM\_PUBLIC\_PROPERTY\_PREFIX +` `"core.identity.delete"` - using seperator constant in key name suffix is not preferred - constant can be simply found by key suffix ("ctrl-f" + "core.identity.delete").

Configure environment properties

Application profiles

We are using Spring profiles: Doc.

Start server under defined profile (add JAVA_OPTS parameters):

```
-Dspring.profiles.active=production
```

Configured devstack profiles

- `default` - the default profile - configured to db h2. If a developer downloads the project from Git, the application will run without any other configuration over h2 database with demo data (by admin user ...). Default profile is used for issuing a demo.
- `dev` - developing profile configured to postgresql. In the future, we can move the configuration itself to special profiles - their combinations (e.g. test+ postgresql or dev + mysql). We will be able to cover more variants of environment versus database.

- `test` - test profile - configured to db h2 and **it's used for unit and intergration testing only**. Don't use this profile for test environment - create your own profiles (testing / production).
- `release` - release profile - all modules in CzechIdM repository are included, they are released together under one version.

External configuration

External configuration uses Spring: [Doc](#).

Start server with external path to configuration ([add JAVA_OPTS parameters](#)):

```
--
spring.config.location=classpath:/default.properties,classpath:/override.pro
perties
```

Environment properties

[Add JAVA_OPTS parameters](#)

Configuration items

Application/ Server

In the application profile (application.properties) and overloadable via ConfigurationService.

```
# Application stage (development, test, production (default))
#
# Public properties - available for frontend without authentication (show
information about app, decorators etc.).
#
# Application stage - development, test, production.
idm.pub.app.stage=
# Application instance / server id - is used for scheduler etc.
# Can be defined in property file only! Overidding via ConfigurationService
is not possible for application instance (~ more instanceos on the same
database)
idm.pub.app.instanceId=idm-primary
# Frontend server url.
# E.g. http://localhost:3000
# Default: The first 'idm.pub.security.allowed-origins' configured value is
used (~ backward compatible).
# @since 12.0.0
idm.pub.app.frontend.url=
# Backend server url.
```

```
# E.g. http://localhost:8080/idm
# Default: Url is resolved dynamically from current servlet request.
# @since 12.0.0
idm.pub.app.backend.url=

# global date format on BE. Used in notification templates, logs, etc. FE
uses localization key 'core:format.date'.
idm.pub.app.format.date=dd.MM.yyyy
# global datetime format on BE. Used in notification templates, logs, etc.
FE uses localization key 'core:format.datetime'.
idm.pub.app.format.datetime=dd.MM.yyyy HH:mm
# Show identifiers (uuid) in frontend application. Empty value by default =>
identifier is shown, when application 'idm.pub.app.stage' is set to
'development'.
idm.pub.app.show.id=
# Show transaction identifiers (uuid) in frontend application.
idm.pub.app.show.transactionId=false
# Show role environment in frontend application for roles (table, role
detail, niceLabel, info components, role select).
idm.pub.app.show.environment=true
# Show role baseCode in frontend application for roles (table, role detail,
niceLabel, info components, role select).
idm.pub.app.show.role.baseCode=true
# Rendered column in role table agenda. Comma is used as separator. Order of
rendered columns is preserved as configured.
# Available columns:
# - name - role name info card with link to detail
# - baseCode - role base code (without environment)
# - environment - role environment
# - disabled
# - description
idm.pub.app.show.role.table.columns=name, baseCode, environment, disabled,
description
# Show role catalogue item code in role catalogue tree
idm.pub.app.show.roleCatalogue.tree.code=false
# Number of items (pagination) in role catalogue tree in root level. Used on
role select and agenda.
idm.pub.app.show.roleCatalogue.tree.pagination.root.size=25
# Number of items (pagination) in role catalogue tree in other levels. Used
on role select and agenda.
idm.pub.app.show.roleCatalogue.tree.pagination.node.size=25
# Number of items (pagination) in tree node structure in root level.
idm.pub.app.show.treeNode.tree.pagination.root.size=50
# Number of items (pagination) in tree node structure in other levels.
idm.pub.app.show.treeNode.tree.pagination.node.size=50
# Available size options for tables in frontend application
idm.pub.app.show.sizeOptions=10, 25, 50, 100
# Show buttons for bulk actions in tables (0 = select box will be shown
only).
# Count of quick access buttons for bulk actions in tables - the first count
of bulk actions will be shown as button - next action will be rendered in
```

drop down select box.
 # Bulk action icon is required for quick access button - action without icon will be rendered in select box.
 # Bulk action can enforce showing in quick access button (by bulk action configuration).
`idm.pub.app.show.table.quickButton.count`=`5`
 # Quick button for bulk actions in tables will be included in drop down select box too (available as button + menu item with text).
 # Number of selected record is shown in drop down select header.
`idm.pub.app.show.table.quickButton.menuIncluded`=`true`
 # Show default form for newly created user.
 # Default form can be disabled => at least one configured form projection is needed.
`idm.pub.app.show.identity.formProjection.default`=`true`
 # Rendered column in identity table agenda. Comma is used as separator. Order of rendered columns is preserved as configured.
 # Available columns:
 # - username - username with link to detail
 # - entityinfo - identity info card
 # - lastName
 # - firstName
 # - externalCode - personal number
 # - email
 # - state
 # - passwordexpiration - information about identity password epiration
 # - description
 # Note: Table in identity agenda can be configured with this property (common identity table with columns is not specified on FE).
 # If you want to configure rendered columns for all tables generalized from identity table (e.g. on role or tree node detail),
 # you can use FE configuration https://wiki.czechidm.com/devel/documentation/application_configuration/dev/frontend
`idm.pub.app.show.identity.table.columns`=`username, lastName, firstName, externalCode, email, state, description`
 # Rendered columns in user roles agenda (Directly assigned roles). Comma is used as separator. Order of rendered columns is preserved as configured.
`idm.pub.app.show.identityRole.table.columns`=`role, roleAttributes, environment, owner, contractPosition, validFrom, validTill, directRole, automaticRole, incompatibleRoles, description, priority`
 # Rendered columns in role requests in the table for assigned roles. Comma is used as separator. Order of rendered columns is preserved as configured.
`idm.pub.app.show.role.request.table.columns`=`name, description, roleAttributes, contractPosition, validFrom, validTill, directRole, automaticRole, action, priority`
 # If is true, then role-request description will be show on the detail.
 # Description will hidden if this property will be false and role request
*# doesn't contains any value in description (can be filled during the approval process).*

```
idm.pub.app.show.roleRequest.description=true
# Show logout content (~ page) with message, after user is logged out.
# @since 12.0.0
idm.pub.app.show.logout.content=false
#
# Configurable application theme
# @since 12.0.0
idm.pub.app.show.theme={ "palette": { "type": "light", "primary":       {
"main": "#5cb85c", "contrastText": "#fff" }, "secondary":         { "main":
"#f50057", "dark": "#c51162", "contrastText": "#fff" }, "success":        {
"main": "#4caf50", "contrastText": "#ffffff" }, "warning":        { "main":
"#ff9800", "contrastText": "#fff" }, "action": {"loading": "rgba(255, 255,
255, 0.7)"}, "background":        { "default": "#fafafa", "paper": "#fff" }
}, "shape": {"borderRadius": 3} }
#
# Configurable application logo (attachment uuid identifier)
# Recommended logo size is 165 x 40 px.
# @since 12.0.0
idm.pub.app.show.logo=
# Footer help link url.
# @since 12.0.0
idm.pub.app.show.footer.help.link=https://wiki.czechidm.com/start
# Footer service desk link url.
# @since 12.0.0
idm.pub.app.show.footer.serviceDesk.link=https://redmine.czechidm.com/projec
ts/czechidmng
#
# Private properties - used on backend only.
#
# Create demo data at application start.
idm.sec.core.demo.data.enabled=true
# Demo data was created - prevent to create demo data duplicitly.
idm.sec.core.demo.data.created=false
# Create init data at application start. Init data (product provided roles)
are updated automatically with pruct updates.
# Set property to false to disable init data creation and updates.
idm.sec.core.init.data.enabled=true
```

