# iKnowBase Installation Guide

Version 8.3

# Table of Contents

# Chapter 1. Introduction

Welcome to the iKnowBase Installation Guide. Note that this installation guide only covers upgrades from iKnowBase 5.7.2 and newer.

| | |
|---|---|
| **NOTE** | If you are upgrading from an older version, you must <u>first</u> upgrade to iKnowBase 5.7.2 using the old upgrade instructions, and <u>then</u> use these upgrade instructions to upgrade to the latest version. |

# Overview

# Chapter 2. Installation topologies

## 2.1. iKnowBase components

[iKnowBase Conceptual Topology] | *images/Overview-ConceptualTopology.png*

*Figure 1. iKnowBase Conceptual Topology*

The diagram shows the various components of an iKnowBase installation:

- The database ("iKnowBase repository") contains configuration, metadata and content. This is deployed to an Oracle Database.
- The iKnowBase application containing all server side components.

The iKnowBase application is modularized. Modules may be activated/deactivated as needed. You can run several instances of the iKnowBase application, each with its own configuration.

See the *Java Applications* section for details about modules and deployment. See the *Configuration* section for details about how the modules can be enabled/disabled and configured.

## 2.2. Deployment options

### 2.2.1. The iKnowBase repository

The iKnowBase repository is installed onto an Oracle database. For production use, we recommend that you always install at least three repositories: development (where you build new functionality), test (where you verify that the functionality works) and production (where live data lives).

For smaller installations, use a single database and create schemas (database users) for the required repositories. This is extremely easy to set up, but provides limited isolation between the environments. Some applications are built using hard coded schema names, and thus require a different database instance for each repository. These instances may all reside on the same Oracle database server, using the same database license.

Advanced installations (typically with large data volumes or stricter security requirements) will want to install the production repository on its own server; the development and test repositories may still be co-located if that is desired.

### 2.2.2. The web tier

The web tier can also be installed in a large variety of ways:

For most installations, use the iKnowBase web server to host the web applications. This server is customized for running iKnowBase, containing all required functionality in a small, easy-to-manage installation.

- For small installations, run a single iKnowBase web server on the same server as the database.

- For larger installations, run multiple iKnowBase web servers, on one or more physical machines.
- For higher security requirements, install separate instances for internal and public use. Run development and similar services only on the internal server.

Customers that already have an established and managed infrastructure using a supported third party application server may also choose to run the web tier on that server. Note that we recommend setting up the iKnowBase web server even then, since the iKnowBase program is also used for repository installation and upgrades, and is a useful management and troubleshooting tool.

We generally recommend installing an Apache (or similar) proxy server in front of the iKnowBase web server. This is useful for things like SSL-termination, serving static content, redirect rules, virtual hosts, etc.

## 2.3. Supported infrastructure

See "Supported platforms" in iKnowBase Release Notes.

# Chapter 3. Quick installation and upgrade overview

This chapter gives a brief overview of the installation or upgrade of an iKnowBase instance.

The process typically have the following steps:

- Download and install the software onto your application server.
- Prepare the instance-specific installation directory.
- Use the iKnowBase program to install or upgrade the database repository.
- Run the iKnowBase web server to verify the installation.
- Optional: Extend and deploy the iKnowBase web application with plugins and patches.

## 3.1. Recommended directory structure

We recommend that you choose a server for storing the iKnowBase software and running the installation. In most scenarios this server would be a server where the web applications will run. Here, install into a directory structure similar to the one below, with one directory for each version the actual distribution (named after the distribution version) and one directory for each iKnowBase installation (each database repository):

| Directory | Purpose |
|---|---|
| /opt/iknowbase | Root for all iKnowBase software. |
| .../distributions | Collection of all distribution files, packed. |
| .../iknowbase-6.4 | Version specific directory for the unpacked software (old version). |
| .../iknowbase-8.3 | Version specific directory for the unpacked software (current version). |
| .../development | Directory for a given instance (example: "development"). |
| .../development/plugins | Directory containing all plugins (.jar files only). |
| .../production | Directory for a given instance (example: "production"). |
| .../production/plugins | Directory containing all plugins (.jar files only). |

## 3.2. Download and install the iKnowBase software

Using the recommended directory structure above, install the iKnowBase software into the proper location. The assumption here is that the distribution file has been downloaded to /tmp.

```
sudo mkdir /opt/iknowbase
sudo chown iknowbase /opt/iknowbase
mkdir /opt/iknowbase/distributions
cp /tmp/iknowbase-8.3-bin.zip /opt/iknowbase/distributions/

# Note: the .zip-file contains top level directory "iknowbase-8.3"
cd /opt/iknowbase
unzip distributions/iknowbase-8.3-bin.zip
```

## 3.3. Prepare the instance-specific home directory and configuration

You will typically have multiple iKnowBase repositories, to handle different phases in the life cycle, such as development, testing and production. We recommend that you set up a directory for each such repository, where you store configuration files, log files, etc. This chapter describes setting up only one such instance, so you should repeat this for e.g. development, test and production.

```
mkdir /opt/iknowbase/production
```

For simplicity, and to avoid accidentally using the wrong version of the iKnowBase program, we also recommend creating an "iknowbase.sh" script that forwards to the proper version. Run the script below from each of the instance-specific directories:

```
cd /opt/iknowbase/production
cat > iknowbase.sh << 'EOF'
#!/bin/bash
../iknowbase-8.3/iknowbase.sh $*
EOF

chmod +x iknowbase.sh
```

For each repository, create a property file `application.properties` with all the required settings for connecting to the database and running the iKnowBase web server. A sample is provided in `etc/application.properties.SAMPLE` in the distribution. Copy the sample as `application.properties`, edit the new file and set proper values for options prefixed with `spring.datasource` and `com.iknowbase.setup`.

```
cd /opt/iknowbase/production
cp ../iknowbase-8.3/etc/application.properties.SAMPLE application.properties
```

- `spring.datasource.url` is the jdbc url to the database, see example below and in SAMPLE.properties.
- `spring.datasource.username` is the name of the database user that contains the iKnowBase

repository that will be upgraded.

- `spring.datasource.password` is the password for the iKnowBase database user.

During installation and upgrade, some scripts will need SYSDBA privilege in the database. We recommend that you do not store the password of the privileged user in the configuration file, but that you instead use the setup option `--sysPassword -`, which will ask for the system password from the console.

Alternatively, use the environment variable `COM_IKNOWBASE_SETUP_DATABASE_SYSPASSWORD` when needed.

However, should you decide that you want to configure this connection in the configuration file, use the following properties:

- `com.iknowbase.setup.database.sysUser` is the name of a user with SYSDBA privilege in the database, typically "SYS"
- `com.iknowbase.setup.database.sysPassword` is the password for the SYS-user in the database.

Note that the name `application.properties` is the default and recommended name. Changing this name requires extra environment or JVM options. See Spring Boot's documentation for details.

## 3.4. Install a new database repository

Use this chapter if you are installing a new iKnowBase repository. If you are upgrading an existing repository, follow the steps in the next chapter.

We encourage to use the AL32UTF8 charset in the database, and use the default tablespace (USERS).

1. Create user and import schema content:

```
cd /opt/iknowbase/production
./iknowbase.sh createUser
./iknowbase.sh uploadFile ../iknowbase-8.3/etc/IKB_MASTER_83.dmp
./iknowbase.sh importFile IKB_MASTER_83.dmp IKB_MASTER_83
```

2. Optionally download and display import log:

```
./iknowbase.sh downloadFile IKB_MASTER_83.log .
cat IKB_MASTER_83.log
```

3. Run upgrade scripts on the newly created installation:

```
./iknowbase.sh upgradeAll
```

4. Set password for ORCLADMIN, so that you may log in with the default security setup on the iKnowBase web server (replace the example password "changeMe" with one of your own

choosing):

```
./iknowbase.sh setIkbPassword orcladmin changeMe
```

## 3.5. Upgrade an existing database repository

Use this chapter if you are upgrading an existing repository. If you are installing a new iKnowBase repository, follow the steps in the previous chapter.

```
./iknowbase.sh exportSource source.zip
./iknowbase.sh configureUser
./iknowbase.sh upgradeAll
```

If you have any custom scripts that need to run, for example to grant permissions to custom code, run these now.

Upload the EXP-IKB_MASTER_<version>.dmp file from the etc/ folder in the iKnowBase zip file and import it from ikb$console to import any new metadata.

## 3.6. Configure and start the iKnowBase Web Application

With the database repository in place, you can run the iKnowBase web applications. This chapter describes how to configure and run the iKnowBase Web Server.

### 3.6.1. Plugins

If you have any plugins that you want to use, create a "./plugins" directory and install them there. Note that the server can only load .jar files. Check the chapter on Java-applications for more information.

| | |
|---|---|
| **NOTE** | iKnowBase Content Studio, iKnowBase Process Studio and iKnowBase REST User Repository applications are distributed as plugins. If you want to use these applications, add them in this step (see the *iKnowBase Process Studio* section. |

```
mkdir /opt/iknowbase/production/plugins

. Copy plugins, e.g. from /where/i/keep/my/plugins/*
. cp /where/i/keep/my/plugins/*.jar /opt/iknowbase/production/plugins/
```

#### iKnowBase REST User Repository plugin

The iKnowBase REST User Repository is deployed as a regular plugin and adds user repository resources to the `/ikb$rest` endpoint. These resources are by default protected by ACL with external

key `IKB_REST_REPOSITORY`.

When deployed, see /ressurs/iknowbase/doc/html/rest/ikb/index.html for details and configuration options.

### 3.6.2. Start the iKnowBase application

To run the iKnowBase web server, use:

```
cd /opt/iknowbase/production
./iknowbase.sh webServer
```

## 3.7. Next steps

Generic Java applications chapter:

- *Java Applications*

Application server specific chapters:

- *iKnowBase web server*

# Chapter 4. Configuration

The iKnowBase web application can be configured to adapt to different needs using configuration properties.

## 4.1. Overview

First, modules in iKnowBase expose a number of *Configuration objects* that control the workings of the module. Each configuration object has one or more named *configuration properties* that can be set by the user. If the user does not set a given configuration property, a default value will be used.

When the iKnowBase web application starts, it populates the configuration objects with property values set by the user, and then uses the configuration objects to adapt the module. Note that this implies that making changes to properties will require a restart of the application server.

At run time, the actual property values can be inspected in the management console, e.g. at /ikb$console.

## 4.2. Property sources

Configuration properties are available from many different sources. When an application requires a property value, it will check these sources in order and the first one that can supply the required property will be used. The order is provided by Spring Boot and described in Spring Boot - Externalized Configuration, however iKnowBase will in addition:

1. If the property `iknowbase.alias` has been configured
    - All properties prefixed with this alias in `$USER_HOME/iknowbase-aliases.properties` will be added to the environment (without the prefix).
    - These properties will take precedence over any other property files (`application.properties` etc).
    - Typically used to specify sensitive configuration properties like database connection details in a CI/build environment.
2. The property source "IKB_INSTALLATION_PROPERTIES" represents values loaded from the database table *ikb_installation_properties* and allows you to set values that apply to multiple application server instances.
    - These properties will take precedence before the default properties (next to last).
    - Requires that database connection details have already been specified in one of the previous property sources.
    - Can be managed using the iKnowBase Administration Console.
    - Some properties are required to be configured in this property source (like Solr's use of Secure Token Engine). This requirement is explicitly stated where applicable.

In practice, you will for regular servers probably use a combination of `application.properties` and the `ikb_installation_properties` tables.

### 4.2.1. The ikb_installation_properties database table

Using the IKB_INSTALLATION_PROPERTY table is shared between all applications and all application server instances, it is possible to add expressions that are checked at runtime in order to select the proper property. This is done using the "instance_qualifier" table column.

At startup, each iKnowBase java web application loads properties from the ikb_installation_table. For each row, it will evaluate the "instance_qualifier" value to decide if this particular property is valid for this particular application instance. The qualifier is interpreted using the Spring Expression Language, which allows for combining various types of tests.

The available variables and methods for the expression is limited to the following logical interface definition:

| Variable | Description |
|---|---|
| hostname | Name of server |
| directory | Startup directory; same as the java property "user.dir" |
| contextPath | Root context path of web application (e.g. "" or "/CustomContextRoot") |
| getSystemProperty(name) | Value of system property |

These can be combined in several ways, to achieve various effects:

| Qualifier | Description |
|---|---|
| `*` | Used by any application, anytime |
| `true` | Used by any application, anytime |
| `hostname == 'tango'` | Used by any application running on a host named "tango" |
| `directory == '/opt/iknowbase/sso'` | Used by applications running from the "/opt/iknowbase/sso"-directory |
| `hostname == 'tango' && contextPath == '/CustomContextRoot'` | Used by an application deployed to /CustomContextRoot, when running on a host named "tango" |
| `hostname == 'tango' && contextPath matches '/Custom.*'` | Used by all "/Custom" prefixed applications, when running on a host named "tango" (/CustomContextRoot, /CustomOtherContextRoot, ...) |

### 4.2.2. Language settings (NLS)

In order to set the session language for the JDBC Oracle connection to a different language than the instance- or database language, you need to add an instruction in the property file.

e.g. to set the language to Norwegian, add the following to `application.properties` :

```
JAVA_OPTIONS=-Duser.language=no -Duser.region=NO
```

# Setup details

# Chapter 5. Database repository

iKnowBase uses an Oracle Database for storing both data, metadata and applications. Inside the database, iKnowBase also stores a lot of system code, as well as public APIs for manipulating the data.

## 5.1. Fresh install

A fresh install of the Oracle repository is pretty simple, and consists of only a few steps:

- The database schema where iKnowBase is installed must be created, and it must be given the proper permissions. Also, an instance specific database package must be created.
- The database schema must be populated with startup data. These data can be loaded from an existing instance (for example, when doing a fresh install of a test environment, where the startup data comes from an existing production environment), or they can be loaded from the iKnowBase distribution.

A full set of typical commands is shown below, and further described in the following chapters.

```
$ cd /opt/iknowbase/production
$ ./iknowbase.sh createUser
$ ./iknowbase.sh uploadFile  IKB_MASTER_83.dmp
$ ./iknowbase.sh importFile IKB_MASTER_83.dmp IKB_MASTER_83
$ # Optionally download and display import log
$ ./iknowbase.sh downloadFile IKB_MASTER_83.log .
$ cat IKB_MASTER_83.log
```

### 5.1.1. Prepare the database schema

Installing the database schema is most easily done using the iKnowBase program. Assuming that you have created a file "production.properties" with the proper information, use the following command:

```
$ ./iknowbase.sh createUser
```

This command will perform three actions:

- First, as user SYS, it will create the user specified in the property file, with the password also specified there.
- Next, as user SYS, it will grant required permissions to that user. A full list of permissions can be seen in the log file after executing the command.
- Finally, as the newly created user, it will create the database package IKB_GLOBAL_PREFS with default variables.

### 5.1.2. Custom step for Oracle database with Pluggable databases (PDB)

For pluggable databases (PDBs), there is an additional step needed, since DATA_PUMP_DIR is defined at the CDB level and does not work properly with PDBs. You must define an explicit Directory object within the PDB after you have created the new schema. Create a new directory and configure iKnowBase to use it:

- Create directory <DIRECTORY_NAME> as '<OS path to where datapump files should be stored>';
- Grant read, write on directory <DIRECTORY_NAME> to <schema name>;
- Add a property to the iKnowBase configuration file e.g. db.dataPumpDirectory=<DIRECTORY_NAME>

### 5.1.3. Import startup data based on an export file

You can import startup data with any mechanism you choose, but once again the iKnowBase program is the preferred and supported mechanism. Using the iKnowBase program has two steps: First you upload the startup data to the Oracle database server, and then you import them into the Oracle database schema:

```
$ ./iknowbase.sh uploadFile IKB_MASTER_83.dmp
$ ./iknowbase.sh importFile IKB_MASTER_83.dmp `IKB_MASTER_83`
```

The first command will upload the file `IKB_MASTER_83`.dmp from the local directory, and store in an Oracle Directory on the server. The default (and recommended) directory is called DATA_PUMP_DIR, and is often available under /app/oracle/admin/INSTANCE/dpdump. Note that if you import data from another existing database, the file may have any other name. This command will run as the iKnowBase-user.

The second command will import the file `IKB_MASTER_83`.dmp from the Oracle Directory into the iKnowBase schema. The iKnowBase program will in fact use Oracle Datapump to perform this import. For the datapump import to succeed, the name of the database user that exported the schema must be specified. In the distribution, and by convention, the name of the datafile reflects the name of the exporting user; here, it is `IKB_MASTER_83`.

If something fails during import, Oracle Datapump will store log messages in a log file in the Oracle Directory on the database server. When using the `importFile` command, the name of the logfile is always the same as the name of the datafile, with a .log-suffix.

Since the file already exist on the database server, you may view it there (i.e. typically /app/oracle/admin/INSTANCE/dpdump/`IKB_MASTER_83`.log). You can also use the following command to download the logfile to your local directory:

```
$ ./iknowbase.sh downloadFile IKB_MASTER_83.log .
```

Note that it is often useful to store a copy of the logfile even when there are no apparent failures, in case you need it later.

## 5.2. Upgrade

Upgrading an iKnowBase-installation is technically more complex than a fresh install, mostly because there are already existing data that must not be deleted. An upgrade therefore have the following steps:

- If you want, take a copy of existing database object definitions for post-upgrade troubleshooting.
- Next, update and verify the user/schema settings.
- Run through schema upgrade scripts for all required versions, and install the latest code (types, packages, functions and procedures).
- Recompile any invalid packages.
- Import any new master-metadata.

```
$ cd /opt/iknowbase/production
$ iknowbase.sh exportSource scripts.zip
$ iknowbase.sh configureUser
$ iknowbase.sh upgradeAll
$ iknowbase.sh compileInvalid
```

### 5.2.1. Export existing scripts

Export existing scripts is entirely optional, but we recommend this for easier post-upgrade troubleshooting. You may use any available tool for this process, but once again the iKnowBase program has an easy-to-use mechanism:

```
$ iknowbase.sh exportSource scripts.zip
```

The above command will use DBMS_METADATA to recreate scripts for all TYPEs, PACKAGEs, PROCEDUREs and FUNCTIONs in the iKnowBase schema, and store it in a zip file. The zip-file will also contain compile-scripts for each of the object types, as well as a compile script that compiles everything.

### 5.2.2. Prepare the database schema

Various versions of iKnowBase require different permissions, and have different information in the IKB_GLOBAL_PREFS-package. It is therefore necessary to configure the database schema to the new requirements:

```
$ iknowbase.sh configureUser
```

This command will perform two actions:

- First, as user SYS, it will grant required permissions to that user. A full list of permissions can be

seen in the log file after executing the command.

- Then, as the iKnowBase user, it will recreate the database package IKB_GLOBAL_PREFS with default values.

### 5.2.3. Upgrade schema and install latest code

The most complex step in the upgrade process is to upgrade the schema and install the latest code. For convenience, use the iKnowBase program's upgradeAll feature:

```
$ iknowbase.sh upgradeAll
```

This command does in fact comprise a number of schema upgrade steps (one for each schema version), and then a single code installation step.

Note that after installing the latest code, open database connections and open cursors may cache database type information that is no longer correct. It is therefore recommended to restart all application servers, email readers, search crawlers, etc. that might have open database connections.

