Snow Owl Server Documentation

B2i Healthcare

Table of Contents

Software requirements

Introduction

This document lists the required set of software components necessary to run Snow Owl Server on the x86_64 platform. Components and the corresponding files to download are listed in the sections below.

Components

Operating system

Snow Owl Server requires a **CentOS** or **Red Hat Enterprise Linux Server (RHEL)** release **6** installation. As of 2015-07-17, the latest available release for CentOS is version 6.6. We recommend starting with a minimal install using the ISO image named `CentOS-6.6-x86_64-minimal.iso` and adding packages later, as required.

Database server software

Terminology contents are persisted using the **5.6** series of **MySQL Community Server**, downloadable from MySQL's yum repository. Select "Red Hat Enterprise Linux / Oracle Linux 6 (Architecture independent), RPM package" with name `mysql-community-release-el6-5.noarch.rpm` for download.

JDK

Snow Owl Server uses version 7 of the **Java SE Development Kit**; note that runtime checks in the core server component prevent it from starting on JDK 8. An installable archive of Critical Patch Update **79** can be downloaded from the JDK7 download page. Select the "Linux x64" edition named `jdk-7u79-linux-x64.rpm`.

LDAP

Authentication and authorization of browsers, terminology editors, reviewers and administrators is performed through an **Apache DS** version **2.0** LDAP server. The package to install can be located by following the download link after `ApacheDS 2.0.0-M12` on the Download Old Versions page. The file to download is `apacheds-2.0.0-M12-x86_64.rpm`.

Browsing and managing ApacheDS instances can be done through the **Apache Directory Studio** application. We recommend installing the latest release from the corresponding Downloads Old Versions page on Apache's website. Download `Apache Directory Studio 2.0.0-M7` for a version compatible with the M12 release of the server.

Task tracking

When using Snow Owl's desktop client, terminology editing tasks can be tracked using **Bugzilla** version **3.6**. (Recent versions are not guaranteed to work correctly with client integration.) The

archived download of Bugzilla can be found at https://ftp.mozilla.org/pub/mozilla.org/webtools/archived/bugzilla-3.6.13.tar.gz.

Bugzilla requires Perl modules not included in CentOS repositories. See the usage instructions for the Extra Packages for Enterprise Linux (**EPEL**) project for adding the required yum repository.

Reasoners

Depending on the complexity of the ontology, you may need to use the FaCT++ reasoner, which relies on a native library component to increase performance. A customized version of FaCT++ exists for Snow Owl, which requires version **2.12** from the **GNU C Library (glibc)** to be present, which is already included in the OS distributions described above.

Note that the downloadable release of the same reasoner is based on a different (newer) glibc version, which makes it unsuitable for use within Snow Owl on CentOS/RHEL 6.

Installation guide

Introduction

This document goes through the installation steps of required software listed in “Software requirements”. After finishing the installation of all items, follow “Configuration guide” to set up each component to work with Snow Owl Server.

Components

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 160 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 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
```

Database server software

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.

After downloading the RPM package, install MySQL's yum repository with the following command:

```
# yum install mysql-community-release-el6-5.noarch.rpm
```

The repository for MySQL 5.6 should be enabled by default. Confirm by opening `/etc/yum.repos.d/mysql-community.repo`:

```
[mysql56-community]
name=MySQL 5.6 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/5/$basearch/
enabled=1 ①
...
```

① Value should be 0 for other releases, 1 for the 5.6 server

Install MySQL Community Server using yum:

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

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

```
# 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
```

Go through the secure installation, which removes example and public databases, and sets a new root password for the database:

```
# mysql_secure_installation
# reboot
```

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

Finally, check the log file `/var/log/mysqld.log` for any errors.

JDK

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

```
# yum install jdk-7u79-linux-x64.rpm
```

LDAP

Download the 64-bit Linux RPM package, and install it with yum:

```
# yum install apacheds-2.0.0-M12-x86_64.rpm
```

Start the service, check if it is running at startup:

```
# service apacheds-2.0.0_M12-default start
Starting ApacheDS - default...

# service apacheds-2.0.0_M12-default status
ApacheDS - default is running (1947).

# chkconfig apacheds-2.0.0_M12-default on
# chkconfig --list apacheds-2.0.0_M12-default
apacheds-2.0.0_M12-default   0:off   1:off   2:on    3:on    4:on    5:on    6:off

# reboot
```

Check the log files in folder `/var/lib/apacheds-2.0.0_M12/default/log` for any errors. `wrapper.log` holds messages from the service wrapper running at startup and shutdown, while `apacheds.log` captures log output from the directory server itself.

Task tracking

As in the case of MySQL, only the steps required are included below, a more complete installation guide for Bugzilla can be found at https://www.bugzilla.org/docs/3.6/en/html/installing-bugzilla.html.

Install the Apache2 web server:

```
# yum install httpd

# service httpd start
Starting httpd:                                            [  OK  ]

# service httpd status
httpd (pid  1638) is running...

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

# reboot
```

Add the following configuration section to `/etc/httpd/conf/httpd.conf`:

```
<Directory /var/www/html/bugzilla>
    AddHandler cgi-script .cgi
    Options +Indexes +ExecCGI
    DirectoryIndex index.cgi
    AllowOverride Limit
</Directory>
```

Reload the httpd service configuration to apply changes:

```
# service httpd reload
```

Extract the downloaded archive of Bugzilla and move contents into folder `/var/www/html`, adjust permissions and SELinux labels:

```
# tar xzvf bugzilla-3.6.13.tar.gz
# mv bugzilla-3.6.13 /var/www/html/bugzilla
# chown -Rv apache:apache /var/www/html/bugzilla
# restorecon -Rv /var/www/html/bugzilla
```

Check the availability of Perl modules required to get Bugzilla running:

```
# cd /var/www/html/bugzilla
# ./checksetup.pl --check-modules
```

Depending on the set of currently installed Perl modules, the check script will list a set of required modules, another set of optional modules and a database module to use for persisting Bugzilla issues. The preferred way of installing them is via yum, but the suggested `perl install-module.pl` commands can also be used for this. On a clean CentOS 6 system, the following set of packages need to be added for a MySQL client, the required modules and modules XML-RPC and LDAP:

```
# yum install perl-DBD-MySQL
# yum install perl-Digest-SHA perl-DateTime perl-TimeDate perl-Template-Toolkit perl-
Email-Send perl-Email-MIME
perl-Email-MIME-Encodings perl-Email-MIME-Modifier perl-URI
# yum install perl-SOAP-Lite perl-LDAP
```

Finally, run `checksetup.pl` again, so Bugzilla can create its configuration file, `/var/www/html/bugzilla/localconfig`:

```
# ./checksetup.pl
```

Snow Owl Server

Unpack the distribution archive into `/opt`, installing `unzip` first if not already present; change permissions on the created folder:

```
# yum install unzip
# unzip snowowl-community-{version}-mysql.zip -d /opt
# chown -Rv snowowl:snowowl /opt/snowowl-community_{version}
```

Configuration guide

Introduction

This document helps configuring each installed component for use with Snow Owl Server.

Components

Operating system

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

WWW (HTTP)

For accessing Bugzilla's SOAP API and web-based UI

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

11389/TCP

Used by the LDAP server

Database server software

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/5.6/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
```

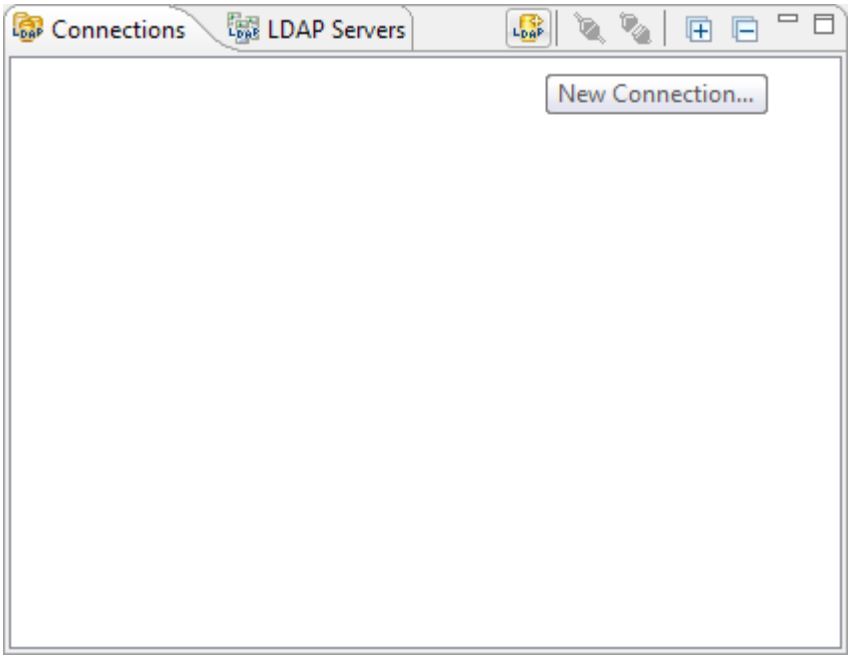
Restart the mysql service to make sure changes are picked up:

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

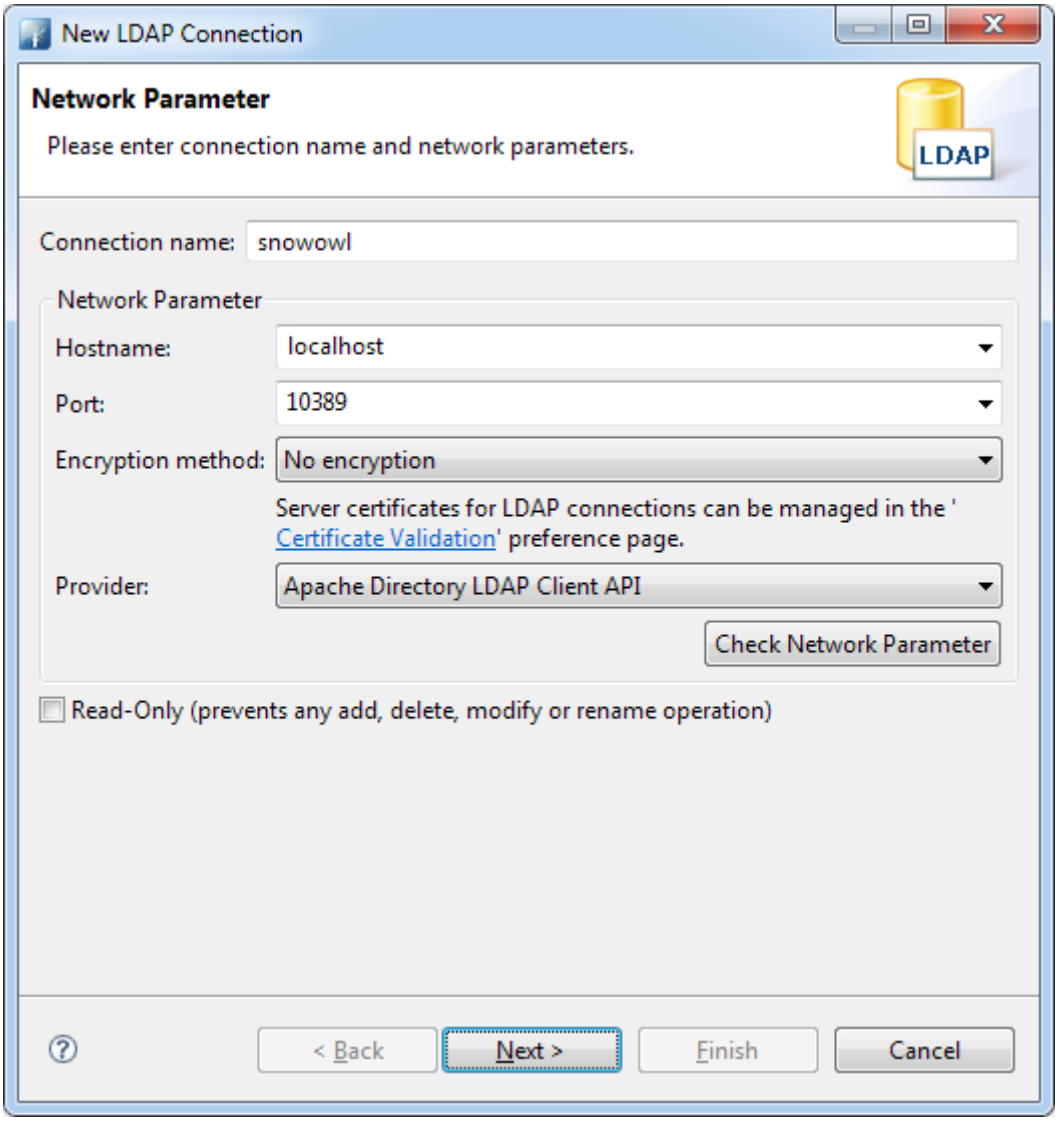
LDAP

Setting up a connection from Apache Directory Studio

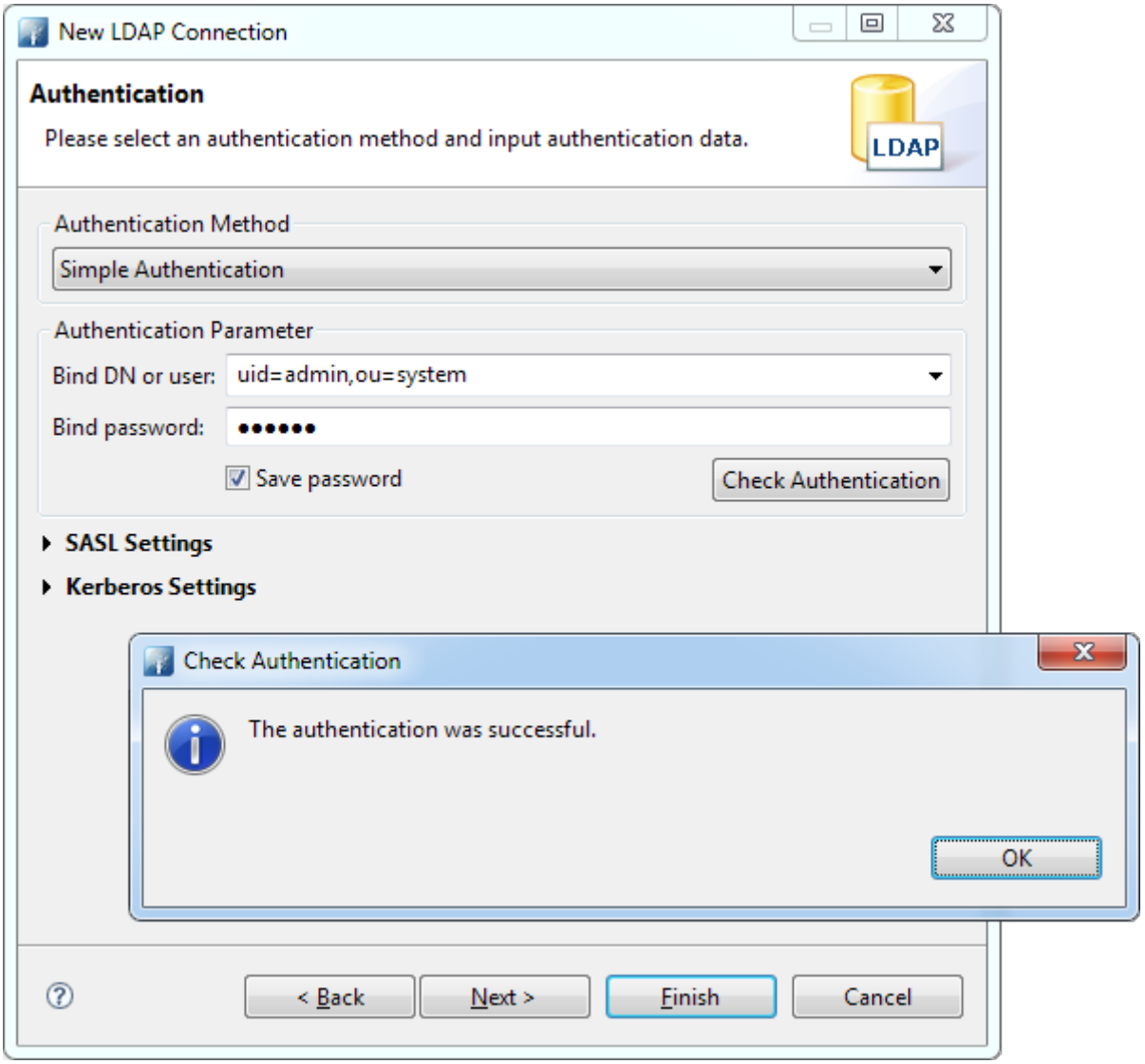
Open Apache Directory Studio, create a new connection using the first button on the “Connections” toolbar:



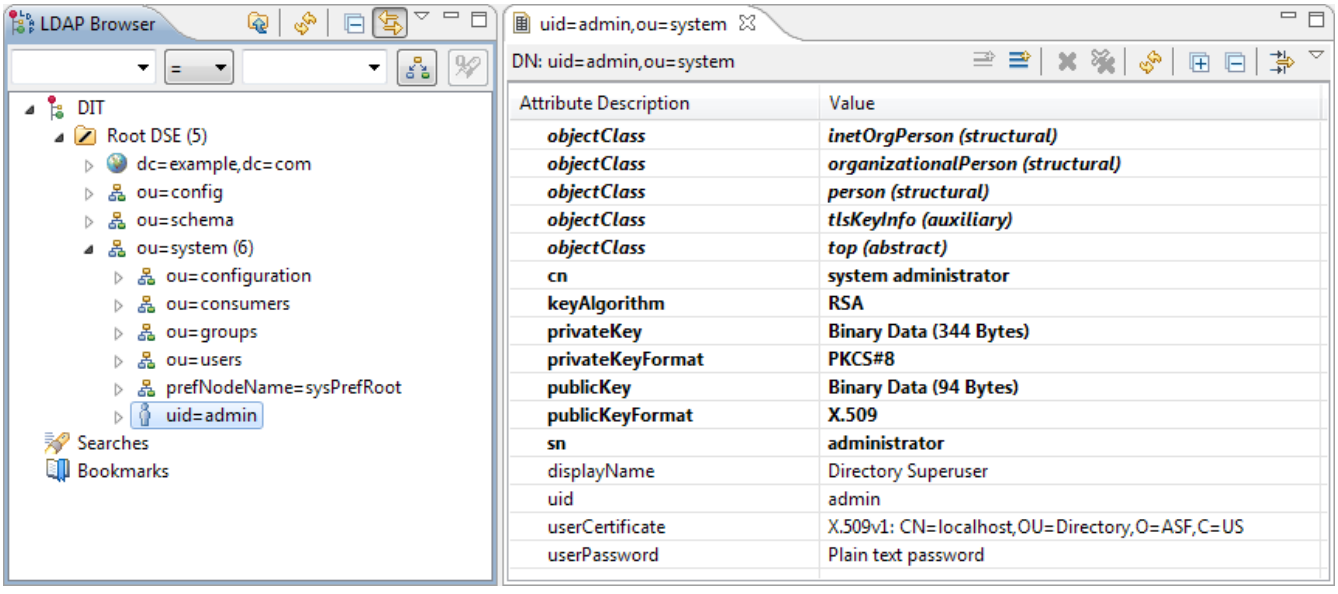
Enter connection name, hostname, and port, then hit `Next >` to go to the next page in the wizard:



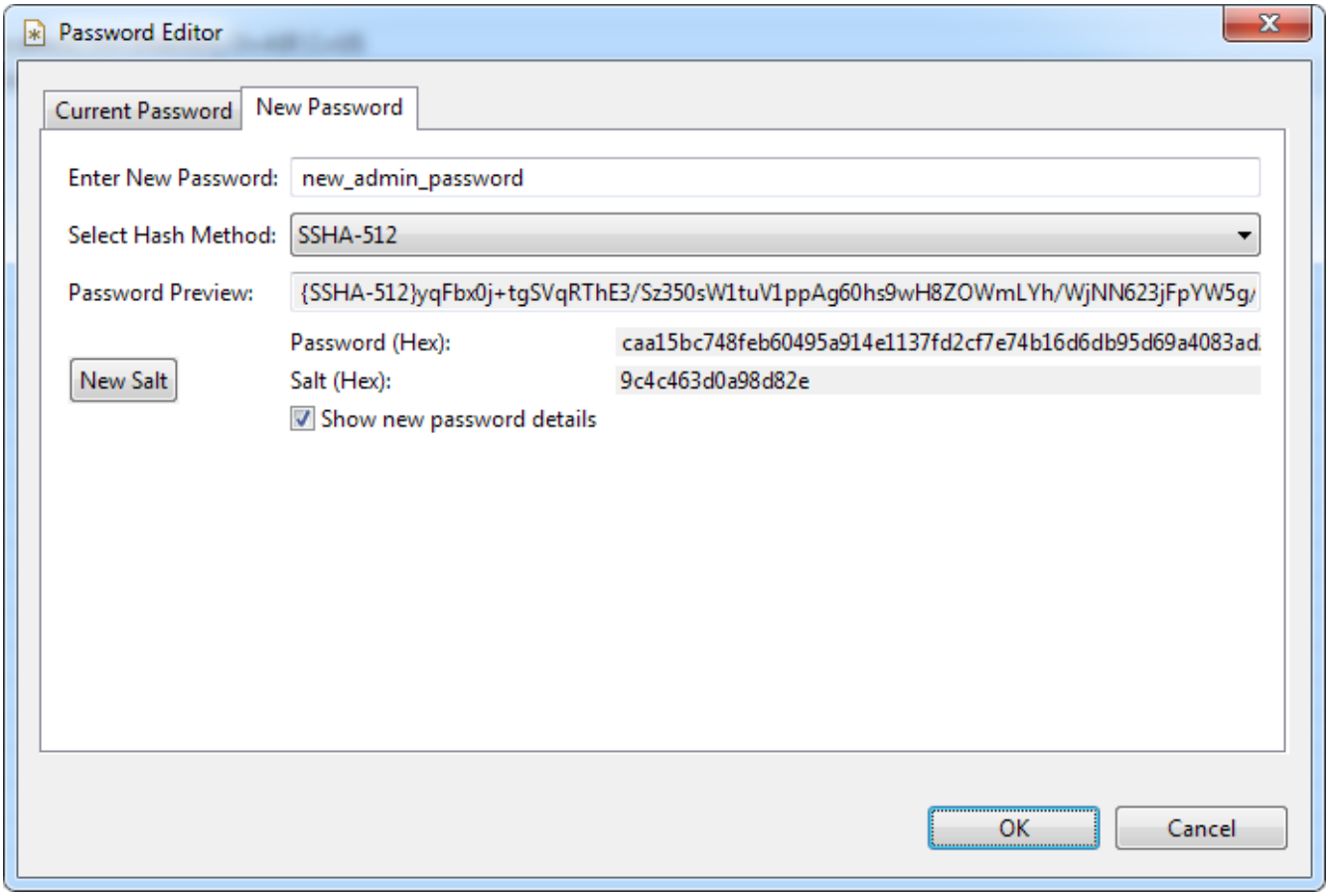
Set authentication method to `Simple Authentication`, enter Bind DN `uid=admin,ou=system` and password `secret`. Click `Check Authentication` to make sure the values are accepted, then dismiss the wizard with `Finish`:



After connecting, you are advised to change the default password for user `system`. In order to do so, expand node `ou=system` and select the person `admin`. The editor will now display related user information:



Double click on `userPassword` to open the password change dialog, select `SSHA-512` as the hash method. To see the new password as it is being entered, check `Show next password details`:

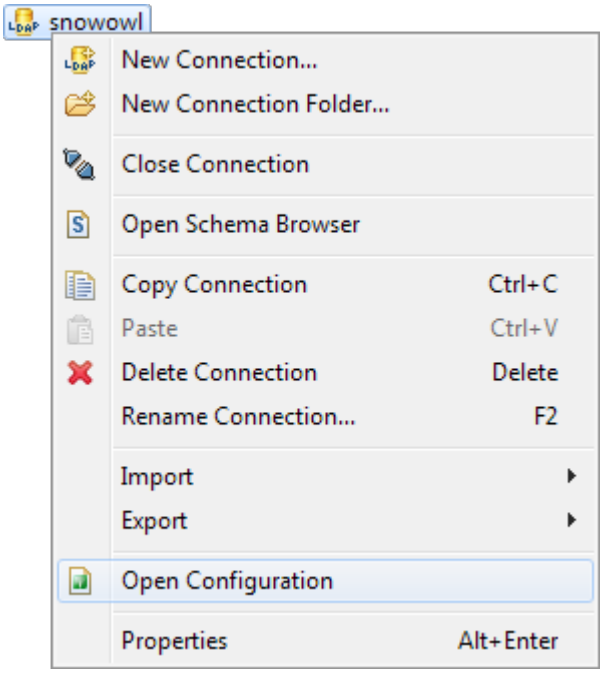


To see if the password was changed successfully:

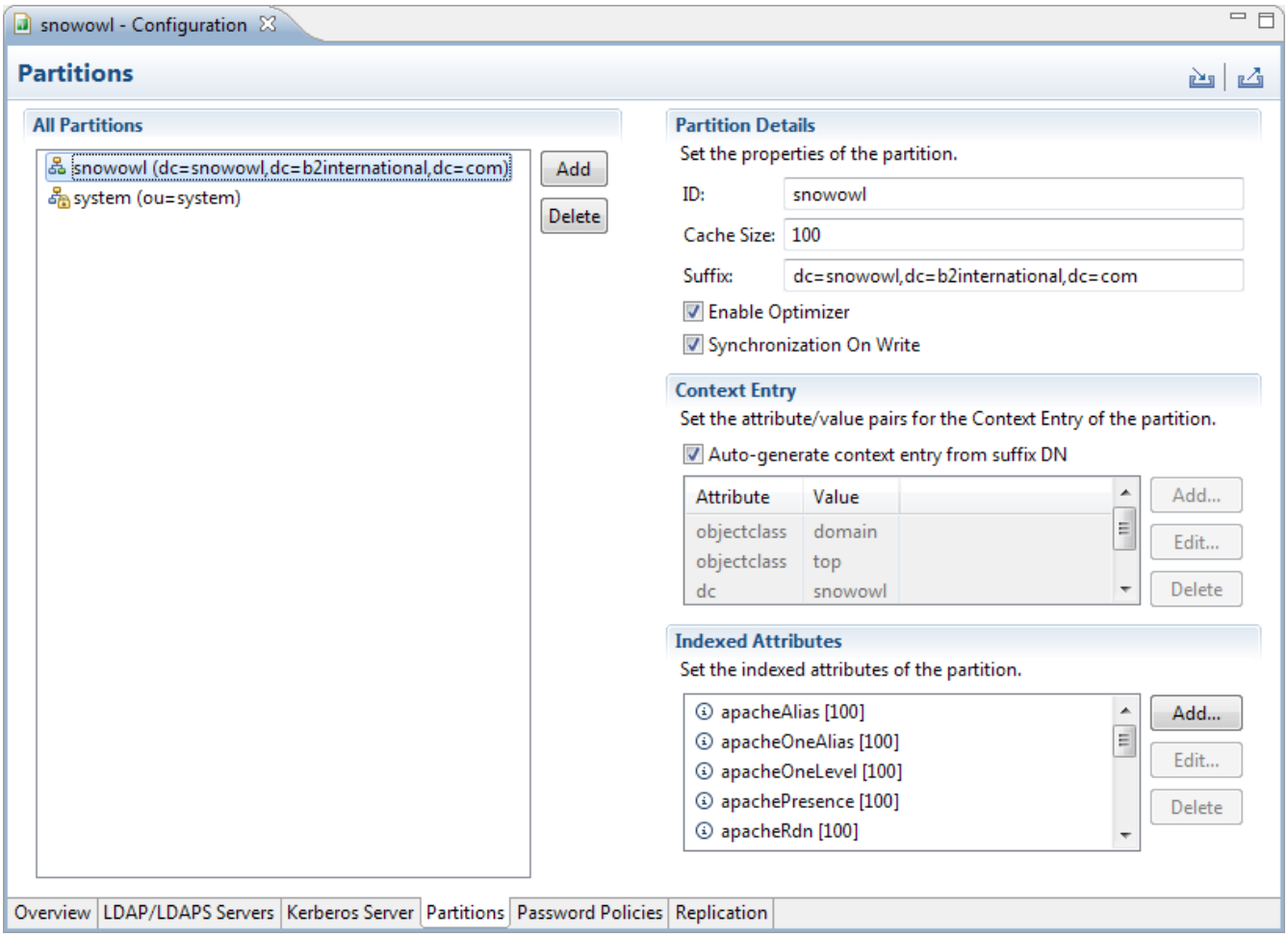
- Close the connection using the third button on the lower left toolbar named `Close Connection`,
- Right-click the item representing the connection to open its properties with the context menu item at the bottom,
- Change the previously entered default bind password to the updated value on the `Authentication` tab,
- Re-connect to the server using `Open Connection`.