Change server for asynchronous processing (switch application instance)

@since 11.1.0

Application instance (server) is used for asynchronus processing - for scheduled tasks, asynchronous long running tasks and events. Instance identifier can be defined in the application profile (application.properties) by property `idm.pub.app.instanceId`. When we want to schedule and process asynchronous tasks and event on other instace (or when one instance shutdown), then we can switch processing by provided bulk action `Change server for asynchronous processing` in configuration agenda:
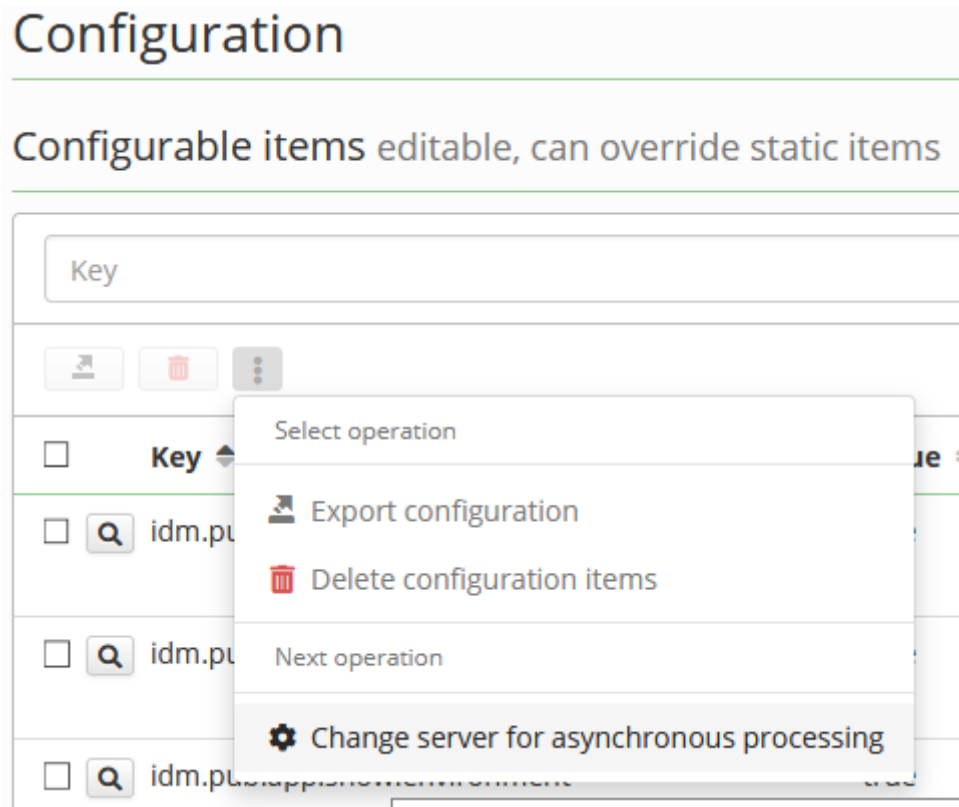
Previous and new instance identifier is required as input parameters. All scheduled tasks and all created (~ not processed) asynchronous long running tasks and events will be moved from previous to new instance and will be processed on new instance (server).

Bulk action is available for logged user with required authorities and permissions:

- CONFIGURATION_UPDATE - configuration property contains instance for asynchronous processing will be changed ⇒ authority and UPDATE base permission for property idm.sec.core.event.asynchronous.instanceId is required.
- SCHEDULER_UPDATE - scheduled tasks and created (~ not processed) asynchronous long running tasks will be changed.
- ENTITYEVENT_UPDATE - created (~ not processed) asynchronous events will be changed.

Jpa

In the application profile (application.properties)

```
# ZonedDateTime is stored in UTC
spring.jpa.properties.hibernate.jdbc.time_zone=UTC
# Driver (e.g. postgres) does not support contextual LOB creation
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
# audit table suffixes
spring.jpa.properties.org.hibernate.envers.audit_table_suffix=_a
spring.jpa.properties.org.hibernate.envers.modified_flag_suffix=_m
# modified flag for all audited columns
spring.jpa.properties.org.hibernate.envers.global_with_modified_flag=true
# prevent to modify attributes created, creator etc.
```

```
spring.jpa.properties.org.hibernate.envers.audit_strategy=eu.bcvsolutions.id
m.core.model.repository.listener.IdmAuditStrategy
spring.jpa.properties.hibernate.session_factory.interceptor=eu.bcvsolutions.
idm.core.model.repository.listener.AuditableInterceptor
# enable / disable audit (envers)
spring.jpa.properties.hibernate.listeners.envers.autoRegister=true
# Spring boot 2 changed default to true, but we are using IDENTITY
identifier generators for mssql database.
spring.jpa.hibernate.use-new-id-generator-mappings=false
#
# DB ddl auto generation by hibernate is disabled - flyway database
migration is used
spring.jpa.generate-ddl=false
spring.jpa.hibernate.ddl-auto=none
#
# DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
spring.datasource.url=jdbc:postgresql://localhost:5432/bcv_idm_storage
spring.datasource.username=*****
spring.datasource.password=*****
spring.datasource.driver-class-name=org.postgresql.Driver
# test connection, when is used from pool (reconnect after db is restarted)
spring.datasource.testOnBorrow=true
spring.datasource.validationQuery=SELECT 1
# Enlarge pool size by default. This property should be revised for each
project. Size should be configured by task and event thread pool size -
should be higher than sum of pool sizes.
spring.datasource.maximumPoolSize=50
```

Additional datasources

As of version 12.2.0 we are no longer using spring-boot datasource autoconfiguration. Instead, we define datasources ourseves. This decision was motivated by our need for multiple independent datasources with separated connection pools, which was previously not possible.

Notable changes:

There are by default two datasources configured

- datasource - default datasource, which is being used for almost all database communication (Flyway, JPA repositories)
- loggingDatasource - This datasource is used by our database logging appender to write logging messages, when databes appender is enabled. The reason why this is done by separate datasource is to prevent database logging to hog database connections and hinder the application performance

Configuration properties, that have changed with introduction of additional datasources:

- *spring.datasource.url* → *spring.datasource.jdbcUrl*

- spring.datasource.hikari.* → spring.datasource.*

Both datasources are required for the app to start.

