iKnowBase®

InstallationGuide

1. iKnowBase Installation Guide

Welcome to the iKnowBase Installation Guide. Note that this installation guide only covers upgrades from iKnowBase 5.7.2 and newer. If you are upgrading from an older version, you must **first** upgrade to iKnowBase 5.7.2 using the old upgrade instructions, and **then** use these upgrade instructions to upgrade to the latest version.

Introduction

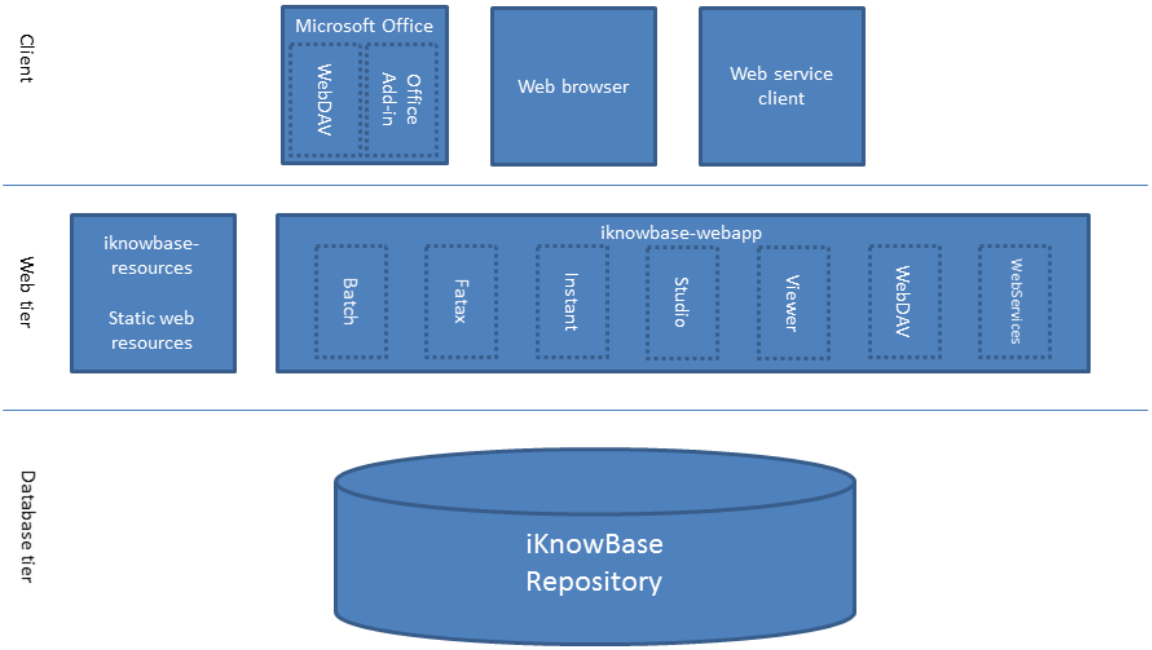
This book is conceptually structure into three parts:

- Part one (comprised of the chapters Installation toplogies, Quick installation and upgrade overview and Configuration overview) gives an overview of the installation. For experience users, the installation and upgrade overview chapter contains most of the required information.
- Part two (comprised of a chapter for the Database repository, one chapter for each of some of the java applications and one chapter for the Solr search server) gives a more detailed understanding of what the installation of these components actually include.
- Part three (comprised of one chapter for each supported servlet- or application server) contains detailed information about installations on that particular platform.

Table of contents

2. Installation topologies

iKnowBase components



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-resource application contains static web resources, such as scripts, css-files and images. This is normally either deployed to a web server in front of the application server or deployed directly to the application server itself.
- The iknowbase-webapp contains all iKnowBase Java web modules, which may be enabled or disabled in the configuration. This is deployed in a java servlet container.

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.

Deployment options

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.

The web tier

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

For most installations, use the iKnowBase web serverserver 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 chose 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.

Supported infrastructure

See “Supported platforms” in iKnowBase Release Notes.

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

We recommend that you use the procedure above even if you intend to install the web application info a third party application server, proceeding with the third party server only when the iKnowBase web server has been verified.

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-7.0.4	Version specific directory for the unpacked software (current version)
.../development	Directory for a given instance (example: “development”)
.../development/plugins	Directory containing all .jar/.war plugins
.../development/wars	Directory containing all instance specific built wars (.war file or exploded war)
.../production	Directory for a given instance (example: “production”)
.../production/plugins	Directory containing all .jar/.war plugins
.../production/wars	Directory containing all instance specific built wars (.war file or exploded war)

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.

```
$ su -
# mkdir /opt/iknowbase
# chown iknowbase /opt/iknowbase
# exit
$ mkdir /opt/iknowbase/distributions
$ cd /opt/iknowbase/distributions
$ cp /tmp/iknowbase-7.0.4-bin.zip /opt/iknowbase/distributions/
```

```
# Note: the .zip-file contains top level directory "iknowbase-7.0.4"
$ cd /opt/iknowbase
$ unzip distributions/iknowbase-7.0.4-bin.zip
```

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 a “iknowbase.sh” script that forwards to the proper version. Run the script below from each of the instance-specific directories:

```
# Run the following lines including line starting with "EOF" all in one
 comman
$ cd /opt/iknowbase/production
$ cat > iknowbase.sh << 'EOF'
#!/bin/bash
../iknowbase-7.0.4/iknowbase.sh $*
EOF

$ chmod +x iknowbase.sh
```

For each repository, create a property file with all the required settings for connecting to the database and running the iKnowBase web server. We recommend naming the property file after the repository instance (“production.properties”), but you may choose any name you want. A sample is provided in etc/SAMPLE.properties in the distribution. Copy the sample, edit the new file and set proper values for db.URL, db.sysPassword, db.ikbUser and db.ikbPassword.

```
$ cp ../iknowbase-7.0.4/etc/SAMPLE.properties production.properties
```

- db.URL is the jdbc url to the database, see example below and i SAMPLE.properties.
- db.sysUser is the name of a user with SYSDBA privilege in the database, typically "SYS"
- db.sysPassword is the password for the SYS-user in the database.
- db.ikbUser is the name of the database user that contains the iKnowBase repository that will be upgraded.
- db.ikbPassword is the password for the iKnowBase database user.

Note that the db.sysUser and db.sysPassword is only required during installation, and may be removed during normal execution, if that is required.

