Configuration reference

Snow Owl Server comes with a predefined default configuration file, which can be used to tweak various system parameters. The configuration file is in YAML format, and located at `/opt/snowowl-{edition}_{version}/snowowl_config.yml`. You can read more about how to create/write such files here: http://en.wikipedia.org/wiki/YAML.

The configuration file has a hierarchical structure, which is defined by modules. Different modules can have different configurations, a module is defined by its name, which should start a line in the file. Configuration parameters in a module should be indented by two spaces following the module's name.

The next section contains the reference of our currently supported configuration parameters. Each parameter should be present in a module configuration as described above.

Snow Owl Server refuses to start if the configuration file contains syntactical or structural errors. The cause of the problem can be found in the `log.log` file, or in the console if you've redirected the output of the server's startup process.

Identity Providers

Snow Owl supports multiple identity provider configurations (by default File and LDAP are available, but others can be added as well).

Name	Default	Description
providers	`[]`	`file`, `ldap` - choose the enabled identity providers, you can select multiple options as well.

```
identity:
  providers:
    - <provider_type>:
        <provider_config_key>: <provider_config_value>
```

To configure an identity provider you must add its configuration to the list of available providers (`identity.providers` config node). The providers list ensures ordered execution of the various providers, the first (top) provider will be invoked first, then the second and so forth. See the next sections on how to configure File and LDAP identity providers.

File Identity Provider

File identity provider supports authentication of a set of users available in the specified configuration file. You can specify the name of the configuration file, which should be placed in the server's configuration directory. Example configuration to use File identity provider based on a property file at `${SERVER_HOME}/configuration/users`:

```
identity:
  providers:
    - file:
        name: users
```

File identity provider does **NOT** support authorization features (such as Roles and Permissions). We recommend the LDAP identity provider for such deployments. To allow a user to authenticate with the File identity provider you must declare the user's username and hashed (with BCrypt hash algorithm) password in the property file in the following form:

Example property file content for File identity provider:

```
snowowl:$2a$10$RtnN9fxuxjz3W7drWtxpT.2BblCxsJ0WL2Fl14XBCjMrMaqq.uenW
```

LDAP Identity Provider

LDAP identity provider uses an external LDAP server (preferably OpenLDAP) to authenticate and retrieve user identities. This provider does support authorization features, such as Roles and Permissions. Example configuration to use LDAP identity provider:

```
identity:
  providers:
    - ldap:
        uri: ldap://localhost:10389
        baseDn: dc=snowowl,dc=b2international,dc=com
        rootDn: cn=admin,dc=snowowl,dc=b2international,dc=com
        rootDnPassword: <adminpwd>
        userIdProperty: uid
```

Configuration options:

Name	Default	Description
uri	``	`ldap://<host>:<port>` - LDAP host and port
baseDn	`dc=snowowl,dc=b2international,dc=com`	Base Dn value of the organization from which user identities will be retrieved.
rootDn	`cn=admin,dc=snowowl,dc=b2international,dc=com`	The root user's DN value, to bind to the LDAP server with.
rootDnPassword	``	The root user's password, to bind to the LDAP server with.

Name	Default	Description
userIdProperty	``	The user ID property which will contain the unique identifier of your users (usually uid, which represent an email address).
usePool	`false`	To use a connection pool under the hood when connecting to the external LDAP server or not.

Repository

Name	Default	Description
host	`0.0.0.0`	The host name to bind to.
port	`2036`	The port of the chosen network interface to use when listening for connections.
numberOfWorkers	`3 x NumberOfCores`	The number of worker threads to assign to a repository during initialization.
revisionCache	`true`	Enable CDO revision cache to keep data returned from the database.
readerPoolCapacity	`7`	The capacity of the reader pool associated with the SNOMED CT store.
writerPoolCapacity	`3`	The capacity of the writer pool associated with the SNOMED CT store.

```
repository:
  host: 0.0.0.0
  port: 2036
```

Database

Name	Default	Description
directory	`store`	The directory of the embedded database inside the global resources folder where the application should look for the database files by default (if no location parameter is given).

Name	Default	Description
type	`h2`	The type of the database adapter to use when connecting to the database.
driverClass	`org.h2.Driver`	The fully qualified name of the driver's Java class to use when connecting to the database.
datasourceClass	`org.h2.jdbcx.JdbcDataSource`	The fully qualified name of the datasource's Java class to use when connecting to the database.
scheme	`jdbc:h2:`	The scheme to use when connecting to the database.
location		The location of the database when connecting to it. If not set then in embedded mode the default directory parameter will be used as location.
username		The username of the database user to use when connecting to the database.
password		The password of the database user to use when connecting to the database.
settings		Other database specific JDBC settings to use when connecting to the database.

```
repository:
  database:
    directory: store
    type: h2
    username: admin
    password: admin
```

Index

Name	Default	Description
commitInterval	`5000`	The commit interval of an index in milliseconds.
queryWarnThreshold	`400`	The threshold of the warn log when querying data.
queryInfoThreshold	`300`	The threshold of the info log when querying data.

Name	Default	Description
queryDebugThreshold	`100`	The threshold of the debug log when querying data.
queryTraceThreshold	`50`	The threshold of the trace log when querying data.
fetchWarnThreshold	`200`	The threshold of the warn log when fetching data.
fetchInfoThreshold	`100`	The threshold of the info log when fetching data.
fetchDebugThreshold	`50`	The threshold of the debug log when fetching data.
fetchTraceThreshold	`10`	The threshold of the trace log when fetching data.
numberOfShards	`3`	Number of shards to use for terminology repositories.
commitConcurrencyLevel	`Number of cores / 4 by default, min 1`	Number of concurrent requests when executing bulk commit operations against a terminology repository index.
clusterUrl		Remote Elasticsearch cluster to connect to and use for terminology repository indexes.
clusterUsername		Username to use when connecting to a remote Elasticsearch cluster that requires Basic authentication credentials. `clusterUrl` is required for this option.
clusterPassword		Password to use when connecting to a remote Elasticsearch cluster that requires Basic authentication credentials. `clusterUrl` is required for this option.
connectTimeout	`1000`	Determines how long should an Elasticsearch request wait for the HTTP connection to be established before failing the request.
socketTimeout	`30000`	Determines how long should an Elasticsearch request wait for a response failing the request.