- By default, both datasources are configured for H2 in-memory database
- If you specify property spring.datasource.jdbcUrl, IdM will no longer use in memory database for main datasource and instead it will configure connection using spring.datasource.* properties
- The same goes for loggingDatasource, which is configured using spring.logging-datasource.* properties

===== Datasource configuration properties

CzechIdM uses HikariCP to manage connections. All possible configuration properties for each datasource can be seen as fields in https://github.com/openbouquet/HikariCP/blob/master/src/main/java/com/zaxxer/hikari/HikariConfig.java class.

Developer

- If you are using `EntityManager` in your code, you will run into the issue with autowiring. In order to fix it, you need to explicitly specify, which `EntityManager` bean you want spring to autowire. You can use
  - @CoreEntityManager` annotation, if you want to autowire main application datasource (in most cases you want to use this
  - @Qualifier("coreEntityManager") annotation, if you want to autowire main application datasource and do not want to explicitly define dependency on core-api module

JNDI datasource

Firstly is needed to configure JNDI resource in the J2EE server. Here is a configuration snippet for Tomcat. It assumes PostgreSQL as the database:

```xml
<Context antiJARLocking="true" path="/idm">
    <Resource
        name="PostgresDS"
        auth="Container"
        type="javax.sql.DataSource"
        username="*****"
        password="*****"
        driverClassName="org.postgresql.Driver"
        url="jdbc:postgresql://localhost:5432/bcv_idm_storage"
        maxActive="8"
            maxIdle="4"/>
</Context>
```

In the application profile (application.properties), update datasource properties:

```
# JNDI location of the datasource. Class, url, username & password are
ignored when set.
spring.datasource.jndi-name=PostgresDS
```

In **logback-spring.xml** configuration (by profile, if db appender is used), update datasource properties:

```
...
<springProperty name="spring.datasource.jndi-name"
source="spring.datasource.jndi-name"/>

<appender name="DB"
class="eu.bcvsolutions.idm.core.exception.IdmDbAppender">
    <connectionSource class="ch.qos.logback.core.db.JNDIConnectionSource">
        <!-- please note the "java:comp/env/" prefix -->
        <jndiLocation>java:comp/env/${spring.datasource.jndi-name}</jndiLocation>
    </connectionSource>
</appender>
...
```

Using SSL

1. Configure PostgreSQL server, documentation:
   https://jdbc.postgresql.org/documentation/head/ssl.html#ssl-server
2. Short example: https://www.howtoforge.com/postgresql-ssl-certificates
3. Create new truststore specifically for the CzechIdM. When starting your Java application you must specify this keystore and password to use -Djavax.net.ssl.trustStore=path/to/mystore -Djavax.net.ssl.trustStorePassword=mypassword. For testing purposes, it is possible to set truststore password to `changeit` which is the Java default - you then have to specify only path to the truststore.

It is technically possible to import certificate into the (systemwide) Java `cacerts` truststore, but this poses significant risk.

While updating custom java deployment:

- Not being a "visible" part of IdM deployment, one can easily omit migrating certificates into the new Java `cacerts` truststore. In this case, IdM will not be able to connect anywhere where the SSL connection is used.

While updating Java OS packages:

- Nowadays, most Linux distros offering packages with OpenJDK, OracleJDK, ... use "extracted" truststore, which is basically `cacerts` truststore located somewhere under /etc/ssl/… and available to every Java distribution on the system. This truststore is constructed by an utility `update-ca-trust` from a list of CA certificates located elsewhere on the filesystem. When updating packages with

> ⚠️ JDK, ca-certs and such, the `update-ca-trust` can be invoked, effectively rewriting the extracted truststore. In this case, any changes made only to the truststore will be lost.

In the application profile (application.properties)

Update datasource properties:

```
# add ssl usage flag, see
https://jdbc.postgresql.org/documentation/head/connect.html
spring.datasource.url=jdbc:postgresql://localhost:5432/bcv_idm_storage?ssl=true
```

Cache

Cache is used for reading configuration values. Value in cache is cleared by an active (save, delete) operation.

In the application profile (application.properties):

```
# Disable cache
# If you are debugging some of code and are you figuring, something is wrong
with the cache, then you can turn the cache off with property.
#spring.cache.type=none
#
# Clusterred cache settings
#idm.sec.cache.terracota.url=localhost:9410,localhost:9420
idm.sec.cache.terracota.resource.name=main
idm.sec.cache.terracota.resource.pool.name=resource-pool
# Size in MB
idm.sec.cache.terracota.resource.pool.size=32
```

Attachment storage

`DefaultAttachmentManager` stores binary files on file system. Binary files can be attached to any entity, which implements `AttachableEntity` interface, read more.

In the application profile (application.properties):

```
# Max file size of uploaded file. Values can use the suffixed "MB" or "KB"
to indicate a Megabyte or Kilobyte size.
# Application server (e.g. Tomcat "maxSwallowSize" connector parameter) has
to be set properly too (e.g. <Connector port="8080" maxSwallowSize="-1" ...)
spring.servlet.multipart.max-file-size=100MB
spring.servlet.multipart.max-request-size=100MB
```

In the application profile (application.properties) and overloadable via `ConfigurationService`:

```
#
## Attachment manager
#
# 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=/opt/data
#
# temporary files for attachment processing (e.g. temp files for download /
upload)
# getStoragePath()/temp will be used if no path is given
# idm.sec.core.attachment.tempPath=/opt/data/temp
#
# temporary file time to live in milliseconds
# older temporary files will be purged, default 14 days
# Temporary file is used mainly for upload files internaly. When upload is
complete, then temporary file is moved into normal IdM attachment (~
temporary file is not reachable, after user session ends).
idm.sec.core.attachment.tempTtl=1209600000
```

Activiti workflow

```
# String boot properties for Activiti workflow engine
#
https://github.com/Activiti/Activiti/blob/master/modules/activiti-spring-boo
t/spring-boot-starters/activiti-spring-boot-starter-
basic/src/main/java/org/activiti/spring/boot/ActivitiProperties.java
# let activiti to manage their schema
spring.activiti.databaseSchemaUpdate=true
# disable automatic jpa entities persisting - dto usage is prefered
spring.activiti.jpaEnabled=false
# Automatic process deployment
spring.activiti.checkProcessDefinitions=true
# path to automatically deployed definitions - should be the same in all
modules
# more locations can be given e.g.
classpath*:eu/bcvsolutions/idm/workflow/,classpath:external/config/wf
# resources in the latest location has the highest priority (last wins) -
workflow definitions are prioritized by file name, don't change definition's
file name, when you want to override some core workflow definition.
# put resource, which has to override some core resource to last location
spring.activiti.processDefinitionLocationPrefix=classpath*:eu/bcvsolutions/i
dm/workflow/
# definitions name pattern - subfolders can be used
spring.activiti.processDefinitionLocationSuffixes=**/**.bpmn20.xml
```

Security

In the application profile (application.properties) and overloadable via ConfigurationService.

```
# allowed origins for FE
idm.pub.security.allowed-origins=http://localhost:3000,http://localhost
# auth token
# - expiration in milis
idm.sec.security.jwt.expirationTimeout=36000000
# - secret jwt password
idm.sec.security.jwt.secret.token=idmSecret
# - extend JWT token expiration period on each successful request
idm.sec.security.jwt.token.extend.expiration=true
# recaptcha
# - recaptchaservice endpoint
idm.sec.security.recaptcha.url=https://www.google.com/recaptcha/api/siteveri
fy
# - secret key, can be generated here https://www.google.com/recaptcha/admin
(generate V2 checkbox)
# - test secret key:
https://developers.google.com/recaptcha/docs/faq#id-like-to-run-automated-te
sts-with-recaptcha-v2-what-should-i-do
idm.sec.security.recaptcha.secretKey=xxx
# Proxy configuration for reCAPTCHA (since version 12.2.5)
idm.sec.security.recaptcha.proxy=12.34.56.78:1234
```

Allowed-origins defines, which resources can use backend API methods. e.g. When there is a web server serving as reverse proxy on the same server as BE, the http://localhost:3000 may be the right value.