### 5.2.4. Recompile invalid packages

After the upgrade step, it may be required to recompile invalid packages in the Oracle schema:

```
$ iknowbase.sh compileInvalid
```

This command utilizes DBMS_UTILITY.RECOMPILE_SCHEMA for recompiling only invalid packages.

### 5.2.5. Import new master metadata

Go to /ikb$console/development/advanced/importjobs#/ and upload the EXP-IKB_MASTER_<version>.dmp file from the etc/ folder in the iKnowBase zip file and run the import. It will add or change any new metadata considered default/required, such as attributes, doctypes, valuelists, ACLs etc.

## 5.3. Clone an installation

Often, you will want to duplicate an existing installation. For example, you may have a development, test and production instance, and you may need to copy from production to dev and test, either for the initial setup, or after a while to make the dev/test-environment prod-like with production data. Note that some of the steps are described in more detail in the surrounding sections about installation and upgrading, so it's kept sparse here.

| | |
|---|---|
| **NOTE** | For simplicity, the target environment is called 'dev' from here on. |

### 5.3.1. Copy data from dev

Unless you are creating the dev environment for the first time, you must make sure you're not loosing any data when overwriting.

Ensure there is no new data or code/changes in dev that only exist there (it will be deleted), and ensure you have a backup of dev.

Then export the dev-specific data:

- Export the table IKB_INSTALLATION_PROPERTIES (e.g. with SQL Developer into an .sql file).
- Make a note/screenshot of the iKB Domain configuration (http://hostname/ikb$console/development/advanced/domains).
- Find and write down or take a screenshot of any differences between prod and dev to users, groups, ACLs, scheduled jobs, events, email setup, newsletters, LDAP config, etc.
- Write it down if the dev schema has any other/fewer grants (see SQL below) or *Network access lists* entries than prod.
- Explore the solution to check if there is any other way than application/installation properties that change the way dev operates (e.g. to prevent it from sending out scheduled emails).

### 5.3.2. Export the prod schema

Export the iKnowBase schema from the prod app server, by (optionally) stopping the service, entering the installation directory and running `exportFile` to dump the schema, then `downloadFile` to transfer it from the DB server into the current directory.

The dump will contain everything except some log tables and the environment specific IKB_INSTALLATION_PROPERTIES table.

```
$ service ikb-production stop
$ cd /opt/iknowbase/production
$ ./iknowbase.sh exportFile <filename>
$ ./iknowbase.sh downloadFile <filename> .
```

### 5.3.3. Import the prod schema into dev

On the dev app server, change to the installation directory of the dev environment, then:

- Stop the dev service (e.g. `service ikb-development stop`)
- Delete the iKnowBase schema with `./iknowbase.sh dropUserCascade` (see *De-installation* for details).
- Create a blank schema to import into with `./iknowbase.sh createUser`
- Upload the prod dump with `./iknowbase.sh uploadFile <filename>`
- Import the prod dump to dev with `./iknowbase.sh importFile <filename> <prod ikb schema name>`

| | |
|---|---|
| **TIP** | If you use plain Oracle Datapump (impdp/expdp) instead of the `iknowbase.sh` scripts to move a schema, be aware that the user may have a custom profile and table space that are not included in the dump, causing the dump's initial create user statement to fail on import. In those cases, run the create user statement manually, then run the import. |

### 5.3.4. Handle any custom schema

If the solution has a custom schema in addition to the iKnowBase schema, inspect it to see if it contains data that must be cloned.

- Cloning a custom schema must be done with regular Oracle Datapump (impdp/expdp). It is somewhat described in the section *Move an installation*.
- Check if there is any configuration (in tables or constants in a PLSQL package) that is supposed to be different in prod and dev, and take a copy of it in dev.

If you are replacing an existing schema, you should delete it before importing. Note that you may have to delete any queue tables before deleting the schema:

```
select distinct queue_table from user_queues;
execute dbms_aqadm.drop_queue_table('TABLE_NAME', true);
```

### 5.3.5. Restore dev-specific data and settings

First, using the SQL client of your choice, import into the dev schema the IKB_INSTALLATION_PROPERTIES table you exported, then run on the dev app server:

```
./iknowbase.sh createGlobalPrefs
```

Restore network ACL entries if any (e.g. for network access or mail sending), see *Network access lists*.

Restore grants for the iKnowBase schema and any custom schema. To list grants, run something like this SQL. If you run it in the old dev, or current prod, you can use the SQL grant statements it creates, in the new dev. If there are any custom schema, run as sys or run it for and as each user to get all the grants.

Restore database links.

If Solr is deployed as a search backend, you should reindex all documents to ensure consistency now that the document collection is changed. It can be done by going to /ikb$console/development/advanced/events, and for all the Indexing Solr events, go to the "index queue status" tab and click the buttons there. To be sure, empty the index, then reindex all documents.

```
select user_tab_privs.*,
'grant ' || privilege || ' on ' || owner || '.' || table_name || ' to ' || grantee ||
';' as sql
from user_tab_privs where grantee in ('IKB', 'IKB_CUSTOM') -- Modify as needed
order by grantee, owner, table_name, privilege;
```

After this, recompile any invalid packages in the schema(s) if needed, now that grants are ok. Then start the iKnowBase dev service.

Enter the iKnowBase console and:

- Restore the iKB Domain configuration (ensure one domain is always set as the default, else you must edit the IKB_PORTAL_PREFERENCES table to fix it).
- Restore any other dev specific configuration you took note of.

Finally: Test all the things.

## 5.4. Move an installation

Moving an installation to a different server is similar to installing a fresh installation, but there are many things to take care of. This guide tries to list all things to take care of when moving everything to a new hosting environment, both a new database server and new application server. If it is only the database schema(s) that is moving to a new DB server, many of the points listed below may be ignored.

| | |
|---|---|
| **NOTE** | Before starting, check whether the different iKnowBase environments (dev/test/prod) are in different databases or in the same. If they are to be moved from several databases into one, they cannot have the same schema name, which they might have when on several databases. Changing a schema name when moving is possible, but might require rewriting code. |

| | |
|---|---|
| **TIP** | There may be several things you must fix, outside of iKnowBase itself. Especially if the new server has a much newer OS version or different distro. E.g. Oracle version (iKnowBase version may be incompatible), Java version (iKnowBase version may be incompatible), version of Apache or other reverse proxy, if used (old config may be incompatible), startup scripts (init vs systemd), email-sending, any integrations. |

| | |
|---|---|
| **TIP** | If the solution has integrations or scheduled jobs for e.g. email, be aware what happens when you start up then new instance. It might end up doing duplicate work that cause problems (e.g. sending out lots of emails, again). |

First step, install the iKnowBase executables on the new server as described in the above section on installing.

Then, for every environment on the old servers:

- Stop iKnowBase.

- If there are custom schemas, export them with Oracle Datapump (`expdp`).
- Export the IKB_INSTALLATION_PROPERTIES table (e.g. with SQL Developer into an .sql file).
- If needed, export log tables the same way, they are not included in the regular iKnowBase export.
- Export the iKnowBase schema.

```
$ cd /opt/iknowbase/production
$ service production stop
$ ./iknowbase.sh exportFile <filename>
$ ./iknowbase.sh downloadFile <filename> .
```

From the old application server, copy to the new application server:

- The exported database dump(s).
  - Tip: In case of SSH access to the DB servers, the dump files can be moved directly from old DB server datapump dir to new DB server datapump dir, to save time moving them through the app servers. If so, ignore the downloadFile/uploadFile commands.
- The entire application folder (consider purging logs first).

On the new application server:

- Create/copy startup scripts (e.g. /etc/init.d).
- Modify application.properties to fit the new environment. Typical changes:
  - DB address/name, sys credentials, ikb credentials if changed
  - Default tablespace name
  - JDK path and options
  - File paths, e.g to static resources, certificates, license files, integrations
- Verify the schema settings, e.g. with `./iknowbase.sh testSysConnection`
- Create blank iKnowBase schema/user.
- Upload the iKnowBase dump file and import it into the empty schema.
- Recompile any invalid packages.

```
$ cd /opt/iknowbase/production
$ vim application.properties
$ ./iknowbase.sh testSysConnection
$ ./iknowbase.sh createUser
$ ./iknowbase.sh uploadFile <filename>
$ ./iknowbase.sh importFile <filename> <old schema name>
$ ./iknowbase.sh compileInvalid
```

On the new DB server:

- Import custom schemas with datapump, if any (see below).
- Copy the table IKB_INSTALLATION_PROPERTIES from the old server, it is not included in the export. Update its properties if needed (URLs etc). Then run `./iknowbase.sh createGlobalPrefs`.
- Restore missing grants and network ACL entries (see SQL below).
  - Grants between the iKnowBase schema and any custom schemas.
  - Grants to system resources, for both the iKnowBase schema and any custom schemas.
  - Network ACL entries if the solution is sending mail, using LDAP or has other networking needs.
- Recompile packages if needed, after grants are in place.
- Enable AQ queues if needed.

If you have done all the steps, but experience issues with missing/invalid DB objects or rights, running `./iknowbase.sh configureUser` or `./iknowbase.sh upgradeAll` might help. But note that upgradeAll will overwrite all standard iKnowBase PLSQL packages. Even though it is bad practice, some solutions have edits to these files, which will be overwritten. Check the documentation and database source repository of the project to look for such cases.

How to import with datapump varies, but here's an example. Note that you may have to remap tablespaces, if the old DB had custom IKB tablespaces and the new one does not.

```
export ORACLE_HOME=/u01/app/oracle/product/18.0.0/dbhome_1
export NLS_LANG=NORWEGIAN_NORWAY.AL32UTF8
$ORACLE_HOME/bin/impdp "system/PASSWORD@pdbdev" directory=DATA_PUMP_DIR dumpfile=dump-
ikb_custom.dmp remap_tablespace=ikb_data:users remap_tablespace=ikb_indexes:users
schemas=ikb_custom
```

To list grants, run something like this in prod as sys to create the SQL grant statements (if you're not sys, you must run it for and as each user if there are more than one):

```
select user_tab_privs.*,
'grant ' || privilege || ' on ' || owner || '.' || table_name || ' to ' || grantee ||
';' as sql
from user_tab_privs where grantee in ('IKB', 'IKB_CUSTOM') -- Modify schema list as
needed
order by grantee, owner, table_name, privilege;
```

Finally:

- Search through all iKnowBase settings and sources (PL/SQL, iKnowBase templates) for URLs, IPs, schema names that must be updated, if any.
- Update the iKnowBase domain configuration if needed (/ikb$console/development/advanced/domains).
- Ensure DB/iKB queues and scheduled jobs are enabled as in the old environment.

- Copy any cronjobs from the old application server.
- Modify your hosts-file to access the new application instance by its real domain name.
- Test all the things.
- Change DNS to point at the new server.

## 5.5. De-installation

De-installation of the iKnowBase installation is pretty simple: Remove the user and all it's data:

```
$ iknowbase.sh dropUserCascade
```

Note that you may have to set the value "allowDropUserCascade=true" in the property file before this command will work.

Note also that while a simple "drop user cascade" from sql*plus may work, it also may not: When a schema has Oracle AQ-tables (Advanced Queing), it is sometimes required to manually drop these queues first. The iKnowBase program handles this, and is therefore the recommended way of deleting a user.

And finally, a word of warning: The dropUserCascade command is utterly unrecoverable, and if you drop a user by accident, you will have to reload data from a backup. Take care!

## 5.6. Advanced topics

### 5.6.1. Global runtime preferences

iKnowBase uses a database package header, IKB_GLOBAL_PREFS, to determine methods and options on certain functionality.

Changing settings is a two-step process:

- First, update changed values in the `ikb_installation_properties`-table, using a database tool or the console page on /ikb$console/development/advanced/installation/property. Use an instance qualifier of `ikb_global_prefs` for these settings.
- Next, run the `createGlobalPrefs` command with the iKnowBase setup program. This will recreate the database package with the updated settings. Note that `configureUser` also will run `createGlobalPrefs`.

Note that you can also add your own variables to this configuration, for example to identify a test or production environment. Any variable set with the `ikb_global_prefs` will be reflected in the database package, and can be used in your own code for whatever purpose you deem necessary.

| Property name | Default Value | _Description |
|---|---|---|
| has_oracle_oid | FALSE | Set to TRUE if users are maintained and integrated from an LDAP directory. When FALSE, the user is maintained fully in iKnowBase. |
| has_oracle_smtp | FALSE | When FALSE, use a Java based mail sender which is the preferred method when sending email. Oracle SMTP uses internal Oracle functions, and they may not be available on all versions of the Oracle Database. |
| has_oracle_http | TRUE | Oracle HTTP may not be available on all versions of the Oracle Database. |
| log_level | ERROR | Defines the log level for iKnowBase when catching errors and debug messages. Valid values are ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL or OFF. |
| has_custom_access_control | FALSE | If the instance is running without a custom access control, the value should be set to FALSE. Will improve performance. |
| attribute_security_function | blank string | Defines the name of the custom function used for attribute security. Set to NULL if no such function exists. If the function exists within a package, use the notation package.function |
| oracleTextSearchParser | blank string | Defines the name of the custom oracle text search parser. Set to NULL if no such function exists. If the function exists within a package, use the notation package.function |
| ignore_illegal_attrib_on_save | FALSE | Used for backward compatibility. |
| use_native_plsql_compiler | TRUE | For Oracle Database 11g and up: use native compilation for PL/SQL code. |
| index_binary_using_oratext | TRUE | Defines whether the oracle text index should contain binary content. Set to FALSE to index only metadata and text content, or leave as TRUE to index both metadata, text content and binary content. |
| index_metadata_using_oratext | TRUE | Defines whether the oracle text index should contain metadata values. Set to FALSE to index disable metadata aggregation and indexing, or leave as TRUE to index the metadata. |
| compat_save_text_as_htmlencode | FALSE | Compatibility flag. Defines whether to HTML-encode text before saving. For compatibility only. Should be FALSE for new installations |

### 5.6.2. Running iKnowBase in an Oracle Enterprise Edition database

By default, an iKnowBase installation is prepared for running in an Oracle Standard Edition database. If you are licensed for running Oracle Database Enterprise Edition then run the following

command to speed up queries displaying (or sorting) document popularity.

```
$ iknowbase.sh dbscript common/mv_log_document.sql logfile
```

To switch back to the standard edition version do:

```
$ iknowbase.sh dbscript common/view_log_document.sql logfile
```

### 5.6.3. Running iKnowBase in an Oracle Cloud autonomous database

When running iKnowBase in an Oracle Cloud autonomous database, make note of the following configuration properties.

Connections are managed with certificates in a wallet, and the datasource url looks a bit different from what it typically is. This is nothing special for iKnowBase, so regular JDBC documentation is applicable.

```
# Example connection string to "iknowbase" database
spring.datasource.url=jdbc:oracle:thin:@iknowbase_medium?TNS_ADMIN=C:/Users/first.last
/wallets/iknowbase_wallet/
```

Oracle Cloud autonomous databases does not provide a sysdba user, instead configure the "admin" user like this:

```
com.iknowbase.setup.database.sysUser=admin
com.iknowbase.setup.database.sysPassword=<admin-password>
com.iknowbase.setup.database.sysPrivilege=normal
```

### 5.6.4. Recreating Oracle Text index

In former distributions, the Oracle Text index would index title, metadata, description, binary- and text content. With SOLR, a text index with Oracle Text should be obsolete, but for some reasons it is still useful for compatibility reasons, but the overall goal is to convert all search functionality to SOLR. Meanwhile, you can create a lightweight Oracle Text index without the binary content which will reduce the index size dramatically and still be able to search. To do this you need to run the following two steps:

```
$ Change ikb_global_prefs.index_binary_using_oratext to FALSE
$ iknowbase.sh dbscript common/create_ikb_freetext.sql logfile
```

To switch back to a full index with also binary content indexed do:

```
$ Change ikb_global_prefs.index_binary_using_oratext to TRUE
$ iknowbase.sh dbscript common/create_ikb_freetext.sql logfile
```

### 5.6.5. Configuring network access lists

Some integration features, such as LDAP sync and sending email, open network connections from the database to other hosts. In a default Oracle database configuration this is not allowed, leading to the error message: `ORA-24247: network access denied by access control list (ACL)`.

To fix, create a network access list that allows the iKnowBase database schema access (use iKnowBase schema for *principal_name*, and required hostname for *host*).

To list network access lists for a schema, run:

```
select * from dba_network_acls join dba_network_acl_privileges using (acl, aclid);
```

To grant network access, run:

```
BEGIN

  -- RESOLVE: To allow looking up hostnames. Often not needed.
  --          Specify server name to look up.
  DBMS_NETWORK_ACL_ADMIN.append_host_ace (
    host       => 'target-host.example.com',
    ace        => xs$ace_type(privilege_list => xs$name_list('resolve'),
                              principal_name => 'IKB_PRODUCTION_SCHEMA_EXAMPLE_NAME',
                              principal_type => xs_acl.ptype_db));

  -- SMTP: To allow sending email from the database, to a server on a given port.
  --          Specify server to send via, as well as port range.
  DBMS_NETWORK_ACL_ADMIN.append_host_ace (
    host       => 'target-host.example.com',
    lower_port => 25,
    upper_port => 25,
    ace        => xs$ace_type(privilege_list => xs$name_list('smtp'),
                              principal_name => 'IKB_PRODUCTION_SCHEMA_EXAMPLE_NAME',
                              principal_type => xs_acl.ptype_db));

  -- CONNECT: To allow arbitrary connections to a server, e.g. for LDAP-sync.
  --          Specify server to connect to, as well as port range.
  DBMS_NETWORK_ACL_ADMIN.append_host_ace (
    host       => 'target-host.example.com',
    lower_port => 443,
    upper_port => 443,
    ace        => xs$ace_type(privilege_list => xs$name_list('connect'),
                              principal_name => 'IKB_PRODUCTION_SCHEMA_EXAMPLE_NAME',
                              principal_type => xs_acl.ptype_db));
END;
```

# Chapter 6. Java applications

## 6.1. Overview

The server side components of iKnowBase are assembled as a single file.

- iknowbase-boot-embedded-8.3-app.jar for running with the embedded iKnowBase Web Server.

The main iKnowBase Java web modules, which may be enabled or disabled in the configuration, are:

- Batch: module is the batch processing server.
- Instant: module is the real time asynchronous messaging server for web clients.
- Studio: module adds content administration, user administration and development capabilities to the Administration Console.
- Viewer: module is the core runtime environment, serving web pages to end users.
- WebServices: module is the Web Services-server, typically to programs and integration engines.

See the module's own chapter for details.

## 6.2. Special requirements

- We strongly recommend deploying iKnowBase with context root set to "/".

## 6.3. Installing plugins and patches

### 6.3.1. iKnowBase Web Server (embedded)

Jar plugins in the `./plugins` directory (subdirectories are not scanned) are automatically included in the application.

If you want to change or add more directories containing jar plugins, you may configure this using

- Linux: Set `loader.path` in application.properties (recommended) or JVM option, or set `LOADER_PATH` environment variable (use only one method).
- Windows: Set `loader.path` JVM option or set `LOADER_PATH` environment variable.

The `loader.path` / `LOADER_PATH` is a comma delimited list of jar files or directories containing jar files. Note that it does not support relative parent directory "../".

## 6.4. Install/upgrade

For the iKnowBase web server, installation is quick and mostly automatic:

- Install and configure the iknowbase instance (software + property file). This is a required step to

install the database, so it is probably done.