A minimal file for repository creation and configuration could look like this:

```
# Database connection information
db.URL = jdbc:oracle:thin:@//localhost:1521/orcl
db.ikbUser = ikb_prod
db.ikbPassword = SECRETPASSWORD
db.sysUser = sys
db.sysPassword = SECRETPASSWORD
```

Install or upgrade the database repository

To *install* a fresh iKnowBase repository, run the following steps:

1. Create user and import schema content

```
$ cd /opt/iknowbase/production
```

```
$ ./iknowbase.sh production.properties createUser
$ ./iknowbase.sh production.properties uploadFile ../iknowbase-7.0.4/etc/
IKB_MASTER_70.dmp
$ ./iknowbase.sh production.properties importFile IKB_MASTER_70.dmp
 IKB_MASTER_70
```

2. Optionally download and display import log

```
$ ./iknowbase.sh production.properties downloadFile IKB_MASTER_70.log .
$ cat IKB_MASTER_70.log
```

3. Run upgrade scripts on the newly created installation

```
$ ./iknowbase.sh production.properties 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 production.properties setIkbPassword orcladmin changeMe
```

To *upgrade* an existing iKnowBase repository, run the following steps instead:

```
$ ./iknowbase.sh production.properties exportSource source.zip
$ ./iknowbase.sh production.properties configureUser
$ ./iknowbase.sh production.properties upgradeAll
```

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

Create an instance specific web application containing all plugins and patches

Optional step if you need to apply any .jar / .war patches or add .jar / .war plugins.

NOTE: iKnowBase Process Studio is distributed as a plugin and is NOT installed by default. Add it in this step if you want to use iKnowBase Process Studio.

First create the instance directories

```
$ mkdir /opt/iknowbase/production/plugins
$ mkdir /opt/iknowbase/production/wars
```

Add all .jar/.war plugins and patches to the plugins directory.

Create an instance specific web application: original iKnowBase web application + all plugins:

```
$ cd /opt/iknowbase/production
$ ./iknowbase.sh production.properties assembleWar ../iknowbase-7.0.4/
wars/iknowbase-webapp-7.0.4.war ./wars/iknowbase-webapp-7.0.4-custom.war ./
plugins/*
```

We recommend always patching from the original application as this ensures you know which plugins you’ve added at all times.

Deploy the newly created iKnowBase web application instead of the original web application. See application server specific chapter for deployment details of custom applications.

Start the iKnowBase application

With the database repository in place, you can run the iKnowBase web applications. There are two main alternatives: Either use the iKnowBase web server, or deploy to one of the supported application servers.

To run the iKnowBase web server, use:

```
$ cd /opt/iknowbase/production
```

```
$ ./iknowbase.sh production.properties webServer
```

Next steps

Generic Java applications chapter:

- *Java Applications*

Application server specific chapters:

- *iKnowBase web server*
- *Oracle WebLogic*

4. Configuration

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

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.

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 property source “IKB_INSTALLATION_PROPERTIES” represents values loaded from the database table _ikb_installation_properties_.
- The property source “servletConfigInitParams” is typically not used, but is provided as a standard source by the underlying technology.
- The property source “servletContextInitParams” is typically not used, but is provided as a standard source by the underlying technology.
- The property source “jndiProperties” is typically not used, but is provided as a standard source by the underlying technology.
- The property source “systemProperties” gets values from the command line used to start the java virtual machine, so that you can directly specify property values when starting the server.
- The property source “systemEnvironment” gets values from the operating system environment (where you find e.g. the PATH environment variable)

In practice, you will probably use either java system properties or the ikb_installation_properties tables:

- Java system properties are easy to set, in particular in an installation with a single application server instance
- The ikb_installation_properties database table is more easily managed, and allows you to set values that apply to multiple application server instances

System properties

System properties is the “standard” java mechanism for configuring an application. They may be set in various ways:

- Using “-Dname=value” as a command-line option to the java program
- Using “System.name=value” in an iKnowBase property file

Thus, to enable the activiti process engine inside iKnowBase, you could use the following iKnowBase property file:

```
db.URL = jdbc:oracle:thin:@//localhost:1521/orcl
db.ikbUser = ikb_prod
db.ikbPassword = SECRETPASSWORD

System.com.iknowbase.process.activiti.enabled=true
```

The ikb_installation_properties database table

Using the IKB_INSTALLATION_PROPERTY table is the most common option. Since this 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, ...)

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.

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 production.properties createUser
$ ./iknowbase.sh production.properties uploadFile IKB_MASTER_70.dmp
$ ./iknowbase.sh production.properties importFile IKB_MASTER_70.dmp
 IKB_MASTER_70
$ # Optionally download and display import log
$ ./iknowbase.sh production.properties downloadFile IKB_MASTER_70.log .
$ cat IKB_MASTER_70.log
```

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 production.properties 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.

Custom step for Oracle 12c database with Pluggable databases (PDB)

If installing on a Oracle 12c database with Pluggable databases (PDB), DATA_PUMP_DIR does not work 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>

Import startup data based on a export file

You can import startup data with any mechanims 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 production.properties uploadFile IKB_MASTER_70.dmp
```

```
$ ./iknowbase.sh production.properties importFile IKB_MASTER_70.dmp
 IKB_MASTER_70
```

The first command will upload the file IKB_MASTER_70.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_70.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_70.

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_70.log). You can also use the following command to download the logfile to your local directory:

```
$ ./iknowbase.sh production.properties downloadFile IKB_MASTER_70.log .
```

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

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

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

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 production.properties 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.

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 production.properties 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.

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 production.properties 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 may have open database connections.

Recompile invalid packages

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

```
$ iknowbase.sh production.properties compileInvalid
```

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

De-installation

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

```
$ iknowbase.sh production.properties 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!

Advanced topics

Global runtime preferences

iKnowBase uses a database package header, IKB_GLOBAL_PREFS, to determine methods and options on certain functionality. To modify the package header you need access to the database with a proper tool e.g SqlDeveloper or Toad.

Property name	Default Value	_Description
has_oracle_workflow	FALSE	Set to TRUE if Oracle Workflow is a integrated part of solution.
has_oracle_oid	FALSE	Set to TRUE if users are maintained and integrated from a LDAP directory. When FALSE, the user is maintained fully in iKnowBase
has_oracle_ordsys	FALSE	When FALSE, use a Java based image scaling which is the preferred way of scaling images.

		Oracle Imtermedia uses internal Oracle functions and the may not be available on all versions of the Oracle Database.
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 wihout 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. It 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. It the function exists within a package, use the notation package.function
ignore_illegal_attrib_on_save	FALSE	Used for backward compability.
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

Duplicate an existing 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 test. Doing so is very simple:

```
$ iknowbase.sh IKB_PROD.properties exportFile IKB_PROD.dmp
$ iknowbase.sh IKB_TEST.properties importFile IKB_PROD.dmp IKB_PROD
```

In the example above, there are two property files. IKB_PROD.properties contain information about the production schema, while IKB_TEST contain information about the test schema. Data is first exported from the production schema, then imported into the test schema.

Note that the first time you do this, you will also need to run the createUser-command. Of course, if the IKB_PROD and IKB_TEST schemas reside on different Oracle-instances, you will also need to move the dump file between the instances, either using a combination of downloadFile and uploadFile, or by using some other file transfer mechanism.

Running iKnowBase in a Oracle Enterprise Edition database

By default, an iKnowBase installation is prepared for running in a Oracle Standard Edition database. It you are licened for running Oracle Database Enterprise Edition then run the following command to speed up queries displaying (or sorting) document popularity.