Flyway

In the application profile (application.properties)

```
# Enable flyway migrations.
# @see
https://proj.bcvsolutions.eu/ngidm/doku.php?id=navrh:databazove_scripty
flyway.enabled=false
```

Module configuration (flyway-core.properties)

```
## Core Flyway configuration
#
# Whether to automatically call baseline when migrate is executed against a
non-empty schema with no metadata table.
# This schema will then be baselined with the baselineVersion before
executing the migrations.
```

```
# Only migrations above baselineVersion will then be applied.
# This is useful for initial Flyway production deployments on projects with
an existing DB.
# Be careful when enabling this as it removes the safety net that ensures
Flyway does not migrate the wrong database in case of a configuration
mistake!
flyway.core.baselineOnMigrate=true
#
# The name of Flyway's metadata table (default **schema_version**).
# By default (single-schema mode) the metadata table is placed in the
default schema for the connection provided by the datasource.
flyway.core.table=idm_schema_version_core
#
# Comma-separated list of locations to scan recursively for migrations. The
location type is determined by its prefix.
# Unprefixed locations or locations starting with classpath: point to a
package on the classpath and may contain both sql and java-based migrations.
# Locations starting with filesystem: point to a directory on the filesystem
and may only contain sql migrations.
# IdmFlywayMigrationStrategy resolves used jdbc database dynamically -
${dbName} in location could be used.
flyway.core.locations=classpath:eu/bcvsolutions/idm/core/sql/${dbName}
```

Module configuration

Information about module can be defined in property file (module-<module>.properties - e.g. module-core.properties). This property file is loaded by `PropertyModuleDescriptor`. Module properties are not editable through `ConfigurationService` (`idm.pub.` prefix is not used).

```
# mapping pom.xml properties by default
# add custom properties if needed
#
# module version
module.<module>.build.version=@project.version@
# build number
module.<module>.build.number=@buildNumber@
module.<module>.build.timestamp=@timestamp@
# module vendor
module.<module>.build.vendor=@project.organization.name@
module.<module>.build.vendorUrl=@project.organization.url@
module.<module>.build.vendorEmail=info@bcvsolutions.eu
# module description
module.<module>.build.name=@project.name@
module.<module>.build.description=@project.description@
```

Swagger

In the application profile (application.properties)

```
## Swagger config
# enable swagger endpoint (can be disabled for development etc.)
springfox.documentation.swagger.enabled=true
# endpoint with exposed documentations. Documentations are exposed by module
e.g. <server>/api/doc?group=core
springfox.documentation.swagger.v2.path=/api/doc
#
# for static documentation generation puprose => internal usage mainly in
test stage. Swagger specification on then rest endpoint is exported to given
file e.g. <module>/target/swagger/swagger.json. Properties with @ are
automatically filled from pom.xml properties.
# output directory and filename for swagger export - other build parts are
dependent on this.
springfox.documentation.swagger.outputDir=@swagger.output.dir@
springfox.documentation.swagger.outputFilename=@swagger.output.filename@
```

Emailer

In the application profile (`application.properties`) - overloadable via `ConfigurationService`.

```
# enable test mode - in this mode, emails are not send
idm.sec.core.emailer.test.enabled=false
 # http://camel.apache.org/mail.html
idm.sec.core.emailer.protocol=smtps
idm.sec.core.emailer.host=smtp.gmail.com
idm.sec.core.emailer.port=465
idm.sec.core.emailer.username=servis.bcvsolutions@gmail.com
idm.sec.core.emailer.password=*****
# The FROM email address.
idm.sec.core.emailer.from=idm@bcvsolutions.eu
```

Templates

In the application profile (application.properties) - overloadable via ConfigurationService.

```
# Templates location
# more locations can be given e.g.
classpath*:/eu/bcvsolutions/idm/template/,classpath*:/external/templates/
# resources in the latest location has the highest priority (last wins).
Resources are prioritized - put resource, which has to override some core
resource to last location
# Locations can be configured
https://docs.spring.io/spring/docs/current/spring-framework-reference/core.h
tml#resources
idm.sec.core.notification.template.folder=classpath*:/eu/bcvsolutions/idm/te
mplate/
```

```
idm.sec.core.notification.template.fileSuffix=**/**.xml # template suffix
```

Scripts

In the application profile (application.properties) - overloadable via ConfigurationService.

```
# Scripts location
# more locations can be given e.g.
classpath*:/eu/bcvsolutions/idm/scripts/,classpath*:/external/scripts/
# resources in the latest location has the highest priority (last wins).
Resources are prioritized - put resource, which has to override some core
resource to last location
# Locations can be configured
https://docs.spring.io/spring/docs/current/spring-framework-reference/core.h
tml#resources
idm.sec.core.script.folder=classpath*:/eu/bcvsolutions/idm/scripts/
idm.sec.core.script.fileSuffix=**/**.xml
```