- If this is your first installation (not an upgrade), you may want to enable a web password for the default ORCLADMIN admin user.
- Start the webServer.

```
$ cd /opt/iknowbase/production
$ ./iknowbase.sh setIkbPassword orcladmin secretpassword
$ ./iknowbase.sh webServer
```

When the server has started, you should be able to navigate to a few interesting links:

- Documentation
- Management Console (requires login)

For further configuration options, see the details:

- *Installation Guide > iKnowBase web server*

# 6.5. Web application security

During or after deployment, you need to configure security for the application (NOT the same as content security).

See *Installation Guide > Web Application Security* for details and examples.

# 6.6. Advanced topics

## 6.6.1. Deploy with alternative context root (/ikbViewer)

In the default and recommended installation, the iKnowBase web application is deployed to "/", so that you can access pages and components without an application-specific prefix. For compatibility reasons, the server will automatically also handle all requests to "/ikbViewer/**", and redirect them to "**/".

While it is technically possible to deploy to a different root (e.g. "/ikbViewer"), **we do not recommend this**. It brings no apparent benefit, but the user experience is negatively impacted. That said, should you choose to do so:

- Deploy to any chosen path.
- Update paths specified in the "Domains" definitions in Development Studio.
- Ensure the infrastructure (reverse proxy) maps the default static resource paths to the updated paths.
  - /ressurs ⇒ /NEW_PATH/ressurs
  - /plugin-resources ⇒ /NEW_PATH/plugin-resources

### 6.6.2. Clustering

The iKnowBase application itself supports clustered deployment, but the "Instant" module does not. If you intend to use this module in an otherwise clustered environment, it must be deployed to a standalone server instance with no clustering.

By default, iKnowBase runs with database persisted sessions. This means that session persistence / sticky sessions are not required. If you change the session configuration to memory sessions (Spring Session or native), you must use session persistence / sticky sessions.

### 6.6.3. Add custom locations for additional static resources

You may configure the iKnowBase application to serve additional static resource locations, typically used for customer specific files (css, js, images).

See options prefixed with `com.iknowbase.server.resource.custom` in application.properies.SAMPLE for configuration details.

### 6.6.4. Limit file upload size

iKnowBase does not by default limit the size of files you may upload. If you want to set a max limit, use Spring Boot's `multipart.*` configuration options.

```
# Max file size (-1 = unlimited)
#multipart.max-file-size=-1
multipart.max-file-size=50MB

# Max total request size (-1 = unlimited)
#multipart.max-request-size=-1
multipart.max-request-size=100MB
```

### 6.6.5. Session configuration

iKnowBase uses Spring Session with database persistence by default. This means that the application itself handles all things related to sessions and does not use the session features provided by the server platform.

All requests use this session setup, except requests for public static resources in /ressurs, /plugin-resources, or any custom-resources you have defined.

The advantages with this setup are:

- The session is still valid after server restart.
  - The users can continue without login interruption.
- No session replication setup is required for seamless failover in a clustered environment.
- Sticky sessions are not required and you may load balance the application using any algorithm you want.

    - NOTE: The Instant module does not support clustering.
- No overhead for static resources.
- Possible to force logout a user by deleting the persisted session.

The disadvantages are:

- A couple of milliseconds overhead for each request to non-static content.
- Minor extra load on database.

See the sub sections below for configuration details.

#### Use native sessions

If for some reason you need to use native sessions provided by the application server, you may disable Spring Session by setting

```
com.iknowbase.spring.session.enabled=false
```

In native session mode, refer to documentation for Spring Boot and the application server for configuration options for sessions. All other sections in this document regarding sessions describe configurations applicable for Spring Session.

#### Specify session cookie domain

By default, the iKnowBase separates session between different domains, so that a user that opens both http://www.example.com and http://intranet.example.com will have to log on to each of these sites. Technically, this is done by using different "session cookie domains" for the session cookie for each of these hosts.

Sometimes, it is desirable to share the session information between sites. To do that, configure the session cookie domain to be used by the server. It is important that this domain is the parent domain of the various hosts used, for example, to share the login session between http://www.example.com and http://intranett.example.com you must specify "example.com"; to share the login session between http://www.iknowbase.com and http://customers.iknowbase.com you must specify "iknowbase.com".

```
com.iknowbase.spring.session.domainName=example.com
```

If you want to support more than one domain name at the same time, you can use a domain name expression instead of the static domainName value.

Example:

- http://www.example.com, http://intranett.example.com with shared cookie for example.com
- http://www.iknowbase.com (separate domain)
- http://localhost:8080 (for testing)

```
# Will match example.com for requests to www.example.com and intranett.example.com.
# Will match iknowbase.com for requests to www.iknowbase.com
# Will not match localhost and for localhost it will not override the cookie domain
name either.
# Note: Double backslash is used in this regular expression configuration
com.iknowbase.spring.session.domainNamePattern=^.+?\\.(\\w+\\.[a-z]+)$
```

You must choose between `com.iknowbase.spring.session.domainName` and `com.iknowbase.spring.session.domainNamePattern`. Configuring both will result in startup error.

#### Specify session cookie name

The iKnowBase application will use session cookie name `IKBSESSION`. In case of session name collision you may change the session cookie name. Session cookie name can be set using:

```
...
com.iknowbase.spring.session.cookieName=JSESSIONID_CUSTOM_NAME
...
```

#### Specify session cookie max idle time

Default max session idle time is set to 2100 seconds / 35 minutes. This can be changed by setting:

```
com.iknowbase.spring.session.maxInactiveInterval=<NUMBER IF SECONDS>
```

#### Avoiding session cookie collision

You may encounter a session cookie collision if you deploy two iKnowBase applications on the same host on different ports or otherwise have additional applications with the same cookie name running behind a load balancer.

As the host name is the same for both requests, the session cookie name will by default be valid for both requests. The session cookie is only valid on one server, which will result in a replaced session cookie and effective log out the authenticated user.

Resolve the issue by changing the session cookie name for one of the applications or use different hostnames for accessing the applications.

# Web Application Modules

# Chapter 7. The web application runtime module

The web application runtime module is the infrastructure upon which all modules rely. This module is mostly "invisible" to users and administratiors alike, but it is possible to configure certain aspects of it.

## 7.1. The in-memory cache manager

The iKnowBase system provides a cache system based on Ehcache (http://www.ehcache.org). This cache helps in maintaining the performance of the system, by storing both metadata and content in memory, saving costly disk and database access.

The caches in iKnowBase are replicated betweeen servers, ensuring that all applications connected to the same repository see the same data. The mechanism used to communicate between the caches is specific to EHCache, and can be configured using the properties shown below.

For information on the meaning of these properties, see the EHCache Product Documentation.

The *CacheManagerConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.cache.EHCache.enableReplication | Enable or disable cache replication. |
| com.iknowbase.cache.EHCache.peerDiscovery | Sets the properties used by Ehcache to discover other nodes. |
| com.iknowbase.cache.EHCache.listenerProperties | Sets the properties used by Ehcache node for cache replication. |
| com.iknowbase.cache.EHCache.sizeOfPolicyMaxDepth | Sets the max depth to traverse during sizeOf statistics before triggering sizeOfPolicyMaxDepthExceededBehavior. |
| com.iknowbase.cache.EHCache.sizeOfPolicyMaxDepthExceededBehavior | CONTINUE (default) or ABORT when sizeOfPolicyMaxDepth is reached. |
| com.iknowbase.cache.EHCache.sizeOfFilterConfigurationFile | Path to an EHCache SizeOf Filter configuration file. Defaults to included filter. |

## 7.2. The SecureToken engine

For most security purposes, iKnowBase uses the secure session mechanisms supported by the

underlying application server. However, there are situations when this session is not available to all clients, and iKnowBase then needs an alternative, secure way of passing identity information. This token is generated using the HMAC_SHA1 message authentication algorithm, based on a secret key along with user identity and time information.

iKnowBase needs to generate a secure token for login information in the following scenarios:

- Client authentication integration between iKnowBase Viewer and iKnowBase Instant.
- Identity-forwarding to the Solr search engine.

By default, a suitable key is automatically generated when the application server starts up. However, for load balanced scenarios with multiple application servers, this would result in different keys and spurious failures during data load. You may then manually specifiy a secret key in the configuration. For optimum security, please use a long phrase with multiple words, numbers and special characters.

The SecureTokenEngineConfiguration_ accepts these configuration properties:

| **Property name** | **Description** |
|---|---|
| com.iknowbase.secureTokenEngine.secureKey | Key value used for generating tokens, when required. |

# Chapter 8. Batch module

iKnowBase comes with a batch module used for processing certain off-line and near-line tasks, such as email processing and file format conversion.

Currently, the Batch Server handles these services:

- The ContentIndexer service submits documents to a Solr search index for indexing.
- The EmailSender service sends emails.
- The PageEngine service runs a specific iKnowBase Page and returns the HTML result. Used by the newsletter module.
- The Transformation service supports various transformation operations on binary data through a set of providers.

The batch server is enabled by default, but may be disabled as needed. In particular, a larger site with multiple servers might want to disable batch processing on the servers handling public traffic. Disabling the batch module will implicitly also disable all services described below.

The *BatchServerConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.batch.enabled | Toggles whether the batch server modules are available. |

## 8.1. ContentIndexer

When using the Solr search engine for content search, the iKnowBase database repository will send indexing requests to the batch server, which will then submit content to the Solr server for actual indexing.

The *ContentIndexerConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.batch.contentIndexer.enabled | Toggles whether to start the contentIndexer. The legal values are either "true" or "false". |
| com.iknowbase.batch.contentIndexer.dequeTimeoutSeconds | Number of seconds each dequeue() shall wait before recycling. |
| com.iknowbase.batch.contentIndexer.spawnPolicy | Decides when the pageEngine starts listening for a new message. Use "immediate" for parallel processing, or "delayed" for serial processing. |

iKnowBase is capable of submitting content to multiple search engines for indexing. The Solr Configuration specified in the Development Console refers to a logical "Search engine name", which is used to look up a client for this engine. See the *SearchClientConfiguration* section for details on

how to configure these connections.

## 8.2. EmailSender

EmailSender is the preferred method for sending emails and is set as default for new installations. An alternative method is available through the iKnowBase repository, see IKB_GLOBAL_PREFS.

The EmailSender configuration is performed using configuration properties, where you define profiles and settings used for sending emails. Whether emails are sent using this service or sent using iKnowBase Repository is controlled by iKnowBase Global Preferences in the iKnowBase Repository.

The *EmailSenderConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.batch.emailSender.enabled | Toggles whether to start the server. The legal values are either "true" or "false". |
| com.iknowbase.batch.emailSender.dequeTimeoutSeconds | Number of seconds each dequeue() shall wait before recycling. |
| com.iknowbase.batch.emailSender.spawnPolicy | Decides when the server starts listening for a new message. Use "immediate" for parallel processing, or "delayed" for serial processing. |

## 8.3. Transformation Server

The Transformation Server provides a database accessible entry point to the Content Transformation Service.

The TransformationServer is reachable through the database package IKB_TRANSFORMATION_API.

See *Content Transformation Providers* for provider specific installation instructions.

The *TransformationConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.batch.transformation.enabled | Toggles whether to start the server. The legal values are either "true" or "false". |
| com.iknowbase.batch.transformation.dequeTimeoutSeconds | Number of seconds each dequeue() shall wait before recycling. |
| com.iknowbase.batch.transformation.spawnPolicy | Decides when the server starts listening for a new message. Use "immediate" for parallel processing, or "delayed" for serial processing. |

| Property name | Description |
| --- | --- |
| com.iknowbase.batch.transformation.replyMessageExpirationSeconds | Number of seconds each reply message shall be valid, before expiring. |

## 8.4. PageEngine

The ikbBatch module contains a page engine server, which can be configured to listen for page rendering requests. This is used by for example the newsletter module, which asks the batch module to render a page to be used as the content.

The *BatchPageEngineConfiguration* accepts these configuration properties:

| Property name | Description |
| --- | --- |
| com.iknowbase.batch.pageEngine.enabled | Toggles whether to start the server. The legal values are either "true" or "false". |
| com.iknowbase.batch.pageEngine.dequeTimeoutSeconds | Number of seconds each dequeue() shall wait before recycling. |
| com.iknowbase.batch.pageEngine.spawnPolicy | Decides when the pageEngine starts listening for a new message. Use "immediate" for parallel processing, or "delayed" for serial processing. |
| com.iknowbase.batch.pageEngine.replyMessageExpirationSeconds | Number of seconds each reply message shall be valid, before expiring. |

# Chapter 9. Development Studio module

iKnowBase comes with a Development Studio used to develop and maintain applications. This module is enabled by default, but may be disabled as needed. In particular, a larger site with multiple servers might want to disable Development Studio on the servers handling public traffic.

The *StudioConfiguration* accepts these configuration properties:

| Property name | Description |
| --- | --- |
| com.iknowbase.studio.enabled | Toggles whether the development studio administration options are available in the Administration Console. |

# Chapter 10. Instant module

iKnowBase comes with client and server side support for creating applications with real time asynchronous messaging support.

See the *iKnowBase Development Guide > Using iKnowBase Instant* for concept and examples.

## 10.1. Installation

The Instant Server is disabled by default, but can be enabled and configured using configuration properties.

Make sure that CORS is enabled in configuration id the clients use Instant on a different name or port than what is used for fetching content from iKnowBase Viewer.

Make sure you also configure *The SecureToken engine* to enable web client single sign on between iKnowBase Viewer and Instant.

### 10.1.1. Special requirements

The connected web-clients will be "always connected" using asynchronous HTTP transport mechanisms and will, compared to traditional HTTP clients, need infrastructure with support for non-blocking I/O or a sufficient high number of supported concurrent connections. One Instant client subscription means one network connection. ikbInstant may be deployed on an application server separate from where the other iKnowBase applications are deployed, as long as it's connected to the same iKnowBase database repository.

If Cross Origin Resource Sharing (CORS) is in use, the CORS address MUST use the same protocol as the webpage containing the JavaScript client. If the web page uses HTTPS, then the CORS address must also be HTTPS.

## 10.2. Configuring the Instant module

The Instant Server is disabled by default and must be explicitly enabled using configuration properties. The *InstantServerConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.instant.enabled | Toggles whether the instant module is enabled. |
| com.iknowbase.instant.suspendTimeoutLP | How long in milliseconds a Long Polling connection is suspended before the server resumes the connection. This will trigger a reconnect by the client. |
| com.iknowbase.instant.enableCORSFilter | Server side support for Cross Origin Resource Sharing (CORS). The legal values are either "true" or "false". |
| com.iknowbase.instant.cleanupTopicThreshold | How often the disconnect cleanup maintenance thread looks for disconnected clients. |

| Property name | Description |
|---|---|
| com.iknowbase.instant.cleanupDisconnectedConnectionsThreshold | How long a connection needs to be in disconnect inactivity queue before it is examined and validated. |
| com.iknowbase.instant.broadcastDelayUserListUsers | Delay between a client subscribe and the issued broadcast for userList is. Must be large enough to allow the client to complete the handshake and enter suspend mode. |
| com.iknowbase.instant.broadcastDelayUserListJoin | Optional delay between a detected join and the issued broadcast. Default no delay. |
| com.iknowbase.instant.broadcastDelayServerRequest | Optional delay between a serverRequest and the issued broadcast. Default no delay. |

**NOTE** Make sure you also configure the SecureTokenEngine to enable authentication for web clients

### 10.2.1. InstantQueueServerConfiguration

The Instant Queue Server is the unit responsible for consuming messages published using Instant's PL/SQL API and delivering them to the specified topic where all web clients are connected. This queue listener can of course be configured through The *InstantQueueServerConfiguration* which accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.instant.aq.enabled | Toggles whether AQ messages is processed by this instance at all. The legal values are either "true" or "false". |
| com.iknowbase.instant.aq.dequeTimeoutSeconds | Number of seconds each dequeue() shall wait before recycyling. |
| com.iknowbase.instant.aq.spawnPolicy | Decides when the AQ server starts listening for a new message. Use "immediate" for parallel processing, or "delayed" for serial processing. |

## 10.3. Testing and troubleshooting

### 10.3.1. Administration console

For developers, the Administration Console provides monitoring and testing details for Instant.

- instant > topics: A list of all topics.
- instant > topics > [topicName]: Detailed information about the specific topic, including all connected clients.
- instant > test: Test page for Instant Server.

### 10.3.2. Session cookie collision when Instant is deployed to a separate instance with CORS

You may encounter a session cookie collision if you

- Deploy iKnowBase with Viewer module to an iKnowBase Quickstart server (e.g. host1:443).
- Deploy iKnowBase with Instant module to a separate iKnowBase Quickstart server on the same host (same hostname seen by web client) and a different port (e.g. host1:8443).
- Use CORS from javascript client.

As the host name is the same for both requests, the session cookie name will by default be valid for both requests. The session cookie is only valid on one server, which will result in a replaced session cookie and effective log out of the authenticated user.

Resolve the issue by changing the session cookie name for one of the applications or use different hostnames for accessing the applications.

# Chapter 11. Viewer module

The viewer module is the main runtime component of iKnowBase, responsible for serving content (pages and documents) to end users over a web interface. The module is almost always enabled, and we rarely recommend that it be disabled.

The *ViewerConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.viewer.enabled | Toggles whether the viewer server modules are available. |

## 11.1. ContentServer

The Content Server is the unit responsible for serving file content, such as word documents or PDFs. The *ContentServerConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.contentServer.contentCacheHeader | Controls cache-related http headers generated by default (when no cache-control is specified in the url). When set to "" (blank), none of the headers `Cache-control`, `Last-Modified` or `ETag` are generated. When set to a non-blank value, the specified value is used for `Cache-control`, while `Last-Modified` and `ETag` are set based on the timestamp of the document.<br><br>The default value of `max-age=0` will make browsers cache the received documents, but re-validate the content before use (re-validation will typically be much faster than a new download). Any proxy server will also cache the content, and use re-validation to verify the cache status. |
| com.iknowbase.contentServer.defaultContentDisposition | Set to "inline" to have the browser try to open documents in-line. Behaviour depends on file type and configured application. Set to "attachment" to have "save as" as the default browser behaviour. |
| com.iknowbase.contentServer.useMimetypeSpecificContentDisposition | Set to "true" to use the content disposition value registered for the mime type if not explicitly provided on the URL. "false" will default to value of the `defaultContentDisposition` setting. |
| com.iknowbase.contentServer.createResponseHeaders | Toggles whether to include HTML-headers for document information. Normally left unchanged. |

## 11.2. PageEngine

The Page Engine is the unit responsible for composing portlets and data, and rendering information to web clients.