Creating a partition for Snow Owl Server

Right-click on the item representing the connection, and select "Open Configuration":



An editor opens with the server configuration; select the `Partitions` tab. An `examples` partition is added by default, which should be removed. Add a new partition named `snowowl` and fill out the details (ID and suffix):



Save the editor and restart the server to apply changes in partitions.

```
# service apacheds-2.0.0_M12-default restart
Stopping ApacheDS - default...
Stopped ApacheDS - default.
Starting ApacheDS - default...
```

Using LDIF dumps

B2i provided LDAP packages include the following content:

schema.ldif
: LDAP schema to use for authorization (contains definitions for permissionId and role)

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 from the contents of folder `/var/lib/apacheds-2.0.0_M12/default`, 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`

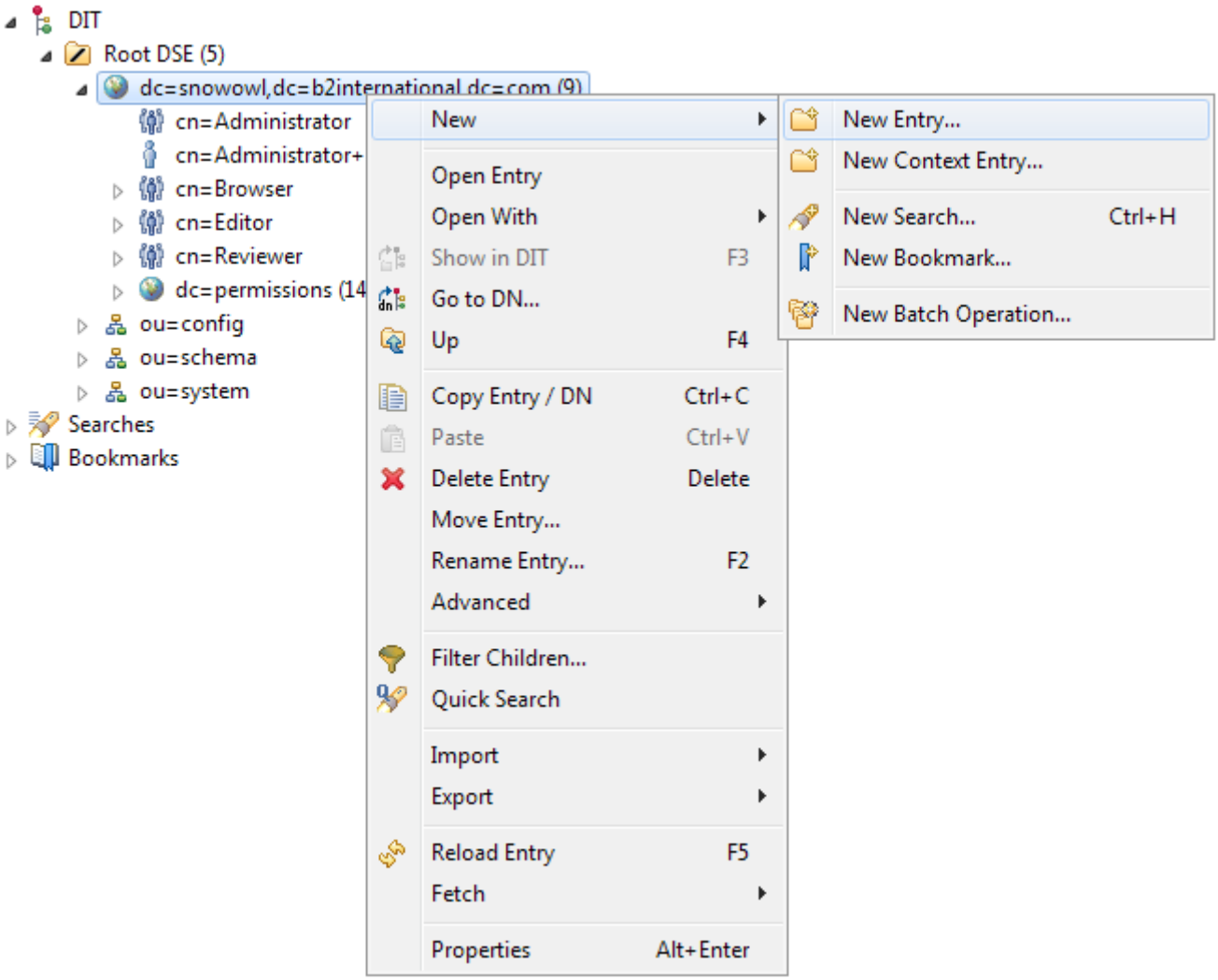
This is expected as most of the time the LDAP instance will already contain an existing definition of some entries and/or schema entities. If you notice other errors (either during script execution or when using the LDAP), roll back your instance to a previous state from a backup.

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:

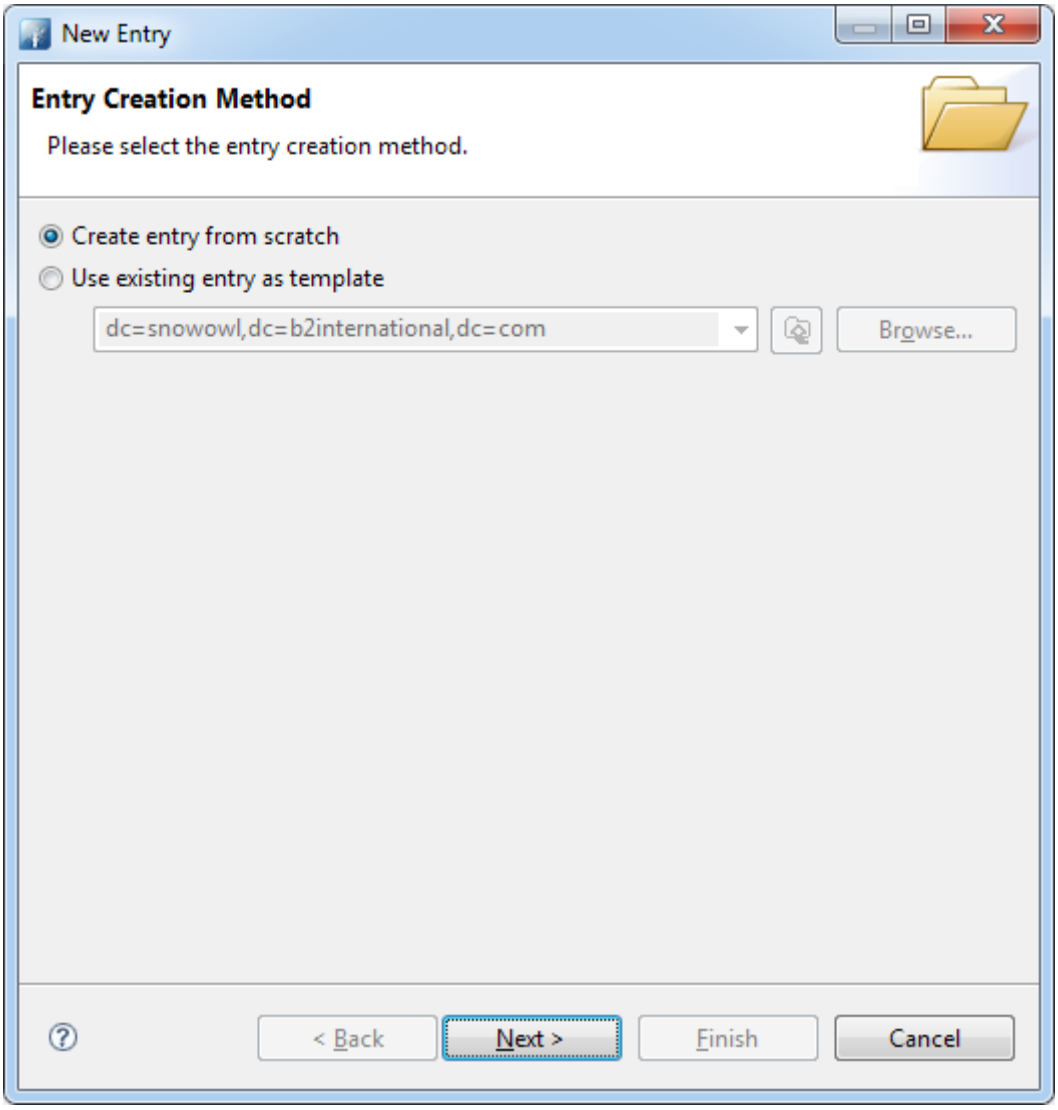
```
# ./update.sh ldap://<host>:<port>
```

Creating a new user from Directory Studio

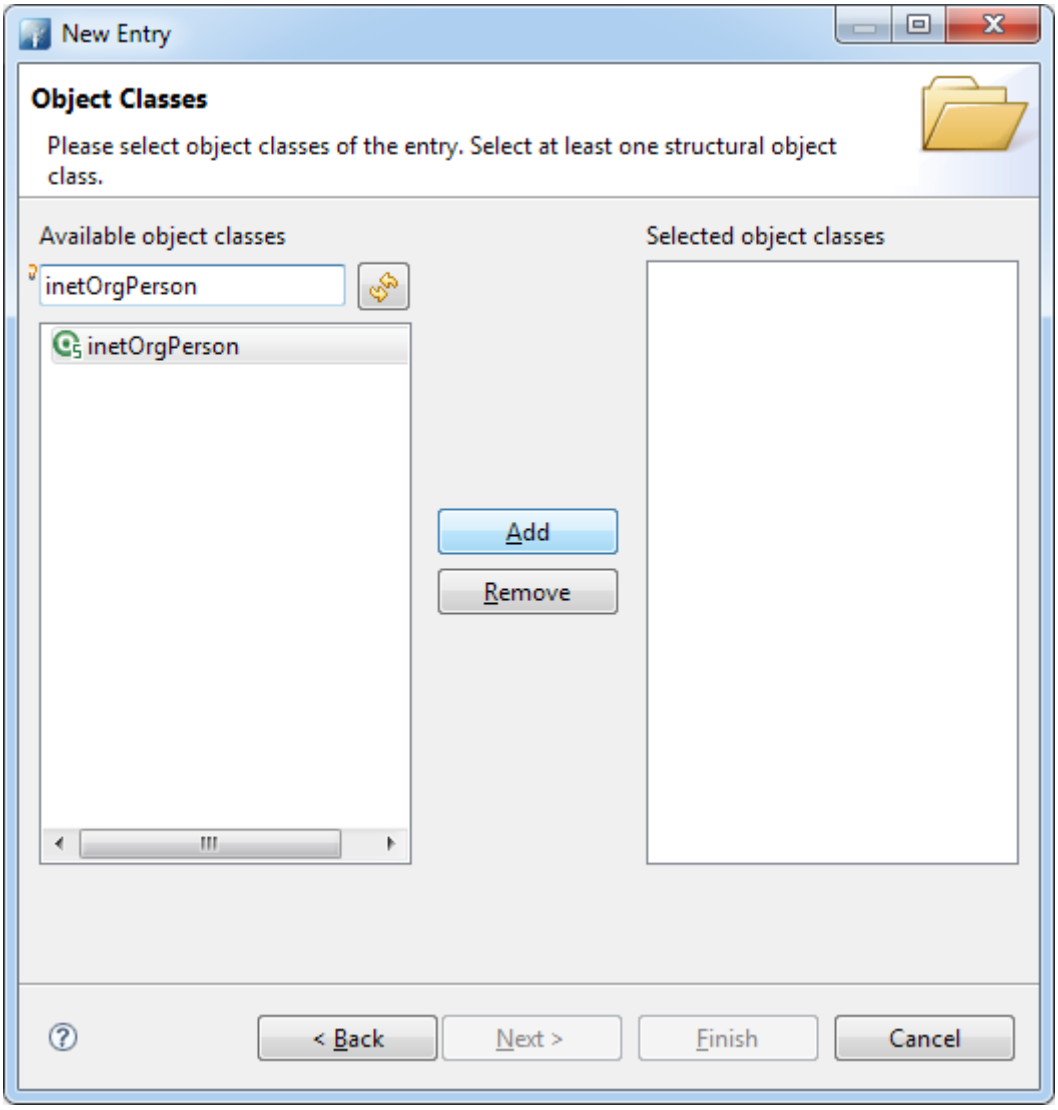
Go to LDAP Browser view, right click on the Domain component (DC) and add new entry via `New` > `New entry`:



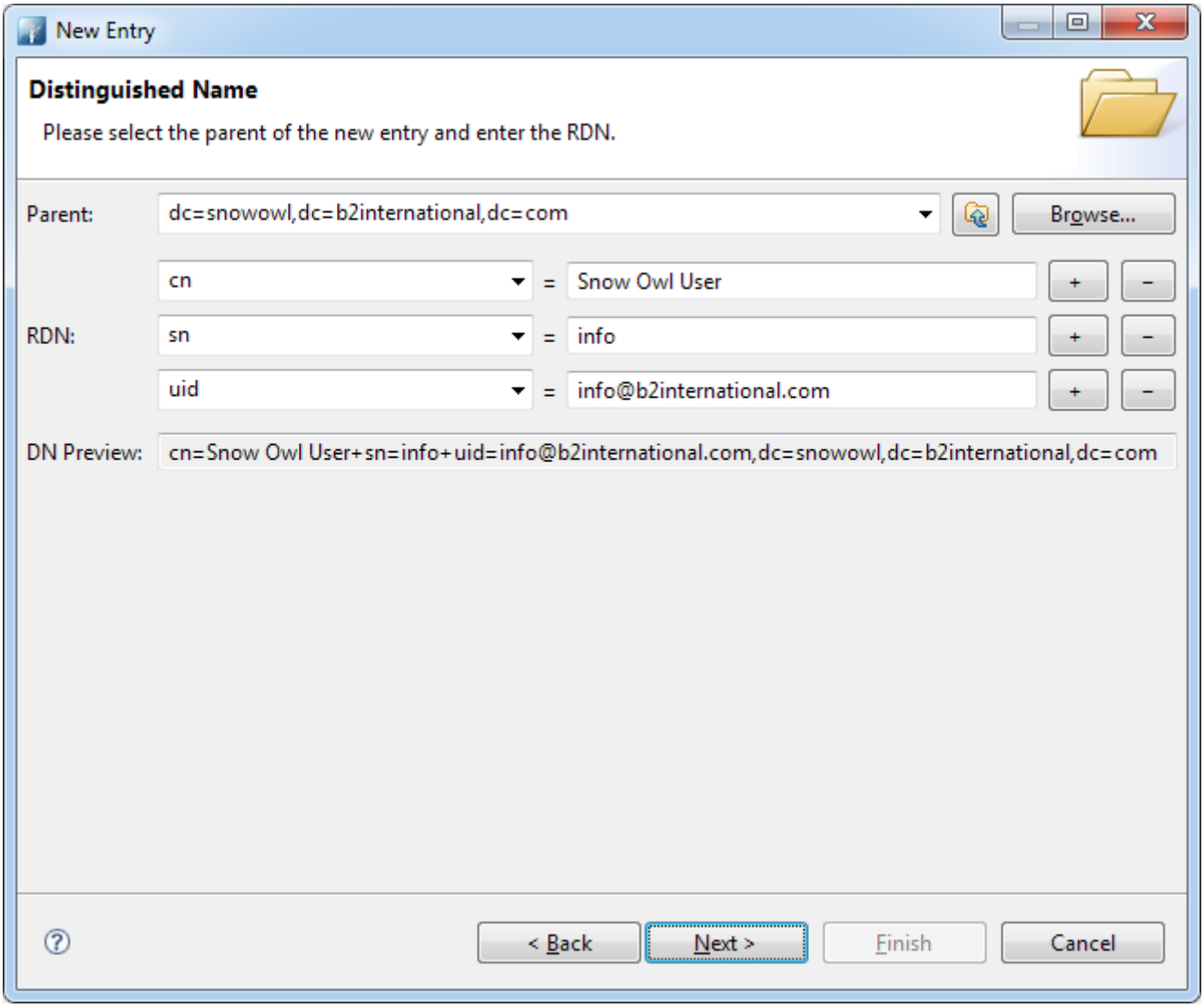
Create a new entry from scratch:



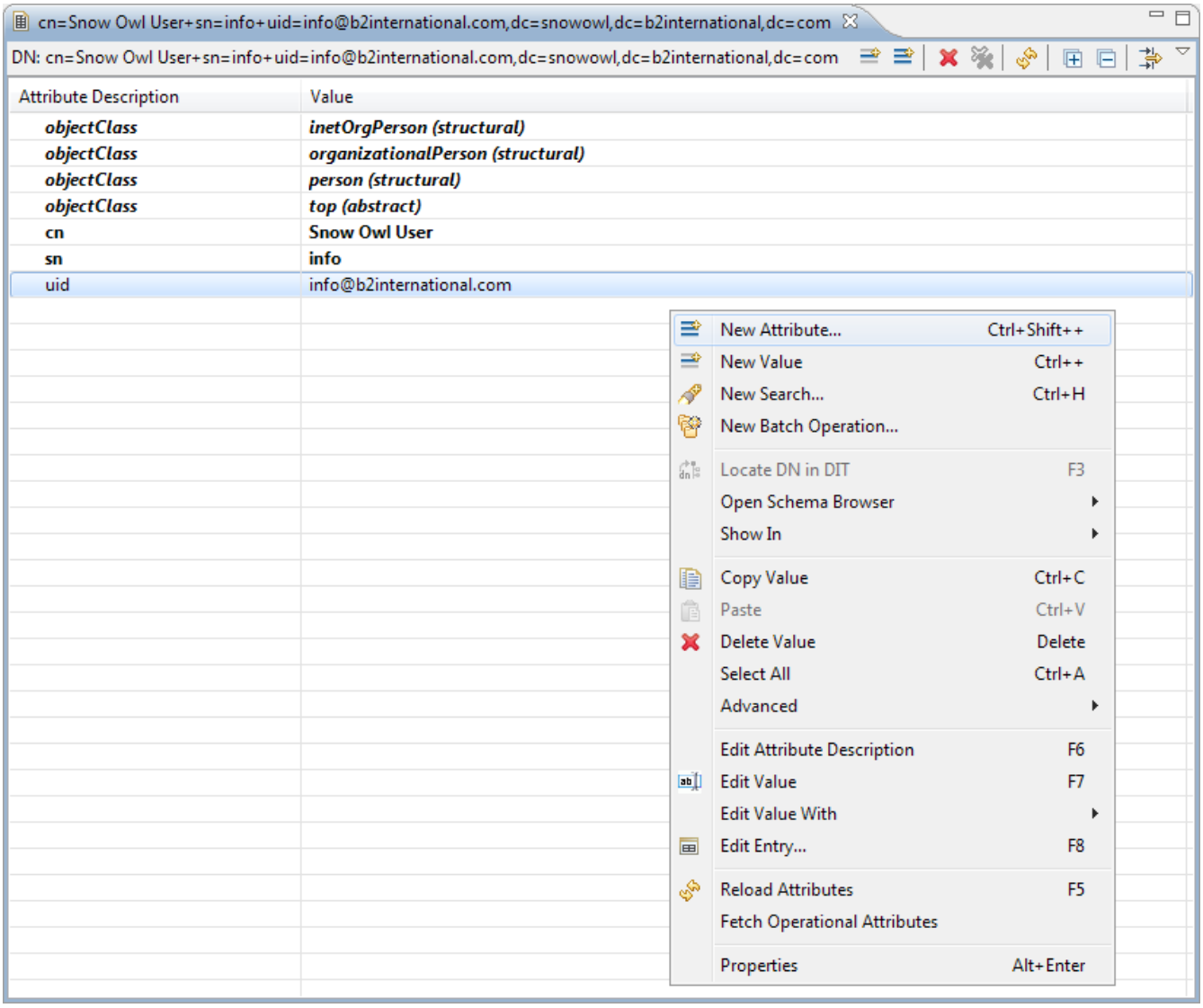
Select `inetOrgPreson` object from the wizard, add it as a selected object class:



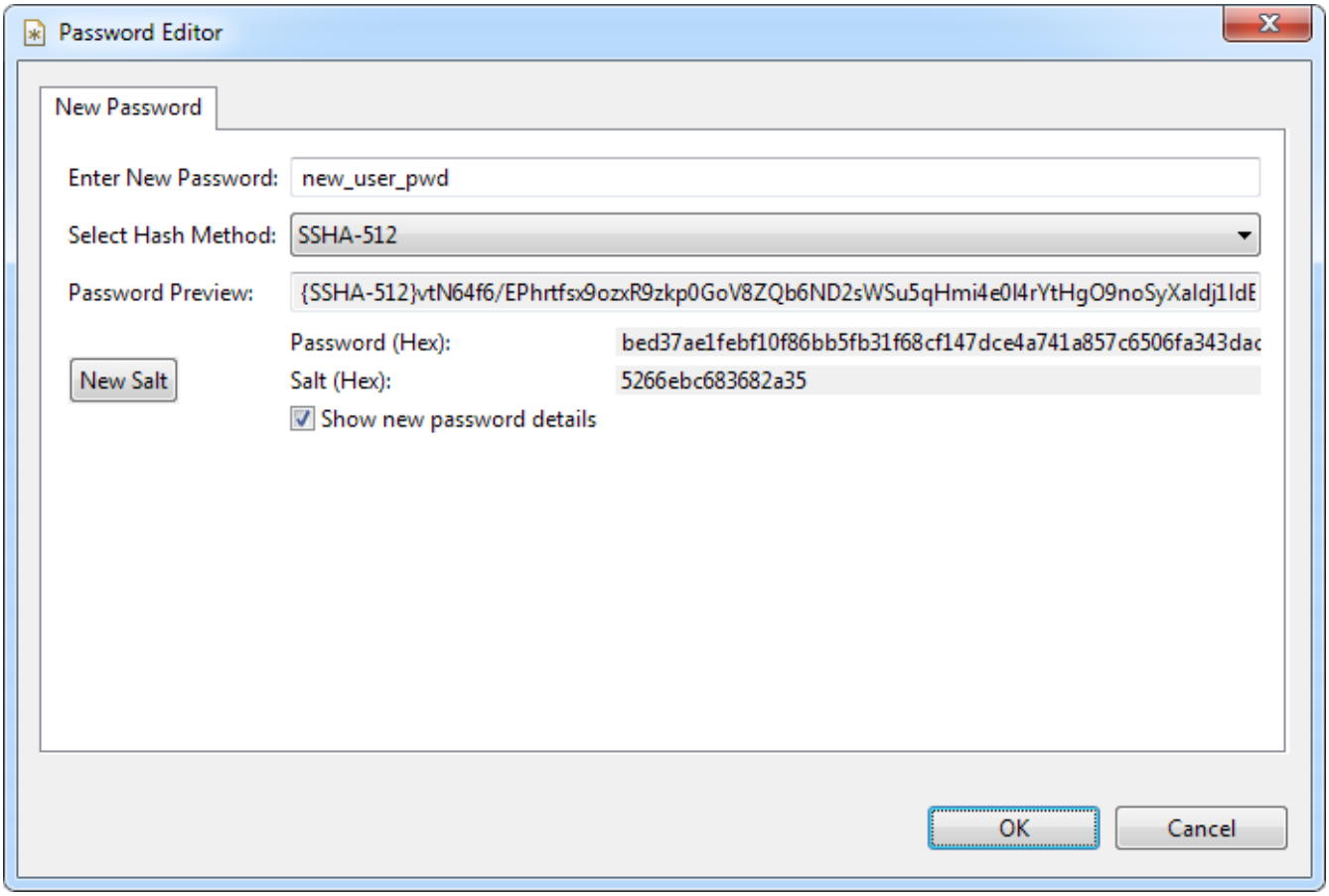
Configure the Relative Distinguished Name (RDN). Specify the common name (CN), surname (SN) and unique ID (uid):



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

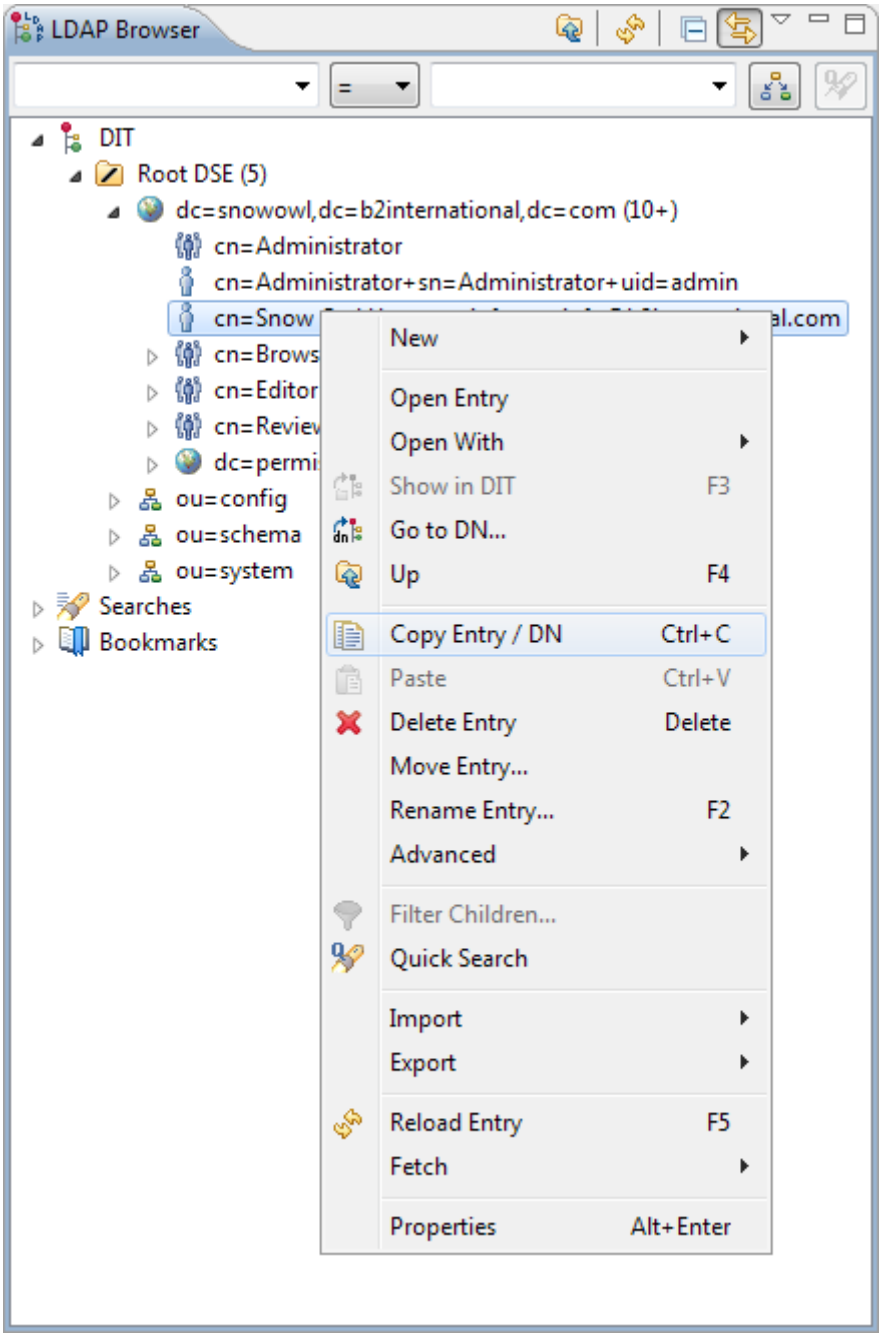


Select `userPassword` attribute, click `Finish`, then enter user password in the `Password Editor` dialog:

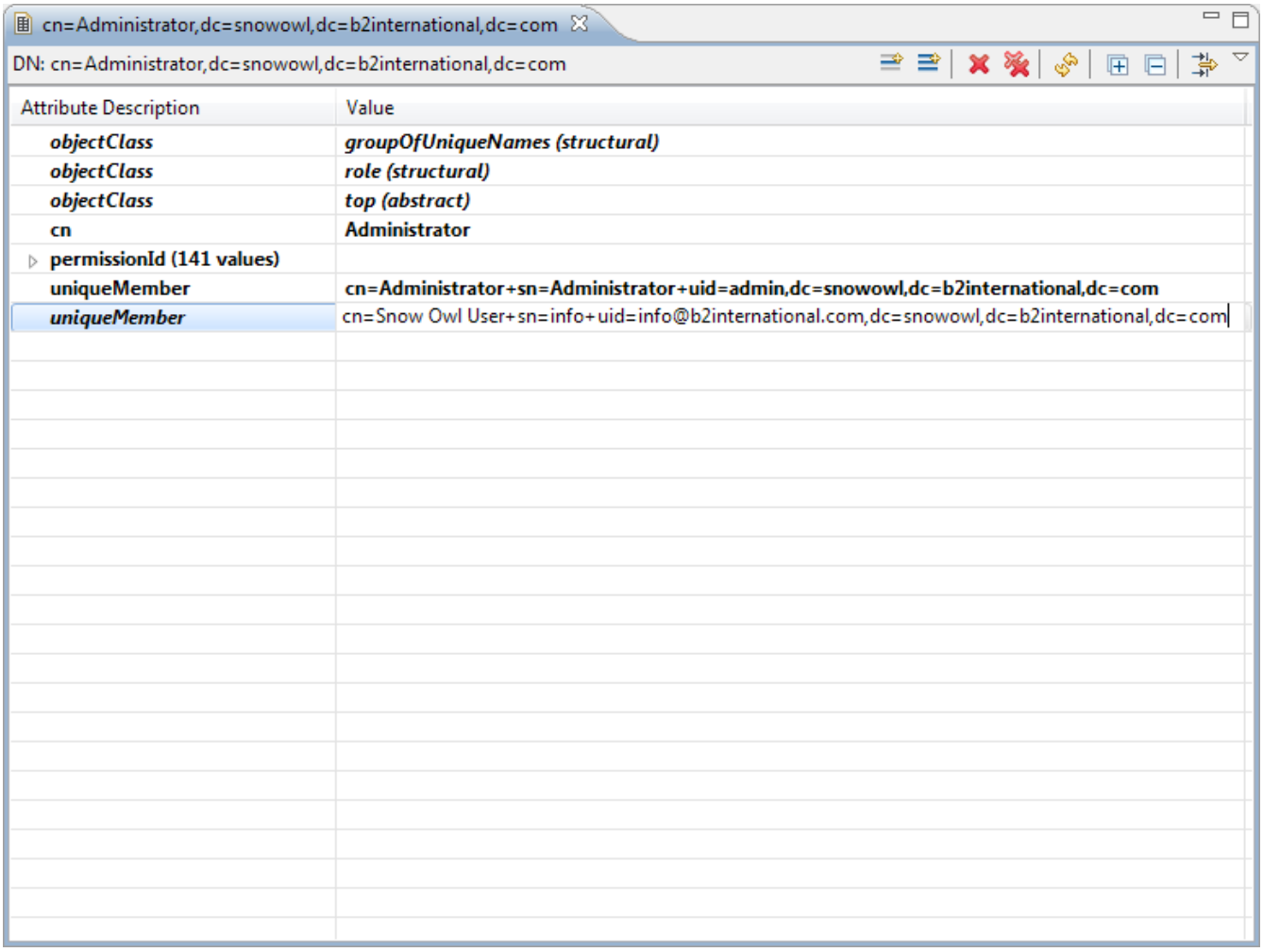


Finally, add a `uniqueMember` attribute to the Administrator group.

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:



The operation takes place after pressing Enter or removing the focus from the edited field.