Name	Default	Description
clusterHealthTimeout	`300000`	Determines how long should the initialization of the embedded Elasticsearch wait for at least yellow cluster health state.

```
repository:
  index:
    commitInterval: 5000
    translogSyncInterval: 1000
    queryWarnThreshold: 400
    fetchInfoThreshold: 100
```

RPC

RPC is a custom protocol implementation used to solve request-response based communication between a client and a server.

NOTE	Changing these settings is not recommended and currently unsupported in production environments.

Name	Default	Description
logging	`false`	`true`, `false`, `ON`, `OFF` - enable or disable verbose logging during RPC communication

```
rpc:
  logging: true
```

Metrics

Snow Owl can measure and report execution times (and other metrics in the future) of executed requests.

Name	Default	Description
enabled	`true`	`true`, `false`, `ON`, `OFF` - enable or disable metrics in the application

SNOMED CT

Configuration of SNOMED CT terminology services.

Name	Default	Description
readerPoolCapacity	`7`	The capacity of the reader pool associated with the SNOMED CT store.
writerPoolCapacity	`3`	The capacity of the writer pool associated with the SNOMED CT store.
language	`en-gb`	`en-gb`, `en-us`, `en-sg` - The language code to use for SNOMED CT Descriptions. Descriptions with membership of the chosen language's reference set will be used runtime.
maxReasonerCount	`2`	The maximum number of reasoners permitted to do computation simultaneously. Minimum `1`, maximum `3` is allowed. If the value is set to `1`, classification requests will be processed in a sequential fashion.
maxReasonerResults	`10`	The number of inferred taxonomies that should be kept in memory after the reasoner completes the computational stage. The user can only choose to save the results of the classification run if the corresponding taxonomy instance is still present.
maxReasonerRuns	`1000`	The number of classification runs of which details should be preserved on disk. Details include inferred and redundant relationships, the list of equivalent concepts found during classification, and classification run metadata (start and end times, status, requesting user, reasoner used for this run).
showReasonerUsageWarning	`true`	'true' will display a dialog if any user selects a non-ELK reasoner, citing memory and compatibility problems, also recommending to contact B2i.
concreteDomainSupport	`false`	'true' will turn on support for concrete domains.

Name	Default	Description
inferredEditingEnabled	`false`	'true' will enable manual editing of inferred relationships and concrete domain elements.

```
snomed:
  language: en-gb
  maxReasonerCount: 1
  maxReasonerResults: 20
  showReasonerUsageWarning: true
  concreteDomainSupport: true
  inferredEditingEnabled: false
```

SNOMED CT Component Identifier Configuration

Snow Owl's SNOMED CT identifier service can be configured to be either using the built-in service or SNOMED International's external Component Identifier Service (CIS). The configuration needs to be placed within the **snomed/ids** section. If omitted, then default configuration will be used, which is the built-in (embedded) service based on the index store allocating ids in a sequential fashion.

Name	Default	Description
service	EMBEDDED	`EMBEDDED` or `CIS` - The service used to generate ids.
source	INDEX	`INDEX` or `MEMORY` - The source of the generated ids. `MEMORY` is used for testing.
strategy	SEQUENTIAL	`SEQUENTIAL` or `RANDOM` - The strategy of the id generation.
cisBaseUrl		The service's URL with port and without context root.
cisContextRoot		The context root of the id generation service.
cisUserName		The registered user name at the CIS site.
cisPassword		The password for the registered user name at the CIS site.
cisClientSoftwareKey	Snow Owl	The client software key to be persisted within CIS as reference.
cisNumberOfPollTries	1	The maximum number of tries when polling jobs of bulk requests.

Name	Default	Description
cisTimeBetweenPollTries	1000	The time to wait between 2 job polling actions It is in milliseconds.
cisNumberOfReauthTries	2	The maximum number of re-authentication attempts when a 401 Not authorized response is received.
cisMaxConnections	100	Maximum number of simultaneous connections that Snow Owl can make to the CIS host via HTTP.
maxIdGenerationAttempts	1000	Maximum number of attempts any non-CIS ID generator will take to generate a single SNOMED CT identifier, if exceeded it throws an exception.

Example for using the built-in service with random ids using the index as the source:

```
snomed:
  ...
  ids:
    service: EMBEDDED
    source: INDEX
    strategy : RANDOM
    cisBaseUrl : <cis_host_and_port>
    cisContextRoot : api
    cisUserName : <your-cis-username>
    cisPassword : <your-cis-password>
    cisClientSoftwareKey : Snow Owl dev. deployment
    cisNumberOfPollTries : 1
    cisTimeBetweenPollTries : 1000
    cisMaxConnections: 100
  ...
```

Example for using IHTSDO's external CIS service:

```
snomed:
  ...
  ids:
    service : CIS
    cisBaseUrl : <cis_host_and_port>
    cisContextRoot : api
    cisUserName : <your-cis-username>
    cisPassword : <your-cis-password>
    cisClientSoftwareKey : Snow Owl dev. deployment
    cisNumberOfPollTries : 1
    cisTimeBetweenPollTries : 1000
    cisMaxConnections: 100
  ...
```