

Snow Owl Server Documentation

B2i Healthcare

Table of Contents

Introduction	1
Installation	1
Installation requirements	1
Client-side requirements	1
Hardware requirements	1
Server-side requirements	1
Hardware requirements	1
Software Requirements	1
Installing	2
Operating system	2
Network	3
Database	4
Configuring database	7
Java	10
LDAP	10
Configuring LDAP	10
Using LDIF dumps	10
Using Apache Directory Studio	12
Creating a new user using Apache Directory Studio	14
Snow Owl Server	22
Update Snow Owl's Configuration	22
Memory settings	23
OSGi console	23
Logging	24
Web Server Configuration	25
Web Server Administrative Console application	26
Virgo documentation	26
Administration	26
Startup and shutdown	26
Importing SNOMED CT content	28
Starting the import	28
Release decriptor file	28
Examples	29
International import	29
Extension import	29
Single administrator operations	30
Steps to perform single admin operations	30
Impersonating users	31

Taking backups	32
"Hot" backups	32
"Cold" backups	33
User and Access control	33
Permissions	33
Default Roles	34
Customizing access control	34
Customize schema	34
Import default configuration	35
Add new user	35
Modify access control	35
Create new Role	35
Modify Roles	35
Console command reference	35
General	36
Disconnecting from the console	37
Available commands	37
Scripting	37
Execute Groovy script	37
Machine-Readable Concept Model	38
Import MRCM rules from XMI file	38
Export MRCM rules to an XMI file	38
Terminology registry	39
List all imported terminologies	39
SNOMED CT OWL Ontology (reasoner)	39
Display available reasoners	39
Change the active reasoner	39
Checking status of available reasoners	40
SNOMED CT	41
Import reference sets in RF2 format from a release text file	41
Import reference sets in DSV format	43
Diagnostics and maintenance	45
Session management	46
Display connected users and session identifiers	47
Send message to users	47
Restrict user logins	47
Disconnect users	48
Repository management	48
Display terminology repositories	48
Display currently held locks	49
Lock and release areas of terminology stores	50

Forcefully unlock stuck locks	52
Remote jobs	53
Display remote jobs	53
Cancel remote job	53
Configuration reference	54
Identity Providers	54
File Identity Provider	54
LDAP Identity Provider	55
Repository	56
Database	56
Index	57
RPC	58
Metrics	59
SNOMED CT	59
SNOMED CT Component Identifier Configuration	60

Introduction

Snow Owl® is a terminology server and a collaborative terminology authoring platform. The authoring platform maintains terminology artifacts developed by a team and supported by business workflows that are driven by external task management systems like Bugzilla and JIRA. With its modular design, the server can maintain multiple terminologies where new terminologies can be plugged-in to the platform. The functionality of Snow Owl is exposed via a REST API.

Clients can easily access and query SNOMED CT, LOINC®, ATC, ICD-10, and dozens of additional terminologies via REST or Java APIs. Collaborative distributed authoring is also supported, including creating and maintaining your own local code systems, mapping between terminologies, and creating terminology subsets.

Snow Owl is maintained by B2i Healthcare.

Installation

This section goes through the required installation steps to get a production ready Snow Owl up and running on your machines.

Installation requirements

Client-side requirements

Hardware requirements

Memory	4 GB
Disk	1 GB free space
Operations System	64-bit Microsoft Windows 7, 8, 8.1, 10

Server-side requirements

Hardware requirements

CPU	20 Core Server (eg. Dual Intel Xeon E5 2650 V3)
Memory	96 GB
Disk	4 x 256 GB Primary SSD Drive RAID 10
Network	1 Gbps Dedicated Port

Software Requirements

	Supported Platforms	Supported Version(s)	Notes
Operating systems	Linux	CentOS (RHEL) 6.8 or Ubuntu 14.04 LTS	We recommend starting with a minimal install and adding packages later, as required. B2i Healthcare only officially supports Snow Owl running on 64-bit derivatives of x86 hardware.
Java	Oracle JDK	1.8.0 update 121	An installable archive can be downloaded from the JDK8 download page . Select the “Linux x64” edition.
Database	MySQL	5.7	Terminology contents are persisted using a MySQL database, downloadable from MySQL’s yum repository
LDAP	OpenLDAP	2.4.x	Authentication and authorization of browsers, terminology editors, reviewers and administrators is performed through an LDAP server. Browsing and managing OpenLDAP instances can be done through the Apache Directory Studio application. We recommend installing the latest release from the corresponding Download Versions page on Apache’s website.

Installing

In this section we’ll run you through installing Snow Owl in a production environment on a CentOS 6.8 machine with an external MySQL database and LDAP based authentication/authorization.

Operating system

Install Red Hat Enterprise Linux or CentOS using the minimal ISO image. As hardware

configurations and the corresponding exact installation steps can be different from machine to machine, please refer to the [installation guide](#) on Red Hat's site for details (covers both distributions).



The text-based installer does not offer all options compared to the graphical one; you may have to connect a physical monitor, or use the built-in KVM management capabilities of the server (if supported) to perform the installation from the graphical environment. The installed system only needs an SSH connection for administration.

When creating the partition layout, keep in mind that Snow Owl Server requires at least 50-150 GB of disk space when branched terminology editing is used extensively.

After logging in to the installed system, update installed packages to the latest version and add EPEL as a package repository for dependencies. For non-CentOS installations, please see the [usage instructions](#) on the EPEL wiki.

```
# yum update
# reboot
# yum install epel-release ①
```

① Works only if CentOS was installed

Create a non-login user for Snow Owl Server to run as:

```
# useradd -r -s /sbin/nologin snowowl
```

For optimal indexing performance, consult the "Production Deployment" sections of [Elasticsearch: The Definitive Guide](#). In particular, the following settings are applicable to an installation of the terminology server:

```
# sysctl settings, to be added to /etc/sysctl.conf or equivalent

vm.swappiness = 1
vm.max_map_count = 262144

# "noop" I/O scheduler, should be set in eg. /etc/rc.local for solid state disks:

echo noop > /sys/block/sda/queue/scheduler
```

Network

Install `system-config-firewall-tui`:

```
# yum install dbus dbus-python system-config-firewall-tui
# reboot
```

Using the text-based UI, enable these Trusted Services:

SSH

For remote administration of the server

Also open access to the following ports:

8080/TCP

Used by Snow Owl Server's REST API

2036/TCP

Used by the Net4J binary protocol connecting Snow Owl clients to the server

Database

An extensive installation guide for getting MySQL Community Edition from a yum repository is available at [the MySQL Documentation Library](#). The required steps are summarized below.

Install MySQL's yum repository with the following command:

```
# yum install -y mysql57-community-release
```

The repository for MySQL 5.7 should be enabled by default. Confirm by opening `/etc/yum.repos.d/mysql-community.repo`:

```
# yum repolist enabled | grep mysql ①
```

① 5.7 should appear somewhere

Install MySQL Community Server using yum (also run an update so packages get replaced with the community version):

```
# yum install -y mysql-community-server
# yum update -y
```

Start the service and wait for first-time initialization to complete:

```
# chkconfig --list mysqld
# service mysqld start
Starting MySQL. [ OK ]
```

After a few minutes, check if the database service is still running and enabled at startup:


```
# service mysqld status
mysqld (pid 1757) is running...
```

```
# chkconfig mysqld on
# chkconfig --list mysqld
mysqld          0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

Get the temporary password and change it

```
# grep 'temporary password' /var/log/mysqld.log
# mysql -uroot -p
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass4!';
```



The entered new root password will be used later for configuration and administrative purposes; do not forget this password.

Edit `/etc/my.cnf` to adjust settings for the MySQL server. Recommended settings are shown below, but there are lots of additional tunable settings to choose from depending on the hardware configuration used; please see <https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html> for the full set of system variables.

/etc/my.cnf

```
#
# The following options will be read by MySQL client applications.
# Note that only client applications shipped by MySQL are guaranteed
# to read this section. If you want your own MySQL client program to
# honor these values, you need to specify it as an option during the
# MySQL client library initialization.
#
[client]
socket = /var/lib/mysql/mysql.sock

[mysqld]

# Remove leading # and set to the amount of RAM for the most important data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
innodb_buffer_pool_size = 5G

# Size of each log file in a log group. You should set the combined size
# of log files to about 25%-100% of your buffer pool size to avoid
# unneeded buffer pool flush activity on log file overwrite. However,
# note that a larger logfile size will increase the time needed for the
# recovery process.
innodb_log_file_size = 1G

# Total number of files in the log group. A value of 2-3 is usually good
```

```

# enough.
innodb_log_files_in_group = 3

# Remove leading # to turn on a very important data integrity option: logging
# changes to the binary log between backups.
# log_bin

# These are commonly set, remove the # and set as required.
# basedir = .....
# datadir = .....
# port = .....
# server_id = .....
socket = /var/lib/mysql/mysql.sock

# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M

sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES

# Minimum word length to be indexed by the full text search index.
# You might wish to decrease it if you need to search for shorter words.
# Note that you need to rebuild your FULLTEXT index, after you have
# modified this value.
ft_min_word_len = 2

# The maximum size of a query packet the server can handle as well as
# maximum query size server can process (Important when working with
# large BLOBs).  enlarged dynamically, for each connection.
max_allowed_packet = 16M

# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

collation-server=utf8_unicode_ci
character-set-server=utf8

lower_case_table_names=1
transaction-isolation=READ-COMMITTED

[mysqldump]
# Do not buffer the whole result set in memory before writing it to
# file. Required for dumping very large tables
quick
max_allowed_packet = 16M

```

Restart the mysql service to make sure changes are picked up:

```
# service mysqld restart
Stopping mysqld:           [ OK ]
Starting mysqld:           [ OK ]
```

Configuring database

Create a MySQL user for the Snow Owl Server by connecting to the DBMS via the console:

```
$ mysql -u root -p
Enter password: root_pwd ①

mysql> CREATE USER 'snowowl'@'localhost' IDENTIFIED BY 'snowowl_pwd'; ②
```

① Replace `root_pwd` with the password for the `root` user in MySQL

② Replace `snowowl_pwd` with a generated password for the `snowowl` user in MySQL

Save the following shell script to an executable file to create databases and grant privileges for user `snowowl`:

snowowl_create_db.sh

```
#!/usr/bin/env bash

#
# Copyright 2015-2017 B2i Healthcare Pte Ltd, http://b2i.sg
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

#
# This script creates all required databases and user for Snow Owl Server US edition.
#

DATABASES=( snomedStore )

MYSQL=`which mysql`
USER="root"
PASSWORD="root_pwd"
```

```

echo -e "\nStarting Snow Owl database setup procedure."

# Create user
${MYSQL} -u${USER} -p${PASSWORD} -e "CREATE USER 'snowowl'@'localhost' identified by
'snowowl';" > /dev/null 2>&1
echo -e "\n\tCreated snowowl/mysql user."

# Create databases
echo -e "\n\tCreating Snow Owl databases:"
for i in "${DATABASES[@]}"
do
    ${MYSQL} -u${USER} -p${PASSWORD} -e "CREATE DATABASE \`${i}\` DEFAULT CHARSET
'utf8';" > /dev/null 2>&1
    echo -e "\t\tCreated Snow Owl database ${i}."
    ${MYSQL} -u${USER} -p${PASSWORD} -e "GRANT ALL PRIVILEGES ON \`${i}\`.* to
'snowowl'@'localhost';" > /dev/null 2>&1
    echo -e "\t\tPrivileges granted on ${i} for snowowl user."
done
echo -e "\tCreation of Snow Owl databases are now complete."