Task Tracking

Finishing the installation

Create a MySQL user for Bugzilla using the command-line client:

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, INDEX, ALTER, CREATE, LOCK TABLES,
       CREATE TEMPORARY TABLES, DROP, REFERENCES ON bugs.*
       TO bugs@localhost IDENTIFIED BY 'bugzilla_pwd'; ①

mysql> FLUSH PRIVILEGES;
```

① Replace `bugzilla_pwd` with a generated password

Return to folder `/var/www/html/bugzilla` and edit `localconfig` to reflect the DB user name and password changes:

/var/www/html/bugzilla/localconfig

```
# Enter your database password here. It's normally advisable to specify
# a password for your bugzilla database user.
# If you use apostrophe (') or a backslash (\) in your password, you'll
# need to escape it by preceding it with a '\' character. (\') or (\)
# (Far simpler just not to use those characters.)
$db_pass = 'bugzilla_pwd';
```

Apply the following patch on `/var/www/html/bugzilla/Bugzilla/DB/Mysql.pm` to make Bugzilla work with MySQL 5.6:

Mysql.pm.patch

```
--- Mysql.pm.old        2015-07-23 22:07:27.797000043 +0200
+++ Mysql.pm    2015-07-23 22:10:49.373999897 +0200
@@ -309,8 +309,8 @@
     # works if InnoDB is off. (Particularly if we've already converted the
     # tables to InnoDB.)
     my ($innodb_on) = @{$self->selectcol_arrayref(
-        q{SHOW VARIABLES LIKE '%have_innodb%'}, {Columns=>[2]})};
-    if ($innodb_on ne 'YES') {
+        q{SHOW ENGINES}, {Columns=>[2]})};
+    if ($innodb_on ne 'YES' && $innodb_on ne 'DEFAULT') {
         print <<EOT;
 InnoDB is disabled in your MySQL installation.
 Bugzilla requires InnoDB to be enabled.
```

Finally, run `./checksetup.pl` again. Bugzilla should be reachable at http://localhost/bugzilla after configuration is completed. Details of the administrator user will be requested at the end of the process:

```
...
Adding a new user setting called 'per_bug_queries'
Adding a new user setting called 'zoom_textareas'
Adding a new user setting called 'csv_colsepchar'
Adding a new user setting called 'state_addselfcc'
Adding a new user setting called 'comment_sort_order'
Adding a new user setting called 'display_quips'

Looks like we don't have an administrator set up yet. Either this is
your first time using Bugzilla, or your administrator's privileges
might have accidentally been deleted.

Enter the e-mail address of the administrator: info@b2international.com ①
Enter the real name of the administrator: Administrator ②
Enter a password for the administrator account: ③
Please retype the password to verify:
info@b2international.com is now set up as an administrator.
Creating initial dummy product 'TestProduct'...

Now that you have installed Bugzilla, you should visit the 'Parameters'
page (linked in the footer of the Administrator account) to ensure it
is set up as you wish - this includes setting the 'urlbase' option to
the correct URL.
```

① Enter the e-mail address of the administrator user

② Add a display name for the Bugzilla administrator

③ Enter the password of the administrator user

Once Bugzilla has created its table structure, you can increase the maximum table size by executing the following commands:

```
mysql> USE bugzilla
mysql> ALTER TABLE attachments
       AVG_ROW_LENGTH=1000000, MAX_ROWS=20000;
```

Administration of Bugzilla

See http://www.bugzilla.org/docs/3.6/en/html/administration.html for a comprehensive list of administrative tasks and options.

After logging in with an account that has administrative privileges, click the `Administration` link on the top. The general administrative page will appear as shown below:

Core parameters can be set by selecting `Parameters` on the top left. The following fields are recommended to be adjusted:

Required Settings

urlbase

Set to the the common leading part of all URLs which are related to Bugzilla (ex.: http://server.domain/bugzilla/)

cookiepath

The common path segment of the URL under which Bugzilla cookies are allowed to be read; as noted in the description above the field, its value should begin with '/' (ex.: /bugzilla/)

General

maintainer

The email address entered here is shown on various pages in Bugzilla where contacting the administrator is suggested

User Authentication

requirelogin

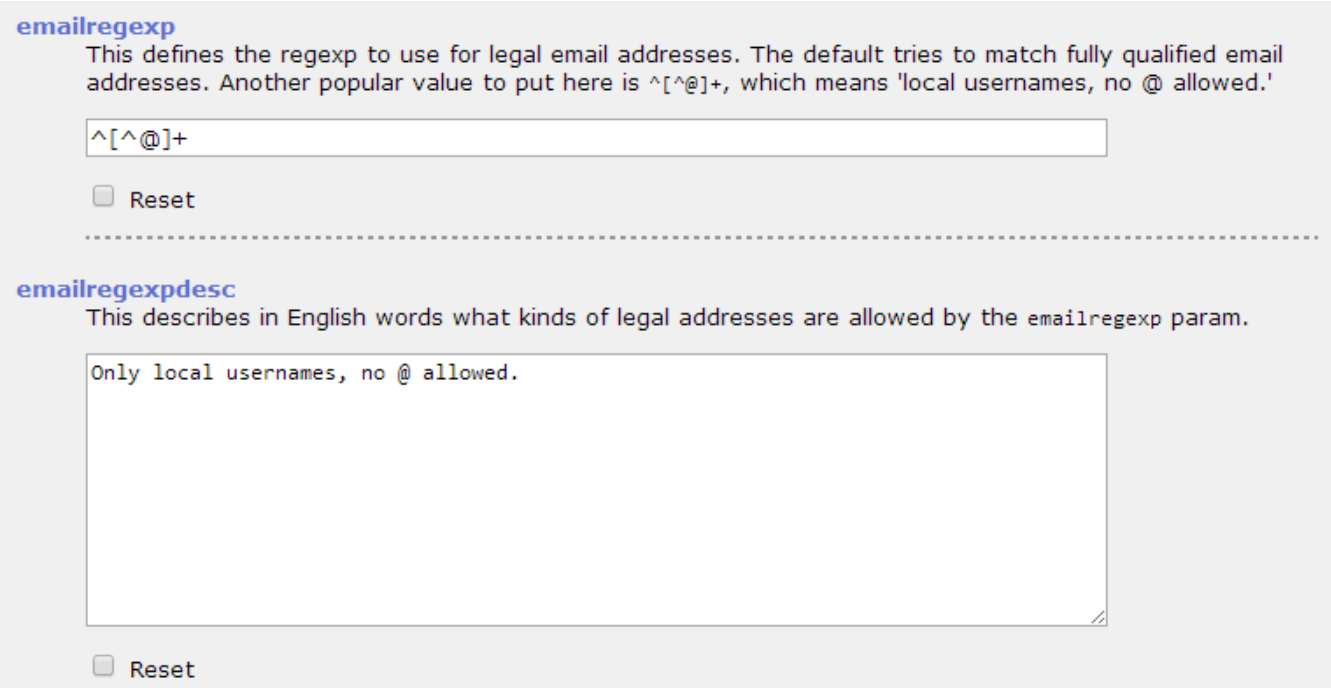
Set to On if you want to limit access to registered users only (disabling anonymous browsing of bugs)

emailregexp and emailregexpdesc

Depending on requirements, the administrator may limit login names to values that are not actual email addresses. In this case, set the fields as suggested in the description above, ie. `^[^@]+` and `Local usernames, no @ allowed.`

createemailregexp

to disable user-initiated registration (requiring the administrator to create each user account by hand), clear the field's contents



Attachments

maxattachmentsize

The maximum size in kilobytes for attachments. Change it to `10240` (10 MB)

Dependency Graphs

webdotbase

To disable relying on an external service for rendering dependency graphs of issues (as populated by default), clear the field's contents

Email

mail_delivery_method

If an SMTP server is available, configure its address and authentication properties below; otherwise, set this value to `None` to disable sending mail altogether

smtpserver

Clear the field's contents if no SMTP server is used

whinedays

Set to 0 if mail delivery is not enabled and/or there's no need to send users regular notifications about their assigned bugs which remained in NEW state

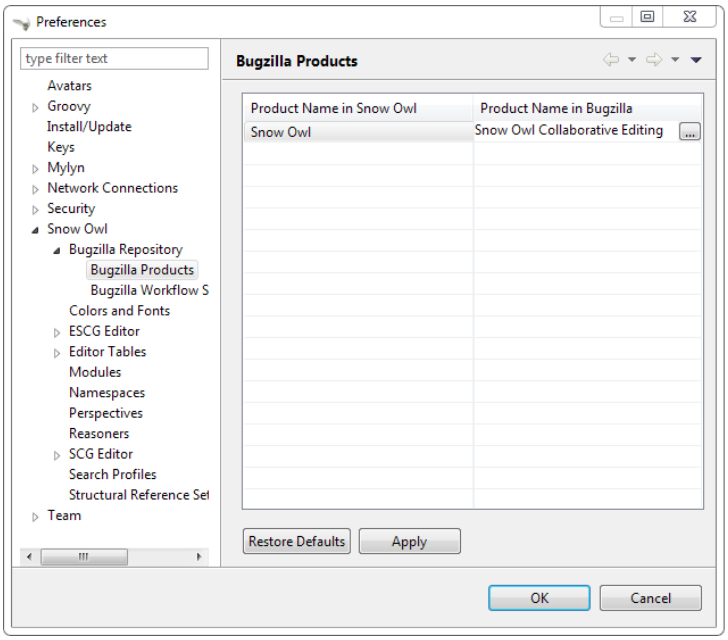
use-mailer-queue

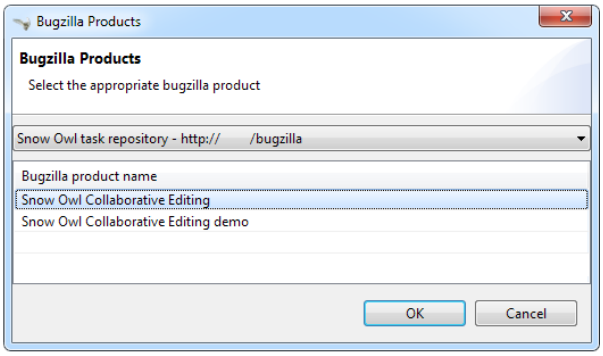
When set to `On`, e-mails are sent asynchronously; to use this feature, the jobqueue.pl daemon needs to be started. For more information on this topic, please see http://www.bugzilla.org/docs/3.6/en/html/api/jobqueue.html.

Product setup

Bugzilla tracks the authoring aspects of Snow Owl clients in multiple products. Per-product configuration is shown in the following parts of the guide.

Opening the preference page `Snow Owl > Bugzilla Products` displays the supported products in the client and their corresponding product names in Bugzilla. If you have different product names added in the issue tracker, you have to adjust the product name as shown in the image. Make sure to press Enter or click in the table to apply the change in the field before hitting `Apply` or `OK` to apply the changes. Products which are not handled by contributed task editors are displayed with an empty context view only:

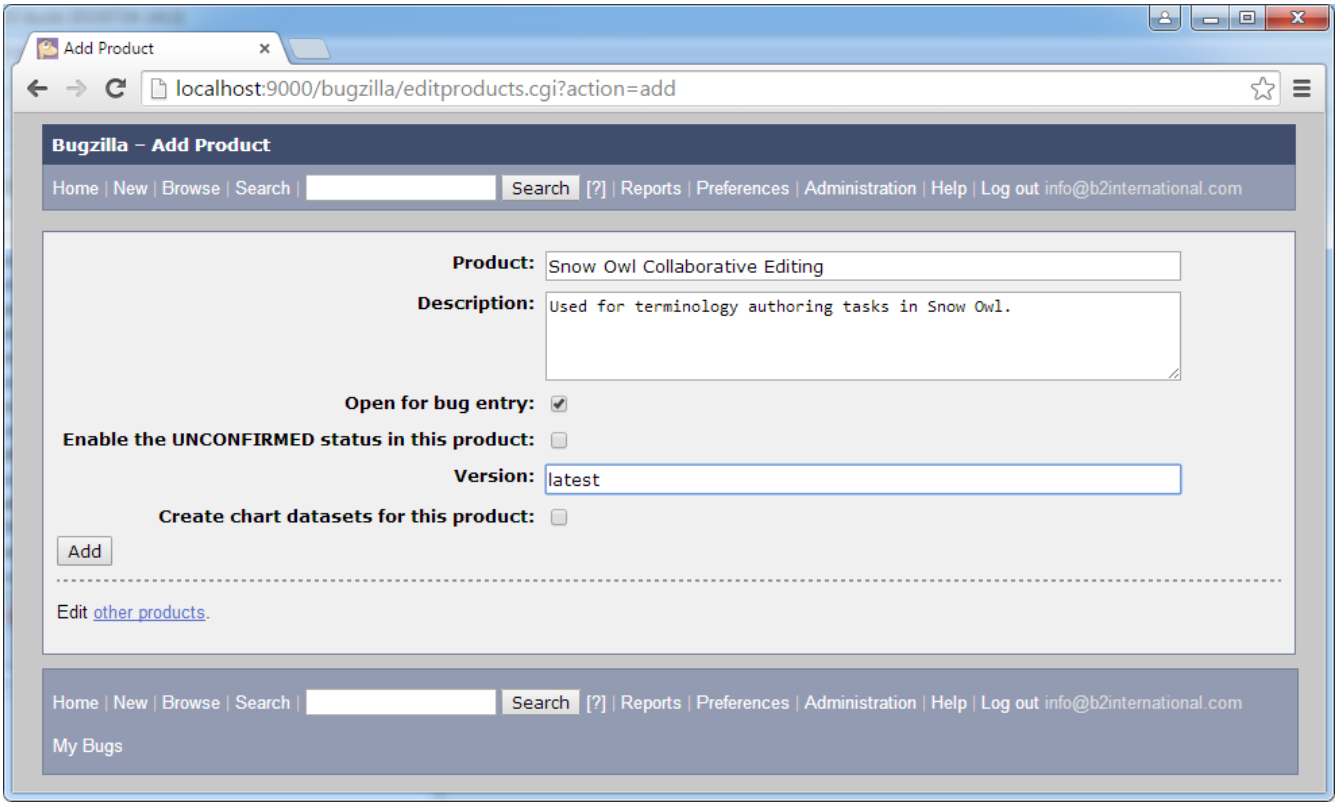


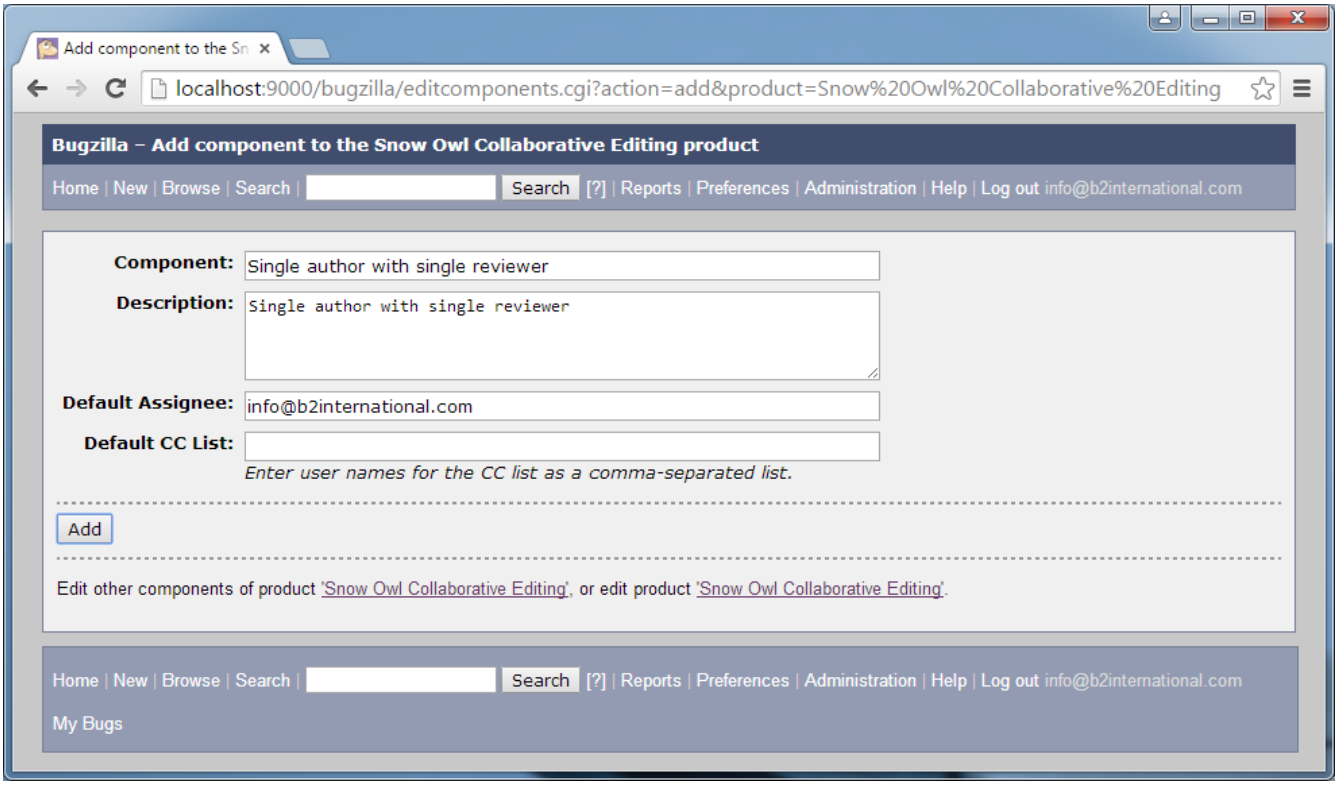


To match the default value set in client preferences, create a product called `Snow Owl Collaborative Editing` by clicking `Products` on Bugzilla's administration page. Add a description, optionally set a version to discern individual releases, and keep `Open for bug entry` checked to allow users to file

issues under this product. After creating the product, a warning will be issued by Bugzilla to create a component as well. Add the following components with the `Component`, `Component description` and `Default assignee` fields populated:

Component name	Description
Single author with single reviewer	Single author with single reviewer
Dual authors with single reviewer – Dual authoring	Dual authors with single reviewer – Dual authoring
Dual authors with single reviewer – Dual blind authoring	Dual authors with single reviewer – Dual blind authoring
Dual authors with dual reviewers – Dual authoring	Dual authors with dual reviewers – Dual authoring
Dual authors with dual reviewers – Dual blind authoring	Dual authors with dual reviewers – Dual blind authoring





Add custom fields through the web interface (`Administration > Custom fields`):

Field name	Description	Sortkey	Type	Editable on Bug Creation	In Bugmail on Bug Creation
cf_artifacttype	Task artifact type	400	Free Text	true	false
cf_author_one	Author one	410	Free Text	true	true
cf_author_two	Author two	420	Free Text	true	true
cf_reviewer_one	Reviewer one	430	Free Text	true	true
cf_reviewer_two	Reviewer two	440	Free Text	true	true
cf_adjudicator	Adjudicator	450	Free Text	true	true
cf_artifact_properties_source	Properties source	991	Free Text	true	false
cf_mappingset_id	Mapping set ID	992	Free Text	true	false
cf_valueset_id	Value domain ID	993	Free Text	true	false
cf_is_promoted	Promoted	995	Free Text	true	false
cf_parent_refset_map_target_component_type	Reference set map target component type	996	Free Text	true	false

Field name	Description	Sortkey	Type	Editable on Bug Creation	In Bugmail on Bug Creation
cf_parent_refset_referenced_component_type	Reference set referenced component type	997	Free Text	true	false
cf_parent_refset_identifierconcept_id	Parent reference set identifier concept id	998	Free Text	true	false
cf_refset_identifierconcept_id	Reference set identifier concept	999	Free Text	true	false

Authentication against LDAP

Bugzilla is capable to authenticate against the external LDAP server which Snow Owl Server will use. Setting it up requires the following steps to be taken:

- Go to `Administration > Parameters > LDAP` and populate the following fields:

  LDAPserver
  : hostname:port pair for contacting the server, eg. `localhost:10389`

  LDAPBaseDN
  : set to `dc=snowowl,dc=b2international,dc=com`

  LDAPuidattribute
  : set to `uid`

  LDAPmailattribute
  : set to `uid`

- Click `Save Changes` to apply changes
- Go to `Administration > Parameters > User Authentication`, scroll down to `user_verify_class` and make `LDAP` the top-most item
- Click `Save Changes` to finish

To test, click the `Log Out` link at the top and try to log in with your bugzilla username and LDAP password. If it was sucessful, you should see bugzilla's landing page. If you see an error message about not able to connect to the LDAP server, then run the following command as root:

```
# setsebool -P httpd_can_network_connect on
```

This will allow Apache to make network connections.

If LDAP is still not working and you are being locked out from bugzilla, you can change back bugzilla to use its internal database for authentication, instead of LDAP. To do so, edit `/var/www/html/bugzilla/data/params`, deleting LDAP from the `user_verify_class` entry:

/var/www/html/bugzilla/data/params

```
...
'user_verify_class' => 'DB',
...
```

If users are already entered in the LDAP server, it is important to synchronize Bugzilla's user database to contents of LDAP so tasks can be assigned to all users. Run the following script to perform synchronization:

```
# cd /var/www/html/bugzilla
# ./contrib/syncLDAP.pl
```

For general questions and documentation, please refer to chapter 3.1.10. LDAP Authentication in the documentation: http://www.bugzilla.org/docs/3.6/en/html/parameters.html.

Snow Owl Server

DB connection

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

```
Unresolved directive in configuration_guide.adoc -
include::scripts/snowowl_create_db.sh[]
```

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

snowowl_load_db.sh

```
Unresolved directive in configuration_guide.adoc -
include::scripts/snowowl_load_db.sh[]
```

Update `snowowl_config.yml` to use LDAP authentication and set the MySQL password for `snowowl`, created earlier:

```
repository:
  ...

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

① Update MySQL username and password, if neccessary

File Authentication

To use file based authentication, configure the `auth` property to `PROP_FILE` and the `fileAuth` configuration object to the desired username/password in the `snowowl_config.yml` file.

Building Snow Owl produces a ready to use, deployable package with the default passwords (`fileAuth` and `database` properties) configured. Usage of the default password configuration is not recommended in production environments.

LDAP Authentication

To use the configured LDAP instance, adjust `snowowl_jaas_configuration.properties` to include the host name of the LDAP server and the LDAP administrator password:

/opt/snowowl-community_{version}/configuration/snowowl_jaas_configuration.properties

```
LDAP {
    org.eclipse.equinox.security.auth.module.ExtensionLoginModule required
        extensionId="com.b2international.snowowl.authentication.ldap.ldapLoginModule"
        ...
        userProvider="ldap://<ldap_host>:10389/" ①
        usePool=false
        snowOwlBase="dc=snowowl,dc=b2international,dc=com"
        bindDnUser="uid=admin,ou=system"
        bindDnPassword="secret" ②
        ...
};
```

① Replace `<ldap_host>` with the host name of the LDAP server to connect to

② Replace `secret` with the LDAP administrator's password

Memory settings

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

```
JAVA_OPTS="$JAVA_OPTS \
    -Xms8g \
    -Xmx10g \
    -XX:MaxPermSize=512m \
```

`Xms` sets the minimum heap size, `Xmx` sets the maximum heap size, and `XX:MaxPermsize` sets the PermGen space used by the JVM.

OSGi console

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

/opt/snowowl-community_{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-3.6.1.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>
```

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-community_{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-community_{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-3.6.1.RELEASE/docs/virgo-user-guide/html/ch09.html and http://www.eclipse.org/virgo/documentation/virgo-documentation-3.6.0.M04/docs/virgo-user-guide/html/ch13s06.html#configuring-authentication.

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-community_{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.

Authentication

Name	Default	Description
type	`LDAP`	`PROP_FILE`, `LDAP` - choose which type of authentication method you want. NOTE: `PROP_FILE` is supported on standalone environments only.

```
authentication:
  type: PROP_FILE
```

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).
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	15000	The hard commit interval of an index in milliseconds.
translogSyncInterval	5000	The sync interval of the transaction log in milliseconds.
queryWarnThreshold	400	The threshold of the warn log when querying data.
queryInfoThreshold	300	The threshold of the info log when querying data.
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.

```
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
compressed	`false`	`true`, `false`, `ON`, `OFF` - enable or disable GZIP compression of the communication.

```
rpc:
  logging: true
  compressed: false
```

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.

Name	Default	Description
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.
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 an be configured to be either using the built-in or IHTSDO'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.
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.

Name	Default	Description
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
  ...
```

Logging

Log files are stored under `./opt/snowowl-community_{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. In case the file reaches a pre-defined maximum size, the system will create additional files named `log_1.log`, `log_2.log`, etc. 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-3.6.1.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.FixedWindowRollingPolicy">
        <FileNamePattern>serviceability/logs/log_%i.log</FileNamePattern>
        <MinIndex>1</MinIndex>
        <MaxIndex>4</MaxIndex>
    </rollingPolicy>
    <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
        <MaxFileSize>10MB</MaxFileSize>
    </triggeringPolicy>
    <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.

Virgo documentation

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

Administration guide

Introduction

This document 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-community_{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
...
[2015-07-24 02:50:12.753] INFO  start-signalling-2 WE0001I Started web bundle [...]
[2015-07-24 02:50:12.756] INFO  start-signalling-2 DE0005I Started bundle [...]
[2015-07-24 02:50:12.756] INFO  start-signalling-2 Thread context class loader [...]
[2015-07-24 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 connect with a telnet client to the server and check that everything is working properly:

```
# yum install telnet ①

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

osgi> snowowl checkservices
Checking core services...
[...]
Core services are registered properly and available for use.

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

① Install the client first (if not already present)

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

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

Importing content from a release archive

To import a `SnomedCT_Release_XXX_xxxxxxxx.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 decriptor 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.

List of single administrator operations

Name	Console command	Snow Owl UI	Admin console
ATC import from ClaML			
ICD-10 import from ClaML			
ICD-10-AM import from .zip archive			
Local Code System import from Excel spreadsheet	`localcodesystem importXL`		
LOINC import from .zip archive	`loinc import`		
Mapping set import from Excel spreadsheet	`mappingset import`		
SNOMED CT release import from RF2 files	`sctimport rf2_release`	(zip only)	
SNOMED CT reference set import from RF2 file	`sctimport rf2_refset`		
SNOMED CT reference set import from delimiter-separated file (includes RF1 subset files)	`sctimport dsv_refset`		
Value domain import from Excel spreadsheet			

Name	Console command	Snow Owl UI	Admin console
Value domain import from UMLS SVS XML file	`valueset import`		
Import MRCM rules from XMI file	`mrcm import`		
Export MRCM rules to XMI file	`mrcm export`		

List of operations that can be executed by regular users

- ATC export to ClaML
- Local Code System export to Excel spreadsheet
- Mapping set export to Excel spreadsheet
- SNOMED CT core components export to OWL 2
- SNOMED CT reference set export to RF1 and RF2
- SNOMED CT reference set export to Delimiter-Separated Values text file
- Value domain export to Excel spreadsheet
- Value domain export to UMLS SVS XML file

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
- Local code system import from Excel

- LOINC import from release archive
- Mapping set import
- Value domain 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.

Backing up and restoring data in the issue tracker

A detailed list of steps are available at the Move Installation page of Mozilla Wiki (which describes moving the installation from one machine to another, but can also be applied for backup and restore on the same server). The important parts to take note of are the commands used for dumping the SQL database:

```
$ mysqldump -u(username) -p(password) bugs > bugzilla-backup.sql
```

Reloading the SQL dump later requires the database to be cleared and recreated from the MySQL console:

```
mysql> DROP DATABASE bugs;
mysql> CREATE DATABASE bugs DEFAULT CHARSET utf8;
```

Applying the dump goes as follows:

```
$ mysql -u (username) -p(password) bugs < /path/to/bugzilla-backup.sql
```

In addition to the contents of the database, the `data` directory and the `localconfig` file from Bugzilla's installation directory should also be preserved. = Console command reference

Introduction

This documents 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

Getting help

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 the terminology server itself:

- Local code system datastore commands — `localcodesystem`

- LOINC datastore commands — `loinc`
- Mapping Set datastore commands — `mappingset`
- Scripting Commands — `script`
- MRCM commands — `mrcm`
- Terminology registry commands — `terminologyregistry`
- SNOMED CT OWL ontology commands — `ontology`
- SNOMED CT importer commands — `sctimport`
- Value Domain datastore commands — `valueset`
- Diagnostic and maintenance commands — `snowowl`
- Session management commands — `session`

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

```
osgi> localcodesystem
--- Local code system datastore commands ---
       localcodesystem exportXL local_codesystems_excelfile_path - Exports local
code...
       localcodesystem importXL local_codesystems_excelfile_path clear_database - ...
```

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.
```

Local code systems

Import local code systems from a spreadsheet

`localcodesystem importXL` imports local code system content from the specified Excel spreadsheet (with `.xls` or `.xlsx` extension). Required arguments are the spreadsheet path and the merge mode, expressed as an integer with three possible values:

1. “clear” — all existing local terminology entries are removed from the terminology store first, then the contents of the spreadsheet are added;
2. “merge” — if the terminology store already contains a local terminology to be imported, metadata groups and local codes with the same identifiers will be unaffected, while new incoming elements (ex.: local codes, metadata keywords) are going to be added to the existing local terminology
3. “replace” — similar to merge, but when an existing local terminology is encountered, the existing content will be replaced by the spreadsheet’s related content.

The administrator is also required to provide a user identifier; the given user ID will be displayed in the commit log as the importing user.

```
osgi> localcodesystem importXL /path/to/local_code_systems.xlsx 0
Impersonate operation as: info@b2international.com

... - Importing local code systems from excel file: /path/to/local_code_systems.xlsx
...
... - Clearing index data...
... - Clearing index data successfully finished.
... - Converted local code system ABCD with 9 codes.
...
... - Updating terminology metadata.
...
```

Export local code systems to a spreadsheet

`localcodesystem exportXL` will generate an Excel file in XLSX format, in which all local terminology metadata and codes will be presented on different worksheets. The only required argument is a destination path; the exported file will be added to this location on the server.

```
osgi> localcodesystem exportXL /path/to/exported_spreadsheet.xlsx

... - Exporting Local Code Systems to Excel started. Server-side file:
/path/to/exported_spreadsheet.xlsx
...
... - Finished exporting Local Code Systems to Excel.
```

LOINC

Import LOINC vocabulary from a release archive

`loinc import` replaces existing LOINC terminology store content with the concepts and multi-axial hierarchy given in the specified input `.zip` archive. This command also requires a user identifier to be entered, which will be presented as the importing user in the commit information view.