Scheduler

In the application profile (application.properties).

```
# Enable scheduler. Enabled by default
scheduler.enabled=true
# Task queue processing period (ms). Default 1000ms.
scheduler.task.queue.process=1000
# Application settings for QUARTZ (for current mvn profile)
scheduler.properties.location=/quartz.properties
# Task executor core pool size. Uses CPU count as default.
scheduler.task.executor.corePoolSize=
# Task executor max pool size. Uses CPU corePoolSize * 2 as default.
# maxPoolSize has to be higher than corePoolSize (IllegalArgumentException
is thrown otherwise).
# When queueCapacity is full, then new threads are created from corePoolSize
to maxPoolSize.
scheduler.task.executor.maxPoolSize=
# Waiting tasks to be processed. Uses 20 as default. {@link
LinkedBlockingQueue} is used for queue => capacity is initialized
dynamically.
# {@link AbotrPolicy} is set for rejected tasks - reject exception has to be
processed by a caller ({@link LongRunningTaskManager}).
scheduler.task.executor.queueCapacity=20
# Thread priority for threads in event executor pool - 5 by default
(normal).
scheduler.task.executor.threadPriority=
# Asynchronous task processing is stopped.
# Asynchronous task processing is stopped, when instance for processing is
switched => prevent to process asynchronous task in the meantime.
```

```
# Asynchronous task processing can be stopped for testing or debugging
purposes.
# Asynchronous task are still created in queue, but they are not processed
automatically - task can be executed manually from ui.
idm.sec.core.scheduler.task.asynchronous.stopProcessing=false
# Event queue processing period (ms). Period to read prepared (~created)
asynchronous entity events from queue.
# Events are processed in batch configured by property
'idm.sec.core.event.asynchronous.batchSize'. If you events are processed
quickly (~provisioning on your environment is quick), then batch size can be
higher or this property can be lower.
# Default 500ms.
scheduler.event.queue.process=500
# Event executor core pool size. Uses CPU count + 1 as default.
scheduler.event.executor.corePoolSize=
# Event executor max pool size. Uses CPU corePoolSize * 2 as default.
# maxPoolSize has to be higher than corePoolSize (IllegalArgumentException
is thrown otherwise).
# When queueCapacity is full, then new threads are created from corePoolSize
to maxPoolSize.
scheduler.event.executor.maxPoolSize=
# Waiting events to be processed. Uses 50 as default - prevent to prepare
events repetitively and use additional threads till maxPoolSize. {@link
LinkedBlockingQueue} is used for queue => capacity is initialized
dynamically.
# {@link AbotrPolicy} is set for rejected tasks - reject exception has to be
processed by a caller ({@link EntityEventManager}).
scheduler.event.executor.queueCapacity=50
# Thread priority for threads in event executor pool - 6 by default (a
little higher priority than normal 5).
scheduler.event.executor.threadPriority=6
```

Identity

In the application profile (`application.properties`) - overloadable via `ConfigurationService`.

```
# supports delete identity. Needed on FE (=> public) to render available
bulk action in table
# @deprecated @since 10.6.0 - action can be disabled by bulk action
configurable api - use 'idm.sec.core.bulk-action.identity-delete-bulk-
action.enabled=false'.
idm.pub.core.identity.delete=true
#
# default password change type for custom users, one of values:
# DISABLED - password change is disable
# ALL_ONLY - users can change passwords only for all accounts
# CUSTOM - users can choose for which accounts change password
# Needed on FE (=> public)
```

```
idm.pub.core.identity.passwordChange=CUSTOM
#
# required old password for change password.
# Needed on FE (=> public)
idm.pub.core.identity.passwordChange.requireOldPassword=true
#
# change password to idm from public pages.
# true - change to IdM and all system
# false - change to all system except IdM
# Needed on FE (=> public)
idm.pub.core.identity.passwordChange.public.idm.enabled=true
#
# Skip identity dashboard content - show full detail directly (link from
table or from info component)
# Needed on FE (=> public)
idm.pub.core.identity.dashboard.skip=
#
# Create default identity's contract, when identity is created.
# Skipped in synchronizations - contract synchronization should be provided.
idm.sec.core.identity.create.defaultContract.enabled=true
# Creates default identity's contract with configured position name.
idm.sec.core.identity.create.defaultContract.position=Default
# Creates default identity's contract with configured state. Valid contract
will be crated by default, other possible values:
# EXCLUDED - Excluded from evidence - remains valid, but roles assigned for
this contract are not added for logged identity.
# DISABLED - Invalid by user - not changed by dates.
idm.sec.core.identity.create.defaultContract.state=
# Number of days related to current date - will be used for set contract
valid till date (current date + expiration in days = valid till).
# Contact valid till will not be set by default (~ contract expiration is
not configured by default).
idm.sec.core.identity.create.defaultContract.expiration=
#
# Profile image max file size in readable string format (e.g. 200KB).
idm.sec.core.identity.profile.image.max-file-size=512KB
```

Identity contract slice

In the application profile (`application.properties`) - overloadable via `ConfigurationService`.

```
# The protected interval can be set using the property
idm.sec.core.contract-slice.protection-interval, where the value is the
number of days.
# If the number of days between the termination of the contract and its
renewal in the following time slice is less than or equal to the number
# of days set in the protection interval, then the date of the contract
validity from the following slice will be used instead of the date of
# termination of the contract from the currently valid slice.
```

```
idm.sec.core.contract-slice.protection-interval=0
```

Role

In the application profile (`application.properties`) - overloadable via `ConfigurationService`.

```
#
# 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.
# Role full code should be given (should contain environment, if it is
used).
# Role authorities are updated automatically, when new IdM version is
installed.
idm.sec.core.role.default=userRole
#
# Admin user role
# Role full code should be given (should contain environment, if it is
used).
# Role authorities are updated automatically, when new IdM version is
installed.
idm.sec.core.role.admin=superAdminRole
#
# Helpdesk user role
# Role full code should be given (should contain environment, if it is
used).
# Role authorities are updated automatically, when new IdM version is
installed.
idm.sec.core.role.helpdesk=helpdeskRole
#
# User manager role
# Role full code should be given (should contain environment, if it is
used).
# Role authorities are updated automatically, when new IdM version is
installed.
idm.sec.core.role.userManager=userManagerRole
#
# Role manager role - role guarantee
# Role full code should be given (should contain environment, if it is
used).
# Role authorities are updated automatically, when new IdM version is
installed.
idm.sec.core.role.roleManager=roleManagerRole
#
# Virtual system implementer  role - product provided role for implementers
(approve vs request etc.).
# Role full code should be given (should contain environment, if it is
```

```
used).
# Role authorities are updated automatically, when new IdM version is
installed.
idm.sec.vs.role.implementer=virtualSystemImplementerRole
#
# Separator for the suffix with environment used in role code.
# Look out: when separator is changed, then all roles should be updated
(manually from ui, by scripted LRT or by change script).
idm.sec.core.role.codeEnvironmentSeperator=|
```

Tree

Tree structures configuration properties.

In the application profile (`application.properties`) - overloadable via `ConfigurationService`.

```
# Default tree type (uuid or code). More in Default organizational structure
doc.
idm.sec.core.tree.defaultType=
# Default tree node (uuid) - is used, when default contract is created. More
in Contractual relationship doc.
idm.sec.core.tree.defaultNode=
```

Internal properties used for tree indexing (forest index) - holds index state:

```
# forest index is valid. Is set to false, when index exception occurs and
tree index has to be rebuild
idm.sec.core.treeType.<tree-code>.valid=true
# rebuild index in progress (true). When tree type index rebuild is in
progress, then tree node cannot be created / updated / deleted.
idm.sec.core.treeType.<tree-code>.rebuild=false
```

Entity events

In the application profile (`application.properties`) - overloadable via `ConfigurationService`.

```
# disable / enable asynchronous event processing. Events will be executed
synchronously, if it's disabled. Enabled by default.
idm.sec.core.event.asynchronous.enabled=true
# Asynchronous event processing is stopped.
# Event processing is stopped, when instance for processing is switched =>
prevent to process instances in the meantime.
# Asynchronous event processing can be disabled for testing or debugging
purposes.
# Events are still created in queue, but they are not processed.
idm.sec.core.event.asynchronous.stopProcessing=false
# Asynchronous events will be executed on server instance with id. Default
is the same as {@link ConfigurationService#getInstanceId()} (current server
```

```
instance).
idm.sec.core.event.asynchronous.instanceId=
# Asynchronous events will be executed in batch - batch will be split for
event with HIGH / NORMAL priority in 70% HIGH / 30% NORMAL.
# If you events are processed quickly (~provisioning on your environment is
quick), then batch size can be higher (in combination with higher
'scheduler.event.queue.process' property).
idm.sec.core.event.asynchronous.batchSize=15
```

Entity event processors

In the application profile (`application.properties`) - overloadable via `ConfigurationService`. Every processor could have his own configuration properties under prefix:

```
# disable / enable event procesor
idm.sec.<module>.processor.<name>.enabled=true
# override event types for given processor
idm.sec.<module>.processor.<name>.eventTypes=CREATE,UPDATE
```

Where `<module>` is processor's module and `<name>` is processor's name (see overridable processor's methods). Filled configuration properties will be shown on processor's content.