${MYSQL} -u${USER} -p${PASSWORD} -e "FLUSH PRIVILEGES;" > /dev/null 2>&1
echo -e "\n\tGrant tables reloaded."

echo -e "\nSnow Owl database setup procedure has finished.\n"

```

B2i provided MySQL dumps (if present) can be found in [/opt/snowowl-{edition}_{version}/resources/*.sql](#) files after unpacking the installation archive. To load terminology data, save and execute the following script:

```
#!/usr/bin/env bash

#
# Copyright 2013-2017 B2i Healthcare Pte Ltd, http://b2i.sg
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# Loads database content for Snow Owl from SQL dumps. Uses the name of the SQL
# file(s) specified as the target database for each file.
#
# Usage: ./snowowl_load_db [dbfile1] [dbfile2] ...

if [[ $# -eq 0 ]]; then
    echo "No SQL files were specified for loading. Exiting."
    exit 1
fi

MYSQL=$(which mysql)
BASENAME=$(which basename)
USER="snowowl"
PASSWORD="snowowl_pwd"

for DBFILE in "$@"
do
    DB="$(${BASENAME} "$DBFILE")"
    ${MYSQL} --batch -u${USER} -p${PASSWORD} "${DB%.sql}" < "${DBFILE}"

    if [ $? -ne 0 ]; then
        echo "Loading from file ${DB} failed."
    else
        echo "Loading from file ${DB} finished successfully."
    fi
done

echo "All files processed."
exit 0
```

Java

Download the “Linux X64” edition, and install it with yum:

```
# yum install jdk-8u121-linux-x64.rpm
```

LDAP

Install the latest (2.4.x) OpenLDAP with yum:

```
# yum -y install openldap compat-openldap openldap-clients openldap-servers openldap-servers-sql openldap-devel
```

Start the service, and enable it to run when system boots up:

```
# systemctl start slapd.service
# systemctl enable slapd.service
```

Next, run the `slappasswd` to create an LDAP root password. Please take note of this root password, the entire hashed value that is returned as output and starts {SSHA}, as you’ll use it throughout this article. Afterwards, you can import the provided LDIF files via `ldapadd/ldapmodify` commands (Role and Permission schema).

Configuring LDAP

Using LDIF dumps

B2i provided LDAP packages include the following content:

permission_schema.ldif

LDAP schema to use for authorization (contains definitions for permissions)

permissions.ldif

All available permissions in the system

roles.ldif

All available roles in the system

pm.ldif

Maps permissions to roles

update.sh

An update script using `ldapmodify` and `ldapadd` commands against a running LDAP instance to update it based on the files above

Optionally the assembly can contain two additional files:

users.ldif

All users available in the system

rm.ldif

Maps roles to users in the system

The update script will also make use of these files if any of them exist.

Install the `openldap-clients` first to make use of the script:

```
# yum install openldap-clients
```

Before updating the LDAP server, it is advised to shut down the service, and create a backup, so it can be restored easily if the script fails.

Restart the server, then create a new `ldif-<version>` folder and unzip the contents of the LDIF archive into this folder. Finally, execute the script to update the contents of LDAP:

```
# chmod u+x update.sh

# ./update.sh
Not specified LDAP URI parameter, using ldap://localhost:10389
adding new entry "cn=permission, ou=schema"
adding new entry "ou=attributeTypes, cn=permission, ou=schema"
...
modifying entry...
```

In case an error occurs, the executed command and the error response will be displayed. Errors will also be logged to a `{file_name}.errors` file, where the `{file_name}` refers to the file being processed (eg. `permissions.errors`).

When executing the script it is possible to get the following errors:

- `ERR_250_ALREADY_EXISTS` (or any synonym of `ALREADY_EXISTS`)
- `ERR_54 Cannot add a value which is already present : snomed:compare:automap`
- `ERR_335 Oid 2.25.128424792425578037463837247958458780603.1 for new schema entity is not unique`

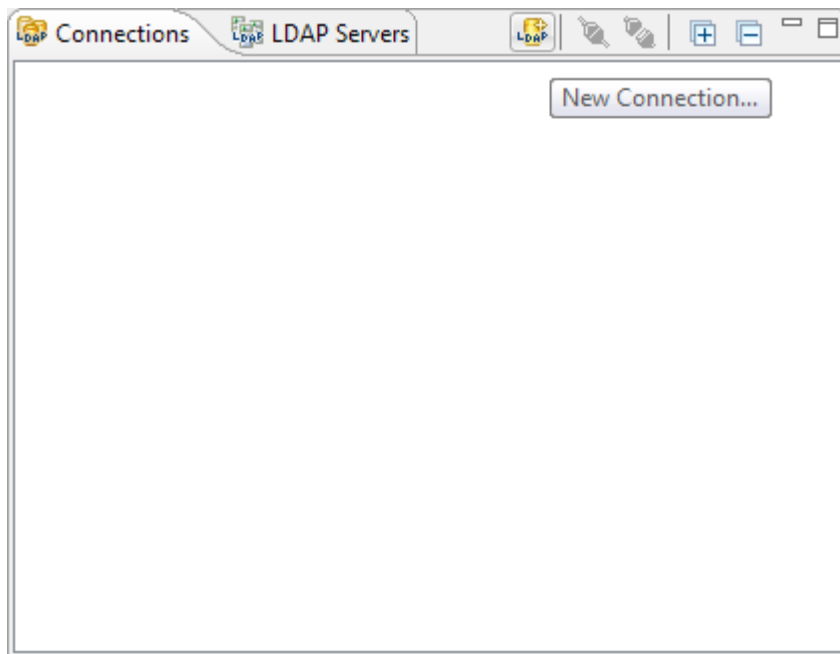
This is expected as most of the time the LDAP instance will already contain an existing definition of some entries and/or schema entities. If you notice other errors (either during script execution or when using the LDAP), roll back your instance to a previous state from a backup.

By default the update script will execute against the LDAP instance running locally at `ldap://localhost:10389`; if you'd like to run the script against a remote LDAP server (or the LDAP is listening on a different port), you can do it by specifying the `LDAP_URI` parameter:

```
# ./update.sh ldap://<host>:<port>
```

Using Apache Directory Studio

Open Apache Directory Studio, create a new connection using the first button on the “Connections” toolbar:



Enter connection name, hostname, and port, then hit **Next >** to go to the next page in the wizard:

New LDAP Connection

Network Parameter

Please enter connection name and network parameters.

Connection name: snowowl

Network Parameter

Hostname: localhost

Port: 10389

Encryption method: No encryption

Server certificates for LDAP connections can be managed in the '[Certificate Validation](#)' preference page.

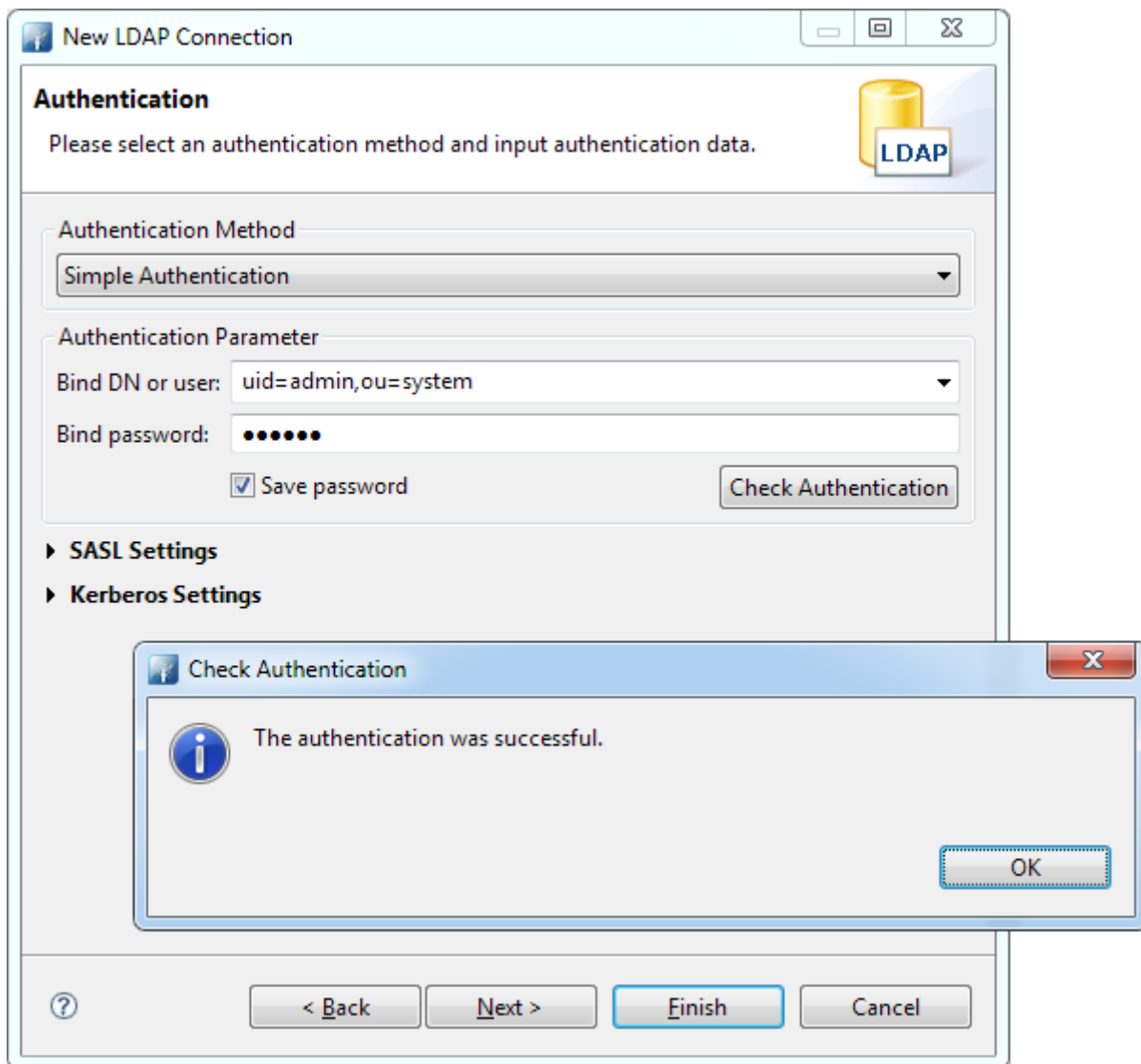
Provider: Apache Directory LDAP Client API

Check Network Parameter

☐ Read-Only (prevents any add, delete, modify or rename operation)

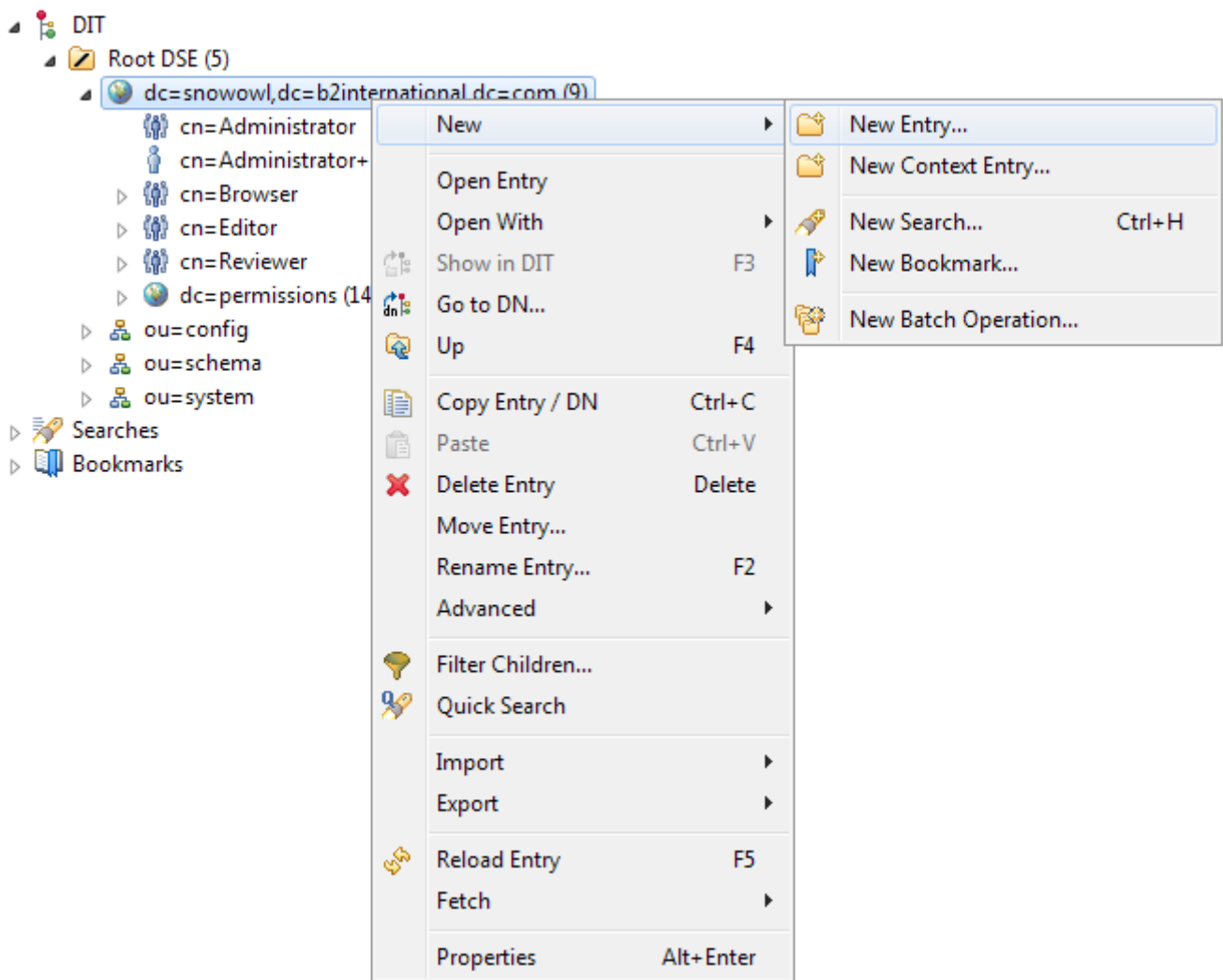
? < Back Next > Finish Cancel

Set authentication method to **Simple Authentication**, enter Bind DN of your root user and its password. Click **Check Authentication** to make sure the values are accepted, then dismiss the wizard with **Finish**:

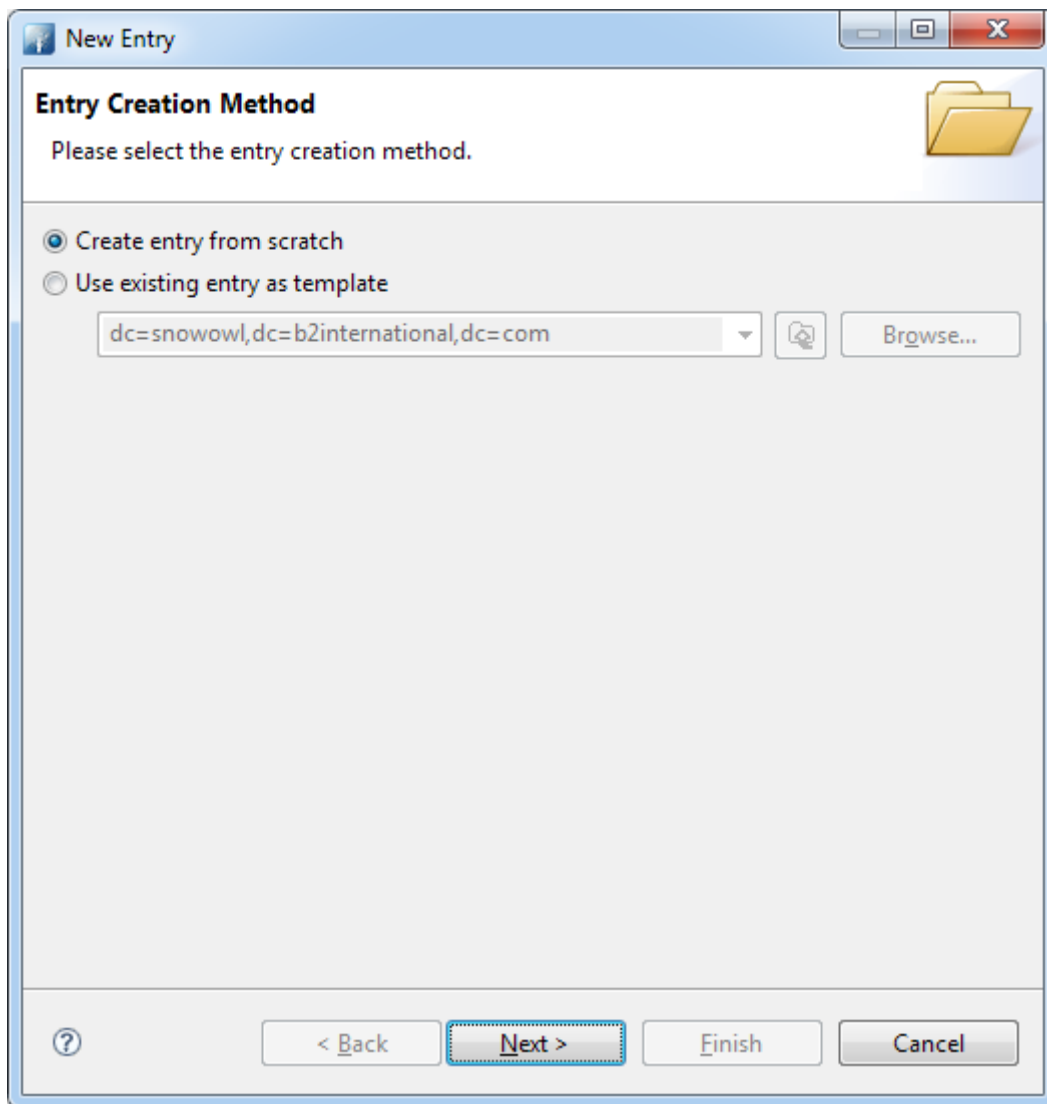


Creating a new user using Apache Directory Studio

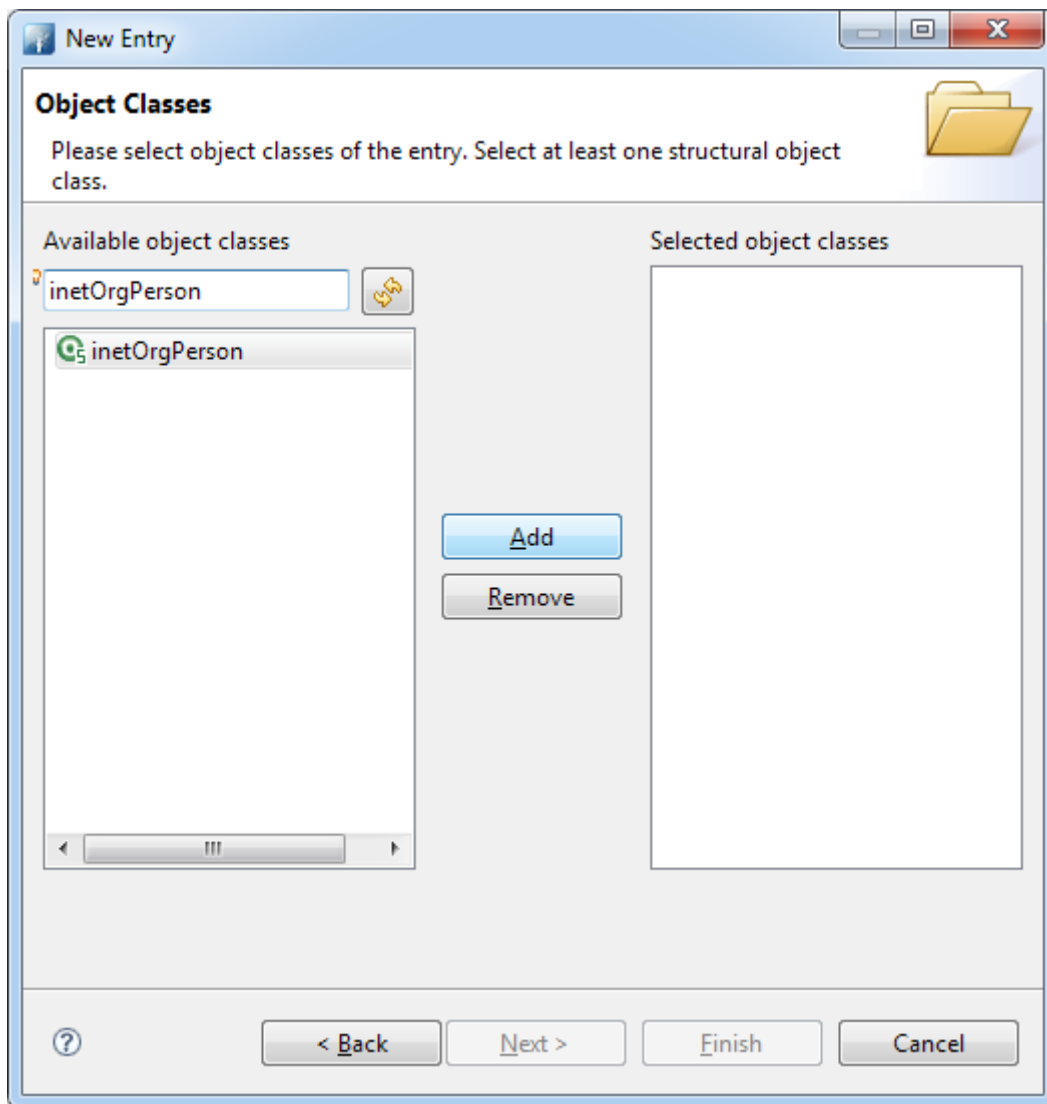
Go to LDAP Browser view, right click on the Domain component (DC) and add new entry via **New > New entry:**



Create a new entry from scratch:



Select **inetOrgPerson** object from the wizard, add it as a selected object class:




Configure the Relative Distinguished Name (RDN). Specify the common name (CN), surname (SN) and unique ID (uid):

New Entry

Distinguished Name


Please select the parent of the new entry and enter the RDN.

Parent: 

RDN:

<input type="text" value="cn"/>	=	<input type="text" value="Snow Owl User"/>	<input type="button" value="+"/>	<input type="button" value="-"/>
<input type="text" value="sn"/>	=	<input type="text" value="info"/>	<input type="button" value="+"/>	<input type="button" value="-"/>
<input type="text" value="uid"/>	=	<input type="text" value="info@b2international.com"/>	<input type="button" value="+"/>	<input type="button" value="-"/>

DN Preview:



Open the added node in an editor, right click in the editor and select **New Attribute**:

cn=Snow Owl User+sn=info+uid=info@b2international.com,dc=snowowl,dc=b2international,dc=com

DN: cn=Snow Owl User+sn=info+uid=info@b2international.com,dc=snowowl,dc=b2international,dc=com

Attribute Description	Value
<i>objectClass</i>	<i>inetOrgPerson (structural)</i>
<i>objectClass</i>	<i>organizationalPerson (structural)</i>
<i>objectClass</i>	<i>person (structural)</i>
<i>objectClass</i>	<i>top (abstract)</i>
cn	Snow Owl User
sn	info
uid	info@b2international.com

- New Attribute... Ctrl+Shift++
- New Value Ctrl++
- New Search... Ctrl+H
- New Batch Operation...
- Locate DN in DIT F3
- Open Schema Browser
- Show In
- Copy Value Ctrl+C
- Paste Ctrl+V
- Delete Value Delete
- Select All Ctrl+A
- Advanced
- Edit Attribute Description F6
- Edit Value F7
- Edit Value With
- Edit Entry... F8
- Reload Attributes F5
- Fetch Operational Attributes
- Properties Alt+Enter

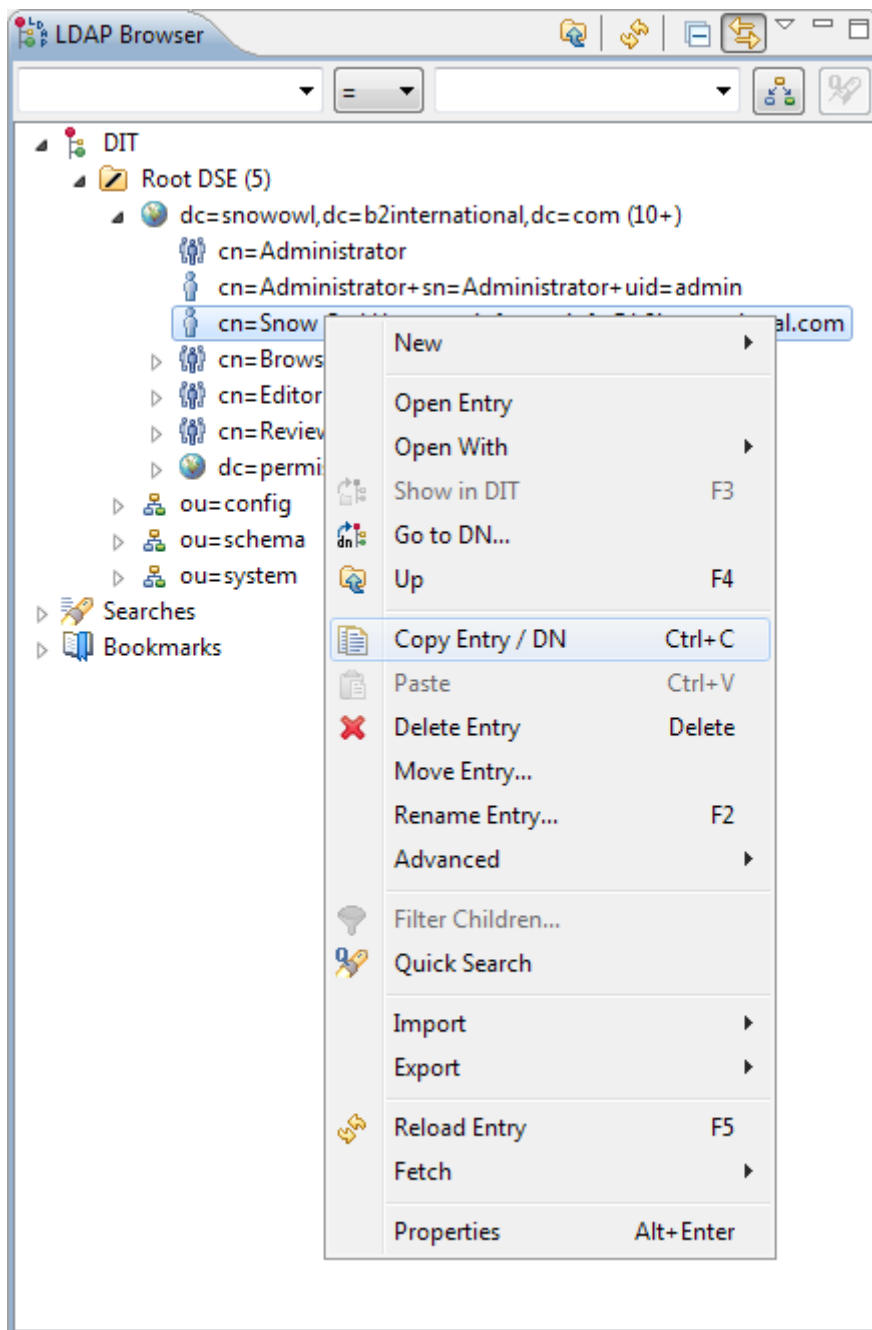
Select **userPassword** attribute, click **Finish**, then enter user password in the **Password Editor** dialog:

The screenshot shows a 'Password Editor' window with a 'New Password' tab. It contains a text field for 'Enter New Password' with the value 'new_user_pwd', a dropdown for 'Select Hash Method' set to 'SSHA-512', and a 'Password Preview' field showing a long alphanumeric string. Below these are fields for 'Password (Hex)' and 'Salt (Hex)', both containing hexadecimal values. A 'New Salt' button is to the left of these fields. A checkbox labeled 'Show new password details' is checked. At the bottom right are 'OK' and 'Cancel' buttons.

Field	Value
Enter New Password:	new_user_pwd
Select Hash Method:	SSHA-512
Password Preview:	{SSHA-512}vtN64f6/EPhtfsx9ozxR9zko0GoV8ZQb6ND2sWSu5qHmi4e0l4rYtHgO9noSyXaldj1IdE
Password (Hex):	bed37ae1febf10f86bb5fb31f68cf147dce4a741a857c6506fa343dac
Salt (Hex):	5266ebc683682a35

Finally, add a `uniqueMember` attribute to the Administrator group.

Select the new user's node in the tree, right click and select `Copy Entry / DN:`



Right click the `uniqueMember` attribute of the `Administrator` node, select `New Value` and paste the previously copied DN of the new user as the value:


```
identity:
  providers:
    - ldap:
        uri: ldap://<host>:<port>
        baseDn: <your-base-dn>
        rootDn: <DN of the ROOT user>
        rootDnPassword: <password of the ROOT user>
        usePool: false

repository:
  ...

database:
  ...
  username: snowowl
  password: snowowl ①
```

① Update MySQL username and password, if neccessary

Memory settings

Heap size used by Snow Owl can be adjusted in `dmk.sh`; look for the following seciton:

```
JAVA_OPTS="$JAVA_OPTS \
-Xms12g \
-Xmx12g \
```

`Xms` sets the minimum heap size, `Xmx` sets the maximum heap size used by the JVM.

OSGi console

The OSGi console can be accessed both via `ssh` and `telnet`. Configuration settings for remote access can be found in `osgi.console.properties`. The default settings are:

`/opt/snowowl-{edition}_{version}/repository/ext/osgi.console.properties`

```
telnet.enabled=true
telnet.port=2501
telnet.host=localhost
ssh.enabled=true
ssh.port=2502
ssh.host=localhost
```

Further information on how to enable/disable the OSGi console can be found here: <http://www.eclipse.org/virgo/documentation/virgo-documentation-3.6.4.RELEASE/docs/virgo-user-guide/html/ch08.html>.

For opening a telnet connection to the server, type:

```
$ telnet localhost 2501
Trying ::1...
Connected to localhost.
Escape character is '^]'.
osgi>
```

Logging

Log files are stored under `./opt/snowowl-{edition}_{version}/serviceability` directory of the Snow Owl server. The following log files are created:

logs/log.log

Generic system trace log file, all log messages are written into this file. Logs files are created for each day with the following file name format `log_%d{yyyy-MM-dd}`. Snow Owl will keep 90 days worth of history in this folder before starting to remove files. This log serves two main purposes:

1. It provides global trace files that capture high-volume information regarding the Virgo's internal events. The files are intended for use by support personnel to diagnose runtime problems.
2. It provides application trace files that contain application-generated output. This includes output generated using popular logging and tracing APIs including the OSGi LogService, as well as output generated by calls to `System.out` and `System.err`. These files are intended for use by application developers and system administrators. An application is defined as a scope so a single bundle will not get its own log file unless it is a Web application Bundle or is included in a scoped plan or a par file.

logs/access/*.log

Web container access log files in the same format as those created by standard web servers. The log files are prefixed with the string `localhost_access_log`, have a suffix of `.txt`, use a standard format for identifying what should be logged, and do not include DNS lookups of the IP address of the remote host.

eventlogs/eventlog.log

The `EVENT_LOG_FILE` appender logs only important events and thus the volume of information is lower.

logs/snowowl/snowowl_user_audit.log

Events with business significance will be logged in this file.

logs/snowowl/snowowl_user_access.log

User access events are logged in this log file. Both authorized and unauthorized access is logged.

logs/snowowl/snowowl_import.log

Import processes log into this file detailed information about import.

logs/snowowl/snowowl_export.log

Export processes log into this file detailed information about export.

Detailed information on the configuration on the logging configuration can be found [here](#):

<http://www.eclipse.org/virgo/documentation/virgo-documentation-3.6.4.RELEASE/docs/virgo-user-guide/html/ch11.html>.

Currently, default logging appenders for the log targets above look like this:

```
<appender name="LOG_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>serviceability/logs/log.log</file>
  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <fileNamePattern>serviceability/logs/log_%d{yyyy-MM-dd}.log</fileNamePattern>
    <!-- keep 90 days' worth of history -->
    <maxHistory>90</maxHistory>
  </rollingPolicy>
  <filter class="ch.qos.logback.core.filter.EvaluatorFilter">
    <evaluator class="ch.qos.logback.classic.boolex.OnMarkerEvaluator">
      <marker>SNOW_OWL_USER_ACCESS</marker>
    </evaluator>
    <onMismatch>ACCEPT</onMismatch>
    <onMatch>DENY</onMatch>
  </filter>
  <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
    <Pattern>[%d{yyyy-MM-dd HH:mm:ss.SSS}] %-5level %-28.28thread %-
64.64logger{64} %X{medic.eventCode} %msg %ex%n</Pattern>
  </encoder>
</appender>
```

In this setting, the administrator can set the location of the log file, the maximum size of the log file and the total number of files rolling over. Documentation on the logging configuration settings can be found here: <http://logback.qos.ch>.

Web Server Configuration

Snow Owl Server uses Tomcat as its built-in web server for administrative and RESTful services. The configuration settings for the web server can be found in `tomcat-server.xml`. Detailed information on configuring the different elements can be found here: <http://tomcat.apache.org/tomcat-7.0-doc/config/index.html>. The most important settings are the port numbers for HTTP and HTTPS protocols:

/opt/snowowl-{edition}_{version}/configuration/tomcat-server.xml

```
<Service name="Catalina">
  <Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" />
  <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="configuration/keystore"
    keystorePass="changeit"/>
```

Web Server Administrative Console application

The Admin Console is a web application for managing the Virgo Server instance powering Snow Owl Server. The default location of the admin console is at <http://localhost:8080/admin>.

The Admin Console is a password-protected page; to configure users allowed to access the Admin Console, change settings in file `org.eclipse.virgo.kernel.users.properties`. The username-password pair configured by default is `user=admin`, `pwd=adminpwd`:

/opt/snowowl-{edition}_{version}/configuration/org.eclipse.virgo.kernel.users.properties