```
$ iknowbase.sh production.properties dbscript source/common/
mv_log_document.sql logfile
```

To switch back to the standard edition version do:

```
$ iknowbase.sh production.properties dbscript source/common/
view_log_document.sql logfile
```

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 compability reasons, but the overall goal is to convert all search functionality to SOLR. Meanwhile, you can create a lightweigh 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 production.properties dbscript source/common/
create_ctx_freetext.sql
```

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 production.properties dbscript source/common/
create_ctx_freetext.sql
```

6. Java applications

Overview

iKnowBase comes with two web applications that comprise the server side components of iKnowBase:

- The iknowbase-resource application contains static web resources, such as scripts, css-files and images. This is normally deployed to a web server in front of the application server or deployed directly to the application server itself.
- The iknowbase-webapp contains all iKnowBase Java web modules, which may be enabled or disabled in the configuration. This is deployed in a java servlet container.
  - The Batch module is the batch processing server.
  - The Instant module is the real time asynchronous messaging server for web clients.
  - The Studio module adds content administration, user administration and development capabilities to the Administration Console.
  - The Viewer module is the core runtime environment, serving web pages to end users.
  - The WebServices module is the Web Services-server, typically to programs and integration engines.
  - The WebDAV module is the WebDAV content server, enabling easy editing of Office documents stored in iKnowBase.
  - The iknowbase-webapp also serves as a host application for Java based plugins and patches.

Special requirements

- iknowbase-webapp application should be deployed with context root set to “/”.
- If you intend to use WebDAV, the iknowbase web application must be deployed with context root set to “/”.
- iKnowBase WebDAV module MUST be mounted to / and will serve all content for the configured WebDAV (virtual) hostnames.

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 production.properties setIkbPassword orcladmin
 secretpassword
$ ./iknowbase.sh production.properties webServer
```

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

- Documentation
- index.html
- Management Console (requires login)

For further configuration options, or if you want to install on an alternative application server, see the details for each of them:

- *Installation Guide > iKnowBase web server*
- _Installation Guide > Oracle WebLogic

Web application security

During or after deployment, you need to configure security for the application (NOT the same as content security). Again, the configuration method will vary from application server to application server.

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

Advanced topics

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.

Note that the WebDAV service will not work when deployed to a context root other than “/”.

Clustering

The iKnowBase application itself supports clustered deployment, but the “Instant” and “WebDAV” modules do not. If you intend to use those modules in an otherwise clustered environment, they must be deployed to a standalone server instance with no clustering.

In a clustered setup, load balancers must use session persistence / sticky sessions so that a single user session is always served by the same host.

7. The web application runtime module

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

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.

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.

8. 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.

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.defaultContentDisposition	Set to “inline” to have the browser try to open documents in-line, or to start Office for viewing. Set to “attachment” to have “save as” as the default browser behaviour.
com.iknowbase.contentServer.createResponseHea	Toggles whether to include HTML-headers for document information. Normally left unchanged.

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”.

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>.serverUrl	URL to the SOLR server, e.g. `http://solr.example.com/solr`.
com.iknowbase.searchEngine.<searchEngineNam	Max number of milliseconds to wait for the connection.
com.iknowbase.searchEngine.<searchEngineName>.maxSearchOperationTimeoutMillis	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.

Activiti BPM Platform

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

iKnowBase Process Studio capable of managing Activiti process definitions, running processes and tasks 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.mailServerDefaultFr	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.

9. 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. The batch server is implemented as a java module in the iknowbase-webapp application.

Currently, the Batch Server handles these services:

- The ContentIndexer service submits documents to a Solr search index for indexing.
- The EmailReader service fetches email from an email account, and submits to iKnowBase for processing
- The EmailSender service sends emails
- The FileConverter service converts various file formats to pdf, html or images. Note that the File converter service is only available for 64-bit Linux (the “x86_64” architecture).
- The ImageEditor service performs image operations such as resize, rotate, flip and crop.
- The PageEngine service runs a specific iKnowBase Page and returns the HTML result. Used by the newsletter module.

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.

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.dequeTimeout	Number of seconds each dequeue() shall wait before recycyling.
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 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:

com.iknowbase.searchEngine.<searchEngineNam	Type of index server. Currently “SOLR” is the only supported value.
com.iknowbase.searchEngine.<searchEngineName>.indexURL	URL to Solr server, e.g. `http://solr.example.com/solr/CoreName`.
com.iknowbase.searchEngine.<searchEngineNam	Max number of milliseconds to wait for the connection.
com.iknowbase.searchEngine.<searchEngineName>.maxIndexOperationTimeoutMillis	Max number of milliseconds to wait for the operation.

com.iknowbase.searchEngine.<searchEngineNam	Max number of seconds before the index server commits an update to the search index.

EmailReader

Most of the EmailReader configuration is performed in the Development Studio, where you define the various email accounts that you want to process, along with the pl/sql packages you want to use for processing the actual messages.

By default, any running iKnowBase batch server will process email messages. This is nice, unless you happen to have multiple BatchServers installed and running in parallel, which might lead to multiple batch servers accessing the same email account at the same time. This is a potential source of trouble, so we recommend that you configure the EmailReader so that only one instance is active at any time.

The *EmailReaderConfiguration* accepts these configuration properties:

Property name	Description
com.iknowbase.batch.emailReader.enabled	Toggles whether the emailReader is enabled or not.

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 fileConverter. The legal values are either “true” or “false”.
com.iknowbase.batch.emailSender.dequeTimeoutSe	Number of seconds each dequeue() shall wait before recycyling.
com.iknowbase.batch.emailSender.spawnPolicy	Decides when the pageEngine starts listening for a new message. Use “immediate” for parallel processing, or “delayed” for serial processing.

FileConverter

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

Note that the FileConverter service is licensed separately from the core iKnowBase product.

Understanding the FileConverter

The FileConverter installs as a service

Usage of the FileConverter works like this:

- In the database, a file (Word document, PowerPoint, etc) is put onto a Queue called BATCH_FILECONVERT_QUEUE, typically through the use of the database package BATCH_FILECONVERT_CLIENT.
- The FileConverter, running in the batch server, will receive this document from the Queue.
- 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.

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
```

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.enabled	Toggles whether to start the fileConverter. The legal values are either “true” or “false”.
com.iknowbase.batch.fileConverter.dequeueTimeoutS	Number of seconds each dequeue() shall wait before recycyling.
com.iknowbase.batch.fileConverter.spawnPolicy	Decides when the fileConverter starts listening for a new message. Use “immediate” for parallel processing, or “delayed” for serial processing.
com.iknowbase.batch.fileConverter.outsideInDirecto	Location of outside in installation. File Converter is disabled when this is not set.
com.iknowbase.batch.fileConverter.replyMessageExpiration	Number of seconds each reply message shall be valid, before expiring.

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.

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
```

ImageEditor

The ImageEditor service performs image operations such as resize, rotate, flip and crop. It is the preferred method for image operations and is set as default for new installations. An alternative method is available through the iKnowBase repository, see IKB_GLOBAL_PREFS.

The *ImageEditorConfiguration* accepts these configuration properties:

Property name	Description
com.iknowbase.batch.pageEngine.enabled	Toggles whether to start the fileConverter. The legal values are either “true” or “false”.
com.iknowbase.batch.pageEngine.dequeTimeoutSe	Number of seconds each dequeue() shall wait before recycyling.
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.replyMessageEx	Number of seconds each reply message shall be valid, before expiring.