Common configuration properties for all processors:

- `enabled` - on / off
- `eventTypes` - list of event types (separated by comma) to which given processor reacts
- `order` - comming soon

Exists processors configuration: implemented proccessors.

Bulk actions

@since 10.6.0

In the application profile (`application.properties`) - overloadable via `ConfigurationService`. Every bulk action could have his own configuration properties under prefix:

```
# disable / enable bulk action
idm.sec.<module>.bulk-action.<name>.enabled=true
```

Where `<module>` is bulk action module and `<name>` is bulk action name.

Common configuration properties for all bulk actions:

- enabled - **true** / false.
- `order` - bulk action order (for FE only). Action provided default order in implementation.
- `icon` - Icon on frontend (for FE only). Icon libraries can be used: `component:`, `fa:`, `glyph:`.

Icon is loaded from FE locale by default.

- `level` - bulk action level ~ button and icon color (for FE only). Available options: `success` (default value), `info`, `warning`, `error`.
- `deleteAction` - true / **false** - Action deletes records (for FE only). Action will be in bottom menu section, is action is included in menu.
- `quickButton` - true / **false** - Render action as quick button (for FE only). The first available actions are rendered as buttons, if icon is defined. This configuration enforces rendering action as quick button (order is ignored).
- `quickButtonable` - **true** / false - Action can be included in quick buttons on FE. Set to **false**, when button should be not rendered ⇒ action will be rendendered in drop down menu only.

Workflow settings for approval of change user roles

```
## WF
# Approve by manager
idm.sec.core.wf.approval.manager.enabled=false
# Approve by security department
idm.sec.core.wf.approval.security.enabled=false
idm.sec.core.wf.approval.security.role=Security
# Approve by helpdesk department
idm.sec.core.wf.approval.helpdesk.enabled=false
idm.sec.core.wf.approval.helpdesk.role=Helpdesk
# Approve by usermanager department
idm.sec.core.wf.approval.usermanager.enabled=false
idm.sec.core.wf.approval.usermanager.role=Usermanager
# Approve a role incompatibilities - If some incompatibilities are found in
request, then this approving will be executed.
idm.sec.core.wf.approval.incompatibility.enabled=true
idm.sec.core.wf.approval.incompatibility.role=Incompatibility
# Approval wf by role priority
idm.sec.core.wf.role.approval.1=approve-role-by-manager
idm.sec.core.wf.role.approval.2=approve-role-by-guarantee
idm.sec.core.wf.role.approval.3=approve-role-by-guarantee-security
# Approval wf for unassign role (one remove WF for whole application)
idm.sec.core.wf.role.approval.remove=approve-remove-role-by-manager
# Approve a change on the role - Is uses in the request of changing a role.
# In the request to create new role is also used.
idm.sec.core.wf.approval.role-change.role=
#
# Default main WF for approve all roles.
idm.sec.core.processor.role-request-approval-processor.wf=approve-identity-
change-permissions
```

Universal requests

```
## Universal requests
# Role
idm.pub.core.request.idm-role.enabled=false
# Defines type of guarantee. Requests will be approving only by guarantee
```

```
with this type.
# If returns null, then all guarantees will be used for approving (no
limitations).
idm.sec.core.request.idm-role.approval.guarantee-type=
```

Notification from Workflow

```
## Global property that allow disable or enable sending notification from WF
idm.sec.core.wf.notification.send=false
## Enable sending notification of changing roles to user, whose account will
be modified
idm.sec.core.wf.notification.applicant.enabled=false
## Enable sending notification of changing roles to user, who made request
idm.sec.core.wf.notification.implementer.enabled=true
```

Confidential storage

Properties **is not** overloadable via `ConfigurationService`. For more info see

```
# Cipher secret key for crypt values in confidential storage
# for crypt values is used secretKey - secret.key
# Can be empty => confidential storage will not be crypted, application
cannot be used in production (dev, test only).
cipher.crypt.secret.key=
# or secretKey defined in the external file - secret.keyPath
# cipher.crypt.secret.keyPath=/path/to/key
```

Entity filters

In the application profile (`application.properties`) - overloadable via `ConfigurationService`.

```
# Enable / disable check filter is properly registered, when filter is used
(by entity and property name).
# Throws exception, when unrecognized filter is used.
idm.sec.core.filter.check.supported.enabled=true
# Check count of values exceeded given maximum.
# Related to database count of query parameters (e.g. Oracle = {@code 1000},
MSSql = {@code 2100}).
# Throws exception, when size is exceeded. Set to {@code -1} to disable this
check.
idm.sec.core.filter.check.size.maximum=500
```

Every registered filter could have his own configuration properties under prefix:

```
# enable / disable filter - enabled by default. When filter is disabled and
```

```
property is filled in filter, then ''disjunction'' criteria is added => no
data will be returned
idm.sec.<module>.filter.<entity>.<name>.enabled=true
# filter implementation
idm.sec.<module>.filter.<entity>.<name>.impl=<beanName>
```

Where:

- `<module>` is filter's module - overriden module has to be used (e.g. default filter is in core module, then `core` module identifier has to be used)
- `<entity>` is entity class simple name - filter will be applied to this domain type (e.g. `IdmIdentity`)
- `<name>` the name of the property name during which the filter is actively evaluated, if it is stated in the filtering criteria (⇒ `get` parameter)
- `<beanName>` is filter's bean name - see implemented filters

Common configuration properties for all filters:

- `enabled` - on / off
- `impl` - contains implementation (Spring bean name) of given filter. When property of given `<name>` will be set for filter, then this implementation will be used for filtering. New module could register new filter for defined entity and name - by this configuration one of provided implementation will be selected and used.

Exists filters configuration: implemented filters.

Notification senders

In the application profile (`application.properties`) - overloadable via `ConfigurationService`. Senders could have his own configuration properties under prefix:

```
# sender implementation
idm.sec.<module>.notification-sender.<notificationType>.impl=<beanName>
```

Where:

- `<module>` is senders's module - overriden module has to be used (e.g. default sender is in core module, then `core` module identifier has to be used)
- `<notificationType>` is notification type, which has to be supported by configured sender by «beanName»

Common configuration properties for all senders:

- `impl` - contains implementation (Spring bean name) of given sender. This sender implementation will be used for sending notifications with `<notificationType>`. New module could register new sender implementation for notification types (even new notification type can be created) - by this configuration one of provided implementation will be selected and used.