The *PageEngineConfiguration* accepts these configuration properties:

| Property name | Description |
| --- | --- |
| com.iknowbase.pageEngine.contentCacheEnabled | Toggles whether content caching is used at all. The legal values are either "true" or "false". |

## 11.3. ContentViewerConfiguration

The *ContentViewerConfiguration* accepts these configuration properties:

| Property name | Description |
| --- | --- |
| com.iknowbase.contentViewer | Decides if and how clueTip scripts are rendered. The legal values are `none` (default), `delayed` (load scripts only if tooltip information is found in the page), `traditional` (always load clueTip scripts). |

## 11.4. PresentationConfiguration

The *PresentationConfiguration* accepts these configuration properties:

| Property name | Description |
| --- | --- |
| com.iknowbase.freemarker.localizedLookup | Decides if localized lookup of templates is enabled, e.g. an import of *header.ftl* might first look for *header_no.ftl*, then *header_en.ftl* and finally *header.ftl* etc. The legal values are either "true" or "false". |

## 11.5. SearchClientConfiguration

The Search Client, using SOLR, handles connections to a SOLR search server. iKnowBase is capable of using multiple search engines for searching. The administrative interface uses a logical "search engine name" when mapping content for searching; for each such there is a corresponding *SearchEngineConfiguration* accepting these configuration properties:

| Property name | Description |
| --- | --- |
| com.iknowbase.searchEngine.<searchEngineName>.search.URL | URL to index server, e.g. `http://solr.example.com/solr`. |
| com.iknowbase.searchEngine.<searchEngineName>.search.protocol | Type of connection, either `BINARY` or `XML`. |
| com.iknowbase.searchEngine.<searchEngineName>.search.connectionTimeoutMillis | Max number of milliseconds to wait for the connection. |

| Property name | Description |
| --- | --- |
| com.iknowbase.searchEngine.<searchEngineName>.search.operationTimeoutMillis | Max number of milliseconds to wait for the operation. |

Note that you may also configure search engines using the iKnowBase Java Development Toolkit. Any managed bean of type "org.apache.solr.client.solrj.SolrClient" will be discovered, and a corresponding search client will be created. Using this mechanism provides the ultimate control of the SolrServer connection parameters.

## 11.6. Activiti BPM Platform

iKnowBase comes with an embedded process engine based on the Activiti BPM Platform. This engine must be explicitly enabled if you want to use it.

You should consider deploying the *iKnowBase Process Studio* which is available as a plugin extension.

The *ActivitiProcessEngineConfiguration* accepts these configuration properties:

| Property name | Description |
| --- | --- |
| com.iknowbase.process.activiti.enabled | Toggles whether the activiti engine is enabled. |
| com.iknowbase.process.activiti.jobExecutorEnabled | Toggles whether the activiti engine will pick up jobs from the database, or only respond to online requests. |
| com.iknowbase.process.activiti.processEngineName | Sets the name of the process engine. Should normally not be changed. |
| com.iknowbase.process.activiti.mailServerDefaultFrom | Sets default "from" address for email sent from activiti. |
| com.iknowbase.process.activiti.mailServerHost | Hostname for smtp server used when sending email. |
| com.iknowbase.process.activiti.mailServerPort | Port number for smtp server. |
| com.iknowbase.process.activiti.mailServerUsername | Username used for logging in to email server when sending email. |
| com.iknowbase.process.activiti.mailServerPassword | Password used for logging in to email server when sending email. |

# Chapter 12. WebServices module

iKnowBase comes with a WebServices server that provides SOAP-access to the Service API. This is disabled by default, but can be enabled when needed.

The *WebServicesConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.ws.enabled | Toggles whether the webservices module is enabled. |
| com.iknowbase.ws.mtomEnabled | Toggles whether MTOM is enabled for the SOAP endpoints. |
| com.iknowbase.ws.security.requireAuthentication | Toggles whether WS-SECURITY based authentication is required to connect to the SOAP server. |
| com.iknowbase.ws.security.loginModuleName | Name of login module to handle login requests. Use default value. |
| com.iknowbase.ws.security.trustedPrincipal | Name of trusted principal (user or group) if only some users are to be given access. |

# Chapter 13. Content Transformation Providers

See *Content Transformation Service* in the *Development Guide* for concept and usage details.

These providers are automatically enabled

- ImageEditor
- Oracle Text

These providers have specific installation requirements detailed below

- FileConverter

## 13.1. FileConverter

The FileConverter is a content transformation provider that converts documents from a number of file formats, to PDF, HTML or a number of image formats.

Note that the FileConverter is licensed separately from the core iKnowBase product. Note that the FileConverter requires 64-bit Linux "x86_64" architecture.

### 13.1.1. Understanding the FileConverter

Usage of the FileConverter works like this:

- In the database, a file (Word document, PowerPoint, etc.) is put onto a Queue through the use of the database package BATCH_FILECONVERT_CLIENT.
- The FileConverter, running in the batch server, will receive this document from the Queue via TransformationServer.
- The FileConverter will write the file to disk, and use Oracle's Outside In technology to convert it to a new format.
- The FileConverter will receive the new, converted file, and put on the queue again, using a correlation ID that lets the client find it
- In the database, the client will receive the document.

The process above implies that for the FileConverter to work, you also need to install a separate Outside In program to the server.

### 13.1.2. Installing Outside In technology

The Outside In programs are delivered separately from iKnowBase, in a zip-file that will typically be named something like fileConverter-linux-x86-64-outsidein-835.zip. Install this file using the following steps:

- Create a directory on the server, where you want to install the fileConverter

- Unzip the file, which should create a subdirectory fileConverter
- Configure the FileConverter with the proper location

```
$ cd /opt/iknowbase
$ unzip fileConverter-linux-x86-64-outsidein-835.zip
```

### 13.1.3. Configuration properties

After installing the outside in technology, you must configure the file converter. The *FileConverterConfiguration* accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.batch.fileConverter.outsideInDirectory | Location of outside in installation. File Converter is disabled when this is not set. |

### 13.1.4. Testing and troubleshooting

#### Running tests

The first step is to verify that the converstion program itself runs. Go to the installation directory, and verify that you may run document conversion from the command line:

```
$ ./exsimple Test.docx Test.pdf pdf.cfg
EX_CALLBACK_ID_PAGECOUNT: The File had 5 pages.
Export successful: 1 output file(s) created.
```

The second step is to run a "local" conversion from the web-application. Using a browser, open the "/ikbBatch" application. In the tab named "fileconverter", you will find a number of links for test conversions. They will convert from a Microsoft Word document and a Microsoft PowerPoint presentation, to a number of export formats. Clicking on these will run the server-side conversion, and return the converted document. Using the tests named "Test.docx (local)" and "Test.pptx (local)" will run the test locally, without any database involvment.

The third step is to run a "queue based" conversion. The procedure is the same as above. Using the tests named "Test.docx (queue)" and "Test.pptx (queue)" will send the document through the database for conversion, the same way as most production usage will work.

#### Missing libraries

A common problem is for conversion to image formats to fail under Linux, due to missing libraries:

```
$ ./exsimple Test.docx Test.pdf pdf.cfg
./exsimple: error while loading shared libraries: libstdc++.so.5: cannot open shared
object file: No such file or directory
./exsimple: error while loading shared libraries: libXm.so.3: cannot open shared
object file: No such file or directory
```

Search for the missing file using the "locate"-command, as shown below. If the file is missing, or only available as a stub, the proper library must be installed.

- If the library libXm.so.3 is missing, this is often due to missing openmotif22. Try the command `up2date openmotif22`.
- For libstdc.so.5, try using `yum install compat-libstdc-33.x86_64`.
- Or, search the web for similar problems, and follow instructions.

#### Fonts not found

If font path is not set you get the following message:

```
[root@build fileConverter]# ./exsimple test.docx test.tiff tiff.cfg
        EX_CALLBACK_ID_PAGECOUNT: The File had 0 pages.
EXRunExport() failed: No valid fonts found (0x0B03)
```

This can be fixed by setting the environment variable `GDFONTPATH` to a true type font directory.

One way of doing this is to add a file named "fonts.sh", with the following content, to the /etc/profile.d directory:

```
export GDFONTPATH=/usr/share/fonts/liberation/
```

#### Missing fonts

Another common problem is missing fonts:

```
[root@ip-10-53-107-93 fileConverter]# ./exsimple Test.docx Test.pdf pdf.cfg
EX_CALLBACK_ID_PAGECOUNT: The File had 1 page.
EXRunExport() failed: The font directory does not contain any font files or the
directory is invalid (0x0B02)
```

This can often be fixed by installing the liberation fonts:

```
$ yum install liberation-fonts-common liberation-mono-fonts liberation-sans-fonts
liberation-serif-fonts libreoffice-opensymbol-fonts
```

# Chapter 14. Plugins

You can install and activate custom plugins and plugins distributed with iKnowBase. iKnowBase is distributed with two plugins: Content Studio and Process Studio.

| NOTE | Neither of the iKnowBase plugins is installed by default. |
|---|---|

For installation instructions, see the *Installing plugins and patches* section.

## 14.1. Content Studio plugin

iKnowBase Content Studio is a suite of web-based tools for publishers. It includes document and image picklists.

The *Content Studio* plugin accepts these configuration properties:

| Property name | Description |
|---|---|
| com.iknowbase.plugin.ikbcontentstudio.changeDocumentTypeQuickLinkGuid | The guid of the quicklink used for chaning document types. Default value 19E06AFF1BD92038E050000A18006129, ie quicklink "CS Publish links". |
| com.iknowbase.plugin.ikbcontentstudio.completeStatusValueExternalKey | The external key of the status value used for completed documents (IKB$STATUS), used as filter on the approval page (/cs/approva). Default value IKB_COMPLETE. |
| com.iknowbase.plugin.ikbcontentstudio.imageDefaultAclExternalKey | The external key of the acl to be used as default acl for images, both in the image form and when uploading an image using drag and drop. No default value. |
| com.iknowbase.plugin.ikbcontentstudio.imageDefaultFolderExternalKey | The external key of the dimension to be used as default folder for images in the image form. No default value. |
| com.iknowbase.plugin.ikbcontentstudio.imageDoctypeExternalKey | The external key used for image documents, used both by image form and content queries. Default value IKB_DOCTYPE_IMAGE. |
| com.iknowbase.plugin.ikbcontentstudio.subtypeAttributeExternalKey | The external key of the subtype attribute, used as filter in the document archive/picklist. Default value IKB_COMMON_SUBLIST. |

## 14.2. Process Studio plugin

iKnowBase Process Studio is a web-based tool for managing Activiti process definitions, starting and monitoring processes and completing user tasks. It requires that the Activiti Process Engine is enabled, see *Activiti BPM Platform*.

### 14.2.1. Troubleshooting

#### java.lang.NoClassDefFoundError: Could not initialize class sun.awt.X11GraphicsEnvironment

If you get the exception `java.lang.NoClassDefFoundError: Could not initialize class sun.awt.X11GraphicsEnvironment`, you should add the following to your application.properties file:

```
JAVA_OPTIONS=-Djava.awt.headless=true
```

# Web Application Security

# Chapter 15. Quick install

For the iKnowBase web server, the default configuration will be

- Form based username and password authentication against the iKnowBase User Repository.
- RememberMe functionality.

If you need to customize authentication, the basic steps are

- Configure and set a default authentication module.
- Optional: Configure any extra authentication modules you need.

# Chapter 16. Overview

The iKnowBase web application security implementation is based on the Spring Security framework and takes care of user authentication and authorization to specific web application protected areas.

An authentication module combines authentication methods with authentication providers.

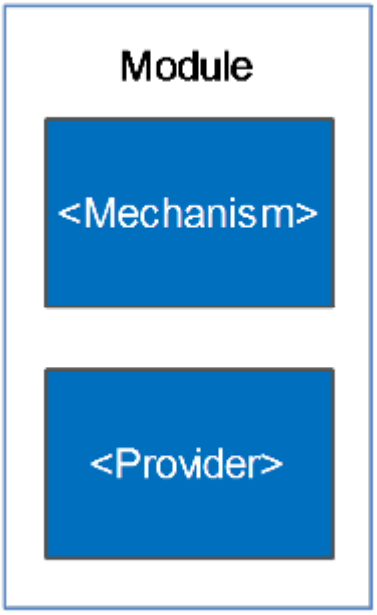


*Figure 2. Authentication Module*

Whereas an authentication mechanism defines how the client interacts with the system to provide the necessary credentials for authentication, the authentication provider verifies the credentials and, if verified, creates the authentication object for the application. The authentication object are then evaluated by the authorization rules associated with the requested resource.

The authentication process follows this flow:

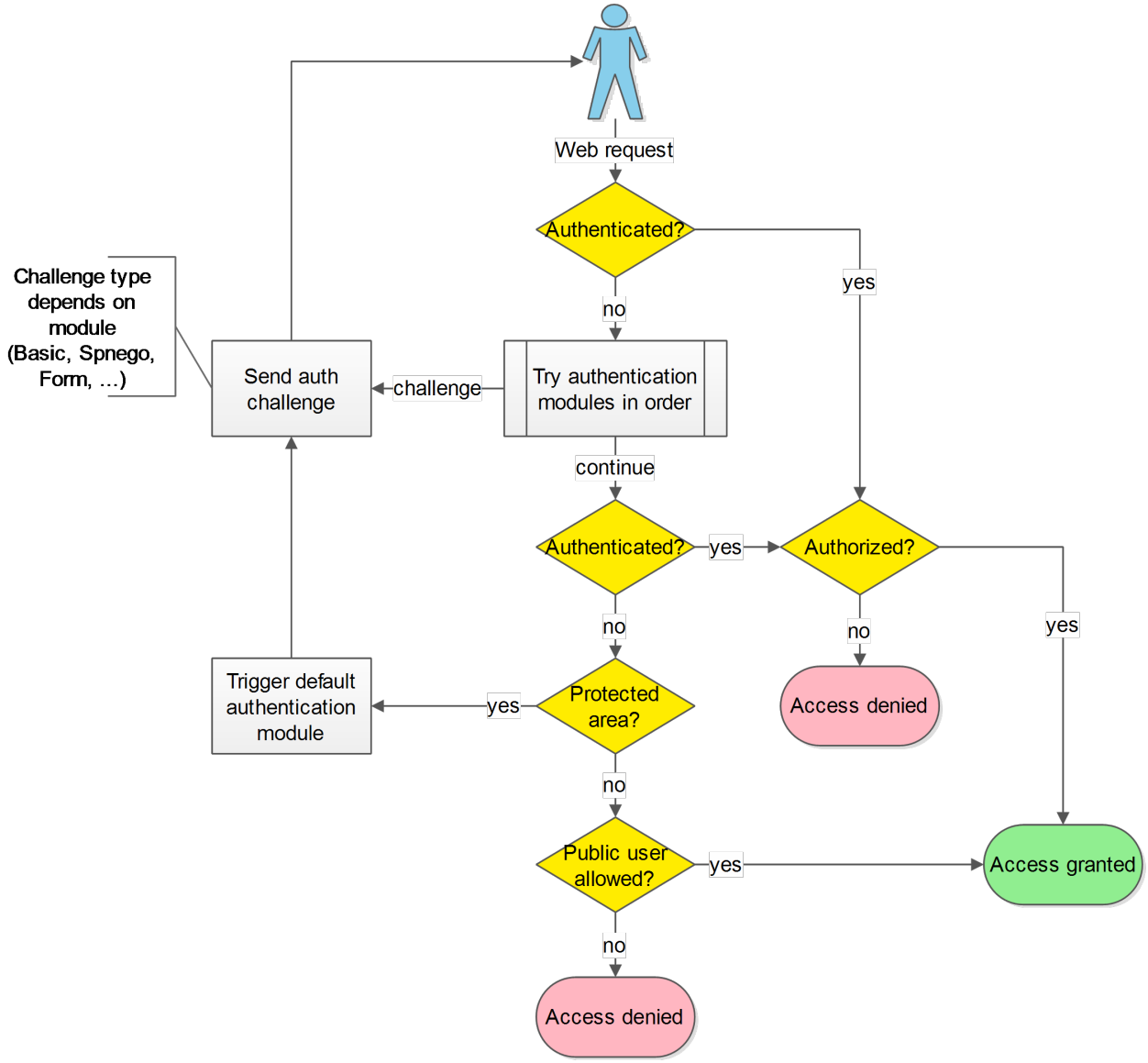


*Figure 3. Authentication Process*

The authentication modules may authenticate the user based on the incoming web request or start an authentication challenge flow.

ALL authentications will ultimately be verified by loading and verifying the user from the iKnowBase User Repository.

Multiple authentication modules (mechanisms and providers) are supported and can be activated at the same time. One is set as the default and the other active modules may be explicitly triggered. This image provides an overview of the possible combinations:

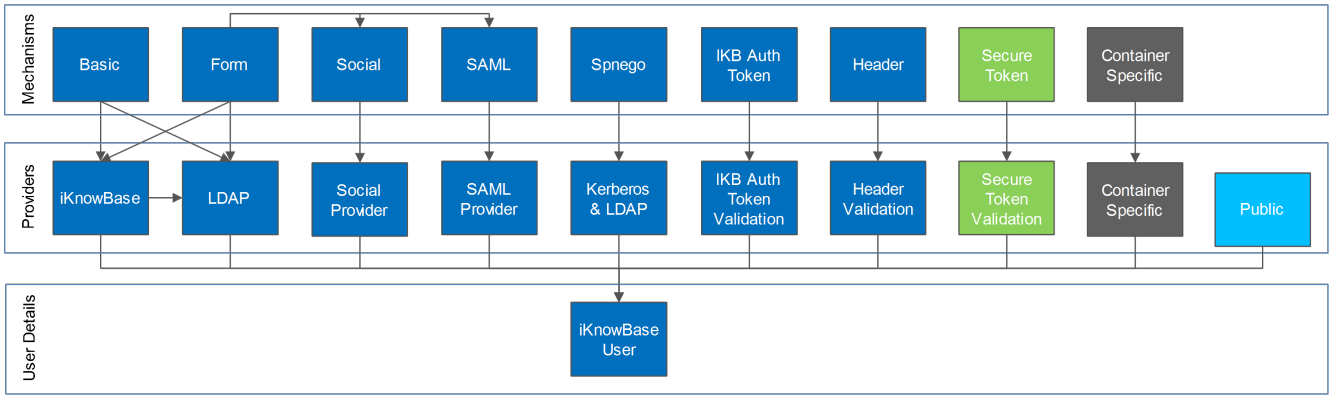


*Figure 4. Supported Authentication Modules*

- The Saml module may be triggered from the form based login module.
- The OAuth2 module is triggered from the form based login module.
- The Secure Token module is intended for application to application authentication.
- The Public module triggers if none of the other modules have authenticated the user.

# Chapter 17. Configuration

Web Application Security is configured using *configuration properties*.

See *SpringSecurityConfiguration* for detailed configuration options.

# Chapter 18. Authentication

## 18.1. Default authentication module

When the users enter a protected area, the default authentication module will be triggered and as part of the authentication process challenge the user for user credentials.

When no explicit default module has been set, the following defaults apply:

| Application Server | Modules | Authentication module | Authentication method | Authentication provider | Comment |
|---|---|---|---|---|---|
| iKnowBase web server | * | Form | Form | iKnowBase User repository | |
| iKnowBase web server | Instant | Basic | Basic | iKnowBase User repository | Due to client compatibility. |

### 18.1.1. Instant

"simple authentication" in this section refers to the following authentication mechanisms in prioritized order (first enabled wins)

- Header
- Spnego
- Basic

iKnowBase Instant requires "simple authentication" if you want to use the `/private` Instant endpoint and this is therefore set as the default. If not, you may reconfigure and use other modules such as Form based authentication.

## 18.2. Force a specific authentication mechanism

A client may trigger a specific type of authentication other than the configured default. iKnowBase supports triggering a specific mechanism using either the path `/ikb$auth/<authentication mechanism>/authenticate` or HTTP request header `X-IKB-AUTH`.