Note that the most image editing features are currently managed from the database, which can also use Oracle ORDSYS for image manipulation. Use the *Database Global Preferences* to enable the use of the batch ImageEditor for image editing.

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 fileConverter. The legal values are either “true” or “false”.

com.iknowbase.batch.pageEngine.dequeTimeoutSe	Number of seconds each dequeue() shall wait before recycyling.
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.replyMessageEx	Number of seconds each reply message shall be valid, before expiring.

10. 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.

11. WebDAV module

iKnowBase comes with a WebDAV server serving iKnowBase documents to WebDAV Clients. This enables direct editing of Microsoft Office documents with a Microsoft Office client.

The WebDAV server is implemented as a java module in the iknowbase-webapp application.

Special requirements

License from Milton.io

The WebDAV server is built with Java WebDAV Server Library from Milton.io. A valid commercial license is required and can be purchased from Milton.io. The WebDAV server must be configured with path to a directory containing the license file.

WebDAV traffic is served from /

Due to client requirements, the WebDAV Server must be available at context root / (the root). Collision with the iKnowBase Viewer module (also deployed to /) is avoided using either User-Agent detection or a separate virtual host for WebDAV traffic.

By default, all known Microsoft Office User-Agents will be served by the WebDAV server (the expression is configurable). This detection mechanism is default and recommended.

For special requirements, you may also use a separate virtual host for the WebDAV traffic (e.g. using webdav.example.com, while other applications are deployed on intranet.example.com). When enabled together with the Viewer module, use the hostnames WebDAV configuration option.

SSL

SSL is strongly recommended and Microsoft Office will require SSL for some authentication types.

Clustering

The WebDAV module does not support clustering.

Installation

Review WebDAV Special Requirements and configure the WebDAV Server module accordingly.

The *WebdavConfiguration* accepts these configuration properties:

Property name	Description
com.iknowbase.webdav.enabled	Toggles whether the WebDAV server modules are available.
com.iknowbase.webdav.userAgentExpr	Regular expression matching the User-Agent headers the WebDAV server will respond to. Defaults to all known Microsoft Office User-Agents.
com.iknowbase.webdav.hostnames	Hostnames the WebDAV server will respond to. Comma delimited.
milton.license.dir	File system directory containing the license files for Milton WebDAV.

See Web Application Security for configuration options for form based authentication for WebDAV.

Authentication

The WebDAV module supports these authentication options in prioritized order (first enabled module wins):

- Header

- Spnego
- Form based (MS-OFBA protocol)
- Basic
- Container (Oracle WebLogic only)

For the iKnowBase web server, the default authentication option is “Form”. For Oracle WebLogic, the default is “Container”.

Presentation configuration

To set up direct editing of Microsoft Office documents, do the following:

- Using the iKnowBase Administration Console, configure the “WebDAV Path” in the relevant Domain configuration with the proper value, e.g. “https://webdav.example.com/”.
- In relevant content viewers, configure an “Edit using WebDAV” action.
- From a supported web browser, click the “Edit using WebDAV” icon.

Launching applications for editing

From iKnowBase 7.0, applications are launched using browser protocol extensions and *Microsoft Office URI schemes*.

If the document type has been registered as a WebDAV type and presentation has been configured to display “Edit using WebDAV”, the registered application will launch when the link is clicked. The link includes information regarding which application to launch.

NOTE: Before iKnowBase 7.0, applications were launched using the Microsoft SharePoint NPAPI plugin (SharePoint.OpenDocuments and ffSharepointPlugin). Support for this plugin has been removed in favor of browser protocol extensions.

WebDAV registration for file types

WebDAV registration for each type file type is managed in iKnowBase administration console under Mimetypes. “Edit using WebDAV” will only be available for types enabled for WebDAV. The associated WebDAV script must match the Microsoft Office URI Scheme for the application that should be used for editing the file.

Troubleshooting

For troubleshooting, do this:

- Find a suitable document in your iKnowBase installation, making note of the document ID. For this example, we assume a document ID of “2804”.
- Using a web browser, go to the url “https://webdav.example.com/ikb$content/2804/document.docx” (using the proper hostname and a suitable extension). Open (or save + open) this document, and verify that it opens properly in e.g. Microsoft Word.
- Using Microsoft Word directly, go to the “Open Document” dialog (Ctrl+F12 will often do the trick). Here, enter the same full URL, including the https-prototcol and all (“https://webdav.example.com/ikb$content/2804/document.docx”) and click “Open”. Verify that it opens properly.
- Make sure that the plugin is not blocked by the browser (in such case you will be prompted)

Microsoft Office 2011 for Mac requires SSLv3 protocol support

Microsoft Office 2011 for Mac requires that the old SSLv3 protocol is enabled. Web sites will typically want to remove support for SSLv3 due to the SSLv3 vulnerability “POODLE”, but this will result in the message “No connectivity with the server” when opening a document using WebDAV.

The WebDAV web site must support SSLv3 until this issue is resolved.

iKnowBase reference: IKB-2867

Microsoft Office 2016 for does not currently support forms based authentication

Office 2016 for Mac clients are not compatible and will currently (by the time of this release) not be able to edit documents when using forms based authentication (MS-OFBA).

iKnowBase reference: IKB-3060

Re-authentication problems with form based login

Form based authentication for WebDAV uses persistent session cookies. The session is valid until server restart or idle timeout.

When a presented session cookie is invalid, Microsoft Office will say that there is a permission problem and offer to save a copy. If you continue with the save copy dialog, you will be able to re-authenticate and save your document.

Note: Microsoft Office also offers the sign in option when a permission error occurs. However, this option will not try to re-authenticate when you still have a persistent cookie.

Note: A restart of the Microsoft Office application will prompt for new authentication.

Word did not save the document (0x80004005)

When a form based authentication is invalid and you re-authenticate (see previous section), this error message may occur if you cancel the re-authentication dialog: “Word did not save the document” with the error code 0x80004005 in event viewer. A hotfix is available at http://support2.microsoft.com/kb/2479169.

Browser warning when launching application for direct editing

By default, all browsers will display a warning when clicking the “Edit using WebDAV” (or similar) link.

Chrome displays an option to remember the decision and further warnings can then easily be avoided.

The warning in Firefox and Internet Explorer can be avoided by adding the WebDAV hostname to Internet Explorer’s Trusted Zone. Note that if you run WebDAV on a separate hostname, then only that hostname is required to be in the Trusted Zone.

Warning: While Intranet Zone will have Compatibility View and Automatic Windows login enabled by default, the Trusted Zone will not. You may add the site to Compatibility View manually (if required) and enable Automatic Windows login by setting Trusted Zone custom level to “Automatic login with current user name and password”.

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.endpointRoot	The web services endpoint root.
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

13. Instant module

iKnowBase comes with client and server side support for creating applications with real time asynchronous messaging support. The Instant server is implemented as a java module in the iknowbase-webapp application.