Read more about notification manager.

Authentication

UUID of system, against which to user will be authenticated. This authentication is from version 10.4.0 deprecated.

```
# ID system against which to authenticate
idm.sec.security.auth.system=
```

Authentication against multiple system wich to user will be authenticated (since 10.4.0) - ID or Code can be used:

```
idm.sec.acc.security.auth.order1.system=
idm.sec.acc.security.auth.order2.system=
```

Maximum system for authentication can be set with the property:

```
idm.sec.acc.security.auth.maximumSystemCount=50
```

More about authenticator can be found there.

Authentication filters

In the application profile (`application.properties`) - overloadable via `ConfigurationService`. Authentication filter could have his own configuration properties under prefix:

```
# enable/ disable filter - enabled by default or by filter implementation.
idm.sec.<module>.authentication-filter.<name>.enabled=true
```

Where:

- `<module>` is filter's module - overriden module has to be used (e.g. default filter is in core module, then `core` module identifier has to be used)
- `<name>` is filter's name - see overridable filter's `#getName()` method. Filter name could be the same as bean name in context.

Common configuration properties for all filters:

- enabled - on / off

SSO authentication filter

Single-Sign-On mechanism can be configured with following properties:

```
# Allow SSO authentication
idm.sec.core.authentication-filter.core-sso-authentication-
filter.enabled=false
```

```
# The name of the header which contains the login of the authenticated user
idm.sec.core.authentication-filter.core-sso-authentication-filter.header-
name=REMOTE_USER
# The suffixes to remove from the login - usually domains
idm.sec.core.authentication-filter.core-sso-authentication-filter.uid-
suffixes=
# The uids that can't be authenticated by SSO
idm.sec.core.authentication-filter.core-sso-authentication-filter.forbidden-
uids=
```

Remote user authentication filter

Login into IdM by preset request remote user by servlet container can be configured with following properties:

```
# Allow remote user authentication
idm.sec.core.authentication-filter.core-remote-user-authentication-
filter.enabled=false
# The suffixes to remove from the login - usually domains
idm.sec.core.authentication-filter.core-remote-user-authentication-
filter.uid-suffixes=
# The uids that can't be authenticated by SSO
idm.sec.core.authentication-filter.core-remote-user-authentication-
filter.forbidden-uids=
```

This authentication filter reuses SSO authentication filter behavior above (`uid-suffixes`, `forbidden-uids`), but application administrator can be logged by this filter (identity with APP_ADMIN authority).

Two-factor authentication

Two-factor authentication can be configured in the application profile (application.properties) with following properties:

```
# Verification secret length
totp.secret.length=32
# Time Period ~ period to generate new authentication code
totp.time.period=30
# Time Discrepancy - number of past (but still valid) authentication codes
(e.g. when code is sent by notification, then user could need more time to
fill it into CzechIdM)
totp.time.discrepancy=1
```

CAS authentication filter

@since 12.0.0 CAS authentication can be configured with following properties:

```
# Enable authentication via CAS. If enabled, "idm.sec.core.cas.url" become
mandatory and must be set for SSO authentication via CAS to work. Default:
false
idm.pub.core.cas.enabled=false
# Other properties
# Base URL where CAS is accessible. Syntax of this field is
https://hostname-of-CAS/URI.
idm.sec.core.cas.url=
# IdM service name configured as service on CAS server.
# When service is configured, then login and logout redirect urls, should be
defined directly in CAS service configuration.
# Default: service name for login / logout is created dynamically by BE
server url (recommended).
idm.sec.core.cas.service=
# Suffix which is, in effect, appended to idm.sec.core.cas.url. Resulting
URL is used for login operation in CAS. It must start with slash (eg.
/login).
idm.sec.core.cas.login-path=/login
# Suffix which is appended to idm.sec.core.cas.url. Resulting URL is used
for single sign-out operation. It must start with slash (eg. /logout).
idm.sec.core.cas.logout-path=/logout
# Ticket can be given as request parameter (recommended, configured by
default).
idm.sec.core.cas.parameter-name=ticket
# Header name in which CAS sends the ticket value. Ticket can be given as
request header. Not configured by default.
idm.sec.core.cas.header-name=
# Path to CzechIdM for the HTTP Referer header used by CAS while redirecting
back to application. This value is concatenated with CAS ticket to form
Referer header. Syntax of this field is
https://hostname-of-CzechIdM/URI/?ticket=. Not configured by default.
idm.sec.core.cas.header-prefix=
```

OIDC authentication

@since 13.1.0 🗨OIDC authentication can be configured with following properties:

```
# Enable authentication via OIDC when false IDM will return 503
SERVICE_UNAVAILABLE on enpoints used for OICD auth, and ignore any Bearer
token. Default: false
idm.pub.core.oidc.enabled=false
# REQIRED configuration
# client-id confugured in CAS Service
idm.sec.core.oidc.client-id=
# client-secret confugured in CAS Service
idm.sec.core.oidc.client-secret=
# Base URL where OICD provider is accessible. Syntax of this field is
https://hostname-of-OICD/URI.
```

```
idm.sec.core.oidc.url=

# OPTIONAL configuration
idm.sec.core.oidc.login-path=/authorize
idm.sec.core.oidc.logout-path=/logout
idm.sec.core.oidc.token-path=/token

     # Configuration for spring.security most is gotten from auto-discover
endpoint (${idm.sec.core.oidc.url}/.well-known/openid-configuration) but can
be overwritten here
spring.security.oauth2.client.registration.cas.client-
id=${idm.sec.core.oidc.client-id}
spring.security.oauth2.client.registration.cas.client-
secret=${idm.sec.core.oidc.client-secret}
spring.security.oauth2.client.registration.cas.scope=openid
spring.security.oauth2.client.registration.cas.redirect-
uri={baseUrl}/api/v1/authentication/oidc-login-response/{registrationId}
#spring.security.oauth2.client.registration.cas.authorization-grant-
type=authorization_code
#spring.security.oauth2.resource.jwk.key-set-
uri=${idm.sec.core.oidc.url}/jwks

spring.security.oauth2.client.provider.cas.issuer-
uri=${idm.sec.core.oidc.url}
#spring.security.oauth2.client.provider.cas.token-
uri=${idm.sec.core.oidc.token-path}
#spring.security.oauth2.client.provider.cas.authorization-
uri=${idm.sec.core.oidc.login-path}
```

Backup

If you want to use redeploy and backup for example in agenda (notification templates, scripts), you must define default backup folder. When redploy is used, then actual templates (or scripsts) are loaded from classpath by configuration (for templates or scripts) and deployed into application. Previous templates (or scripts) are backup too.

```
# Configuration property for backup files.
# Configured attachment storage patrh ( see
'idm.sec.core.attachment.storagePath') is used as default.
idm.sec.core.backups.default.folder.path=/tmp/backup
```

Http proxy

For outgoing http communication, you can set a proxy.