As an example, while the default authentication is form based, a client may trigger the HTTP Basic mechanism by accessing `/ikb$auth/basic/authenticate` or by issuing the HTTP request header `X-IKB-AUTH: Basic`.

| | |
|---|---|
| **NOTE** | Path trigger has lower case "basic", but header and default module setting has the initial letter in uppercase "Basic". |

## 18.3. Available authentication modules

Available authentication mechanisms:

| Authentication mechanism | Path trigger | Header trigger | Can be used as default | Authentication provider | Description |
|---|---|---|---|---|---|
| Basic | /ikb$auth/basic/authenticate | X-IKB-AUTH: Basic | YES | UsernamePassword | Basic authentication for username and password. |
| Form | /ikb$auth/form/authenticate | X-IKB-AUTH: Form | YES | UsernamePassword, OAuth2, Saml | Form based authentication. |
| Header | /ikb$auth/header/authenticate | X-IKB-AUTH: Header | YES | Header validation | Request header based authentication. |
| Spnego | /ikb$auth/spnego/authenticate | X-IKB-AUTH: Spnego | YES | Kerberos realm and LDAP | SPNEGO based authentication for Kerberos single sign on (i.e. Microsoft Windows domain single sign on). |
| Saml | /ikb$auth/saml/authenticate | X-IKB-AUTH: Saml | YES | SAML2 identity provider | SAML2 based authentication with SAML2 compatible identity providers (i.e. ADFS, Feide, Salesforce). |
| NoChallenge | N/A | X-IKB-AUTH: NoChallenge | YES | N/A | Only returns HTTP status 401 Unauthorized. No challenge provided. Intended used by javascript clients. |

| Authentication mechanism | Path trigger | Header trigger | Can be used as default | Authentication provider | Description |
|---|---|---|---|---|---|
| OAuth2 | /ikb$auth/oauth2/authorization/providerId | N/A | NO | OAuth2 | Uses form based module with Spring Security OAuth2. Will verify a preexisting user account connection. |
| iKB Secure Token | N/A | N/A | NO | Secure token validation | iKnowBase secure token based authentication. |
| iKB Auth Token | N/A | N/A | NO | Auth token validation | iKnowBase Auth token based authentication. |
| RememberMe | N/A | N/A | NO | RememberMe | Uses form based module. Will verify RememberMe cookies (created during form based login if checked). |

The authentication provider token "UsernamePassword" must be processed by an authentication provider capable of validating username and password.

All modules must ultimately validate the user against the iKnowBase User Repository using username lookup for an authentication to be considered successful.

### 18.3.1. Username and password capable Providers

iKnowBase supports multiple UsernamePassword providers for verifying the submitted username and password.

Available authentication providers for UsernamePassword:

| Provider | Default enabled | Order | Description |
|---|---|---|---|
| LDAP | NO | 1 | LDAP user repository. |
| iKnowBase | YES | 2 | iKnowBase User repository. |

Both modules may be enabled at once and will be tried in the specified order.

### 18.3.2. SAML capable Providers

An external account hosted by a SAML2 identity provider can be linked to an existing iKnowBase user account and used for authentication.

In the SAML module, iKnowBase acts as a service provider that authenticates the user with an identity provider.

Basic steps to enable iKnowBase as a service provider:

- Setup HTTPS for the iKnowBase website (may be terminated in front of iKnowBase).
- Create / reuse a Java Keystore with a private key and certificate for signing, encryption and validation of SAML metadata and messages.
- Configure iKnowBase: Enable SAML.
- Configure iKnowBase: Use the keystore with the specified private key(s).
- Start iKnowBase and use the SAML metadata generator to generate the metadata along with configuration settings.
- Store the metadata XML file to the application server's file system.
- Configure iKnowBase: According to the generator's configuration settings.

Basic steps to enable authentication with an identity provider:

- Configure iKnowBase: Add the identity provider.
- Configure identity provider: Add the iKnowBase service provider and map exchanged SAML attributes for the authenticated user.

SAML support is implemented using "spring-security-saml" (*Spring Security SAML documentation*).

#### SAML account connection

A SAML account can be mapped to an iKnowBase account using either iKnowBase Auth Token or the auto connect feature.

iKnowBase auth token "ACTIVATION" is a one time token that can be issued to a pre-existing iKnowBase user. When such as token is used, the user can choose authentication method. After successful authentication with an external identity provider, the token is used to map the accounts.

The SAML auto connect feature can be used if the identity provider knows the iKnowBase username and can issue this username attribute in the SAML response. iKnowBase will connect the accounts if a user was found with the specified username.

| | |
|---|---|
| **NOTE** | Account connections leverage the account connection infrastructure. Existing connections per user can be viewed in administration console. |

#### SAML and multiple identity providers

iKnowBase supports multiple identity providers. The user may choose identity provider using either the login form or SAML identity provider discovery mode.

The login form provided with iKnowBase will add login buttons for each configured identity provider. The login form may be fully custom implemented.

The discovery mode redirects the user to a discovery area where the default provided implementation displays a list of valid identity providers. The user may choose a provider and start the authentication process. The discovery area also takes an optional parameter that specifies which identity provider to use and will, if specified, start the authentication process automatically. The discovery area can be fully customized and allow custom automatic detection of the correct identity provider.

#### SAML and multiple service providers

Multiple service providers may be specified and the default strategy for selecting a service provider is configured using the "spSelectionStrategy" option. The default strategy is to resolve using the request's serverName, which normally will be the value of the HTTP Host header. An alternative is to use the alias path strategy where the Service Provider Entity ID must be present on the login path on the form .../alias/@localEntityId@.

#### SAML services and endpoints

SAML administration services:

| Path | Description |
|---|---|
| /ikb$auth/saml/display | Metadata web display (for iKnowBase service provider metadata). |
| /ikb$auth/saml/generate | Metadata generator (for iKnowBase service provider metadata). |
| /ikb$auth/saml/metadata | Metadata download URL (for iKnowBase service provider metadata). Supports additional alias=entityId path parameter if multiple SPs are present. |

SAML endpoints:

| Path | Description |
|---|---|
| /ikb$auth/saml/authenticate | Explicit SAML authentication trigger. Common iKnowBase auth trigger. |
| /ikb$auth/saml/discovery | IdP discovery service. |
| /ikb$auth/saml/login | Explicit SAML authentication trigger. Supports multiple SPs using "alias" path parameter. |
| /ikb$auth/saml/logout | Global logout. Use common /ikb$auth/logout for local logout. |

| Path | Description |
|---|---|
| /ikb$auth/saml/SingleLogout | Single Logout processing. (Called during global logout) |
| /ikb$auth/saml/SSO | Web Single Sign-on processing. (Called during single sign-on) |
| /ikb$auth/saml/HoKSSO | Web Single Sign-on Holder of Key processing. (Called during single sign-on) |

**SAML verified identity providers**

iKnowBase has been verified against these providers:

- Microsoft ADFS 2.0 and 3.0 using HTTP POST binding
- Feide OpenIdP using HTTP POST binding
- Salesforce using HTTP POST binding
- OpenAM using HTTP POST binding

### 18.3.3. OAuth2 and OpenID Connect

iKnowBase relies upon Spring Security OAuth2.

Each OAuth2 identity provider is configured and enabled separately and require that you configure them with the registered application id and secret. Refer to the external identity provider's documentation for application registration.

The OAuth2 identity providers require that you specify the authorized Redirect URL(s). Use `<absolute url to site>/ikb$auth/oauth2/code/<providerid>`. Example: `https://www.example.com/ikb$auth/oauth2/code/google`.

### 18.3.4. Trusted HTTP request header as authentication

The iKnowBase Header authentication module supports authentication based on a trusted HTTP request header. This means that if you have a reverse proxy in front of iKnowBase that handles the authentication and is able to provide the username in a guaranteed and trusted HTTP request header, the user will be logged in to iKnowBase as well.

It is extremely important that the reverse proxy guarantees the header, meaning that if an end client (user) sends the trusted header it is to be discarded from the HTTP request and not be allowed to reach the iKnowBase web applications.

You may configure the header module with restrictions that must be satisfied before a header is trusted, like server name, ip-address and secret header sent by the reverse proxy.

### 18.3.5. iKnowBase Auth Token

iKnowBase supports a token called "ikbAuthToken" that can be used if the end user does not know the password associated with the account. The following types and privileges are supported:

| TokenType | Consumed after use | Privilege description |
|---|---|---|
| LOGIN | NO | Valid non expired token allows automatic login. |
| ACTIVATION | YES (successful only) | Valid non expired token allows the end user to choose between multiple authentication activation options. |

Tokens are generated using IKB_AUTH_API, see *iKnowBase PL/SQL ContentServices API*, and used as either URL parameter `ikbAuthToken` or HTTP header `X-IKB-AUTH-TOKEN`.

Example with URL parameter: http://www.example.com?ikbAuthToken=the-generated-token

**iKnowBase Auth Token: LOGIN**

Enables authentication with only a web link (no username, no password) as long as the token has not expired.

| | |
|---|---|
| **WARNING** | Anyone with a valid non-expired trusted login token will be automatically logged in as the user associated with the token! |

**iKnowBase Auth Token: ACTIVATION**

The available activation options are:

- Set password for the account in the iKnowBase User Repository and proceed with username and password login to iKnowBase.
- Connect aa external (oauth2 or saml) account from one of the installed providers to the newly created iKnowBase account and authenticate using external (oauth2 or saml) authentication.

| | |
|---|---|
| **WARNING** | Anyone with a valid non-expired trusted activation token will be allowed to set password / connect using ANY external (oauth2 or saml) account! |

Requests containing an activation token will be intercepted and will redirect the user to the link without the activation token after successful activation.

### 18.3.6. Authentication token processing

Before an authenticated user has been established all enabled modules will examine the request for authentication information in the order they are defined. Some modules will only do so on specific path's (path scoped), while other modules will look no matter where the request is sent. A disabled module will skip processing.

If all modules are enabled, they will be called in the following order:

| Module | Authentication token | Path scoped |
|---|---|---|
| ikbAuthToken | ikbAuthToken URL param | NO |
| ikbAuthToken | X-IKB-AUTH-TOKEN Header | NO |

| Module | Authentication token | Path scoped |
|---|---|---|
| OAuth2 Login | HTTP request for oauth2 login | /ikb$auth/oauth2/authorization/<providerId> |
| Secure Token | _ikbUserToken HTTP header or URL param | NO |
| Header | HTTP header: [Configurable] | NO |
| Spnego | HTTP header: Authorization: Negotiate | NO |
| Form | HTTP POST j_username, j_password | /j_spring_security_check |
| Basic | HTTP header: Authorization: Basic | NO |
| Saml | SAML2 messages | /ikb$auth/saml/* |
| RememberMe | HTTP RememberMe Cookie | NO |
| Public | | NO |

Public authentication specific for iKnowBase will be used if no other authentication has been established and public access is allowed.

# Chapter 19. Authorization

A client may have one of three privilege levels:

1. Public
2. Normal
3. Administrator

If a client uses the Public level and tries to access a web area that requires Normal authentication or higher, the user will be asked to authenticate.

The iKnowBase modules have the following access requirements:

| Module | Area | Required level |
|---|---|---|
| iKnowBase Viewer | Default (NO ACL) | 1 |
| iKnowBase Viewer | ACL with public user | 1 |
| iKnowBase Authentication area | /ikb$auth | 2 |
| iKnowBase Viewer | ACL without public user | 2 |
| iKnowBase Viewer | /private | 2 |
| iKnowBase WebServices | Default | 2 and TRUSTED |
| iKnowBase Administration console | /ikb$console | 3 |

iKnowBase WebServices requires that the user is registered as a trusted principal through membership in the trusted principal ACL. See the *WebServices module* section for configuration details.

In addition, content (web modules or documents) may also be protected by iKnowBase ACLs as discussed in *iKnowBase Development Guide > Security Administration* and *iKnowBase User Administration Reference > Access Control Lists*.

## 19.1. Administrator

A user receives administrator privileges if the user is flagged as an administrator in the iKnowBase User Repository, see *iKnowBase User Administration Reference > Users*.

## 19.2. Development toolkit

Restricting access in development toolkit is discussed in the *Security* section of the *iKnowBase Development Guide > Development Toolkit*.

## 19.3. iKnowBase 6.5 and earlier versions

iKnowBase 6.5 and earlier versions required various roles for allowing access to the application,

such as IKB_USERS, IKB_DEVELOPERS and IKB_SYSADMINS. These roles are no longer in use.

From iKnowBase 6.6 the security privileges are:

| Privilege | Previous applicable role | Current requirement |
|---|---|---|
| Normal (2) | IKB_USERS | Any user that has authenticated and been verified as enabled in iKnowBase User Repository. |
| Administrator (3) | IKB_DEVELOPERS, IKB_SYSADMINS | User marked as Admin in iKnowBase User Repository. |

# Chapter 20. Switch user

An authorized user may switch to a different identity. This can be used to greatly speed up troubleshooting as an administrator/developer can be granted permission to act as the user that reported the issue.

| | |
|---|---|
| **WARNING** | Enabling switch user functionality can have severe security implications. Make sure these are understood and approved before activating this module. |

Switch user support is enabled with (See configuration reference):

- configuration enable flag
- access check procedure
- an optional audit procedure

## 20.1. Switch user access check procedure

The access check procedure is mandatory and must have the following signature:

```
procedure [procedure_name] (
    p_switch_user       ot_switch_user,
    p_return_code       out number,     // 0=PERMIT, others are user defined error
codes
    p_return_message    out varchar2    // User defined error message
);
```

The input object ot_switch user is described below. A return code of 0 will allow access. Any other will deny access and both return code and return message will be logged.

## 20.2. Switch user audit procedure

Whenever a user switches to or exits from a user happens an INFO message will be logged and the optionally defined audit procedure will be called.

```
procedure [procedure_name] (
    p_switch_user       ot_switch_user
);
```

## 20.3. Switch user database object ot_switch_user

The `ot_switch_user` object has the following properties:

| Property name | Value |
| --- | --- |
| authenticated_username | Username of the real authenticated user. |
| switch_to_username | Username of the user that is being switched to- |
| switch_type | 1=SWITCH, 2=EXIT (Only applicable for audit function). |

## 20.4. Trigger switch user

When switch user has been enabled and configured, a user can switch by sending a POST request to the path `/j_spring_security_switch_user?j_username=[USERNAME TO SWITCH TO]` and exit with path `/j_spring_security_exit_user`. Both services support an optional `redirect` URL parameter that defaults to "/".

A user may switch to different users provided access is granted for the switch. The maximum switch depth is 1, i.e. if USER1 has switched to USER2, the user may switch to USER3 provided USER1 is granted the switch. An implicit exit from USER2 is executed before the switch to USER3.

# Chapter 21. Logout

Logout service is available at `/ikb$auth/logout` with an optional `redirect` URL param, which supports both relative and absolute URLs.

Note that you are specifically authenticated for each deployed application. If you deploy iKnowBase multiple times, then each will have a `[Application context path]/ikb$auth/logout` service.

If you use Basic authentication, you may still logout, but the browser will automatically log in again if needed until the browser is closed.

If you use Spnego authentication, you may still logout, but the browser will automatically log in again if needed.

# Chapter 22. Custom security implementation

You may provide your own security implementation. Contact the iKnowBase Product Development team for implementation requirements.

# Chapter 23. Examples

This section covers common setup scenarios and explains the necessary setup.

## 23.1. Set password for users in iKnowBase User Repository

This is most often done from inside ikbStudio, but for the first user you will have to do it using the command line:

```
cd /opt/iknowbase/production
./iknowbase.sh setIkbPassword orcladmin SECRETPASSWORD
```

You can now log in using the orcladmin user, with the password SECRETPASSWORD.

## 23.2. Form based authentication against iKnowBase User Repository

For the iKnowBase web server, this is the default. There is no need to change anything.

## 23.3. Custom login form

To use a custom login form instead of the default provided with iKnowBase, adjust the form module configuration with

- Where the login form is located.
- Where the error page is located.

Login form requirements for username and password:

- Username input MUST be named "j_username"
- Password input MUST be named "j_password"
- Form action must be `/j_spring_security_check`

Login form OAuh2 requirements:

- Form action must be `/ikb$auth/oauth2/authorization/<providerId>?action=ikb$auth`

Login form RememberMe requirements:

- RememberMe input MUST be named "_spring_security_remember_me"

With database persisted sessions (default), sessions are valid for all applications using the same database repository, and you may deploy multiple iKnowBase applications and reuse the sessions. If you have changed session persistence (non-default) to memory sessions please note that an

authenticated session is valid for one web application only. If you deploy multiple applications, e.g. iknowbase-1, iknowbase-2, etc., the login form needs to POST to /j_spring_security_check relative to the Web Application you want to log in to.

- /j_spring_security_check (if deployed to the default /)
- /custom1/j_spring_security_check (if iknowbase is deployed to /custom1)

RememberMe functionality can be used to cross application borders.

The Form login module will remember the original path that triggered authentication and redirect after successful login.

If you submit directly to j_spring_security_check without being redirected to a login form first, i.e. you have implemented login functionality on a public page, you will be redirected to /. You may use an additional parameter "redirect" to explicitly set the redirect location after successful login.

## 23.4. Basic authentication against iKnowBase User Repository

- Change the default authentication module to: Basic

## 23.5. Username and password authentication against LDAP User Repository

- Change the default authentication module to a username and password capable authentication: Form or Basic (if required, see previous examples)
- Enable and configure the "LDAP UsernamePassword authentication provider"

If you don’t want fallback to iKnowBase User Repository

- Disable the "iKnowBase UsernamePassword authentication provider"

## 23.6. Authentication against LDAP User Repository with mapping for the iKnowBase username

The username mapped to iKnowBase user may also be set to any attribute name.

Given the user

```
DN:                     CN=My Name,CN=Users,DC=ad,DC=example,DC=com
sAMAccountName:         MYNAME1234
someCustomAttribute:    ORCLADMIN
```

You will be logged in to iKnowBase as "ORCLADMIN" when authenticating as "MYNAME1234".

# 23.7. Windows single sign on

It is possible to configure iKnowBase to provide single sign-on authentication on Windows workstations that are already logged into Active Directory using the SPNEGO protocol. There are several steps to this process:

These main steps will be discussed in the following section

- Change the default authentication module to: Spnego
- Enable and configure the "Spnego module"
- Enable and configure the "LDAP UsernamePassword authentication provider"
    - Optionally enable and configure the user sync feature

If you don’t want fallback to iKnowBase User Repository

1. Disable the "iKnowBase UsernamePassword authentication provider"

## 23.7.1. Prerequisites

Certain things needs to be known for SPNEGO to work. Let us assume the following configuration:

- We have windows active directory running on a server called "ad-server.example.com", serving an active directory domain called "ad.example.com".
- We will install on a server known as "webserver-01.example.com", with the IP-address 10.10.10.10, where it will serve requests to "www.example.com" and "intranet.example.com".

First, we need to make sure that the DNS (domain name system) is set up properly:

- There must be a single A(Address)-record for the webserver, with the proper IP-address. In the example, this would be an A-record for "webserver-01.example.com", pointing to the IP-address 10.10.10.10.
- All other names must be aliases, or CNAME-records, pointing to the real server. In the example, this would be two CNAME-records for "www.example.com" and "intranet.example.com", both pointing to "webserver-01.example.com".