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

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 SecureTokenEngine *InstallationGuide#SecureTokenEngine* to enable web client single sign on between iKnowBase Viewer and Instant. An alternative without single sign on between iKnowBase Viewer and Instant is to use explicit login – see *Authenticating with direct application container authentication*.

Special requirements

The connected web-clients will be “always connected” using asycnchronous 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.

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 miliseconds 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.
com.iknowbase.instant.cleanupDisconnectedConnectionsThreshold	How long a connection needs to be in disconnect inactivity queue before it is examined and validated.
com.iknowbase.instant.broadcastDelayUserListUser	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.broadcastDelayServerReque	Optional delay between a serverRequest and the issued broadcast. Default no delay.

Note: Make sure you also configure SecureTokenEngine to enable authentication for web clients

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.

Testing and troubleshooting

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

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 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.

WebLogic: WARN AtmosphereFrameworkInitializer – WebLogic 12c unable to retrieve Servlet. Please make sure your servlet-name is 'AtmosphereServlet' or set org.atmosphere.servlet to the current value

This warning can safely be ignored.

WebLogic: Log messages related to BlockingIO

The current Atmosphere library incorrectly switches to BlockingIO if JSR356AsyncSupport is not available.

```
ERROR   DefaultAsyncSupportResolver - Failed to create comet support
 class: class org.atmosphere.container.JSR356AsyncSupport, error:
 java.lang.reflect.InvocationTargetException
```

```
ERROR   DefaultAsyncSupportResolver - Switching to BlockingIO
```

This has been registered as Atmosphere Issue 2018 and fixed for future versions. iKnowBase 7.0 ships with a workaround to avoid BlockingIO and you should see the previous ERROR messages immediately followed by:

```
WARN    IKBAsyncSupportResolver - Atmosphere issue 2018 detected. Overriding
 asyncSupport class from BlockingIOCometSupport to Servlet30CometSupport to
 avoid Blocking IO
```

14. Web Application Security

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 are installing on Oracle WebLogic, the default will be

- HTTP basic username and password authentication against the internal WebLogic user repository.

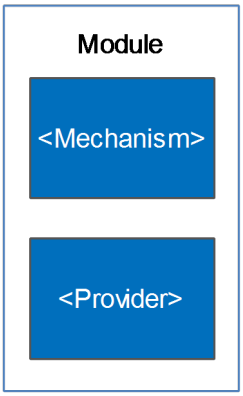
If you need to customize authentication, the basic steps are

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

Overview

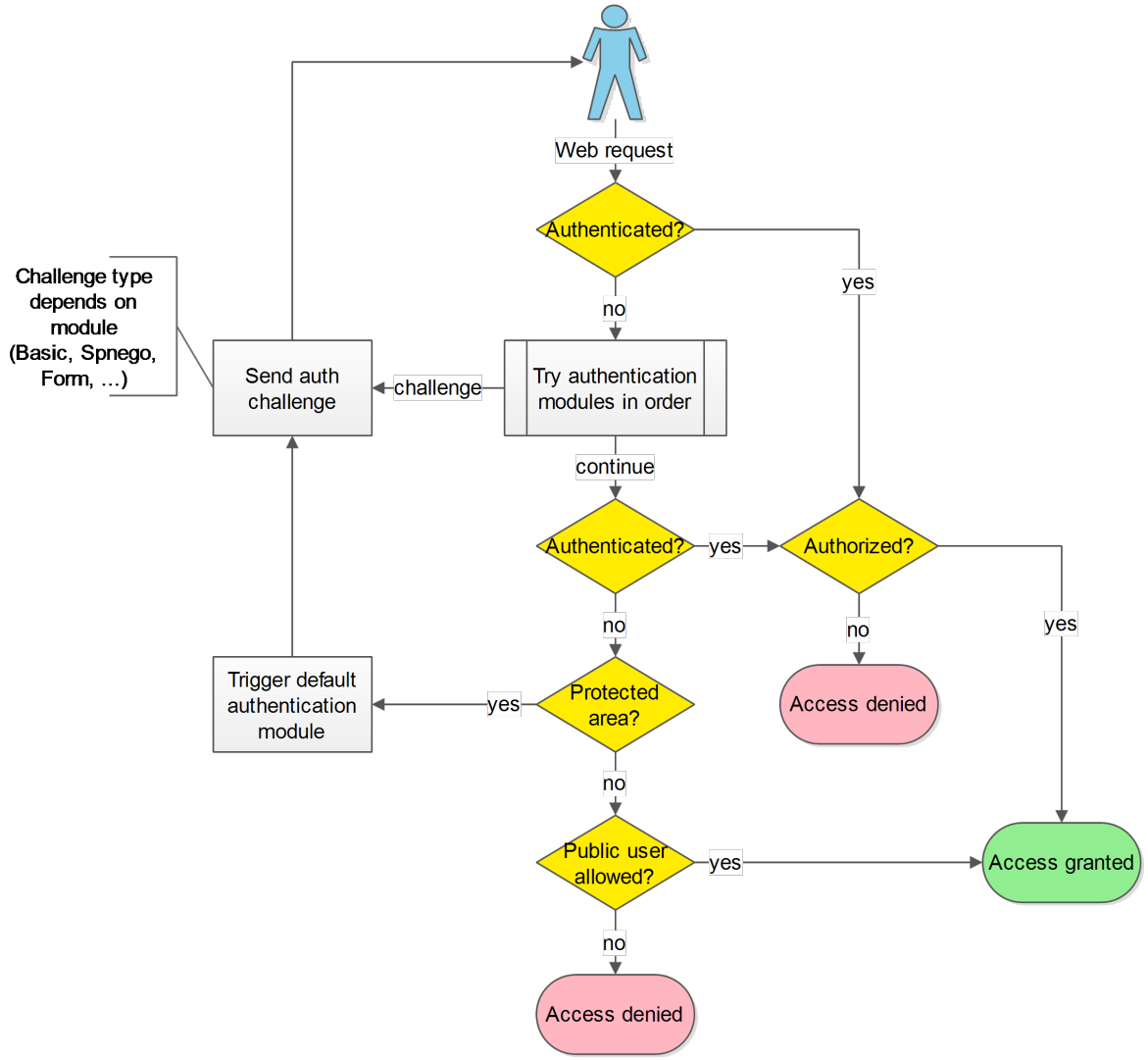
The iKnowBase web application security implementation is based on the Spring Security framework and takes care of user authentication and authorization to specific web application protected areas.

An authentication module combines authentication methods with authentication providers.



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.