```
#####
# User definitions
#####
user.admin=adminpwd

#####
# Role definitions
#####
role.admin=admin
```

More information on administrative user access control can be found on the following pages: <http://www.eclipse.org/virgo/documentation/virgo-documentation-3.6.4.RELEASE/docs/virgo-user-guide/html/ch09.html> and <http://www.eclipse.org/virgo/documentation/virgo-documentation-3.6.4.RELEASE/docs/virgo-user-guide/html/ch13s06.html#configuring-authentication>.

Virgo documentation

Complete documentation of the Virgo OSGi server can be found here: <http://www.eclipse.org/virgo/documentation>.

Administration

This section covers administrative and maintenance tasks in Snow Owl Server.

Startup and shutdown

Scripts for starting and stopping a Snow Owl Server instance are located in `/opt/snowowl-{edition}_{version}/bin/*.sh` files. Change the active user to `snowowl` when starting the server:

```
# su snowowl -s /bin/bash -c "nohup bin/startup.sh > /dev/null" &
[1] 12473
nohup: ignoring input and redirecting stderr to stdout

# jobs
[1]+  Running    su snowowl -s /bin/bash -c "nohup bin/startup.sh > /dev/null" &
```

Note that startup will continue in the background and the server stays running even if the user

disconnects from the server.

You can follow the startup sequence by inspecting `serviceability/logs/log.log`:

```
# tail -f serviceability/logs/log.log
...
[2017-01-24 02:50:12.753] INFO start-signalling-2 WE0001I Started web bundle [...]
[2017-01-24 02:50:12.756] INFO start-signalling-2 DE0005I Started bundle [...]
[2017-01-24 02:50:12.756] INFO start-signalling-2 Thread context class loader [...]
[2017-01-24 02:50:12.757] INFO start-signalling-2 DE0005I Started plan
'osgi_server.plan' version '1.0.0'.
```

Snow Owl Server finishes the startup procedure after displaying the lines from above. You can verify that everything is working properly by checking the server's version, the available databases and their health states via the OSGi console:

```
# yum install telnet ①

# telnet localhost 2501
Trying ::1...
Connected to localhost.
Escape character is '^]'.

osgi> snowowl --version ②
5.8.6

osgi> snowowl repositories
-----
|id           |health      |diagnosis    |
-----
|snomedStore  |GREEN       |-            |
-----

osgi> disconnect
Disconnect from console? (y/n; default=y) y
Connection closed by foreign host.
```

- ① Install the client first (if not already present)
- ② Should always print the currently running server's version

The server can be stopped cleanly by running `bin/shutdown.sh`:

```
# cd /opt/snowowl-{edition}_{version}
# bin/shutdown.sh
```

Importing SNOMED CT content

To import a `SnomedCT_Release_XXX_XXXXXXX.zip` release archive, the file must be present in the Snow Owl Server host file system, which we will refer to as `{import_archive}`.

Starting the import

Use the following OSGi command to start the import process:

```
osgi> sctimport rf2_release -t <type> -b <branch> -v /path/to/{import_archive}  
/path/to/{release_descriptor_file}
```

-t <type>

One of **FULL**, **SNAPSHOT** or **DELTA**, depending on the import type the administrator wants to use. The parameter is case-insensitive.

-b <branch>

The existing branch to import the content onto. In case of extension import, an effective time from the base SNOMED CT release (e.g. 2016-01-31). If omitted **MAIN** will be used.

-v

Creates versions for each effective time found in the release archive. If omitted no versions will be created.

/path/to/{import_archive}

Specifies the release archive to import. Must be a **.zip** file with a supported internal structure, such as the release archive of the International Release.

/path/to/{release_descriptor_file}

The path to the release descriptor **.json** file.

Release descriptor file

The release descriptor **.json** file could contain the following attributes as key/value pairs:

Name

Descriptive name of the code system.

Short name

Short name of the code system. Must **not** contain any whitespaces, must be **unique** among the other code systems already present in Snow Owl.



See **terminologyregistry listall** OSGi command to list all existing code systems.

Language

Three letter language code of the code system.

Code system OID

The OID of the code system.

Extension of

The short name of the base code system / release this SNOMED CT release depends on.

Maintaining organization link

Maintaining organization link.

Release type

The type of the release. Either **INTERNATIONAL** or **EXTENSION**.

Citation

Citation.

Examples

International import

```
osgi> sctimport rf2_release -t FULL /path/to/{international_import_archive}  
/path/to/snomed_ct_international.json
```

Where **snomed_ct_international.json** looks like:

```
{  
  "name": "Systematized Nomenclature of Medicine Clinical Terms International Version",  
  "shortName": "SNOMEDCT",  
  "language": "ENG",  
  "codeSystemOID": "2.16.840.1.113883.6.96",  
  "maintainingOrganizationLink": "http://www.ihtsdo.org",  
  "citation": "SNOMED CT contributes to the improvement of patient care by  
underpinning the development of Electronic Health Records that record clinical  
information in ways that enable meaning-based retrieval. This provides effective  
access to information required for decision support and consistent reporting and  
analysis. Patients benefit from the use of SNOMED CT because it improves the recording  
of EHR information and facilitates better communication, leading to improvements in  
the quality of care."  
}
```

Extension import

An extension import based on the 2015-01-31 version of the international release:

```
osgi> sctimport rf2_release -t FULL -b 2015-01-31 -v  
/path/to/{extension_import_archive} /path/to/snomed_ct_extension.json
```

Where `snomed_ct_extension.json` looks like:

```
{
  "name": "SNOMED CT Special Extension",
  "shortName": "SNOMEDCT-SE",
  "language": "ENG",
  "codeSystemOID": "2.16.840.1.113883.6.96.2",
  "extensionOf": "SNOMEDCT",
  "maintainingOrganizationLink": "http://www.snomed-special-extension.org",
  "citation": "Long citation about the details of SNOMED CT special extension"
}
```



While the import is running, feedback might be delayed on the OSGi console. Log output can be observed throughout the import session in the file `serviceability/logs/log.log`. The import may take several hours depending on your hardware and JVM configuration.

Single administrator operations

Import and export processes are dedicated single administrator operations. As these operations are long-running, administrators need to ensure that during these processes no other users should be connected to the system. The following steps describe how to disconnect all clients from the server and how to ensure that no one else, but the administrator is the only connected user while performing any import/export operations.

Steps to perform single admin operations

Checking the connected users from the OSGi server console, to list all connected users one should perform the following command:

```
osgi> session users
User: info@b2international.com ,session id: 9
```

Before starting to gracefully disconnect users, the administrator should disable non-administrator user logins to the server. To check the login status on the server:

```
osgi> session login status
Non-administrative logins are currently enabled.
```

As the response states above, there is no login restrictions applied. To restrict non-administrator logging, one should execute the following command:

```
osgi> session login disabled
Disabled non-administrative logins.
```

Now any users with insufficient privileges (in other words; users without 'Administrator' role) will be refused by the server when trying to connect.



None of the currently connected users will be disconnected. Connected users have to be disconnected by the administrator via the OSGi console as described later.

The administrator can send an informational message from the OSGi console to connected clients, so users can be informed about the upcoming maintenance:

```
osgi> session message ALL Server is going down in 10 minutes due to a SNOMED CT
publication process. Please commit all your unsaved changes.
Message sent to info@b2international.com
```

To disconnect all currently connected users:

```
osgi> session disconnect ALL
User: info@b2international.com ,session id: 9 was disconnected.
```



In this case, all clients, including the administrator will be logged out from the server, but the administrator may reconnect to the server as only non-administrative users are locked out.

After disabling non-administrator user login, notifying and disconnecting users, double-check of the current status and the connected users at the server:

```
osgi> session login status
Non-administrative logins are currently disabled.
```

```
osgi> session users
osgi>
```

It is now safe to perform any single administrator operations, such as an RF2 import. When finished, enable non-administrative connections again:

```
osgi> session login enabled
Enabled non-administrative logins.
```

Impersonating users

Snow Owl Server will ask for a user identifier for server-side import operations in the following cases:

- SNOMED CT RF2 import

- SNOMED CT MRCM import

The user identifier will be used for associating commits to the terminology repository with a user in the commit information view.

Taking backups

"Hot" backups

The example shell script `snowowl_hot_backup_mysql.sh` exercises all functionality mentioned above, and produces a .zip archive containing database dumps and copies of index folders in the directory it is started from. Please update the variable `SNOW_OWL_SERVER_HOME` so that it points to the installation folder of Snow Owl Server before running the script.

The return value is 0 for successful backups, and 1 if an error occurs while backing up content from the server. The script produces timestamped diagnostic output on its standard output; error messages are directed to the standard error output.

To create backups regularly, add a dedicated non-login user for backups as root:

```
# useradd -r -M -d / -s /sbin/nologin -c "Snow Owl Backup" snowowl-backup
```

Create and/or update access privileges of the backup destination, log output, and the location of the singleton instance lock file:

```
# mkdir -pv /storage/backups /var/log/snowowl-backup /var/run/snowowl-backup
mkdir: created directory '/storage/backups'
mkdir: created directory '/var/log/snowowl-backup'
mkdir: created directory '/var/run/snowowl-backup'

# chown -v root:snowowl-backup /storage/backups /var/log/snowowl-backup
/var/run/snowowl-backup
changed ownership of '/storage/backups' to root:snowowl-backup
changed ownership of '/var/log/snowowl-backup' to root:snowowl-backup
changed ownership of '/var/run/snowowl-backup' to root:snowowl-backup

# chmod -v 775 /storage/backups /var/log/snowowl-backup /var/run/snowowl-backup
mode of '/storage/backups' changed to 0775 (rwxrwxr-x)
mode of '/var/log/snowowl-backup' changed to 0775 (rwxrwxr-x)
mode of '/var/run/snowowl-backup' changed to 0775 (rwxrwxr-x)
```

Save the backup script in an accessible place, set the owner to snowowl-backup, and make it executable:

```
# chown -v snowowl-backup: /storage/backups/snowowl_full_backup_mysql.sh
changed ownership of '/storage/backups/snowowl_full_backup_mysql.sh' to snowowl-
backup:snowowl-backup

# chmod -v 744 /storage/backups/snowowl_full_backup_mysql.sh
mode of '/storage/backups/snowowl_full_backup_mysql.sh' changed to 0744 (rwxr--r--)
```

Add the script to the backup user's crontab (the example runs the script at 4 AM, and outputs log entries to logfiles with a year-month-date suffix in /var/log/snowowl-backup):

```
# EDITOR=nano crontab -e -u snowowl-backups

<nano opens; add the content below to the opened file, save, and exit the editor>