Server restart is needed to apply this configuration change.

```
# Proxy for HTTP requests
```

```
idm.sec.core.http.proxy=12.34.56.78:1234

# For reCAPTCHA is used since version 12.2.5 new configuration. Backward
compatibility with original configuration still exists.
# Proxy configuration for reCAPTCHA
idm.sec.security.recaptcha.proxy=12.34.56.78:1234
```

CGLIB

CGLIB for creating proxies has to be enforced. Is possible to use annotations on methods, which is not defined in service interface. Prevent to use some logic in service constructors (will be called twice) and always define annotations in implementation class, read more.

```
# use cglib for proxies by default
spring.aop.proxy-target-class=true
```

Virtual system

VS configurations allows define implementers via assigned IdM role or directly by selected identities. If you do not define none directly implementers and none role in VS configuration, then will be used implementers from default role. Default role can be defined in configuration:

```
# If you do not define default role, then will be used **superAdminRole** as
default!
idm.sec.vs.role.default=<some-code-of-role>
```

Long polling

```
# Long polling
idm.pub.app.long-polling.enabled=true
```

You can disable long polling for all types of entites with use value `false`.

Provisioning

```
# It's possible to send additional attributes, when password is changed
(mapped attributes with flag sendOnPasswordChange)
# - true: additional password attributes will be send in one provisioning
operation together with password
# - false: additional password attributes will be send in new provisioning
operation, after password change operation
idm.sec.acc.provisioning.sendPasswordAttributesTogether=true

# It's possible to automatic mapped existed account on the target system. It
```

```
means, before create new account (call create on the connector),
# we try to found account (by generated UID) on the target system. If
account will be
# returned, then will be mapped on the IdM account. Target account will be
reused and only updated by connector.
# - true: for reusing account
# - false: for not reusing account
# - Default value is 'true'
idm.sec.acc.provisioning.allowedAutoMappingOnExistingAccount=true

# Default provisioning timeout in milis - every longer provisioning
operations will ends with timeout exception (prevent to stuck running
operations).
# 3 minutes by default.
# Timeout has to be configured>= 1000, otherwise default will be returned.
idm.sec.acc.provisioning.timeout=180000
```

Provisioning global break

For enable global provisioning break you must set configurations properties defined below, otherwise global provisioning break will not be activated.

```
# Global break for update disabled/enabled (values: true/false)
idm.sec.acc.provisioning.break.update.disabled
# Global break for update checked period (integer values)
idm.sec.acc.provisioning.break.update.period
# Global break for update disable limit (integer values)
idm.sec.acc.provisioning.break.update.disableLimit
# Global break for update disabled template (ID of template, if will by null
default template will be used)
idm.sec.acc.provisioning.break.update.templateDisable
# Global break for update warning limit (integer values)
idm.sec.acc.provisioning.break.update.warningLimit
# Global break for update warning template (ID of template, if will by null
default template will be used)
idm.sec.acc.provisioning.break.update.templateWarning
# Global break for update. Existing identity recipients (identity username
or id, split by ',')
idm.sec.acc.provisioning.break.update.identityRecipients
# Global break for update. Recipient will be solved as identities that has
assigned defined role/s (role code or id, split by ',')
idm.sec.acc.provisioning.break.update.roleRecipients
#
#
# Global break for create disabled/enabled (values: true/false)
idm.sec.acc.provisioning.break.create.disabled
# Global break for create checked period (integer values)
idm.sec.acc.provisioning.break.create.period
```

```
# Global break for create disable limit (integer values)
idm.sec.acc.provisioning.break.create.disableLimit
# Global break for create disabled template (ID of template, if will by null
default template will be used)
idm.sec.acc.provisioning.break.create.templateDisable
# Global break for create warning limit (integer values)
idm.sec.acc.provisioning.break.create.warningLimit
# Global break for create warning template (ID of template, if will by null
default template will be used)
idm.sec.acc.provisioning.break.create.templateWarning
# Global break for create. Existing identity recipients (identity username
or id, split by ',')
idm.sec.acc.provisioning.break.create.identityRecipients
# Global break for create. Recipient will be solved as identities that has
assigned defined role/s (role code or id, split by ',')
idm.sec.acc.provisioning.break.create.roleRecipients
#
#
#
# Global break for delete disabled/enabled (values: true/false)
idm.sec.acc.provisioning.break.delete.disabled
# Global break for delete checked period (integer values)
idm.sec.acc.provisioning.break.delete.period
# Global break for delete disable limit (integer values)
idm.sec.acc.provisioning.break.delete.disableLimit
# Global break for delete disabled template (ID of template, if will by null
default template will be used)
idm.sec.acc.provisioning.break.delete.templateDisable
# Global break for delete warning limit (integer values)
idm.sec.acc.provisioning.break.delete.warningLimit
# Global break for delete warning template (ID of template, if will by null
default template will be used)
idm.sec.acc.provisioning.break.delete.templateWarning
# Global break for delete. Existing identity recipients (identity username
or id, split by ',')
idm.sec.acc.provisioning.break.delete.identityRecipients
# Global break for delete. Recipient will be solved as identities that has
assigned defined role/s (role code or id, split by ',')
idm.sec.acc.provisioning.break.delete.roleRecipients
```

Reports

Report executor

In the application profile (`application.properties`) - overloadable via `ConfigurationService`.
Every report executor (~report) could have his own configuration properties under prefix:

```
# disable / enable report
idm.sec.<module>.report-executor.<name>.enabled=true
```

Where <module> is report's module a <name> is report's name.

Common configuration properties for all reports:

- enabled - on / off

Report renderer

In the application profile (`application.properties`) - overloadable via `ConfigurationService`. Every report renderer could have his own configuration properties under prefix:

```
# disable / enable renderer
idm.sec.<module>.report-renderer.<name>.enabled=true
```

Where <module> is renderer's module a <name> is renderer's name.

Common configuration properties for all renderers:

- enabled - on / off

Logger

In the application profile (`application.properties`):

```
# Show thread name configured by thread pools (task, event) in logs
(generated name is shown otherwise)
# Two appenders 'console' and 'file' are provided by product. Same
configuration is needed for your custom appenders (added in logback.xml).
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss.SSS} %5level %relative ---
[%thread] %logger{60}.%M : %msg%n
logging.pattern.file=%d{yyyy-MM-dd HH:mm:ss.SSS} %5level %relative ---
[%thread] %logger{60}.%M : %msg%n
```

Logger levels can be configured programmatically (override `logback.xml` file with default logger levels configuration).

In the application profile (`application.properties`) - overloadable via `ConfigurationService`:

```
idm.sec.core.logger.<packageName>=<level>
```

Where <packageName> is package name to set logger <level>.

Example:

```
idm.sec.core.logger.eu.bcvsolutions=DEBUG
```

Monitoring

Monitoring evaluator

In the application profile (`application.properties`) - overloadable via `ConfigurationService`.

```
# disable / enable monitoring evaluator
idm.sec.<module>.monitoring-evaluator.<name>.enabled=true
```

Where <module> is monitoring's module a <name> is monitoring's name.

Common configuration properties for all monitorings:

- `enabled` - true / false

From:
https://wiki.czechidm.com/ - **CzechIdM Identity Manager**

Permanent link:
https://wiki.czechidm.com/devel/documentation/application_configuration/dev/backend

Last update: **2023/12/01 13:10**