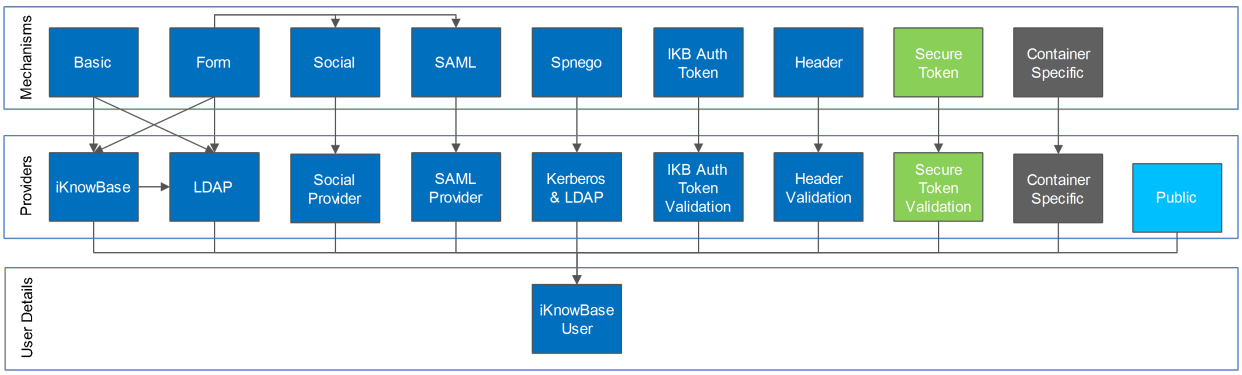
The authentication process follows this flow:



The authentication modules may authenticate the user based on the incoming web request or start a 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:



- The Saml module may be triggered from the form based login module.
- The Social module is triggered from the form based login module.
- The Secure Token module is intended for application to application authentication.

- The Container Specific module delegates to the application server.
- The Public module triggers if none of the other modules have authenticated the user.

Configuration

Web Application Security is configured using *configuration properties*.

See *iKnowBase Installation Guide* > *Configuration* > *Spring Security Configuration* for detailed configuration options.

Authentication

Default authentication module

When the users enters 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.
Oracle WebLogic	*	Container	Container dependant	Container dependant	

Instant

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

1. Header
2. Spnego
3. 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.

WebDAV

Form based authentication for WebDAV uses the MS-OFBA protocol. See WebDAV specific configuration options for form based authentication.

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 the either 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”.

Available authentication modules

Available authentication mechanisms:

Authentication mechanism	Path trigger	Header trigger	Can be used as default	Authentication provider	Description

Basic	/ikb$auth/basic/authenticate	X-IKB-AUTH: Basic	YES	UsernamePassword	Basic authentication for username and password.
Container	/ikb$auth/container/authenticate	X-IKB-AUTH: Container	YES (WebLogic only)	Container	Container based authentication (depends on security setup in the application server).
Form	/ikb$auth/form/authenticate	X-IKB-AUTH: Form	YES	UsernamePassword, Social, Saml	Form based authentication.
FormWebdav	/ikb$auth/formWebdav/authenticate	X-IKB-AUTH: FormWebdav	YES (WebDAV only)	UsernamePassw Social, Saml	Form based authentication for WebDAV (MS-OFBA protocol authentication page traffic).
Msofba	/ikb$auth/msofba/authenticate	X-IKB-AUTH: Msofba	YES (WebDAV only)	UsernamePassword, Social, Saml	Form based authentication for WebDAV (MS-OFBA protocol WebDAV traffic).
AutoForm	/ikb$auth/autoform/authenticate	N/A	YES	UsernamePassw	Form based authentication using an autogenerated login form (handy as a backup if the normal login form is out of order).
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$auth/saml/authenticate	X-IKB-AUTH: Saml	YES	SAML2 identity provider	SAML2 based authentication with SAML2 compatible identity providers (i.e. ADFS, Feide, Salesforce).
Google	/ikb$auth/google	N/A	NO	Social connection	Uses form based module.

					Will verify a preexisting social connection (created using account activation link).
Twitter	/ikb$auth/twitter	N/A	NO	Social connection	Uses form based module. Will verify a preexisting social connection (created using account activation link).
Facebook	/ikb$auth/facebook	N/A	NO	Social connection	Uses form based module. Will verify a preexisting social connection (created using account activation link).
LinkedIn	/ikb$auth/linkedin	N/A	NO	Social connection	Uses form based module. Will verify a preexisting social connection (created using account activation link).
Microsoft Live	/ikb$auth/live	N/A	NO	Social connection	Uses form based module. Will verify a preexisting social connection (created using account activation link).
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.

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.

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 web site (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 Social (social infrastructure is required by the SAML module)
- 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 user name 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 social 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$auth/logout for local logout.
/ikb$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

Social capable Providers

An external social account can be linked to an existing iKnowBase user account and used for authentication. iKnowBase supports multiple social providers.

The social authentication module requires that “Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files” are installed. Download and install “Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files” from http://www.oracle.com/technetwork/java/javase/downloads/index.html.

When enabling social authentication, you are required to set password and salt used for encrypting the secret user tokens received from the social provider.

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

Some social providers require that you specify the OAuth 2.0 Redirect URL. Use `<absolute url to site><iknowbase-webapp contextPath>/ikb$auth/<providerid>`. Example: `https://www.example.com/ikb$auth/google` or `https://www.example.com/ikb$auth/google`.

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 provide the user name 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.

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 PL/SQL IKB_AUTH_API and used as URL parameter “ikbAuthToken”.

Example: 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 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 a external (social or saml) account from one of the installed providers to the newly created iKnowBase account and authenticate using external (social or saml) authentication.

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

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

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 paths (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
Social Login	HTTP request for social login	/ikb$auth/<socialProvider>
Secure Token	_ikbUserToken HTTP header or URL param	NO
Header	HTTP header: [Configurable]	NO
Spnego	HTTP header: Authorization: Negotiate	NO
Container	Container dependant	Container dependant
Form	HTTP POST j_username, j_password	/j_spring_security_check
Basic	HTTP header: Authorization: Basic	NO
Saml	SAML2 messages	/ikb$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.

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 web applications have the following access requirements:

Application	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 WebDAV	Default	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 *iKnowBase Installation Guide > Configuration > Web Services Security Configuration* 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 Guide > Access-Control-Lists*.

Administrator

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

Development toolkit

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

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

Switch user

An authorized user may switch to a different identity. This can be used to greatly speed up troubleshooting as a 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

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.

Switch user audit procedure

Whenever a user switch to or exit 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
);
```

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

Trigger switch user

When switch user has been enabled and configured, a user can switch by accessing the path `/j_spring_security_switch_user?j_username=[USERNAME TO SWITCH TO]` and exit with path `/j_spring_security_exit_user`. Both services supports 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.

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-webapp multiple times, then each will have a `/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.

Custom security implementation

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

Examples

This section covers common setup scenarios and explains the necessary setup. Refer to the *Configuration* section for detailed configuration names and options.

WebLogic: Note that container mode for WebLogic is supported and you may also as an alternative accomplish the required authentication using WebLogic's own security modules. These examples does not cover detailed WebLogic container configuration.

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 production.properties setIkbPassword orcladmin SECRETPASSWORD
```

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

Form based authentication against iKnowBase User Repository

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

WebLogic only:

- Change the default authentication module to: Form

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 [contextPath]/j_spring_security_check

Login form Social requirements:

- Form action must be [contextPathForIKnowBaseWebApp]/ikb$auth/<social_provider>
- MUST have a input named “scope” for specifying social provider permissions.