```
osgi> loinc import /path/to/loinc_archive.zip
Impersonate operation as: info@b2international.com

... - LOINC import: 0%
... - Processed LOINC multi-axial hiearchy lines: 0
... - Processed LOINC multi-axial hiearchy lines: 10000
... - LOINC import: 1%
...
... - Set children for number of beans: 1000
... - Set children for number of beans: 2000
... - LOINC import: 10%
... - Set children for number of beans: 3000
...
... - Processed and normalized the LOINC hierarchy file: 76675 entries.
... - Processed LOINC table rows: 10000
... - LOINC import: 16%
... - Processed LOINC table rows: 20000
...
... - Processed the LOINC table file: 71464 lines.
... - Processed 10000 skeleton components.
...
... - Processed the total of 76675 skeleton components.
...
... - Processed 50000 concepts with hierarchy info.
...
... - Processed 76675 concepts with hierarchy info.
... - Commit notification received for user info@b2international.com.
... - Clearing index data...
... - Clearing index data successfully finished.
...
... - LOINC import: 96%
... - Processed 40000 index entries.
... - LOINC import: 97%
...
... - LOINC import: 100%
... - LOINC import completed.
```

Mapping sets

Import mapping sets from a spreadsheet

`mappingset import` allows the administrator to import one or more mapping sets from the specified spreadsheet; the command also requires a merge setting (given as an integer). The selection and meaning of merge options are the same as the ones given in Import local code systems from a spreadsheet.

```
osgi> mappingset import /path/to/mapping_sets.xlsx 1
Impersonate operation as: info@b2international.com

... - Importing mapping sets from excel file: /path/to/mapping_sets.xlsx
... - Processed excel sheet 1. First spreadsheet for mapping set
... - Branch: MAIN Event: MappingSet:  new components added: 12
... - Commit notification received for user info@b2international.com.
```

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: ABC Local Test Dictionary short name: ABCD OID: 9.8.7.6.54321 organization:
http://localhost/abcd language: ENG last version: 0.1
Name: International Classification of Diseases short name: ICD-10 OID:
2.16.840.1.113883.6.3 organization: http://www.who.int/classifications/icd/en/
language: ENG last version: 1
Name: Logical Observation Identifiers Names and Codes short name: LOINC OID:
2.16.840.1.113883.6.1 organization: http://loinc.org language: ENG last version: 2
Name: Anatomical Therapeutic Chemical Classification System short name: ATC OID:
2.16.840.1.113883.6.73 organization: http://www.who.int/classifications/atcddd/en/
language: ENG last version: 1
Name: Australian Modification of the International Classification of Diseases short
name: ICD-10-AM OID: 2.16.840.1.113883.6.135 organization:
http://sydney.edu.au/health-sciences/ncch/about.shtml language: ENG last version: 1
Name: Systematized Nomenclature of Medicine Clinical Terms International Version short
name: SNOMEDCT OID: 2.16.840.1.113883.6.96 organization: http://www.ihtsdo.org
language: ENG last version: 2013-07-31
```

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.
```

Value Domain

Import value domains from a UMLS SVS XML file

`valueset import` works with value domain terminology content and also supports the three merge modes mentioned at the local code system import command. The administrator will be prompted for an importing user identifier, which is required for identification in the commit information view.

The two required arguments are the absolute path of the source XML file, and the selected merge mode (represented by an integer in the range of 0..2).

```
osgi> valueset import /path/to/svs_import_file.xml 0
Impersonate operation as: info@b2international.com

[2013-12-12 16:50:04.660] [Worker-11] INFO  c.b.s.d.s.i.AbstractTerminologyImportJob -
Importing value domains from UMLS file /path/to/svs_import_file.xml.
[2013-12-12 16:50:04.660] User: info@b2international.com Event: Importing value
domains from UMLS file /path/to/svs_import_file.xml.
```

```
[2013-12-12 16:50:04.660] [Worker-11] INFO  c.b.s.d.s.i.AbstractTerminologyImportJob -
Deleting existing value domains from database...
[2013-12-12 16:50:04.660] User: info@b2international.com Event: Deleting existing
value domains from database...
[2013-12-12 16:50:04.755] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionBeforeCommitting() start
[2013-12-12 16:50:04.755] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionBeforeCommitting() lock acquired for BranchPath{Path='MAIN'}
[2013-12-12 16:50:04.776] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionBeforeCommitting() end
[2013-12-12 16:50:04.785] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionAfterCommitted() start
[2013-12-12 16:50:04.787] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionAfterCommitted() end
[2013-12-12 16:50:04.787] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionAfterCommitted() lock released for BranchPath{Path='MAIN'}
[2013-12-12 16:50:04.956] [Worker-11] INFO  c.b.s.d.PostStoreUpdateManager - Commit
notification received for user info@b2international.com.
[2013-12-12 16:50:05.083] [Worker-11] INFO  c.b.s.d.s.i.AbstractTerminologyImportJob -
Processed value domain Hospital Measures-Joint Commission Mental Disorders.
[2013-12-12 16:50:05.083] User: info@b2international.com Event: Processed value domain
Hospital Measures-Joint Commission Mental Disorders.
[2013-12-12 16:50:05.141] [Worker-11] INFO  c.b.s.d.s.i.AbstractTerminologyImportJob -
Processed value domain Hospital Measures-Comfort Measures Only Intervention.
[2013-12-12 16:50:05.141] User: info@b2international.com Event: Processed value domain
Hospital Measures-Comfort Measures Only Intervention.
...
[2013-12-12 16:50:15.075] [Worker-11] INFO  c.b.s.d.s.i.AbstractTerminologyImportJob -
Processed value domain Ethnicity.
[2013-12-12 16:50:15.075] User: info@b2international.com Event: Processed value domain
Ethnicity.
[2013-12-12 16:50:17.352] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionBeforeCommitting() start
[2013-12-12 16:50:17.352] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionBeforeCommitting() lock acquired for BranchPath{Path='MAIN'}
[2013-12-12 16:50:28.973] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionBeforeCommitting() end
[2013-12-12 16:50:39.420] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionAfterCommitted() start
[2013-12-12 16:50:40.085] User: info@b2international.com Branch: MAIN Event:
ValueSetFolder:  new components added: 131646
[2013-12-12 16:50:40.086] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionAfterCommitted() end
[2013-12-12 16:50:40.086] [Worker-11] INFO  c.b.s.d.s.CDOServerChangeManager -
handleTransactionAfterCommitted() lock released for BranchPath{Path='MAIN'}
[2013-12-12 16:50:40.965] [Worker-11] INFO  c.b.s.d.PostStoreUpdateManager - Commit
notification received for user info@b2international.com.
[2013-12-12 16:50:41.005] [Worker-11] INFO  c.b.s.d.s.i.AbstractTerminologyImportJob -
Commited value domains.
[2013-12-12 16:50:41.005] User: info@b2international.com Event: Commited value
domains.
```

```
[2013-12-12 16:50:41.009] [Worker-11] INFO  c.b.s.v.d.s.i.ValueSetIndexInitializerJob
- Clearing indexes for value domain import...
[2013-12-12 16:50:41.009] User: web Event: Clearing indexes for value domain import...
[2013-12-12 16:50:41.020] [Worker-11] INFO  c.b.s.v.d.s.i.ValueSetIndexInitializerJob
- Cleared indexes for value domain import
[2013-12-12 16:50:41.020] User: web Event: Cleared indexes for value domain import
[2013-12-12 16:50:41.023] [Worker-11] INFO  c.b.s.v.d.s.i.ValueSetIndexInitializerJob
- Processed 2 value domain member index entries.
[2013-12-12 16:50:41.023] User: web Event: Processed 2 value domain member index
entries.
...
[2013-12-12 16:50:56.176] [Worker-11] INFO  c.b.s.v.d.s.i.ValueSetIndexInitializerJob
- Processed 20 index entries for folders.
[2013-12-12 16:50:56.176] User: web Event: Processed 20 index entries for folders.
[2013-12-12 16:50:58.125] [Worker-11] INFO  c.b.s.d.s.i.AbstractTerminologyImportJob -
Completed importing value domains from UMLS file /path/to/svs_import_file.xml.
[2013-12-12 16:50:58.125] User: info@b2international.com Event: Completed importing
value domains from UMLS file /path/to/svs_import_file.xml.
```

Diagnostics and maintenance

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

```
osgi> snowowl listrepositories
Repositories:
    snomedStore
```

`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 namspace 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.

Supporting indexes

Supporting indexes store additional data that is referenced across terminology stores, such as previous choices of users, bookmarks, and task metadata. Access to these items cannot be locked, however, a consistent snapshot of their contents can be taken and saved for backup purposes. Snapshots are only kept while the server is running; after a restart, only the latest data will be available. It is advised to release snapshots straight after a successful backup.

The provided backup script performs the steps below for all available supporting indexes automatically.

Display supporting indexes

To list all available supporting index identifiers, issue:

```
osgi> index list

Index service identifier
--------------------------------
  previous_picks
  bookmarks
  tasks
--------------------------------
```

Create snapshot

Consistent snapshots can be referenced by their UUIDs, which is displayed when a new snapshot is created.

```
osgi> index createSnapshot tasks
Snapshot '885bded0-5e93-4f20-bdbd-aafd5434a41a' for service 'tasks' has been
successfully created.
```

List snapshots

To get the list of currently available snapshots for any supporting index, run:

```
osgi> index listSnapshots tasks

Index snapshot identifier
--------------------------------
 885bded0-5e93-4f20-bdbd-aafd5434a41a
--------------------------------
```

List files in snapshots

To collect all files that make up a particular snapshot, execute the following command:

```
osgi> index listSnapshotFiles tasks 885bded0-5e93-4f20-bdbd-aafd5434a41a

 Files in snapshot '885bded0-5e93-4f20-bdbd-aafd5434a41a'
--------------------------------
 tasks/_1.si
 tasks/_0.si
 tasks/_1.cfs
 tasks/_1.cfe
 tasks/segments_3
 tasks/_0.cfs
 tasks/_0.cfe
--------------------------------
```

Files are displayed relative to the Snow Owl Server installation's `resources/index` folder.

Release created snapshot

To unreference files that might only be in use because a snapshot was taken, and free up disk space, run:

```
osgi> index releaseSnapshot tasks 885bded0-5e93-4f20-bdbd-aafd5434a41a
Snapshot 885bded0-5e93-4f20-bdbd-aafd5434a41a has been successfully released.
```

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        |
Started          | Status
-------------------------------------------------------------------------------------
----------------------
   0 | Batch ontology generation        | info@b2intern... | 2014-03-20 11:03 | 2014-
03-20 11:03 | Finished
-------------------------------------------------------------------------------------
----------------------
   1 | Classifying the ontology on MAIN | info@b2intern... | 2014-03-20 11:13 | 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.

Administrative REST API reference

Introduction

This document provides a list of available REST endpoints, which can be used for taking scripted backups of a Snow Owl Server instance.

REST API

The following URLs are exposed to assist the administrator in creating backups without stopping the server (accepted and produced content type is `text/plain` in all cases except errors, where an HTML message may be returned):

Write access control via repository locks

- `GET /snowowl/admin/repositories` — Display list of available terminology repositories
  - Parameters:
    - none
  - Returns: the list of registered repositories by identifier (one per line, separated by LF)
  - Responses:
    - `200 (OK)`
- `POST /snowowl/admin/repositories/lock` — Lock all terminology repositories for exclusive access
  - Parameters:
    - `timeoutMilis` - lock timeout in milliseconds (optional; default is 5 seconds)
  - Returns: empty response body
  - Responses:
    - `204 (No Content)` - if the operation succeeded
    - `400 (Bad Request)` - if the format or number of expected arguments is incorrect
    - `409 (Conflict)` - if another participant is already holding a lock (eg. a long-running operation has exclusive access)
- `POST /snowowl/admin/repositories/unlock` — Release previously acquired lock on all terminology repositories
  - Parameters:
    - none
  - Returns: empty response body
  - Responses:
    - `204 (No Content)` - if the lock has been successfully released
    - `400 (Bad Request)` - if releasing the lock fails for some reason (eg. when no lock was held by the system on the specified repository)
- `POST /snowowl/admin/repositories/{id}/lock` — Lock specific repository for exclusive access
  - Parameters:
    - `id` (path parameter) - the repository identifier (as returned by the call to the first URL)

    - `timeoutMilis` - lock timeout in milliseconds (optional; default is 5 seconds)
  - Returns: empty response body
  - Responses:
    - `204 (No Content)` - if the operation succeeded
    - `409 (Conflict)` - if another participant is already holding a lock (eg. a long-running operation has exclusive access)

- `POST /snowowl/admin/repositories/{id}/unlock` — Release previously acquired lock on specific repository
  - Parameters:
    - `id` (path parameter) - the repository identifier
  - Returns: empty response body
  - Responses:
    - `204 (No Content)` - if the lock has been successfully released
    - `400 (Bad Request)` - if the format or number of expected arguments is incorrect, or if releasing the lock fails for some reason (eg. when no lock was held by the system on the specified repository)

User notification

- `POST /snowowl/admin/messages/send` — Send a message to connected users
  - Parameters:
    - `message` - the message to send
  - Returns: empty response body
  - Responses:
    - `400 (Bad Request)` - if the format or number of expected arguments is incorrect

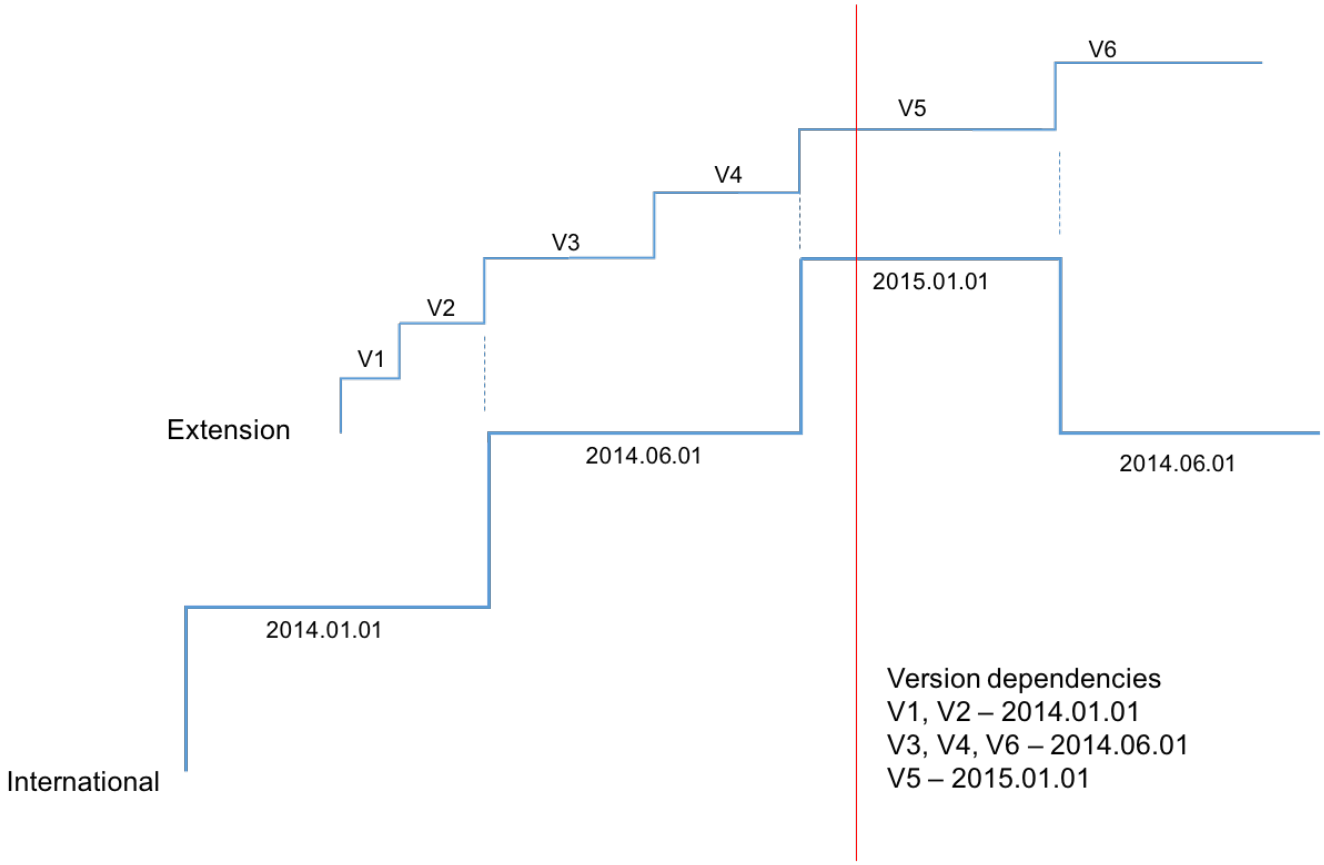
Versioning and terminology indexes

- `GET /snowowl/admin/repositories/{id}/versions` — Display list of registered versions for the specified repository
  - Parameters:
    - `id` (path parameter) - the repository identifier
  - Returns: the list of versions for the specified repository (one per line, separated by LF; always including `MAIN` as a placeholder for the mainline of the repository, from which a version may be created later)
  - Responses:

- `200 (OK)` - if the operation succeeded
- `404 (Not Found)` - if a repository with the given identifier could not be found = SNOMED CT Extension management

Introduction

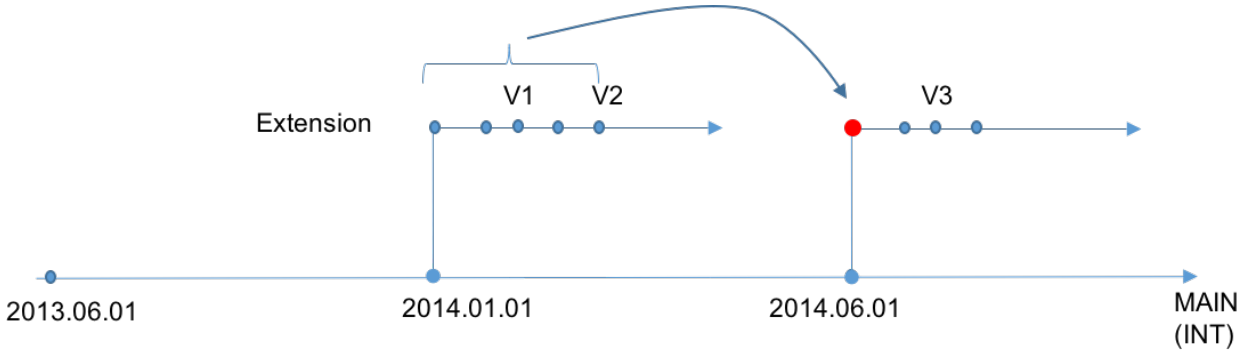
Snow Owl is capable of managing multiple SNOMED CT extensions on a single server. This document highlights how each extension is managed in a dedicated area and lists the main authoring scenarios. Similarly to generic SNOMED CT authoring, content is developed via the help of a workflow where the work performed on a task is promoted to the repository after review. Promoting (donating) content from an extension to the international content or between extensions is not supported. Extension authors can follow their own release schedule and extension content can be versioned at any time during the authoring process. Extensions are based on a single international release version. The base international release can be upgraded 'underneath' the extension content. Extensions do not have an additional isolated area for long running development projects (project branch). The import and export of standard RF2 serialized content is supported. The international content is not edited on the server, only updated via delta releases as consecutive releases are available from IHTSDO's own authoring process. Search will be performed for both the native language and English as is currently supported. There will be no special search features to accommodate specific language requirements, such as word decompoundization.



Branching arrangement

The implemented branching arrangement closely follows the current branching preferences where the international release is imported onto the MAIN branch and each Member Country will have a

dedicated branch for the extension content. The only commits on the MAIN branch are the contents of the delta release for a particular version. These commit points are versioned during the import process. The dedicated branch will be forked off the MAIN branch at the point representing the base international release as shown in Figure 1.



Versioning

The extension content is marked with a version tag for future reference with a new empty branch. All new components will receive an effective time.

Upgrading/Downgrading

Upgrading means to increase the version of the base international release. During the upgrade process, the current extension branch is forward-rebased via creating a new branch and copying over the entire content as a single commit. As shown in Figure 2, the V1 and V2 versioned content is based on the 2016.01.01, whereas the V3 version is based on the 2016.06.01 version of the international release. The main advantage is that the old branch still maintains the earlier versions of the extension, this way the version dependencies between the extension and international release can be implicitly represented via the branches. If the user decides to switch back to the read-only older V1 version, content is properly displayed with international content from 2016.01.01. To avoid content with incorrect dependencies, only versioned extensions should be allowed to be upgraded. Downgrading means to decrease the version of the base international release. The same process described above for upgrade applies.

International version update

When new international release is available, the corresponding delta is imported onto the MAIN branch.

Data provisioning example for a single extension

The following example demonstrates how a blank server is provisioned for extension management. The example is carried out using both the command line OSGi interface as well as the REST API.

OSGi command line interface

First, the full International Release is imported with version tags created (*-v*). The selected

Language Refset is English.