The reason for this requirement is that the web browser will use the canonical name when creating its secret "ticket", which the server will then decode. It is therefore important that the client uses the name expected by the server. The configuration can be verified using the "ping" command on a client:

```
C:\>ping www.example.com
Pinging webserver-01.example.com [10.10.10.10] with 32 bytes of data:
Reply from 10.0.0.24: bytes=32 time=10ms TTL=63
...
C:\>ping intranett.example.com
Pinging webserver-01.example.com [10.10.10.10] with 32 bytes of data:
Reply from 10.0.0.24: bytes=32 time=10ms TTL=63
...
```

Internet Explorer will, by default, only attempt SPNEGO logins if the client uses a hostname without any dots. Thus, for maximum interoperability, make sure that it can reach the hosts "www" and "intranet":

```
C:\>ping www
Pinging webserver-01.example.com [10.10.10.10] with 32 bytes of data:
Reply from 10.0.0.24: bytes=32 time=10ms TTL=63
...
C:\>ping intranett
Pinging webserver-01.example.com [10.10.10.10] with 32 bytes of data:
Reply from 10.0.0.24: bytes=32 time=10ms TTL=63
...
```

To enable SPNEGO for hostnames with dots, they must be added to IE's Local Intranet Sites.

For the web server to be able to verify login requests, it will need to communicate with the active directory server. For that, it will need a dedicated user in active directory. The name of that user is arbitrary, but we recommend a name that matches the name of the webserver:

- The user should be named "iknowbase.webserver-01"

### 23.7.2. Configure Active Directory (Windows Server 2008 R2)

In active directory, create a user with the following properties:

- Set username to "iknowbase.webserver-01"
- Set a secret password. Use a long password with multiple words, such as "SecretPassword!GlobalMilk"
- Set "User cannot change password"
- Set "Other encryption options", "This account supports kerberos AES 128" and "... AES 256"
    - NOTE: AES encryption is recommended, but the iKnowBase web server supports other encryption types as well through Java GSS-API.

On the acti ve directory server, generate a keytab file for the user. The keytab file contains a username and password in encrypted format, and is used so that the web server can log in without specifying a password in a property file. The parameter to "-princ" is very important, and **must** follow this precise syntax: First the string "HTTP/", then the canonical name of the web server, as

seen by clients, and finally an at-sign followed by the active directory domain name.

```
C:\>ktpass -out iknowbase.webserver-01.example.com.krb5.keytab -princ HTTP/webserver-
01.example.com@AD.EXAMPLE.COM -pass SecretPassword!GlobalMilk -mapUser
iknowbase.webserver-01@AD.EXAMPLE.COM -pType KRB5_NT_PRINCIPAL -crypto ALL /kvno 0
Targeting domain controller: ad-server.example.com
Using legacy password setting method
Successfully mapped HTTP/webserver-01.example.com to webserver-01.example.com.
Key created.
Key created.
Key created.
Key created.
Key created.
Output keytab to webserver-01.example.com.krb5.keytab:
Keytab version: 0x502
keysize 68 HTTP/webserver-01.example.com@AD.IKNOWBASE.COM ptype 1 (KRB5_NT_PRINCIPAL)
vno 0 etype 0x1 (DES-CBC-CRC) keylength 8 (0x80b685bf1f52ec5b)
keysize 68 HTTP/webserver-01.example.com@AD.IKNOWBASE.COM ptype 1 (KRB5_NT_PRINCIPAL)
vno 0 etype 0x3 (DES-CBC-MD5) keylength 8 (0x80b685bf1f52ec5b)
keysize 76 HTTP/webserver-01.example.com@AD.IKNOWBASE.COM ptype 1 (KRB5_NT_PRINCIPAL)
vno 0 etype 0x17 (RC4-HMAC) keylength 16 (0x8d32128c7d08747ccc61c3b343c93c47)
keysize 92 HTTP/webserver-01.example.com@AD.IKNOWBASE.COM ptype 1 (KRB5_NT_PRINCIPAL)
vno 0 etype 0x12 (AES256-SHA1) keylength 32
(0xdd23498cd95fdb4b7ae7355b81c7999712bb4d590137af137922af97482ee45d)
keysize 76 HTTP/webserver-01.example.com@AD.IKNOWBASE.COM ptype 1 (KRB5_NT_PRINCIPAL)
vno 0 etype 0x11 (AES128-SHA1) keylength 16 (0x4296070ee7889d4b26d15986f8190df4)
```

### 23.7.3. Configure Web Application Security (SPNEGO and LDAP)

Move the keytab file you generated on the AD-server to the web server. Note that this is a sensitive file, since it can be used to log in on the AD-server: Use a secure transferring mechanism, and keep the file protected while on the web server.

Configure the Spnego module with the following settings: (More options are available)

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.spnego.enabled | true |
| com.iknowbase.spring.security.spnego.keytab | [PATH_TO]/iknowbase.webserver-01.example.com.krb5.keytab |
| com.iknowbase.spring.security.spnego.targetName | HTTP/webserver-01.example.com@AD.EXAMPLE.COM |
| com.iknowbase.spring.security.spnego.debug | true (optional) |

Configure the LDAP UsernamePassword authentication provider: (More options are available)

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.ldap.enabled | true |
| com.iknowbase.spring.security.ldap.serverUrl | ldap://ad-server.example.com:389/DC=ad,DC=example,DC=com (Multiple space-separated URLs with equal BaseDN may be provided as failovers.) |
| com.iknowbase.spring.security.ldap.clientUserDn | CN=iknowbase.webserver-01.example.com,CN=Users,DC=ad,DC=example,DC=com |
| com.iknowbase.spring.security.ldap.clientUserPassword | SecretPassword!GlobalMilk |
| com.iknowbase.spring.security.ldap.userSearchBase.1 | CN=Users |
| com.iknowbase.spring.security.ikbauth.enabled | false (optional if you don't want fallback to iKnowBase User Repository) |

Optionally, specify the LDAP sync profile. If this is specified, user synchronization will happen during login to add new users automatically. If it is omitted, a user who tries to log on must already exist in iKnowBase (typically through a scheduled LDAP Sync). The property "profile" is the externalKey of the "LDAP Sync" profile as specified in ikbStudio: (More options are available)

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.ldap.sync.enabled | true |
| com.iknowbase.spring.security.ldap.sync.profileExternalKey | ADSYNC_PROFILE_EXTERNALKEY |

Next, startup iKnowBase. The Spnego module will verify the settings when starting up. The results are logged to the configured log files.

### 23.7.4. Configure Active Directory for end users

Out of the box, all iknowbase objects are owned by a user called "orcladmin". We recommend that you create a corresponding user in Active Directory, but you may also skip this step if you have enabled fallback authentication to iKnowBase User Repository:

- Create an AD-user called "orcladmin". No group memberships are required.

### 23.7.5. Configure user synchronization for Active Directory users

For a user to be allowed access to iKnowBase, the user must exist in the iKnowBase user directory. This is done through a directory synchronization. Use the "LDAP Profile" to configure a profile, and

"LDAP Sync" to configure synchronization frequency.

### 23.7.6. Using an alternative username

By default, logging in using Active Directory will expose the "sAMAccountName" attribute (the windows domain user account name) as the username in iKnowBase. This is the default behaviour, and most often the correct one.

In some scenarios, however, you may want to expose a different username to the iKnowBase application. For example, a new corporate directory may specify account names (login names) based on employee numbers (emp10523), while you want the iKnowBase application to use the historical usernames (which could be based on first initial and last name, e.g. "EPresley"). This is easily solvable, using the following steps:

- Designate an attribute in the Active Directory to store the username you want to expose to iKnowBase. You may create a new attribute, or you may choose to reuse an existing (but unused) attribute for this purpose. Creating a new attribute is often more work and requires some effort on the Active Directory side, while reusing an existing attribute is easy but creates a mismatch between attribute purpose and actual value (some customer have chosen to use the "ipPhone" attribute, since it is most often not used for anything else).
- Update Active Directory with the proper information (e.g. for windows logon account "53310761" insert the value "EPresley" into the designated attribute).
- Update the LDAP UsernamePassword authentication provider with a username mapping as shown below:

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.ldap.ikbUsernameAttribute | ipPhone |

With this setup, a user logging in with the windows login "53310761" will be known as "EPresley" to iKnowBase.

Note that if an LDAP user does not have the ikbUsernameAttribute attribute set, iKnowBase will use the usernameAttribute attribute as username.

### 23.7.7. Configuring multiple and separate user dn patterns

The most frequently used setup is to have all users located in the same Active Directory subtree (in the examples at the beginning of this chapter, this is "CN=Users,DC=ad,DC=example,DC=com"). However, the iKnowBase security framework allows multiple user bases. The application uses the following logic when looking for these values:

- Always look for `com.iknowbase.spring.security.ldap.userSearchBase.1`.
- Keep looking for incremented numbers as long as they exist (e.g. ".2", ".3", etc., but if ".3" is missing, don't look further).
- The default value of "CN=Users" is set if no configuration was found.

This is the simplest possible configuration (also the default), with a single userbase. For users in CN=Users,DC=ad,DC=example,DC=com (DC=ad,DC=example,DC=com is set as base in the LDAP server url)

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.ldap.userSearchBase.1 | CN=Users |

This a more complex configuration, with three userbases

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.ldap.userSearchBase.1 | CN=Employees |
| com.iknowbase.spring.security.ldap.userSearchBase.2 | CN=Premium Members |
| com.iknowbase.spring.security.ldap.userSearchBase.3 | CN=Basic Members |

### 23.7.8. Combined Windows single sign on and iKnowBase User Repository

You may use a combination of Active Directory and iKnowBase login. This is by default enabled. When enabled, the server and client will first attempt to do a single sign on using the Active Directory. If this fails, the client will ask for user information which will be used to first try LDAP authentication against the Active Directory and then authenticate against the internal iKnowBase user repository. This is useful for scenarios where certain administrative users (such as the "orcladmin" user) should not exist in Active Directory.

Building on the previous example, this mode is by default enabled, but you may disable the iKnowBase User Repository by setting

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.ikbauth.enabled | false |

### 23.7.9. Conditional SPNEGO support

You may add restrictions for which requests should trigger authentication negotiation. If negotiation is disabled due to a restriction, this module will fallback to a username and password based authentication.

To restrict the negotiation to a set of specific hosts:

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.spnego.serverNameExpr | webserver-01.example.com (server name from http request) |

To restrict the negotiation to a set of specific IP addresses (X-Forwarded-For IPs are supported):

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.spnego.remoteAddrExpr | ^10\..* |

iKnowBase web server only: To restrict the negotiation to the real immediate client IP (typically a reverse proxy):

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.spnego.realRemoteAddrExpr | ^192\.168\..* |

To restrict the negotiation to a specific request header (any request header):

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.spnego.requestHeaderName | User-Agent |
| com.iknowbase.spring.security.spnego.requestHeaderExpr | `.*Firefox/25.*` |

### 23.7.10. SPNEGO fallback

The default fallback mode if SPNEGO fails is using redirect to form based login where the user can provide username and password to be validated against the enabled UsernamePassword providers (LDAP, iKnowBase).

If you want fallback to Basic authentication instead, set

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.spnego.sendAdditionalHttpBasicChallenge | true |

| Property name | Value |
| --- | --- |
| com.iknowbase.spring.security.spnego.fallbackRedirectEnabled | false |

This will first send both a Negotiate and a Basic challenge. The client will normally try Negotiate if it supports SPNEGO. If SPNEGO fails, the next challenge will be Basic only.

# 23.8. Explicit authentication trigger with redirect

The explicit authentication trigger `/ikb$auth/<authentication module>/authenticate` supports the URL parameter `redirect`. If the default module is Form based and you want to use the Basic trigger and be redirected to a specific URL afterwards, you would use `/ikb$auth/<authentication module>/authenticate?redirect=[Absolute_or_relative_url]`.

# 23.9. Integrating with ADFS using SAML

Authentication with ADFS is supported using SAML.

This example demonstrates SAML authentication on:

- ADFS 2.0.
- Account linking using both ikb Auth Token ACTIVATION or automatic link on the Name ID claim.
- Form based and default login options.

Prerequisites:

- Active Directory (AD) with users with a corresponding user account in iKnowBase.
- Active Directory Federation Services (ADFS) installed and configured to use AD and one of the available authentication options (i.e. Windows SSO using Kerberos, Basic auth, Form auth).
- HTTPS for iKnowBase website (may be terminated in front of iKnowBase).

## 23.9.1. Set up iKnowBase as a service provider

Create or reuse a Java keystore for SAML signing and encryption purposes. This example uses a private key with a self signed certificate with two year validity.

```
keytool -genkey -alias myprivatekeyalias1 -keyalg RSA -keystore
<path_to_keystore>/keystore.jks -validity 720 -keysize 2048
<specify a keystore and a key passoword (may be the same)>
```

Configure iKnowBase with SAML and specify this keystore.

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.saml.enabled | true |
| com.iknowbase.spring.security.saml.keyManager.defaultKeyAlias | myprivatekeyalias1 (must be lower case) |
| com.iknowbase.spring.security.saml.keyManager.privateKey.1.alias | myprivatekeyalias1 (keyIndex = 1) (must be lower case) |
| com.iknowbase.spring.security.saml.keyManager.privateKey.1.password | <password for myprivatekeyalias1> (keyIndex = 1) |
| com.iknowbase.spring.security.saml.keyManager.storeFile | <path to keystore/keystore.jks> |
| com.iknowbase.spring.security.saml.keyManager.storePassword | <password for keystore.jks> |

Start iKnowBase and create service provider metadata using the SAML metadata generator at /ikb$auth/saml/generate (requires admin privileges). The default settings in the generator are normally sufficient, but please check if the identity provider administrator has any special requirements.

The generator displays an XML file containing the SAML service provider metadata. Store the file a file system reachable by the application server.

The generator also displays the configuration needed. Add to iKnowBase Installation properties (or set in any other of the supported property areas). The metadataProvider index must be chosen - select the first available index >=1 (if this is the first provider added, use index 1).

### 23.9.2. Register ADFS identity provider with iKnowBase

Download the ADFS metadata from https://YOUR_ADFS_SERVER/FederationMetadata/2007-06/FederationMetadata.xml and store the file in a file system reachable by the application server.

Add the identity provider to the iKnowBase configuration with the next available metadata provider index (if the SP was .1, this should be .2).

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.saml.metadataProvider.<index>.idpLoginFormShortName | <Name to display on form login button> |

| Property name | Value |
| --- | --- |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.idpProviderEntityId | <Entity ID in ADFS XML metadata, i.e. http://YOUR_ADFS_SERVER/adfs/services/trust |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.providerEntityType | IdP |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.resourcePath | <path to ADFS XML metadata> |

### 23.9.3. Register iKnowBase service provider with ADFS

The iKnowBase service provider metadata must be registered in the ADFS identity provider.

- In ADFS console > Trust Relationships > Relying Party Trusts choose Add Relying Party Trust.
- You may now specify a local XML file containing the iKnowBase service provider metadata or (if reachable) directly download from https://YOUR_IKNOWBASE_SERVER/ikb$auth/saml/metadata.
- Choose a display name.
- Choose permit all users to access this relying party (to allow all authenticated users to access).
- Finish the wizard.

Select properties for the newly added relying party trust and set the secure hash algorithm in the advanced tab. Match the iKnowBaser service provider metadata (defaults to SHA-1).

### 23.9.4. Map identity provider user account attributes

Edit claim rules for the newly added relying party trust and add a new issuance transform rule.

- Claim rule template: Send LDAP Attributes as Claims
- Claim rule name: NameId
- Attribute store: Active Directory
- LDAP Attribute: Choose existing attribute (or type attribute name if the source attribute is not in the list) containing the user's username that will be asserted as NameID
- Outgoing Claim Type: Name ID

iKnowBase supports using the iKnowBase Auth Token for linking iKnowBase and external user accounts. See description of the iKnowBase Auth Token "ACTIVATION". The default iKnowBase setting will link using the provided Name ID, but you may also configure it to use a specific claim attribute instead.

iKnowBase also supports automatic account linking IF the identity provider can assert a claim containing the iKnowBase username. The default iKnowBase setting will link using the provided Name ID, but you may also configure it to use a specific claim attribute instead. Automatic linking is enabled using:

| Property name | Value |
|---|---|
| com.iknowbase.spring.security.saml.metadataProvider.<index>.idpAccountAutoConnectEnabled | true |

### 23.9.5. Login options and verify setup

Setup is now complete and a login button for the identity provider has been added to the login form provided by iKnowBase on /ikb$auth/form/show.

Saml is by default not the default authentication provider, but you may choose to set it as the default by configuring "Saml" as the default module.

## 23.10. Switch user database procedures

This is an example implementation of the switch user procedures.

### 23.10.1. Package spec

```
create or replace
package custom_switch_user is
    procedure access_check (p_switch_user ot_switch_user, p_return_code out number,
p_return_message out varchar2);
    procedure audit_user  (p_switch_user ot_switch_user);
end;
/
```

### 23.10.2. Package body

```
create or replace
package body custom_switch_user is

    procedure access_check (p_switch_user ot_switch_user, p_return_code out number,
p_return_message out varchar2) is
    begin
        -- log to IKB_ERROR_TAB

ikb_common_procs.log_error('CUSTOM_SWITCH_USER.ACCESS_CHECK','authenticated_username:
'||p_switch_user.authenticated_username||', '|
            'switch_to_username: '||p_switch_user.switch_to_username);

        -- implement access check logic here

        p_return_code := 0; -- permit access
    end;

    procedure audit_user  (p_switch_user ot_switch_user) is
    begin
        -- log to IKB_ERROR_TAB

ikb_common_procs.log_error('CUSTOM_SWITCH_USER.AUDIT_USER','authenticated_username:
'||p_switch_user.authenticated_username||', '|
            'switch_to_username: '||p_switch_user.switch_to_username||' switch_type =
'||p_switch_user.switch_type);

        -- do more logging / registrations here, if you need to
    end;

end;
/
```

# 23.11. Enable OAuth2 authentication with user activation link

The steps needed to enable oauth2 authentication with user activation link and default form authentication are:

- Register an application at the oauth2 provider(s) to get key and secret for using the oauth2 identity provider's API. iKnowBase only needs the minimal set of permissions for authentication.
- Configure iKnowBase authentication modules (see OAuth2 and OpenID Connect) and restart iKnowBase web application
- Test the authentication setup by generating an activation token for an existing user and access the website on any area with the "ikbAuthToken=<your token>" URL parameter. Choose the oauth2 provider and authentication should complete.

When inviting users to activate / connect using a oauth2 account, you'll need to implement

- Create user (user information, groups, ACLs, ..).
- Send activation link.
  - Create an email to send.
  - If you allow the user to activate by setting account password, you must add the user's username to this email.
  - Generate the token and insert into email body (token should NOT be presented to the user sending this email).
  - Send mail to user.

## 23.12. Best practise regarding authentication for JavaScript and other external clients

The automatic redirect and/or authentication challenge behaviour when authentication is required is not very convenient when developing JavaScript clients. The Ajax requests will typically follow any redirect, and you will need to examine the resulting page or information to determine if authentication was required. Getting an HTTP 401 with an HTTP basic challenge is even worse, since the browser will intercept the challenge without any interception possibilities by the application.