# MAILTO="local-user"
#
# Minute - Hour - Day of month - Month - Day of week - Command
0 4 * * * cd /storage/backups && ( ./snowowl_full_backup_mysql.sh >> /var/log/snowowl-
backup/log-`date +%Y%m%d` 2>&1 )
```

(If the standard error output is not redirected with the "2>&1" part of the command, errors will be captured by cron and mailed to the snowowl-backup user's mailbox. The destination can be changed by uncommenting the MAILTO parameter and setting it to a different address.)

"Cold" backups

When the server is shut down, the above mentioned REST service for enumerating store content and getting exclusive write locks for the repositories is not available, so a separate script, `snowowl_cold_backup_mysql.sh` is being provided for this case.

User and Access control

Snow Owl supports role based access control by organizing **Users** into a set of **Roles**. A **Role** is essentially a group that has a set of assigned **Permissions**. **Permissions** allow users to do certain things in the system. If **User A** belongs to a **Role** called *R* that has a permission called *P*, then user *A* can perform operations that require permission *P*.

Permissions

- browse:* - to be able to browse/search/view terminology content
- edit:* - to be able to create/modify/delete terminology content
- import:* - to be able to modify terminology content via import operation
- export:* - to be able to export terminology content into various formats
- version:* - to be able to create a terminology version

- promote:* - to be able to merge task changes to the terminology working branch (aka MAIN)

Default Roles

Snow Owl comes with a set of predefined **Roles** that can be used to organize your users in order to restrict them from performing something they should not be allowed to do. Each role has a default set of assigned permissions

- Browser - browse, export
- Editor - browse, edit, import, export
- Reviewer - browse, export, promote
- Administrator - browse, edit, import, export, promote, version

Administrator role is a special **Role** that is required by Snow Owl. It must be kept as is in order to run Snow Owl without issues. Snow Owl uses this role to identify users who can version and release content of a terminology and edit anything on the terminology's working branch (aka MAIN or equivalent).

Customizing access control

Snow Owl must be configured to use *LDAP* as external source of user and access control information in order to customize it to your needs.



If you're using an ApacheDS LDAP service (as mentioned in the install guide earlier than 5.10.11), then please switch to the more secure and robust OpenLDAP service. To install OpenLDAP please see the provided `ldap/docker-compose.yml` file in the distribution archive or this Dockerfile (<https://github.com/osixia/docker-openldap/blob/stable/image/Dockerfile>) if you would like to install OpenLDAP manually. Also you can install OpenLDAP manually by following a guide matching your deployment environment (CentOS 7.x/Ubuntu 14.04/Other).

Once you have a running OpenLDAP service you can connect to it with the ApacheDS Studio you've used for the ApacheDS LDAP service.

Customize schema

Snow Owl requires a permission schema object to be present in the LDAP schema in order to represent permissions in LDAP:

1. Connect to the running OpenLDAP service via an LDAP client using the `cn=admin,cn=config` user and `config` password.
2. Open the `permission_schema.ldif` in the ApacheDS Studio via Open File... menu
3. Execute the LDIF file against the OpenLDAP service

Import default configuration

After successfully modifying the schema we can import Snow Owl's permissions and the default roles.

1. Connect to the running OpenLDAP service via an LDAP client using the `cn=admin,dc=snowowl,dc=b2international,dc=com` user and configured admin password.
2. Open and execute the following LDIF files:
 - a. `permissions.ldif` - contains permissions recognized by Snow Owl
 - b. `roles.ldif` - contains default roles
 - c. `pm.ldif` - contains default permission → role mapping

Add new user

See Section [Creating a new user from Directory Studio](#) in Snow Owl installation guide on how to add new users to the LDAP repository.

Modify access control

To customize access control, first connect to your running OpenLDAP service using the `cn=admin,dc=snowowl,dc=b2international,dc=com` user and configured admin password.

Create new Role

1. Create a new child entry under `dc=snowowl,dc=b2international,dc=com` using the *Administrator* Role as template
2. Change the name of your new **Role**
3. Finish the wizard

Modify Roles

Select the **Role** you would like to edit

- You can edit name by double clicking on it then changing it
- You can add new permissions by **Right Click** → **New Attribute**
- You can remove permissions by **Right Click** → **Delete Value** on a permissionId row
- You can assign users via **Right Click on uniqueMember** → **New Value**
- You can unassign users via **Right Click on a uniqueMember row** → **Delete Value**

Console command reference

This section lists commands available to the system administrator when connecting to the OSGi console of a running Snow Owl Server instance using telnet or SSH. Instructions for setting up the console can be found in the section titled “OSGi console” of the Configuration guide.



Depending on log configuration, the output shown following user-entered commands in the usage examples might only appear in the service log, located at `serviceability/logs/log.log` in the Snow Owl Server installation folder, or the export/import logs in folder `serviceability/logs/snowowl`.

General

To get a quick overview of all available commands, type `help` into the console:

```
osgi> help
---CDO commands---
    cdo list - list all active repositories
    cdo start - start repositories from a config file
    cdo stop - stop a repository
...
service - list all services in the service registry
    scope: vsh
    parameters:
        String operation (list)

service - examine a service in the service registry
    scope: vsh
    parameters:
        String operation (examine)
        long service id

shutdown - shut down the kernel
    scope: vsh
```

The displayed list of commands are a mixture of application-specific and framework-specific operations; the following categories provide functionality related to Snow Owl Server itself:

- Diagnostic and Maintenance commands — `snowowl`
- Terminology Registry commands — `terminologyregistry`
- Session Management commands — `session`
- Scripting commands — `script`
- SNOMED CT Importer commands — `sctimport`
- SNOMED CT OWL Ontology commands — `ontology`
- MRCM commands — `mrcm`

To narrow the command list down to a certain category, type the category name given above:

---Snow Owl commands---

`snowowl --version` - returns the current version

`snowowl dbcreateindex [nsUri]` - creates the CDO_CREATED index on the proper DB tables for all classes contained by a package identified by its unique namespace URI.

`snowowl listrepositories` - prints all the repositories in the system.

`snowowl listbranches [repository] [branchPath]` - prints all the child branches of the specified branch path in the system for a repository. Branch path is MAIN by default and has to be full path (e.g. MAIN/PROJECT/TASK)

`snowowl reindex [repositoryId] [failedCommitTimestamp]` - reindexes the content for the given repository ID from the given failed commit timestamp (optional, default timestamp is 1 which means no failed commit).

`snowowl optimize [repositoryId] [maxSegments]` - optimizes the underlying index for the repository to have the supplied maximum number of segments (default number is 1)

`snowowl purge [repositoryId] [branchPath] [ALL|LATEST|HISTORY]` - optimizes the underlying index by deleting unnecessary documents from the given branch using the given purge strategy (default strategy is LATEST)

`snowowl repositories [repositoryId]` - prints all currently available repositories and their health statuses

Disconnecting from the console

To leave the console (and keep the server running), type **disconnect**, then confirm the operation by pressing Enter. Alternatively, you can type "y" to confirm, or "n" to back out.

```
osgi> disconnect
Disconnect from console? (y/n; default=y) y

Connection closed by foreign host.
```

Available commands

Scripting

Execute Groovy script

script run parses and executes the specified Groovy script. Services will be provided by the running server instance, similarly to the Groovy editing environment within the Snow Owl client.

Make sure to inspect the server log for any issues, as they might not be printed to the console output depending on the log configuration.

```

osgi> script run /home/user/IndexExample.groovy

[Systemic blood pressure]
Number of results for 'hyp' query term: 3566
[Hypertensive disorder, Hyperlipidaemia, Asthma]
Number of results for 'hyp' AND 'blood cell' query term: 100
Autologous peripheral blood stem cell transplant
White blood cell disorder
...
Red blood cell count, manual, peritoneal fluid
Red blood cell folate borderline high
ID          Label
425983008   Autologous peripheral blood stem cell transplant
54097007    White blood cell disorder
...
442218004   Red blood cell folate borderline high

```

Machine-Readable Concept Model

Import MRCM rules from XMI file

mrcm import reads and applies rules from the specified source file. Note that as with regular editing of MRCM rules, only SNOMED CT concept and reference set editors opened after the import will display their output with the changes considered.

This command also requires the specification of a user identifier, which will be presented as the importing user in the commit information view.

```

osgi> mrcm import /path/to/mrcm_20131212143528517.xmi
Impersonate operation as: info@b2international.com

... - Importing MRCM rules...
...
... - Persisting changes...
... - Changes have been successfully persisted.
... - Branch: MAIN Event: SNOMED CT Changes: changed concepts: [123037004:Body
structure], ...
...
... - MRCM rule import successfully finished.

```

Export MRCM rules to an XMI file

mrcm export creates a file named **mrcm_{timestamp}.xmi** in the directory given by the administrator. The command requires a user identifier which will be recorded in the user audit log.

```
osgi> mrcm export /path/to/export/folder
Impersonate operation as: info@b2international.com

... - Exporting MRCM rules...
...
... - MRCM rule export successfully finished.
```

Terminology registry

List all imported terminologies

`terminologyregistry listall` displays information about terminologies and terminology extensions imported into the running server instance, including individual local code systems.

```
osgi> terminologyregistry listall

Name: SNOMED CT
Short name: SNOMEDCT
Code System OID: 2.16.840.1.113883.6.96
Maintaining organization link: http://www.snomed.org
Language: ENG
Last version: 0
Current branch path: MAIN
```

SNOMED CT OWL Ontology (reasoner)

Display available reasoners

To list the available reasoners and the preferred one (marked with an ***** symbol) one has to perform the following:

```
osgi> ontology list

0 None [version: 4.1.0] (org.protege.editor.owl.NoOpReasoner)
1 ELK 0.3.2 [version: 0.3.2] (org.semanticweb.elk.elk.reasoner.factory)
* 2 MORE A (0.1.3) [version: 0.1.3] (org.semanticweb.more.MORE.reasoner.factory)
3 MORE B (0.1.3) [version: 0.1.3] (org.semanticweb.more.MORERLrew.reasoner.factory)
4 FaCT++ [version: 1.6.2] (uk.ac.manchester.cs.owl.factplusplus.factplusplus-
factory)
```

Change the active reasoner

To change the preferred reasoner on the server (in our case from **MORE A** to **FaCT++**), use the following command:

```
osgi> ontology select 4
```

```
0 None [version: 4.1.0] (org.protege.editor.owl.NoOpReasoner)
1 ELK 0.3.2 [version: 0.3.2] (org.semanticweb.elk.elk.reasoner.factory)
2 MORE A (0.1.3) [version: 0.1.3] (org.semanticweb.more.MORE.reasoner.factory)
3 MORE B (0.1.3) [version: 0.1.3] (org.semanticweb.more.MORERLrew.reasoner.factory)
* 4 FaCT++ [version: 1.6.2] (uk.ac.manchester.cs.owl.factplusplus.factplusplus-
factory)
```

Note that this setting does not affect ongoing computations if they were started using a different reasoner.

Checking status of available reasoners

The command checks the presence and availability of all reasoners available on the server side. In case of the response below, all reasoners are available.