```
> sctimport rf2_release -t full -v /path/to/SnomedCT_RF2Release_INT_20160131.zip
/path/to/snomed_ct_international.json
```

Where the `snomed_ct_international.json` release descriptor file is:

```
{
  "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."
}
```

After the import process, to check the terminology registry issue the command:

```
> terminologyregistry listall
```

To see all the version branches created execute:

```
> snowowl listbranches snomedStore
```

To import an extension terminology based on the *2016-01-31* International release, issue the command:

```
> sctimport rf2_release -t full -b 2016-01-31 -v
/path/to/SnomedCT_Extension_Release.zip /path/to/snomed_ct_extension.json
```

Where the `snomed_ct_extension.json` release descriptor file is:

```
{
  "name": "B2i Healthcare SNOMED CT extension",
  "shortName": "SNOMEDCT_B2i",
  "language": "ENG",
  "codeSystemOID": "2.16.840.1.113883.6.961",
  "extensionOf": "SNOMEDCT",
  "maintainingOrganizationLink": "http://www.b2i.sg",
  "citation": "This is an example SNOMED CT extension by B2i Healthcare."
}
```

Again, after the import process, to check the terminology registry issue the command:

```
> terminologyregistry listall
```

To see all the version branches created execute:

```
> snowowl listbranches snomedStore
```

There should be a branch named *MAIN/2016-01-31/SNOMEDCT_B2i* dedicated for the initial content of the extension.

REST API

Importing the terminology and creating an entry in the terminology registry require two separate REST endpoints to be invoked.

Upload the import configuration:

```
POST /snowowl/snomed-ct/v2/imports
{
  "createVersions": true,
  "type": "FULL",
  "branchPath": "MAIN",
  "codeSystemShortName": ""
}
```

Note: For backward compatibility, missing or empty *codeSystemShortName* result in a SNOMED CT International Release entry in the terminology registry, no explicit operation is required.

Import the International archive:

```
POST /snowowl/snomed-ct/v2/imports/{importId}/archive
```

After the import process, check the terminology registry for the entry representing the International release:

```
GET /snowowl/admin/codesystems
```

Create new branch for extension:

```
POST /snowowl/snomed-ct/v2/branches
{
  "metadata": {},
  "parent": "MAIN/2016-01-31",
  "name": "Extension_branch"
}
```

To check the branch created (*MAIN/2016-01-31/Extension_branch*):

```
GET /snowowl/snomed-ct/v2/branches
```

Create new Code System for the extension. The extension will initially be based on the *2016-01-31* International release:

```
POST /snowowl/admin/codesystems
{
    "oid": "2.16.840.1.113883.6.961",
    "name": "B2i Healthcare SNOMED CT extension",
    "shortName": "SNOMEDCT_B2i",
    "organizationLink": "http://www.b2i.sg",
    "primaryLanguage": "ENG",
    "citation": "This is an example SNOMED CT extension by B2i Healthcare.",
    "iconPath": "icons/b2i_extension_icon.png",
    "terminologyId": "com.b2international.snowowl.terminology.snomed",
    "repositoryUuid": "snomedStore",
    "branchPath": "MAIN/2016-01-31/Extension_branch",
    "extensionOf": "SNOMEDCT"
}
```

Check the terminology registry for the extension entry:

```
GET /snowowl/admin/codesystems
```

Upload import configuration for extension import:

```
POST /snowowl/snomed-ct/v2/imports
{
  "createVersions": true,
  "type": "DELTA",
  "branchPath": "MAIN/2016-01-31/Extension_branch",
  "codeSystemShortName": "SNOMEDCT_B2i"
}
```

Import the extension:

```
POST /snowowl/snomed-ct/v2/imports/{importId}/archive
```

Versioning extension content

There is no command-line command available for versioning, the functionality is only accessible via the REST interface:

```
POST /snowowl/admin/codesystems/SNOMEDCT_B2i/versions
{
  "version": "Extension:2016-06-01",
  "description": "Extension test version",
  "effectiveDate": "20160601"
}
```

Upgrading extension content

There is no command-line command available for upgrading, the functionality is only accessible via the REST interface.

Import the new International DELTA archive to the MAIN branch in two steps:

First create the import configuration.

```
POST /snowowl/snomed-ct/v2/imports
{
  "createVersions": true,
  "type": "DELTA",
  "branchPath": "MAIN",
  "codeSystemShortName": "SNOMEDCT"
}
```

Then upload the archive to start the import.

```
POST /snowowl/snomed-ct/v2/imports/{importId}/archive
```

After the import process, create a new branch for extension:

```
POST /snowowl/snomed-ct/v2/branches
{
  "metadata": {},
  "parent": "MAIN/2016-07-31",
  "name": "Extension_branch"
}
```

Then copy over the entire content onto the target extension branch. If the merge fails because of conflicts, fix the errors and then try again.

```
POST /snowowl/snomed-ct/v2/merges
{
  "source": "MAIN/2016-01-31/Extension_branch",
  "target": "MAIN/2016-07-31/Extension_branch",
  "commitComment": "Upgrade extension content."
}
```

The above call will return the unique ID of the merge process which can be used to poll its status (e.g. *IN_PROGRESS*, *COMPLETED*, *CONFLICTS*, etc).

```
GET /snowowl/snomed-ct/v2/merges/{mergeId}
```

Finally update the branch path of the extension Code System to point to the target extension branch.

```
PUT /snowowl/admin/codesystems/SNOMEDCT_B2i
{
  "repositoryUuid": "snomedStore",
  "branchPath": "MAIN/2016-07-31/Extension_branch"
}
```

Downgrading extension content

There is no command line command available for downgrading, the functionality is only accessible via the REST interface.

Copy over the current extension content onto the target extension branch. If the merge fails because of conflicts, fix the errors and then try again.

If no new content was added to the current branch since the merge, you can skip this step.

```
POST /snowowl/snomed-ct/v2/merges
{
  "source": "MAIN/2016-07-31/Extension_branch",
  "target": "MAIN/2016-01-31/Extension_branch",
  "commitComment": "Downgrade extension content."
}
```

Then update the branch path of the extension Code System to point to the target extension branch.

```
PUT /snowowl/admin/codesystems/SNOMEDCT_B2i
{
  "repositoryUuid": "snomedStore",
  "branchPath": "MAIN/2016-01-31/Extension_branch"
}
```

SNOMED CT Swedish Extension management example

The following example demonstrates how a blank server is provisioned for the Swedish extension management through the REST API.

First create the import configuration for the International release.

```
POST /snowowl/snomed-ct/v2/imports
{
  "createVersions": true,
  "type": "FULL",
  "branchPath": "MAIN",
  "codeSystemShortName": "SNOMEDCT"
}
```

Then upload the RF2 archive file to start the import.

```
POST /snowowl/snomed-ct/v2/imports/{importId}/archive
```

After the import process, create a new branch for the Swedish extension which is based on version 2015-01-31.

```
POST /snowowl/snomed-ct/v2/branches
{
  "metadata": {},
  "parent": "MAIN/2015-01-31",
  "name": "SNOMEDCT-SE"
}
```

Create a new Code System for the Swedish extension. The branch path points to the previously created branch.

```
POST /snowowl/admin/codesystems
{
    "oid": "2.16.840.1.113883.6.962",
    "name": "SNOMED CT Swedish Extension",
    "shortName": "SNOMEDCT-SE",
    "organizationLink": "http://www.socialstyrelsen.se/",
    "primaryLanguage": "SWE",
    "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.",
    "iconPath": "icons/snomed.png",
    "terminologyId": "com.b2international.snowowl.terminology.snomed",
    "repositoryUuid": "snomedStore",
    "branchPath": "MAIN/2015-01-31/SNOMEDCT-SE",
    "extensionOf": "SNOMEDCT"
}
```

Create the import configuration for the Swedish import.

```
POST /snowowl/snomed-ct/v2/imports
{
  "createVersions": true,
  "type": "FULL",
  "branchPath": "MAIN/2015-01-31/SNOMEDCT-SE",
  "codeSystemShortName": "SNOMEDCT-SE"
}
```

Upload the Swedish RF2 archive file to start the extension import.

```
POST /snowowl/snomed-ct/v2/imports/{importId}/archive
```

Create a new branch based on version 2016-01-31. The Swedish extension will be merged onto this branch.

```
POST /snowowl/snomed-ct/v2/branches
{
  "metadata": {},
  "parent": "MAIN/2016-01-31",
  "name": "SNOMEDCT-SE"
}
```

Start a merge to upgrade the extension content onto the previously created branch.

```
POST /snowowl/snomed-ct/v2/merges
{
  "source": "MAIN/2015-01-31/SNOMEDCT-SE",
  "target": "MAIN/2016-01-31/SNOMEDCT-SE",
  "commitComment": "Upgrade Swedish extension to international version 2016-01-31"
}
```

The POST operation will return a *mergeId* that can be used to monitor the progress of the merge via:

```
GET /snowowl/snomed-ct/v2/merges/{mergeId}
```

Which operation will return with failed status and the following conflicts:

```
Relationship with ID '6271000052128' has a conflict of type 'HAS_INACTIVE_REFERENCE'
on target branch, conflicting attributes are: [destinationId -> 373245004].
Relationship with ID '16741000052126' has a conflict of type 'HAS_INACTIVE_REFERENCE'
on target branch, conflicting attributes are: [destinationId -> 373245004].
Relationship with ID '44001000052128' has a conflict of type 'HAS_INACTIVE_REFERENCE'
on target branch, conflicting attributes are: [destinationId -> 237968007].
Relationship with ID '1929491000052120' has a conflict of type
'HAS_INACTIVE_REFERENCE' on target branch, conflicting attributes are: [destinationId
-> 237968007].
```

After fixing the conflicts, start another merge (same as before) and then update the branch path of the Swedish Code System.

```
PUT /snowowl/admin/codesystems/SNOMEDCT-SE
{
  "repositoryUuid": "snomedStore",
  "branchPath": "MAIN/2016-01-31/SNOMEDCT-SE"
}
```

Developer notes

During the upgrade/downgrade process, conflicts are identified between the terminology to be upgraded/downgraded and the international release. Using the **mergeConflictRuleProvider** extension point, rule providers can be registered the provide conflict rules. The currently registered rule provider is **com.b2international.snowowl.datastore.server.snomed.merge.SnomedMergeConflictRuleProvider.java**.