Starting with iKnowBase 7.6, iKnowBase supports and recommends:

1. The client should accept new or updated session cookie from the server (default name `IKBSESSION`) and send this cookie on all requests.
2. The client should send `X-IKB-AUTH: NoChallenge` HTTP header on all requests. If authentication is required, the server will respond with HTTP status `401 Unauthorized` without any challenge. The browser will not interrupt this call and the client can start the authentication flow.
3. Unless the client needs to force a specific authentication flow, the default authentication flow is triggered by redirecting the user to `/private/login?redirect=<where you want to go after authentication>`. If you want to preserve hash parameters, remember to encode the `redirect` parameter value using `encodeURIComponent("<where you want to go after authentication>#a-hash-parameter")`
4. For mobile apps, you may want to generate and store a fallback authentication token if the session expires. Once authenticated, request a login token and send it using `X-IKB-AUTH-TOKEN: <the-token-value>` HTTP header.
   a. If you have no session or the session has expired, the token will authenticate without interruption and you will get a new authenticated session.
   b. If both the session and the fallback token has expired the server will respond with HTTP status `401 Unauthorized` if you've also sent `X-IKB-AUTH: NoChallenge` HTTP header or redirect to a `activationFailedURL` (configuration) if not.
   c. See IKB Auth Token section regarding server configuration requirements.

# Chapter 24. Troubleshooting

## 24.1. I only want to change the configuration for a specific web application

A wildcard instance qualifier for a configuration option will match all applications. As an example, if you only want to match a specific application, set the instance qualifier to match the context path ("/" or "/MyCustomContextRoot"). See *Installation Guide > Configuration > The ikb_installation_properties table*.

## 24.2. 'AES-256-bit is not supported', 'java.security.InvalidKeyException: Illegal key size' or 'Unable to initialize due to invalid secret key'

All these messages points to that you will need to update the java installation to handle higher security levels.

You will encounter this requirement if you use any of the following:

- SPNEGO authentication with AES-256 encryption

Download and install "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files" from http://www.oracle.com/technetwork/java/javase/downloads/index.html.

## 24.3. On demand LDAP Sync during login fails

LDAP sync log can be viewed in /ikb$console/development/advanced/ldapsync. Select your sync profile and go to the "show log" tab.

## 24.4. SAML custom ADFS claim as iKnowBase username is not picked up

When using the `idpIkbUsernameAttribute` setting with ADFS, make sure that you use the claim type as `idpIkbUsernameAttribute` and NOT the name.

Example:

- DO: http://schemas.xmlsoap.org/ws/2005/05/identity/claims/ipphone
- DON'T: ipPhone

## 24.5. Kerberos: Encryption type DES CBC mode with MD5 is not supported/enabled

Java 8 has by default disabled weak crypto and will by default not support DES-CBC-MD5. A stronger encryption mechanism is **strongly** recommended. It is still possible to allow the weak crypto mechanisms by editing krb5.conf and setting allow_weak_crypto to true.

See *The Kerberos 5 GSS-API Mechanism* for more information.

# Chapter 25. SpringSecurityConfiguration

Application security in iKnowBase relies on the Spring Security Framework and provides multiple modules for authentication and authorization.

See *iKnowBase Installation Guide > Web Application Security* for additional explanations.

See [Application context path]/ikb$console/config/configurations for default and active web application security configuration. All sections start with `com.iknowbase.spring.security`.

## 25.1. Debug

Spring Security provides a debug mode documented as:

"Enables Spring Security debugging infrastructure. This will provide human-readable (multi-line) debugging information to monitor requests coming into the security filters. This may include sensitive information, such as request parameters or headers, and should only be used in a development environment."

| Property name | Description |
|---|---|
| com.iknowbase.spring.security.debug | Enable or disable Spring Security debug mode |

For debugging, you may also want to enable trace logging adjust the logger levels to trace for `org.springframework.security` and `com.iknowbase.spring.security`.

Note that this particular debug flag is not visible under /ikb$console/config/configurations.

## 25.2. Default authentication module

iKnowBase supports having multiple active authentication modules at the same time and these can be explicitly triggered, however, one must be set as the default.

| Property name | Description |
|---|---|
| com.iknowbase.spring.security.init.defaultAuthenticationModule | Name of the authentication module to use as the default |

## 25.3. Authentication modules

Module specific configuration is provided in the following sections.

### 25.3.1. Basic module configuration

No configuration options available.

### 25.3.2. Form module configuration

| Property name | Description |
|---|---|
| com.iknowbase.spring.security.form.loginPageCSS | Path to css (absolute or relative) used by the form based login page. |
| com.iknowbase.spring.security.form.loginPageURL | Login URL (absolute or relative). |
| com.iknowbase.spring.security.form.loginErrorURL | Error URL (absolute or relative) if authentication does not succeed. |
| com.iknowbase.spring.security.form.rememberMeEnabled | Enable (true) or disable (false) RememberMe functionality. |
| com.iknowbase.spring.security.form.rememberMeTokenValiditySeconds | RememberMe token cookie validity in seconds. |
| com.iknowbase.spring.security.form.rememberMeTokenCleanupThresholdSeconds | RememberMe token cookie cleanup threshold in seconds. Must be higher than the highest token validity for this iKnowBase database repository. |
| com.iknowbase.spring.security.form.rememberMeCookieName | Set the cookie name used for RememberMe tokens. |
| com.iknowbase.spring.security.form.rememberMeCookiePath | Set the cookie path used for RememberMe tokens. Defaults to / (all apps on host). |
| com.iknowbase.spring.security.form.rememberMeCookieDomain | Set the cookie domain used for RememberMe tokens. Defaults is to send the cookie to the full host name that issued it. |
| com.iknowbase.spring.security.form.rememberMeParameterName | The HTTP parameter used to indicate to remember the user at time of login. Defaults to _spring_security_remember_me. |
| com.iknowbase.spring.security.form.rememberMeProviderKey | Key to identify tokens created for remember me authentication. Default is a secure randomly generated key and is normally OK. |
| com.iknowbase.spring.security.form.exceptionMapping.<index> | Authentication exception to url mapping. `Exception full class name:url to redirect to`. |

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.form.signupURL | If OAuth2 authentication is not connected to local user account you will be redirected here (if specified) and you may then perform signup including connecting the accounts. |

### 25.3.3. Header module configuration

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.header.enabled | Enable (true) or disable (false) module. |
| com.iknowbase.spring.security.header.serverNameExpr | Regular expression restriction matching against the host name of the server to which the request was sent. |
| com.iknowbase.spring.security.header.remoteAddrExpr | Regular expression restriction matching against the end client's IP address. |
| com.iknowbase.spring.security.header.realRemoteAddrExpr | Regular expression restriction matching against the immediate client's IP address (typically a reverse proxy). |
| com.iknowbase.spring.security.header.headerNameUser | Name of HTTP request header containing the authenticated username. |
| com.iknowbase.spring.security.header.headerNameUserExtractExpr | If necessary, specify a regular expression for extracting the username from the specified request header's value. |
| com.iknowbase.spring.security.header.headerNameSecret | Name of HTTP request header containing the authentication shared secret. |
| com.iknowbase.spring.security.header.headerValueSecret | Value of the shared authentication secret (if any). |
| com.iknowbase.spring.security.header.requestHeaderName | Restriction defining required HTTP request header name (if any), see requestHeaderExpr for validation expr of value. |
| com.iknowbase.spring.security.header.requestHeaderExpr | Regular expression restriction matching against the HTTP request header specified in requestHeaderName. |
| com.iknowbase.spring.security.header.usernameExpr | Regular expression restriction matching against the username. |

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.header.authSchemeName | The scheme name in use when the server sends an authentication challenge. |
| com.iknowbase.spring.security.header.sendAdditionalHttpBasicChallenge | Also send a HTTP Basic authentication challenge. |
| com.iknowbase.spring.security.header.unauthorizedStatusCode | HTTP response status code for authentication challenge. |
| com.iknowbase.spring.security.header.unauthorizedResponseHeaders | Pipe delimited set of response headers. |

### 25.3.4. Spnego module configuration

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.spnego.enabled | Enable (true) or disable (false) module. |
| com.iknowbase.spring.security.spnego.serverNameExpr | Regular expression restriction matching against the host name of the server to which the request was sent. |
| com.iknowbase.spring.security.spnego.remoteAddrExpr | Regular expression restriction matching against the end client's IP address. |
| com.iknowbase.spring.security.spnego.realRemoteAddrExpr | Regular expression restriction matching against the immediate client's IP address (typically a reverse proxy). |
| com.iknowbase.spring.security.spnego.requestHeaderName | Restriction defining required HTTP request header name (if any), see requestHeaderExpr for validation expr of value. |
| com.iknowbase.spring.security.spnego.requestHeaderExpr | Regular expression restriction matching against the HTTP request header specified in requestHeaderName. |
| com.iknowbase.spring.security.spnego.keytab | Path to keytab file. |
| com.iknowbase.spring.security.spnego.targetName | targetName corresponding to the name registered in keytab (HTTP/[NAME]@[DOMAIN]). |

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.spnego.sendAdditionalHttpBasicChallenge | Also send a HTTP Basic authentication challenge. |
| com.iknowbase.spring.security.spnego.fallbackRedirectEnabled | Fallback to form based authentication instead of authentication challenge if spnego cannot be completed. |

### 25.3.5. LDAP UsernamePassword authentication provider

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.ldap.enabled | Enable (true) or disable (false) module. |
| com.iknowbase.spring.security.ldap.serverUrl | URL to LDAP server. May use LDAPS. TLS is not supported. Multiple space-separated URLs with equal BaseDN may be provided as failovers. |
| com.iknowbase.spring.security.ldap.clientUserDn | Full DN for client LDAP user used for user lookup. |
| com.iknowbase.spring.security.ldap.clientUserPassword | Password for client LDAP user. |
| com.iknowbase.spring.security.ldap.userSearchBase.<index> | Search base for users within the base DN given in server url. Index >= 1. Will search subtree from this base. |
| com.iknowbase.spring.security.ldap.usernameAttribute | Attribute matching the authentication username submitted by the web client. |
| com.iknowbase.spring.security.ldap.ikbUsernameAttribute | Attribute used as iKnowBase authenticated user (if ikb-username differs from LDAP username). If this attribute is missing in the LDAP user, iKnowBase will look for an iKnowBase user with the username in the above usernameAttribute instead. |
| com.iknowbase.spring.security.ldap.serverNameExpr | Regular expression restriction matching against the host name of the server to which the request was sent. |
| com.iknowbase.spring.security.ldap.remoteAddrExpr | Regular expression restriction matching against the end client's IP address. |
| com.iknowbase.spring.security.ldap.realRemoteAddrExpr | Regular expression restriction matching against the immediate client's IP address (typically a reverse proxy). |

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.ldap.requestHeaderName | Restriction defining required HTTP request header name (if any), see requestHeaderExpr for validation expr of value. |
| com.iknowbase.spring.security.ldap.requestHeaderExpr | Regular expression restriction matching against the HTTP request header specified in requestHeaderName. |
| com.iknowbase.spring.security.ldap.usernameExpr | Regular expression restriction matching against the username. |
| com.iknowbase.spring.security.ldap.passwordExpr | Regular expression restriction matching against the password. |

The LDAP provider may also be used in conjunction with the LDAP sync service. If the user was not found, the LDAP sync service will make an attempt at synchronizing the user into the iKnowBase User Repository.

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.ldap.sync.enabled | Enable (true) or disable (false) module. |
| com.iknowbase.spring.security.ldap.sync.profileExternalKey | External key to LDAP synchronization profile defined in iKnowBase. |
| com.iknowbase.spring.security.ldap.sync.executionUserName | Execution user. Should normally not be changed. |

### 25.3.6. iKnowBase UsernamePassword authentication provider

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.ikbauth.enabled | Enable (true) or disable (false) module. |
| com.iknowbase.spring.security.ikbauth.serverNameExpr | Regular expression restriction matching against the host name of the server to which the request was sent. |
| com.iknowbase.spring.security.ikbauth.remoteAddrExpr | Regular expression restriction matching against the end client's IP address. |
| com.iknowbase.spring.security.ikbauth.realRemoteAddrExpr | Regular expression restriction matching against the immediate client's IP address (typically a reverse proxy). |

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.ikbauth.requestHeaderName | Restriction defining required HTTP request header name (if any), see requestHeaderExpr for validation expr of value. |
| com.iknowbase.spring.security.ikbauth.requestHeaderExpr | Regular expression restriction matching against the HTTP request header specified in requestHeaderName. |
| com.iknowbase.spring.security.ikbauth.usernameExpr | Regular expression restriction matching against the username. |
| com.iknowbase.spring.security.ikbauth.passwordExpr | Regular expression restriction matching against the password. |

### 25.3.7. SAML authentication provider

See also: *Spring Security SAML documentation*.

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.saml.defaultFailureUrl | URL to redirect to in case of authentication failure. |
| com.iknowbase.spring.security.saml.enabled | Enable (true) or disable (false) module. |
| com.iknowbase.spring.security.saml.idpSelectionCSS | Path to css (absolute or relative) used by the page presenting available SAML identity providers for login. |
| com.iknowbase.spring.security.saml.idpSelectionURL | URL presenting available SAML identity providers for login if not using the default or custom login form . |
| com.iknowbase.spring.security.saml.keyManager.defaultKeyAlias | Default key alias to use for signing and encryption |
| com.iknowbase.spring.security.saml.keyManager.privateKey.<keyIndex>.alias | Alias for the private key in the key store. Configuration index >= 1. |
| com.iknowbase.spring.security.saml.keyManager.privateKey.<keyIndex>.password | Password for the private key in the key store. Configuration index >= 1. |

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.saml.keyManager.storeFile | Path to the Java keystore containg private key(s). |
| com.iknowbase.spring.security.saml.keyManager.storePassword | Password for accessing the Java keystore. |
| com.iknowbase.spring.security.saml.loginDefaultTargetUrl | Default URL to redirect to if no saved request was found after successful login. Defaults to / |
| com.iknowbase.spring.security.saml.logoutDefaultTargetUrl | Default URL to redirect to if no saved request was found after successful logout. Defaults to / |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.enabled | Enable (true) or disable (false) provider. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.requireArtifactResolveSigned | If true, received artifactResolve messages will require a signature, sent artifactResolve will be signed. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.requireLogoutRequestSigned | If true logoutRequests received will require a signature, sent logoutRequests will be signed. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.requireLogoutResponseSigned | Flag indicating whether entity in question requires logout response to be signed. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.alias | Alias for choosing a specific service provider (if more than one). |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.encryptionKey | Alias to private key used for encryption. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.idpDiscoveryEnabled | Enable (true) or disable (false) identity provider discovery mode. |

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.idpDiscoveryResponseURL | Identity provider discovery response URL. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.idpDiscoveryURL | Identity provider discovery URL. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.local | If true, entity is treated as locally deployed and will be able to accept messages on endpoints determined by the selected alias. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.securityProfile | metaiop = "Uses cryptographic data from the metadata document of the entity in question. No checks for validity or revocation of certificates is done in this mode. All keys must be known in advance". pkix = Signatures are deemed as trusted when credential can be verified using PKIX with trusted keys of the peer configured as trusted anchors. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.signMetadata | Flag indicating whether local metadata will be digitally signed. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.signingAlgorithm | Algorithm used for creation of digital signatures of this entity. Only used for metadata signatures. Only valid for local entities. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.signingKey | Signing key used for signing messages or verifying signatures of this entity. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.sslHostnameVerification | Hostname verifier to use for verification of SSL connections, e.g. for ArtifactResolution. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.sslSecurityProfile | Security profile used for SSL/TLS connections of the local entity. metaiop or pkix. |

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.tlsKey | Key used to authenticate instance against remote peers when specified on local entity. When specified on remote entity the key is added as a trust anchor during communication with the entity using SSL/TLS. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.extended.trustedKeys | Pipe delimited string of key alias for keys included as trusted anchors during PKIX evaluation. All keys in the keyStore are used as trust anchors with null value. Keys are only used with PKIX security profile. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.idpAccountAutoConnectEnabled | Enable (true) or disable false) automatic connect/link between iKnowBase and external user account. The iKnowBase username must be present in the SAML response (claim) using Name ID or a specific attribute. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.idpAccountSignupUrl | If no linked account could be found, redirect the user to this URL. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.idpIkbUsernameAttribute | iKnowBase username attribute in SAML response (claim). Defaults to Name ID claim. Custom attributes for ADFS uses the claim type defined in ADFS > Service > Claim Descriptions. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.idpLoginFormCssClass | iKnowBase default login form: Specify css class for the login button. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.idpLoginFormShortName | iKnowBase default login form: Specify name that will appear on the login button. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.idpProviderEntityId | EntityID, as specified in the provider's metadata XML. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.metadataRequireSignature | When set to true metadata from this provider should only be accepted when correctly signed and verified. Metadata with an invalid signature or signed by a not-trusted credential will be ignored. |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.metadataTrustCheck | If enabled, verifies metadata signature using pkix signature trust engine. |

| Property name | Description |
|---|---|
| com.iknowbase.spring.security.saml.metadataProvider.<index>.providerEntityType | Type of provider. SP = Service Provider. IdP = Identity Provider |
| com.iknowbase.spring.security.saml.metadataProvider.<index>.resourcePath | File system path to XML metadata for this provider. |
| com.iknowbase.spring.security.saml.metadataProvider.hostedSpName | Name of the default service provider for this application. Must match Entity ID of the service provider. |
| com.iknowbase.spring.security.saml.metadataProvider.spSelectionStrategy | Strategy for selecting Service Provider. SERVER_NAME or ALIAS_PATH. Defaults to SERVER_NAME (HTTP Request ServerName), which must match Entity ID of the service provider. |

### 25.3.8. OAuth2 and OpenID Connect authentication provider

See Spring Security OAuth2 sign in configuration for Spring Boot.