```
osgi> ontology check
All reasoner instances are available and ready for use.
```

Whenever any of the reasoners is not available, the output should contain the problematic reasoner identifier. Please note that reasoner identifiers may vary; also, if more than one reasoner reports a problem, a list of reasoner identifiers will be printed to the console:

```
osgi> ontology check
Couldn't initialize reasoner factory for ID 'unique.id.of.the.reasoner'.
```

For getting the original cause of the reasoner availability issue, one could dump the exception by appending the **-d** flag:

```

osgi> ontology check -d
Couldn't initialize reasoner factory for ID 'unique.id.of.the.reasoner'.

com.b2international.snowowl.snomed.reasoner.exceptions.ReasonerException: Couldn't
initialize reasoner factory for ID 'unique.id.of.the.reasoner'.
    at
com.b2international.snowowl.snomed.reasoner.server.preferences.ReasonerPreferencesServ
ice.createReasonerInfo(ReasonerPreferencesService.java:306)
    at
com.b2international.snowowl.snomed.reasoner.server.preferences.ReasonerPreferencesServ
ice.checkAllAvailableReasoners(ReasonerPreferencesService.java:270)
    at
com.b2international.snowowl.snomed.reasoner.server.console.SnomedOntologyCommandProvid
er$Command$3.execute(SnomedOntologyCommandProvider.java:65)
    at
com.b2international.snowowl.snomed.reasoner.server.console.SnomedOntologyCommandProvid
er._ontology(SnomedOntologyCommandProvider.java:150)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
    at java.lang.reflect.Method.invoke(Method.java:597)
    at
org.eclipse.osgi.framework.internal.core.FrameworkCommandInterpreter.execute(Framework
CommandInterpreter.java:209)
...

```

SNOMED CT

Import reference sets in RF2 format from a release text file

Use `sctimport rf2_refset` to import one or more reference sets from an RF2 text file. All RF2 import modes (**FULL**, **SNAPSHOT** and **DELTA**) are available; certain reference set members can be excluded from being imported based on their reference set identifiers.

The following example imports a snapshot release file from the SNOMED CT International RF2 release, excluding two reference sets by identifier:

```

osgi> sctimport rf2_refset /path/to/der2_Refset_SimpleSnapshot_INT_20130731.txt -t
SNAPSHOT -x 447566000 447565001

[2013-12-12 17:10:47.987] [OSGi Console] INFO  c.b.s.s.i.rf2.util.ImportUtil - SNOMED
CT import started from RF2 release format.
[2013-12-12 17:10:47.987] User: web Event: SNOMED CT import started from RF2 release
format.
Importing release files...: 0% [0ms]
[2013-12-12 17:10:47.988] [OSGi Console] INFO  c.b.s.s.i.rf2.util.ImportUtil -
Validating release files...
[2013-12-12 17:10:47.988] [OSGi Console] INFO  c.b.s.s.i.rf2.util.ImportUtil -

```

Validating RF2 release files.

[2013-12-12 17:10:47.988] User: web Event: Validating RF2 release files.

[2013-12-12 17:10:47.988] [OSGi Console] INFO c.b.s.s.i.rf2.util.ImportUtil -
Creating staging directory '...' for simple type reference set member validation.

Preparing simple type reference set members validation: 5% [96ms]

[2013-12-12 17:10:48.083] [OSGi Console] INFO c.b.s.s.i.rf2.util.ImportUtil -
Validating 'der2_Refset_SimpleSnapshot_INT_20130731.txt' release file.

[2013-12-12 17:10:48.083] User: web Event: Validating
'der2_Refset_SimpleSnapshot_INT_20130731.txt' release file.

Validating simple type reference set members...: 11% [868ms]

[2013-12-12 17:10:48.954] [OSGi Console] INFO c.b.commons.db.JdbcUtils - Connected to
database '...'.

Finishing simple type reference set members validation: 17% [3ms]

[2013-12-12 17:10:48.954] [OSGi Console] INFO c.b.s.s.i.r.m.AbstractSnomedImporter -

Preparing simple type reference set member import

[2013-12-12 17:10:48.954] User: web Event: Preparing simple type reference set member
import

[2013-12-12 17:10:48.954] [OSGi Console] INFO c.b.s.s.i.rf2.util.ImportUtil -
Creating staging directory '...' for simple type reference set member import.

[2013-12-12 17:10:48.954] [OSGi Console] INFO c.b.s.s.i.r.m.AbstractSnomedImporter -
Creating staging directory '...' for simple type reference set member import.

[2013-12-12 17:10:48.954] User: web Event: Creating staging directory '...' for simple
type reference set member import.

Preparing simple type reference set member import: 20% [1ms]

[2013-12-12 17:10:48.955] [OSGi Console] INFO c.b.s.s.i.rf2.util.ImportUtil -
Populating storage keys for simple type reference set member import.

[2013-12-12 17:10:48.955] [OSGi Console] INFO c.b.s.s.i.r.m.AbstractSnomedImporter -
Populating storage keys for simple type reference set member import.

[2013-12-12 17:10:48.955] User: web Event: Populating storage keys for simple type
reference set member import.

Preparing simple type reference set member import: 21% [2ms]

Preparing simple type reference set member import: 22% [595ms]

Preparing simple type reference set member import: 23% [8382ms]

[2013-12-12 17:10:58.002] [OSGi Console] INFO c.b.s.s.i.r.m.AbstractSnomedImporter -
Collecting simple type reference set member import units

[2013-12-12 17:10:58.002] User: web Event: Collecting simple type reference set member
import units

Collecting simple type reference set member import units: 24% [92ms]

...

[2013-12-12 17:10:58.148] [OSGi Console] INFO c.b.s.s.i.r.m.AbstractSnomedImporter -
Processing simple type reference set members

[2013-12-12 17:10:58.148] User: web Event: Processing simple type reference set
members

...

Processing simple type reference set members: 88% [17ms]

[2013-12-12 17:10:59.325] [OSGi Console] INFO c.b.s.s.i.r.m.AbstractSnomedImporter -
Finishing simple type reference set member import

[2013-12-12 17:10:59.325] User: web Event: Finishing simple type reference set member
import

[2013-12-12 17:10:59.325] [OSGi Console] INFO c.b.s.s.i.rf2.util.ImportUtil -
Creating indexes for simple type reference set member import.

```

[2013-12-12 17:10:59.325] [OSGi Console] INFO c.b.s.s.i.r.m.AbstractSnomedImporter -
Creating indexes for simple type reference set member import.
[2013-12-12 17:10:59.325] User: web Event: Creating indexes for simple type reference
set member import.
[2013-12-12 17:10:59.334] [OSGi Console] WARN c.b.s.s.i.rf2.util.ImportUtil -
Couldn't create or drop index SNOMEDREFSET_SNOMEDREFSETMEMBER_IDX1000 for table
SNOMEDREFSET_SNOMEDREFSETMEMBER.
[2013-12-12 17:10:59.334] [OSGi Console] WARN c.b.s.s.i.rf2.util.ImportUtil -
Couldn't create or drop index SNOMEDREFSET_SNOMEDREFSETMEMBER_IDX1001 for table
SNOMEDREFSET_SNOMEDREFSETMEMBER.
Finishing simple type reference set member import: 89% [10ms]
[2013-12-12 17:10:59.335] [OSGi Console] WARN c.b.s.s.i.rf2.util.ImportUtil -
Couldn't create or drop index SNOMEDREFSET_SNOMEDREFSETMEMBER_IDX1002 for table
SNOMEDREFSET_SNOMEDREFSETMEMBER.
Finishing simple type reference set member import: 90% [1ms]
[2013-12-12 17:10:59.336] [OSGi Console] WARN c.b.s.s.i.rf2.util.ImportUtil -
Couldn't create or drop index SNOMEDREFSET_SNOMEDREFSETMEMBER_IDX1003 for table
SNOMEDREFSET_SNOMEDREFSETMEMBER.
Finishing simple type reference set member import: 91% [1ms]
[2013-12-12 17:10:59.336] [OSGi Console] INFO c.b.s.s.i.rf2.util.ImportUtil -
Removing staging directory '...' from simple type reference set member import.
[2013-12-12 17:10:59.336] [OSGi Console] INFO c.b.s.s.i.r.m.AbstractSnomedImporter -
Removing staging directory '...' from simple type reference set member import.
[2013-12-12 17:10:59.336] User: web Event: Removing staging directory '...' from
simple type reference set member import.
Finishing simple type reference set member import: 94% [1ms]
Finishing simple type reference set member import: 100% [0ms]
[2013-12-12 17:10:59.337] [OSGi Console] INFO c.b.s.s.i.rf2.util.ImportUtil - SNOMED
CT import successfully finished.
[2013-12-12 17:10:59.337] User: web Event: SNOMED CT import successfully finished.

```

Note that messages related to not being able to create or drop database indexes are not an indication of a failed import process.

Import reference sets in DSV format

`sctimport dsv_refset` imports a single reference set from text files in Delimiter Separated Values (DSV) format. The syntax is as follows:

```
sctimport dsv_refset <path> <hasHeader> <skipEmptyLines> <parentConcept>
```

<path>

Specifies the file to be used for importing

<hasHeader>

Set to `true` if the source text file has a header row, `false` otherwise

<skipEmptyLines>

Set to `true` if the source text file has empty lines which should be ignored, `false` otherwise

<parentConcept>

Set to an integer value specifying the parent of the imported reference set's identifier concept

Accepted values for parentConcept are:

1. Simple type
2. B2i examples
3. KP Convergent Medical Terminology
4. CORE Problem List
5. Infoway Primary Health Care

The reference set name is determined by the input file name; as an example, `CamelCase.csv` will be converted to `Camel Case reference set`. An attempt will be made to interpret the first column of each line as a SNOMED CT concept identifier. If the identifier can be resolved, a member will be added to the reference set, otherwise an exception is thrown.

```
osgi> sctimport dsv_refset /path/to/SampleConcepts.txt false true 0
Impersonate operation as: info@b2international.com

Importing Interesting Reference Set...
[2013-12-12 17:22:03.154] [OSGi Console] INFO c.b.s.d.s.CDOServerChangeManager -
handleTransactionBeforeCommitting() start
[2013-12-12 17:22:03.154] [OSGi Console] INFO c.b.s.d.s.CDOServerChangeManager -
handleTransactionBeforeCommitting() lock acquired for BranchPath{Path='MAIN'}
[2013-12-12 17:22:03.155] [Worker-36] INFO
c.b.s.s.r.s.c.SnomedReasonerChangeProcessor - >>> Processing OWL ontology changes
[2013-12-12 17:22:03.155] [Worker-36] INFO
c.b.s.s.r.s.c.SnomedReasonerChangeProcessor - --- Processing OWL ontology changes:
change processing skipped, no ontology instance present for branch or running in
embedded mode
[2013-12-12 17:22:03.156] [Worker-36] INFO
c.b.s.s.r.s.c.SnomedReasonerChangeProcessor - <<< Processing OWL ontology changes
[249.6 µs]
[2013-12-12 17:22:03.156] [Worker-43] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Processing and updating changes...
[2013-12-12 17:22:03.156] [Worker-43] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Retrieving taxonomic information from store.
[2013-12-12 17:22:03.313] [Worker-42] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Processing changes taxonomic information.
[2013-12-12 17:22:03.313] [Worker-34] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Building taxonomic information.
[2013-12-12 17:22:03.315] [Worker-42] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Rebuilding taxonomic information based on the changes.
[2013-12-12 17:22:03.573] [Taxonomy difference processor] INFO
c.b.s.d.s.s.i.SnomedCDOChangeProcessor - Calculating taxonomic differences...
[2013-12-12 17:22:03.586] [Worker-43] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Updating reference set membership changes...
```



```
[2013-12-12 17:22:03.586] [Worker-43] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Updating taxonomy...
[2013-12-12 17:22:03.602] [Taxonomy difference processor] INFO
c.b.s.d.s.s.i.SnomedCDOChangeProcessor - Calculating taxonomic differences
successfully finished.
[2013-12-12 17:22:03.602] [Worker-43] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Processing and updating changes successfully finished.
[2013-12-12 17:22:03.602] [OSGi Console] INFO c.b.s.d.s.CDOServerChangeManager -
handleTransactionBeforeCommitting() end
[2013-12-12 17:22:03.628] [OSGi Console] INFO c.b.s.d.s.CDOServerChangeManager -
handleTransactionAfterCommitted() start
[2013-12-12 17:22:03.628] [Worker-36] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Persisting changes...
[2013-12-12 17:22:03.689] [Worker-36] INFO c.b.s.d.s.s.i.SnomedCDOChangeProcessor -
Changes have been successfully persisted.
[2013-12-12 17:22:03.689] User: info@b2international.com Branch: MAIN Event: SNOMED CT
Changes: new concepts added: [745288891000154109:Sample Concepts reference set],
changed concepts: [446609009:Simple type reference set], new reference sets:
[745288891000154109:Sample Concepts reference set],
[2013-12-12 17:22:03.690] [OSGi Console] INFO c.b.s.d.s.CDOServerChangeManager -
handleTransactionAfterCommitted() end
[2013-12-12 17:22:03.690] [OSGi Console] INFO c.b.s.d.s.CDOServerChangeManager -
handleTransactionAfterCommitted() lock released for BranchPath{Path='MAIN'}
[2013-12-12 17:22:03.769] [OSGi Console] INFO c.b.s.d.PostStoreUpdateManager - Commit
notification received for user info@b2international.com.
All concepts were imported.
```

Diagnostics and maintenance

snowowl `--version` prints the current server version.