Login form RememberMe requirements:

- RememberMe input MUST be named "_spring_security_remember_me"

An authenticated sessions is valid for one web application only. If you deploy multiple applications, e.g. iknowbase-webapp-1, iknowbase-webapp-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-webapp 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.

Basic authentication against iKnowBase User Repository

- Change the default authentication module to: Basic

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"

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”.

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"

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 "intranett.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 a 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 “intranett.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 interoperatibiliy, make sure that it can reach the hosts “www” and “intranett”:

```
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"

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.
  - NOTE: AES 256 requires that Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files are added to the JVM.

On the active 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 presice 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)
```

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.enabled	true
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
com.iknowbase.spring.security.ldap.clientUserDn	CN=iknowbase.webserver-01.example.com,CN=Users,DC=ad,DC=
com.iknowbase.spring.security.ldap.clientUserPassv	SecretPassword!GlobalMilk
com.iknowbase.spring.security.ldap.userSearchBase	CN=Users
com.iknowbase.spring.security.ikbauth.enabled	false (optional if you don’w want fallback to iKnowBase User Repository)

Optionally, specify the LDAP sync profile. If this is specified, user synchronization will happend during login to add new users automatically. If it is ommited, a user who tries to log on must already exist in iKnowBase (typically through a scheduled LDAP Synch). 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.profileExte	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.

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.

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.

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.

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

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

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.requestHeaderValueRegExp	`.*Firefox/25.*`

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 agains the enabled UsernamePassowrd providers (LDAP, iKnowBase).

If you want fallback to Basic authentication instead, set

Property name	Value
com.iknowbase.spring.security.spnego.sendAdditionalHeader	httpBasicChallenge
com.iknowbase.spring.security.spnego.fallbackRedirect	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.

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]`.

Integrating with Oracle SSO 10g

The Header authentication module can be used if you want to integrate with the Oracle SSO 10g platform. When using the Header authentication module it is extremely important that the trusted HTTP header is guaranteed by the reverse proxy in front of the iKnowBase web applications.

Guarantee integrity of HTTP server Osso-User-Dn

The header Osso-User-Dn must be guaranteed by the reverse proxyies in front of iKnowBase. The Oracle HTTP Server 10g does not have the capabilities to guarantee that the client cannot send this header, so you must have an additional reverse proxy in front of the Oracle HTTP Server 10g that removes the HTTP header Osso-User-Dn from all incoming requests.

Rely on Oracle HTTP Server OSSO plugin

The iKnowBase Web Application must be deployed to a server behind the Oracle HTTP Server with OSSO plugin and the following paths must be protected and require valid-user.

- /private
- /ikbInstant/private
- /ikb$console
- /ikb$content
- /ikb$runner
- /ikbWebServices

An example Oracle HTTP Server configuration:

```
NameVirtualHost *:7779
<VirtualHost *:7779>
    Port 7778
    ServerName example.iknowbase.com
    RewriteEngine On
    RewriteOptions inherit
    OssoConfigFile /app/oracle/asPortal/Apache/Apache/conf/osso/
osso_example.iknowbase.com.conf
    OssoIpCheck off
    <LocationMatch "^(/private|/ikbInstant/private|/ikb\$console|/ikb\
$content|/ikb\$runner|/ikbWebServices)">
        #HTTP Basic authentication
        AuthType Basic
        #We only require a valid user - no group/roles check
        require valid-user
    </LocationMatch>
    ProxyPass / http://ikb-server.example.iknowbase.com:8080/
    ProxyPassReverse / http://ikb-server.example.iknowbase.com:8080/
</VirtualHost>
```

Configure the iKnowBase Header authentication module

Base configuration:

Property name	Value
com.iknowbase.spring.security.header.enabled	true
com.iknowbase.spring.security.header.headerName	Osso-User-Dn
com.iknowbase.spring.security.header.headerNameUserExtractor	/usr=([^,]+)/$1
com.iknowbase.spring.security.header.sendAddition	true
com.iknowbase.spring.security.header.unauthorizedStatusCode	499
com.iknowbase.spring.security.header.unauthorized	Osso-Paranoid:true

If users can access the iKnowBase server directly over the network, you should add additional configuration to ensure the trusted requests must come from the Orache HTTP Server's IP address:

Property name	_Value
com.iknowbase.spring.security.header.realRemoteAddrExpr	[IP Address to Oracle HTTP Server]

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 web site (may be terminated in front of iKnowBase)

Enable social infrastructure

SAML connection leverage the social connection infrastructure, so we need to configure and enable that:

Property name	_Value
com.iknowbase.spring.security.social.enabled	true
com.iknowbase.spring.security.social.encryptionPas	<A password for sosial token encryption.>
com.iknowbase.spring.security.social.encryptionSalt	<Even number of 8 or more hex characters.>

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.de	myprivatekeyalias1 (must be lower case)
com.iknowbase.spring.security.saml.keyManager.pri	myprivatekeyalias1 (keyIndex = 1) (must be lower case)
com.iknowbase.spring.security.saml.keyManager.pr	<password for myprivatekeyalias1> (keyIndex = 1)
com.iknowbase.spring.security.saml.keyManager.sto	<path to keystore/keystore.jks>
com.iknowbase.spring.security.saml.keyManager.st	<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).

Register ADFS identity provider with iKnowBase

Download the ADFS metadata from https://YOUR_ADFS_SERVER/FederationMetadata/2007-06/FederationMetadata.xml and store the file 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>.displayFormName	<Name to display on form for login button>
com.iknowbase.spring.security.saml.metadataProvider.<index>.idpEntityID	<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>.idpMetadataFile	<path to ADFS XML metadata>

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).

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

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.

Switch user database procedures

This is an example implementation of the switch user procedures.

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;
/
```

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;
/
```

Enable Social authentication with user activation link

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

- Register an application at the social provider(s) to get key and secret for using the social provider's API.
  - We only need the minimal set of permissions for authentication.
- Configure iKnowBase authentication modules and restart iKnowBase web application:

Property name	_Value
com.iknowbase.spring.security.social.enabled	true
com.iknowbase.spring.security.social.encryptionPas	<A password for sosial token encryption.>
com.iknowbase.spring.security.social.encryptionSalt	<Even number of 8 or more hex characters.>
com.iknowbase.spring.security.social.<provider>.	true
com.iknowbase.spring.security.social.<provider>.clientId	<API key from the application registration>
com.iknowbase.spring.security.social.<provider>.	<API secret from the application registration>
com.iknowbase.spring.security.ikbauthtoken.activation	true enabled

- Test the authentication setup by generating an activation token for an existing user and access the web site on any area with the “ikbAuthToken=<your token>” URL parameter. Choose the social provider and authentication should complete.

When inviting users to activate / connect using a social 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

Troubleshooting

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 > Qualifier_

‘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
- Social authentication

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

Error creating bean with name 'aesBytesEncryptor'

For all these messages:

- Hex-encoded string must have an even number of characters
- The salt parameter must not be empty
- Constructor threw exception; nested exception is java.lang.NullPointerException

Both encryption password and salt must be configured in the social module.

The configured encryption salt set using configuration property “com.iknowbase.spring.security.social.encryptionSalt” must be set to a non empty even number of hex characters.