Note that an alternative to auto connect is using `signupURL` and handle the account connection in code. `signupURL` is configured as part of the form configuration.

| Property name | Description |
|---|---|
| com.iknowbase.spring.security.oauth2.enabled | Enable (true) or disable false oauth2 support |
| com.iknowbase.spring.security.oauth2.client.provider.<PROVIDERID>.accountAutoConnectEnabled=true | Enable (true) or disable (false) automatic connect/link between iKnowBase and external user account. The iKnowBase username must be present in the OAuth2 user and mapped as `user-name-attribute` response (attributes claim). Note: Here is a bug, try client registration name instead of providerid. |

## 25.4. Secure Token authentication

**NOTE** The iKnowBase Secure Token Engine configuration is described in *The SecureToken engine*.

| Property name | Description |
|---|---|
| com.iknowbase.spring.security.securetoken.maxAgeSeconds | Max age allowed when validating the iKnowBase secure token. |

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.securetoken.serverNameExpr | Regular expression restriction matching against the host name of the server to which the request was sent. |
| com.iknowbase.spring.security.securetoken.remoteAddrExpr | Regular expression restriction matching against the end client's IP address. |
| com.iknowbase.spring.security.securetoken.realRemoteAddrExpr | Regular expression restriction matching against the immediate client's IP address (typically a reverse proxy). |
| com.iknowbase.spring.security.securetoken.requestHeaderName | Restriction defining required HTTP request header name (if any), see requestHeaderExpr for validation expr of value. |
| com.iknowbase.spring.security.securetoken.requestHeaderExpr | Regular expression restriction matching against the HTTP request header specified in requestHeaderName. |
| com.iknowbase.spring.security.securetoken.usernameExpr | Regular expression restriction matching against the username. |

## 25.5. Anonymous / Public authentication provider

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.authentication.ikbpublic.anonymousKey | Shared key used by the anonymous authentication provider. Should normally not be set. |

## 25.6. iKnowBase User Details

All authentication attempts must ultimately load a user from the iKnowBase User Repository before the authentication is fully accepted.

| Property name | Description |
| --- | --- |
| com.iknowbase.spring.security.userdetails.executionUserName | Execution user. Should normally not be changed. |

## 25.7. Switch User

| Property name | Value |
| --- | --- |
| com.iknowbase.spring.security.switchuser.enabled | Enable (true) or disable (false) module. |
| com.iknowbase.spring.security.switchuser.accessCheckProcedureName | Name of check access database procedure. |
| com.iknowbase.spring.security.switchuser.auditProcedureName | Name of audit database procedure (optional). |

## 25.8. User Account Activation

| Property name | Value |
| --- | --- |
| com.iknowbase.spring.security.activation.showActivationOptionsCSS | Path to css (absolute or relative) used by the activation page. |
| com.iknowbase.spring.security.activation.showActivationOptionsURL | Activation options URL (absolute or relative) to redirect the user to if a valid activation token is presented. |
| com.iknowbase.spring.security.activation.activationFailedURL | Failure URL (absolute or relative) to redirect the user to if the user presented an invalid activation token. |
| com.iknowbase.spring.security.activation.ikbProviderSetPasswordEnabled | Enable users with activation token to set the user's password in the iKnowBase User Repository. |
| com.iknowbase.spring.security.activation.ikbProviderSetPasswordAlgorithm | Override default password encryption algorithm when setting password. |

## 25.9. IKB Auth Token

| Property name | Value |
| --- | --- |
| com.iknowbase.spring.security.ikbauthtoken.activation.enabled | Enable (true) or disable (false) processing of token type ACTIVATION. |
| com.iknowbase.spring.security.ikbauthtoken.login.enabled | Enable (true) or disable (false) processing of token type LOGIN. |

# External applications

# Chapter 26. Apache Solr Search Server

iKnowBase comes with ready-to-use components for integration with the Apache Solr open source enterprise search platform (http://lucene.apache.org/solr/).

We support and recommend the Solr server version matching the version of the Solr client built into iKnowBase. However, the Solr client and Solr server are largely version independent, so other combinations are also likely to work fine. See the release notes for which Solr version is used in each iKnowBase release.

## 26.1. Installation

To install, do the following:

- Download and unzip the latest supported version of SOLR. e.g. from https://lucene.apache.org/solr/downloads.html.

You should now have a top level structure like.

```
solr-<version>
solr-<version>/server
```

- Unzip the file `iknowbase-<version>-solr-distribution.zip` to `solr-<version>/server`. This will create four folders:

```
lib\ext         - jdbc related files
iknowbase       - core definition files
iknowbase\lib   - iKnowBase security plugin
iknowbase\conf  - Core configuration
```

- Security-plugin

iKnowBase ships with a Solr-plugin that verifies document access for all documents returned from the Solr search engine. The principles behind this plugin is that iKnowBase will add security information to the search query sent to Solr, which will then be intercepted by the Solr engine during search. For this to work, two items must be in place. First, the plugin must be able to connect to the iKnowBase database, and second, the iKnowBase viewer application and the Solr plugin must share a common secret used to encrypt the security information.

The common secret is known as a secure key, and is configured either in the database or in the Solr configuration. Please note that the secure key **must** be identical in the iKnowBase client and in the Solr plugin, or the client will not be able to authenticate properly.

If you want the secure key stored in the database, stored it in the ikb_installation_properties table in the database, using a property name of `com.iknowbase.secureTokenEngine.secureKey` and an instance_qualifier that can be used by the iKnowBase web application (a single star, "*", will always work).

Alternatively, store the key directly in the Solr server configuration using the Solr configuration. Solr configuration is read from "solrconfig.xml", but this file can again refer to properties stored in either `solrcore.properties` or `configoverlay.json` See for example https://solr.apache.org/guide/8_11/configuring-solrconfig-xml.html for details.

| Property | Description |
|---|---|
| URL | This points to the database where iKnowBase is installed, eg. jdbc:oracle:thin:@//hostname.example.com:portnumber/service_name. |
| Username | The database-user where iKnowBase is installed. |
| Password | The password to the iKnowBase database user. |
| Secure key | The secure key used to compute the SecureToken |
| instanceQualifier | The instance_qualifier used to look up the proper secure token engine configuration from installation_properties in the iKnowBase database. |

The properties can be stored in one of several locations. The plugin will check them in order and use the first one found.

- An environment variable containing the actual value, according to the table below
- An environment variable containing the name of a file with the actual value
- A Solr property containing the actual value, according to the table below
- A Solr property containing the name of a file with the actual value

| Property | Environment variable | Environment variable for filename | Solr config | Solr config for filename |
|---|---|---|---|---|
| URL | IKNOWBASE_DATASOURCE_URL | IKNOWBASE_DATASOURCE_URL_FILE | dbURL | dbURLFile |
| Username | IKNOWBASE_DATASOURCE_USERNAME | IKNOWBASE_DATASOURCE_USERNAME_FILE | dbUser | dbUserFile |
| Password | IKNOWBASE_DATASOURCE_PASSWORD | IKNOWBASE_DATASOURCE_PASSWORD_FILE | dbPassword | dbPasswordFile |
| Secure key | IKNOWBASE_SECURE_KEY | IKNOWBASE_SECURE_KEY_FILE | secureKey | secureKeyFile |
| Instance | IKNOWBASE_INSTANCE_QUALIFIER | IKNOWBASE_INSTANCE_QUALIFIER_FILE | instanceQualifier | instanceQualifierFile |

Example using values set directly in the Solr configuration:

```
"dbURL": "jdbc:oracle:thin:@//myserver.example.com:1521/orcl.example.com",
"dbUser":"<username>",
"dbPassword":"<password>",
"instanceQualifier":"*"
```

Or, using values stored as Docker secrets:

```
"dbURL": "jdbc:oracle:thin:@//myserver.example.com:1521/orcl.example.com",
"dbUser": "ikb_production",
"dbPasswordFile": "/run/secrets/ikbDatabasePassword",
"secureKeyFile": "/run/secrets/ikbSecureTokenKey"
```

- Start the new solr instance

```
cd solr-<version>
bin/solr start
or
bin/solr start -p <port number>
```

- Create a new core based on a basic config set defined under `solr-<version>/server/iknowbase/conf`.

```
cd solr-<version>
bin/solr create_core -c iknowbase -d server/iknowbase/conf/
```

- Change solr-version to the correct value:

`<luceneMatchVersion>*solr-version*</luceneMatchVersion>`

If you are unsure of the valid value, verify the value given in `solr-<version>/server/solr/configsets/basic_configs/conf/solrconfig.xml`

- Verifiy the installation

Verify it starts without any errors and access `<hostname>:<solr port number>` from a browser. Make sure the core is available.

## 26.2. Upgrade an existing SOLR instance

To upgrade an existing installation, please follow the steps as descriped below:

- Do the two first steps described under the Installation chapter above.
- Copy the core from the former solr release.

```
$ cd solr-<version>/server/solr
$ cp -r <former solr release>/solr/<core>/ .
```

- Change solr-version to the correct value:

`<luceneMatchVersion>*solr-version*</luceneMatchVersion>`

If you are unsure of the valid value, verify the value given in `solr-<version>/server/solr/configsets/basic_configs/conf/solrconfig.xml`

- Update the data index to the new version

```
$ export dir=/fullpath/solr-<version>/server/solr-webapp/webapp/WEB-INF/lib

Verify the version of the index:

$ java -cp $dir/lucene-core-<version>.jar:$dir/lucene-backward-codecs-<version>.jar
org.apache.lucene.index.CheckIndex -fast /fullpath/solr-
<version>/server/solr/<core>/data/index/

Update the index to the latest version:

$ java -cp $dir/lucene-core-<version>.jar:$dir/lucene-backward-codecs-<version>.jar
org.apache.lucene.index.IndexUpgrader -delete-prior-commits /fullpath/solr-
<version>/server/solr/<core>/data/index/
```

- Stop the former solr release.
- Start the new solr version and make a verification.

```
cd solr-<version>
bin/solr start
```

- If the file <core>/conf/configoverlay.json doesn't exist, it must be created. From solr version 6.x this is the preferred way of storing user variables.

Prior to 6.x, solrcore.properties was used for this, but it will be deprecated in the future. Recreate all variables defined in the old solrcore.properties, as a mininum the following user variables must be created:

```
$ curl http://server:port/solr/<core>/config -H'Content-type:application/json' -d
'{"set-user-property" : {"jdbcUrl":"jdbc:oracle:thin:@//<server>:<port>/<service
name>"}}'
$ curl http://server:port/solr/<core>/config -H'Content-type:application/json' -d
'{"set-user-property" : {"dbUsername":"<iknowbase schema>"}}'
$ curl http://server:port/solr/<core>/config -H'Content-type:application/json' -d
'{"set-user-property" : {"dbPassword":"<db password>"}}'
```

Verify it starts without any errors and access <hostname>:<port number> from a browser. Make sure the core is available.

## 26.3. Starting and stopping

Create a start/stop script for linux. It can be placed under `/etc/init.d`. To add Solr as a linux service, use the systemctl tool, or chkconfig if you use an older linux version.

Start Solr and use a web browser to see the Admin Console: `http://hostname.example.com:8983/solr/admin`. If Solr is not running, your browser will complain that it cannot connect to the server.

## 26.4. Configuration

Before use, the Solr-installation must be configured. Similarly, the iKnowBase applications that will index and search must be configured.

### 26.4.1. SolrCloud

Apache Solr includes the ability to set up a cluster of Solr servers that combines fault tolerance and high availability. Called SolrCloud, these capabilities provide distributed indexing and search capabilities, supporting the following features:

- Central configuration for the entire cluster.
- Automatic load balancing and fail-over for queries.
- ZooKeeper integration for cluster coordination and configuration.

SolrCloud is the preferred method when it comes to load balancing, fail-over and replication.

We refer to documentation from Apache Solr e.g. https://cwiki.apache.org/confluence/display/solr/SolrCloud for more information on this.

### 26.4.2. Configure the iKnowBase applications

- Configure the ContentIndexer, see the *ContentIndexer* section
- Configure the SearchClient, see the *SearchClientConfiguration* section

# 3rd party application severs

# Chapter 27. iKnowBase web server

| NOTE | This chapter assumes the iKnowBase database repository has been created, as outlined in the *Quick installation and upgrade overview* section. |
|---|---|

iKnowBase comes with an embedded web server based on the Eclipse Jetty server. This is a fast and capable server which requires little resources and is very easy to manager. This is the recommended web server for most installations.

## 27.1. Spring Boot

Both the iKnowBase application and the embedded web server are developed using *Spring Boot* and configuration handling and options in Spring Boot for Jetty are applicable for iKnowBase.

We've also extended Jetty with settings that are not available in Spring boot. These are prefixed `com.iknowbase.server.jetty`.

## 27.2. Preparations

Verify \/ review the recommended installation structure outlined in the *Quick installation and upgrade overview* section.

## 27.3. Configure the iKnowBase instance

If you've not already done so, set up the iKnowBase instance configuration file described in outlined in the *Quick installation and upgrade overview* section.

For the examples further down we assume you've used the default configuration file name `application.properties` for a configuration named "production".

## 27.4. Run and test the iKnowBase instance

With this configuration, you should be able to easily run and test the applications:

```
cd /opt/iknowbase/production
./iknowbase.sh webServer
```

Start a web-browser, and navigate to http://YOUR_SERVER_NAME:8080/ressurs/iknowbase (to see documentation) or http://YOUR_SERVER_NAME:8080/ikb$console (for the Management Console). The first should open immediately, while the second one will require login, which requires configuration as shown below.

## 27.5. Configure modules

The iKnowBase application itself is distributed as a Spring Boot executable jar file. Review

configuration sections for the available modules and configure them as needed.

# 27.6. Configure Web Application Security

The recommended sequence for configuration is to first set up iKnowBase using the internal user directory, and verify sure that you have access to the iKnowBase web applications. After that, you can change to any supported authentication setting you want, but with the knowledge that iKnowBase does indeed work properly with a simple login service.

See the *Web Application Security* section for additional explanations.

# 27.7. Configure SSL

We strongly recommend using SSL (https) for all production sites.

## 27.7.1. Terminating SSL in an external proxy

If you terminate SSL in an external proxy, that proxy will typically use HTTP (an unsecured connection) to talk to the application server. Then, the application server will not be aware that the browser sees a secure connection, and will by default generate links to an unsecure site. To avoid this, note the following items:

- Configure the load balancer to generate an HTTP header called "X-Forwarded-Proto" with the value "https", ref http://en.wikipedia.org/wiki/List_of_HTTP_header_fields.
- Configure the iKnowBase server to use a scheme-mapping that understands this header, as shown below.
- Verify this setup by loading /ikb$console/java/request using from a secure connection; the value "Requested URL" should indicate a https-scheme.

If using Apache httpd for ssl-termination, the following configuration in httpd.conf should set the required header:

```
<Virtualhost ...>
...
RequestHeader set X-Forwarded-Proto "https"
...
</Virtualhost>
```

To configure the iKnowBase web server, set the property `server.forward-headers-strategy` (called `server.use-forward-headers=true` in iKB 8.0 and older) in the property file:

```
...
server.forward-headers-strategy=NATIVE
...
```

### 27.7.2. Configuring SSL listener in iKnowBase web server

The iKnowBase web server can be configured to listen for HTTPS traffic with the following options:

```
...
server.port=8443
server.ssl.key-store=iknowbase.keystore
server.ssl.key-store-password=<KEYSTORE_PASSWORD>
server.ssl.key-password=<KEY_PASSWORD>
...
```

If you require both HTTP and HTTPS listeners at the same time, do:

```
...
server.port=8443
server.ssl.key-store=iknowbase.keystore
server.ssl.key-store-password=<KEYSTORE_PASSWORD>
server.ssl.key-password=<KEY_PASSWORD>
com.iknowbase.server.jetty.httpListener.port=8080
...
```

The iknowbase.keystore is a standard java keystore in JKS format generated by Java keytool.

The following example demonstrates generating the iknowbase.keystore using preexisting private key, a server certificate and a CA chain file.

```
$ sudo openssl pkcs12 -inkey ./www_example_com.key -in ./www_example_com.crt -certfile
./ca_chain_file.crt -export -out ./www_example_com.pkcs12 -passout
pass:<KEYSTORE_PASSWORD>
$ $JAVA_HOME/bin/keytool -keystore ./iknowbase.keystore -storepass <KEYSTORE_PASSWORD>
-importkeystore -srckeystore ./www_example_com.pkcs12 -srcstoretype PKCS12
-srcstorepass <KEYSTORE_PASSWORD>
```

### 27.7.3. Multiple certificates (SNI)

Support for multiple certificates is as simple as repeating the previous step for the next certificate and importing into the same keystore.

## 27.8. HTTP/2

HTTP/2 is supported and requires that the embedded web server is configured with SSL/TLS. HTTP/2 will only be enabled on the SSL/TLS connector. It will still also handle HTTP/1.1.

1. Configure embedded web server with SSL/TLS.
2. Set `server.http2.enabled=true` in application.properties.

## 27.9. Advanced topics

### 27.9.1. Specify temp directory

During normal execution, the iKnowBase web server needs access to a temporary directory. This defaults to value of the system property `java.io.tmpdir`. If for some reason you need to use a different temporary directory, change this system property.

Note that iKnowBase before version 7.2 unpacked the web application into a work directory and if that temporary directory was deleted it would fail to serve the unpacked content. From iKnowBase 7.2 this is no longer the case, and you may use the normal temp directory.

### 27.9.2. Configure logging

iKnowBase logs only to console by default. If you want to write the logs to file, set `logging.file.path` to absolute or relative path to the log directory you want. We also recommend that you enable the iKnowBase specific log configuration using `logging.config=classpath:log4j2-default.properties`.

```
# logging.file.path=
logging.file.path=./work/logs
# logging.config=
logging.config=classpath:log4j2-default.properties
```

The iKnowBase Setup program writes logs to ./work/logs by default, however you may override this by setting:

```
# com.iknowbase.setup.log.logDirectory=./work/logs
com.iknowbase.setup.log.logDirectory=<A LOG DIRECTORY>
```

Access logging are by default enabled using daily rolling of the access log file. See application.properties.SAMPLE for details.

### 27.9.3. Setting max form size

Max form size that can be submitted to the iKnowBase web server is by default set to 200 000 bytes and 1000 keys. This is intended to help in denial of service attack scenarios where malicious clients send huge amount of data.

Please note that this limitation does NOT affect file uploads using `multipart/form-data`. See multipart.* configuration options in the Java Applications section.

If you need to POST forms larger than 200 000 bytes and/or 1000 keys override the max value in the iknowbase property file:

```
...
# com.iknowbase.server.jetty.maxFormContentSize=200000
# com.iknowbase.server.jetty.maxFormKeys=1000
...
```

### 27.9.4. Shutdown capability and faster restarts

If you enable the shutdown capabilities by setting `com.iknowbase.server.jetty.shutdown.token` to a string token like `com.iknowbase.server.jetty.shutdown.token=YouShallNotPass!!!`, any additional start of the application will send a shutdown signal to the currently running application.

Compared to the normal restart (stop + start) procedure, this will accomplish a restart with minimal downtime since the new container will be close to ready when the shutdown signal is sent.

You may also invoke the command "stopWebServer" to just stop the existing one without activating a new server.

| | |
|---|---|
| **NOTE** | Shutdown call is only accepted from localhost. This means that the shutdown handler cannot be invoked from a remote location even if the token is known. |

## 27.10. Troubleshooting

### 27.10.1. Database connections through firewall or on an unreliable network

When accessing a database through a firewall or on an unreliable network, use the Oracle Net connection descriptor syntax with ENABLE=BROKEN instead of the standard JDBC URL syntax as the database connection string.

Default JDBC URL:

```
jdbc:oracle:thin:@//localhost:1521/ORCL
```

Using Oracle Net connection descriptor syntax:

```
jdbc:oracle:thin:@(DESCRIPTION = (ENABLE = BROKEN)(ADDRESS_LIST = (ADDRESS = (PROTOCOL
= TCP)(HOST = localhost)(PORT = 1521)))(CONNECT_DATA =(SERVER =
DEDICATED)(SERVICE_NAME = ORCL)(FAILOVER_MODE =(TYPE = SESSION)(METHOD = BASIC))))
```

### 27.10.2. Unexpected error occurred: java.lang.IllegalStateException: Form too large

See Setting max form size.

### 27.10.3. WARN: bad HTTP parsed: 400 HTTP/0.9 not supported for HttpChannelOverHttp

iKnowBase 7.0 upgraded the embedded web server Jetty from 9.2 to 9.3, which uses updated HTTP specifications.

See *Header parse error after upgrade to Jetty 9.3* for details:

- Jetty 9.2 followed RFC2616 (Now Obsoleted by: RFC7230, RFC7231, RFC7232, RFC7233, RFC7234, RFC7235 and Updated by: RFC2817, RFC5785, RFC6266, RFC6585).
- Jetty 9.3 follows the updates to the venerable (from 1999!) RFC2616 spec. Many things that were valid in the past are no longer valid. We also dropped support for HTTP/0.9 in Jetty 9.3.

This update should not affect end user clients, but it might affect systems requesting information from iKnowBase over HTTP. The systems need to adjust to follow updated HTTP specs.