```
osgi> snowowl --version
5.8.0
```

snowowl `repositories [id]` prints information about all repositories or just a single repository

```
osgi> snowowl repositories
-----
|id           |health      |diagnosis    |
-----
|snomedStore  |GREEN       |-             |
-----
```

snowowl `listbranches [repository]` prints all the branches in the system for a repository.

```

osgi> snowowl listbranches snomedStore
Branches for repository snomedStore:
|---MAIN
|   |---2011-10-01
|   |---2012-01-31
|   |---2012-07-31
|   |---2013-01-31
|   |---2013-07-31
|   |---2014-01-31
|   |---2014-07-31
|   |---2015-01-31
|       |---SNOMED-CT-SE
|           |---2012-12-21
|           |---2013-05-31
|           |---2013-11-30
|           |---2014-05-31
|           |---2014-11-30
|           |---2015-05-31
|---2015-07-31
|---2016-01-31
|       |---SNOMED-CT-SE

```

snowowl dbcreateindex [nsUri] creates the CDO_CREATED index on the proper DB tables for all classes contained by a package identified by its unique namespace URI.

snowowl reindex [repositoryId] recreates the entire index for the content of the given repository. This long-running process requires the the server to be shut-down, the index to be deleted manually and a restart before invoking this command.

```
osgi> snowowl reindex snomedStore
```

snowowl optimize [repositoryId] [maxSegments] optimizes the underlying index for the repository to have the supplied maximum number of segments (default number is 1).

```
osgi> snowowl optimize snomedStore 6
```

snowowl purge <repositoryId> <branchPath> <purgeStrategy> optimizes the underlying index by deleting unnecessary documents from the given branch using the given purge strategy (default strategy is LATEST, available strategies are ALL, LATEST, HISTORY)

```
osgi> snowowl purge snomedStore MAIN/2016-01-31 ALL
```

Session management

Display connected users and session identifiers

To list all connected users (and the unique session ID), run the following OSGi command:

```
osgi> session users

User: akitta@b2international.com | session ID: 3
User: obali@b2international.com | session ID: 4
User: zstorok@b2international.com | session ID: 5
User: apeteri@b2international.com | session ID: 6
```

Send message to users

To send message to all connected users, use the following command:

```
osgi> session message ALL Some message from the administrator.

Message sent to akitta@b2international.com
Message sent to obali@b2international.com
Message sent to zstorok@b2international.com
Message sent to apeteri@b2international.com
```

All connected client will receive the message via a dialog.

For sending message to a subset of recipient users, execute the following OSGi command:

```
osgi> session message obali@b2international.com,zstorok@b2international.com Message
from the administrator to Orsi and Zsolt.

Message sent to obali@b2international.com
Message sent to zstorok@b2international.com
```

Restrict user logins

Administrator may restrict non-administrator user log in to the sever with the following:

```
osgi> session login disabled

Disabled non-administrative logins.
```

Users will not be able to connect to the server while non-administrator log in is disabled. Clients will receive the following when trying to connect to the server from the splash screen:

```
Logging in for non-administrator users is temporarily disabled.
```

Invoking this command will not disconnect any of the connected non-administrator users. The way how to disconnect clients from the server will be discussed below.

To re-enable non-administrator log in onto the server refer to the following command:

```
osgi> session login enabled  
  
Enabled non-administrative logins.
```

The status can be checked with the following command:

```
osgi> session login status  
  
Non-administrative logins are currently enabled.
```

Disconnect users

To disconnect a subset of connected users from the server, the following command should be performed:

```
osgi> session disconnect akitta@b2international.com,apeteri@b2international.com  
  
User: akitta@b2international.com ,session id: 3 was disconnected.  
User: apeteri@b2international.com ,session id: 6 was disconnected.
```

All disconnected users will receive a message about the lost connection. Then client application could be closed gracefully. This will not prevent users to reconnect the server.

The recommended way to ensure that none of the users are connected to the server when performing any single system administrator task is the following:

- Disable non-administrator log in in the server
- Notify users about the upcoming system admin operation
- Disconnect all users

Repository management

Display terminology repositories

To list all available repositories and their identifiers, one should execute the following command:

```
osgi> session repositories
```

```
LOINC Store [ID: loincStore]
Local Code System Store [ID: localterminologyStore]
Terminology Metadata Store [ID: terminologyregistryStore]
ICD-10 Store [ID: icd10Store]
Value Set Store [ID: valuesetStore]
ATC Store [ID: atcStore]
SNOMED CT Store [ID: snomedStore]
Mapping Set Store [ID: mappingsetStore]
ICD-10-AM Store [ID: icd10amStore]
```

Display currently held locks

To view a table of acquired locks, their targets and owners, execute the command:

```
osgi> session showlocks
```

No locks are currently granted on this server.

```
osgi> session lock allrepositories
```

Acquired lock for all repositories.

```
osgi> session showlocks
```

Id	Lvl	Created on	Locked area	Owner context
0	1	2014-01-31 21:16	All repositories	Lock owner: System
				Performing maintenance from the server console

```
osgi> session lock allrepositories
```

Acquired lock for all repositories.

Id	Lvl	Created on	Locked area	Owner context
0	2	2014-01-31 21:16	All repositories	Lock owner: System
				Performing maintenance from the server console

Locks can be acquired for different purposes, such as:

- administrative maintenance

- backup
- saving editors
- classification

Their area of effect can also vary:

- all terminology stores
- a single terminology store
- a single branch of a particular terminology store

Once a lock owner obtains a lock, the associated area is available for their use only; others will receive indications that someone else is already working on something which requires uninterrupted access to the target area. Lock attempts on the same or overlapping areas will not be able to complete until the lock is released.

The "Lvl" column indicates the "nesting" count of a granted lock; when someone holds a lock, they can lock the same area multiple times. Ownership is only released when the level decreases to 0.

Lock and release areas of terminology stores

To lock all terminology stores simultaneously, issue the following command:

```
osgi> session lock allrepositories  
Acquired lock for all repositories.
```

If a conflicting lock has already been acquired by a different owner, the reason for not granting this request will be displayed in the response.

To lock all branches of a single terminology store, refer to the repository identifiers returned by the `session repositories` command:

```
osgi> session lock snomedStore  
Acquired lock for repository 'snomedStore'.
```

Similarly, for locking a single branch of a single repository, type:

```
osgi> session lock snomedStore MAIN/a
Acquired lock for branch 'MAIN/a' of repository 'snomedStore'.
```

```
osgi> session showlocks
```

Id	Lvl	Created on	Locked area	Owner
context				

0	1	2014-01-31 21:16	All repositories	Lock owner:
System				Performing
				maintenance from the server console

1	1	2014-01-31 21:30	Repository 'snomedStore'	Lock owner:
System				Performing
				maintenance from the server console

2	1	2014-01-31 21:30	Branch 'MAIN' of repository 'snomedStore'	Lock owner:
System				Performing
				maintenance from the server console

The branch path argument is case sensitive; as an example, `main`, `Main`, `main/a` and `Main/a` branch paths would be invalid arguments.

Releasing an owned lock can be performed by executing a corresponding `session unlock` command:

```
osgi> session unlock snomedStore MAIN
Released lock for branch 'MAIN' of repository 'snomedStore'.
```

```
osgi> session showlocks
```

Id	Lvl	Created on	Locked area	Owner context
0	1	2014-01-31 21:16	All repositories	Lock owner: System
				Performing maintenance from the server console
1	1	2014-01-31 21:30	Repository 'snomedStore'	Lock owner: System
				Performing maintenance from the server console



Regular save operations try to get the lock for their target repository and branch. If the administrator has already taken over an area by using the commands above, a dialog will be displayed when the user tries to save after approx. 5 seconds of waiting for the lock to be granted.

Forcefully unlock stuck locks

If an operation gets stuck, or otherwise fails to release locks for which the System user is not the owner, other users may be blocked indefinitely as a result. To resolve the situation, one can forcefully unlock such locks by referring to their identifier shown in the table:

```
osgi> session forceunlock 1
Forcefully released lock with identifier 1.
```

```
osgi> session showlocks
```

Id	Lvl	Created on	Locked area	Owner context
0	1	2014-01-31 21:16	All repositories	Lock owner: System
				Performing maintenance from the server console

To forcefully release all locks, use the command with the **all** argument instead of the identifier.

Remote jobs

Remote jobs are long-running operations intended to be executed on the server; the requesting user's client need not be kept open while the job is active. The result of these computations can be checked immediately, or at a later time, if the results are still available for review. The administration console provides two commands for listing currently running remote jobs, and requesting cancellation of these items.

Display remote jobs

To list all currently scheduled, running, or finished remote jobs, type:

```
osgi> remotejobs list
```

Id	Description	Owner	Scheduled	Status
0	Batch ontology generation	info@b2intern...	2014-03-20 11:03	Finished
1	Classifying the ontology on MAIN	info@b2intern...	2014-03-20 11:13	Finished

(Note that identifiers in the first column can change at any time if, for example, the initiator of the task removes a completed job from their list.)

Cancel remote job

To signal a remote job that it should finish its work at the closest possible occasion without completing it fully, use the following command with the identifier from the list displayed above. If `remotejobs list` was not invoked earlier, the following message will be printed to the console:

```
osgi> remotejobs cancel 0
Please retrieve the list of currently scheduled or running jobs first.
```

Otherwise, when a valid job identifier is given, the following output should appear:

```
osgi> remotejobs cancel 0
Requesting job 0 to cancel.
```

...and an additional invocation of `remotejobs list` should list the given job's status as "Cancel requested".



Not all remote jobs are able to react to cancel requests.

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	<code>[]</code>	<code>file</code> , <code>ldap</code> - 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.2Bb1CxsJ0WL2F114XBCjMrMaqq.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	``	<code>ldap://<host>:<port></code> - LDAP host and port
baseDn	<code>dc=snowowl,dc=b2international,dc=com</code>	Base Dn value of the organization from which user identities will be retrieved.
rootDn	<code>cn=admin,dc=snowowl,dc=b2international,dc=com</code>	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	<code>h2</code>	The type of the database adapter to use when connecting to the database.
driverClass	<code>org.h2.Driver</code>	The fully qualified name of the driver's Java class to use when connecting to the database.
datasourceClass	<code>org.h2.jdbcx.JdbcDataSource</code>	The fully qualified name of the datasource's Java class to use when connecting to the database.
scheme	<code>jdbc:h2:</code>	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	<code>5000</code>	The commit interval of an index in milliseconds.
queryWarnThreshold	<code>400</code>	The threshold of the warn log when querying data.
queryInfoThreshold	<code>300</code>	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 when using the Elasticsearch based index module.
commitConcurrencyLevel	Number of cores / 4 by default, min 1	Number of concurrent requests when executing bulk commit operations against a terminology repository index.

```

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.



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.

Name	Default	Description
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.
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.

Name	Default	Description
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.
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
...

```