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.

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

java.lang.IllegalArgumentException: encryptionPassword is required

SAML require the social connection infrastructure. Please review documentation for the ADFS sample and enable social login (no external social providers needed).

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.

WebLogic: Basic authentication is not validated by iKnowBase Spring Security

WebLogic will by default intercept and validate HTTP Basic credentials even if the resource is not protected by WebLogic (iKnowBase is not). To allow HTTP basic credentials to pass through WebLogic to iKnowBase Spring Security, see WebLogic documentation on *Understanding BASIC Authentication with Unsecured Resources*.

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`.

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.

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

Authentication modules

Module specific configuration is provided in the following sections.

Basic module configuration

No configuration options available.

Container module configuration

Property name	Description
com.iknowbase.spring.security.container.enabled	Enable or disable module. Should not be necessary to change this setting.

Form module configuration

Property name	Description
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.rememberMeTo	RememberMe token cookie validity in seconds.
com.iknowbase.spring.security.form.rememberMeTokenCleanupThreshold	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.rememberMeC	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.rememberMeC	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.exceptionMappingAuthentication	Authentication exception to url mapping. `Exception full class name:url to redirect to`.
com.iknowbase.spring.security.form.webdavFormLo	Enable (true) or disable (false) form based login for WebDAV. Defaults to false.
com.iknowbase.spring.security.form.webdavFormLoginPageURL	Login URL (absolute or relative) for the form based login page. Defaults to standard form login page with RememberMe disabled.
com.iknowbase.spring.security.form.webdavFormLo	Login URL (absolute or relative) with redirect to success URL for WebDAV.
com.iknowbase.spring.security.form.webdavFormLogonSuccessURL	Success URL (absolute or relative). MUST match the URL the user is redirected to if authentication succeeds.
com.iknowbase.spring.security.form.webdavFormLo	Dialog size for the form authentication dialog. Defaults to “800x600”, i.e. width=800px,height=600px.
com.iknowbase.spring.security.form.webdavFormLoginUserAgentExp	Regular expression for matching browser’s User-Agent for WebDAV form based authentication. Defaults to all Microsoft Office clients.

FormAuto module configuration

No configuration options available.

Header module configuration

Property name	Description
com.iknowbase.spring.security.header.enabled	Enable (true) or disable (false) module.
com.iknowbase.spring.security.header.serverNameR	Regular expression restriction matching against the host name of the server to which the request was sent.
com.iknowbase.spring.security.header.remoteAddrExp	Regular expression restriction matching against the end client's IP address.
com.iknowbase.spring.security.header.realRemoteA	Regular expression restriction matching against the immediate client's IP address (typically a reverse proxy). ONLY available for iKnowBase web server.
com.iknowbase.spring.security.header.headerNameUser	Name of HTTP request header containing the authenticated username.
com.iknowbase.spring.security.header.headerName	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.headerValue	Value of the shared authentication secret (if any).
com.iknowbase.spring.security.header.authSchemeName	The scheme name in use when the server sends an authentication challenge.
com.iknowbase.spring.security.header.sendAddition	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.unauthorized	Pipe delimited set of response headers.

Spnego module configuration

Property name	Description
com.iknowbase.spring.security.spnego.enabled	Enable (true) or disable (false) module.
com.iknowbase.spring.security.spnego.serverNameR	Regular expression restriction matching against the host name of the server to which the request was sent.
com.iknowbase.spring.security.spnego.remoteAddrExp	Regular expression restriction matching against the end client's IP address.
com.iknowbase.spring.security.spnego.realRemoteA	Regular expression restriction matching against the immediate client's IP address (typically a reverse proxy). ONLY available for iKnowBase web server.
com.iknowbase.spring.security.spnego.requestHeaderName	HTTP request header (if any) that must be present on the request for spnego to be enabled.
com.iknowbase.spring.security.spnego.requestHead	Regular expression for validating the specified HTTP request header.
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]).
com.iknowbase.spring.security.spnego.sendAdditionalAuthenticate	Also send a HTTP Basic authentication challenge.
com.iknowbase.spring.security.spnego.fallbackRedi	Fallback to form based authentication instead of authentication challenge if spnego cannot be completed.

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.
com.iknowbase.spring.security.ldap.clientUserDn	Full DN for client ldap user used for user lookup.
com.iknowbase.spring.security.ldap.clientUserPassv	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.usernameAttribu	Attribute matching the authentication username submitted by the web client.
com.iknowbase.spring.security.ldap.ikbUsernameAttribute	Attribute used as iKnowBase authenticated user.

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.profileExte	External key to LDAP synchronization profile defined in iKnowBase.
com.iknowbase.spring.security.ldap.sync.executionUserName	Execution user. Should normally not be changed.

iKnowBase UsernamePassword authentication provider

Property name	Description
com.iknowbase.spring.security.ikbauth.enabled	Enable (true) or disable (false) module.

SAML authentication provider

See also: *Spring Security SAML documentation*.

Property name	Description
com.iknowbase.spring.security.saml.defaultFailureUr	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.idpSelectionURL	URL presenting available SAML identity providers for login if not using the default or custom login form
com.iknowbase.spring.security.saml.keyManager.de	Default key alias to use for signing and encryption
com.iknowbase.spring.security.saml.keyManager.privateKey.index	Alias for the private key in the key store. Configuration index >= 1.
com.iknowbase.spring.security.saml.keyManager.pr	Password for the private key in the key store. Configuration index >= 1.
com.iknowbase.spring.security.saml.keyManager.storeFile	Path to the Java keystore containg private key(s).
com.iknowbase.spring.security.saml.keyManager.st	Password for accessing the Java keystore.
com.iknowbase.spring.security.saml.loginDefaultTarget	Default URL to redirect to if no saved request was found after successful login. Defaults to /
com.iknowbase.spring.security.saml.logoutDefaultTa	Default URL to redirect to if no saved request was found after successful logout. Defaults to /
com.iknowbase.spring.security.saml.metadataProvider	Enable (true) or disable (false) provider.

com.iknowbase.spring.security.saml.metadataProvic	If true, received artifactResolve messages will require a signature, sent artifactResolve will be signed.
com.iknowbase.spring.security.saml.metadataProvider.requireLogoutRequestSigned	If true, received logoutRequests will require a signature, sent logoutRequests will be signed.
com.iknowbase.spring.security.saml.metadataProvic	Flag indicating whether entity in question requires logout response to be signed..
com.iknowbase.spring.security.saml.metadataProvider.alias	Alias for the extended specific service provider (if more than one).
com.iknowbase.spring.security.saml.metadataProvic	Alias to private key used for encryption
com.iknowbase.spring.security.saml.metadataProvider.enableExtendedMetadata	Enable (true) extended (false) identity provider discovery mode.
com.iknowbase.spring.security.saml.metadataProvic	Identity provider discovery response URL.
com.iknowbase.spring.security.saml.metadataProvider.identityProviderDiscoveryURL	Identity provider discovery URL.
com.iknowbase.spring.security.saml.metadataProvic	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.metaInitializerOptions	Extended options for parsing metadata 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.metadataProvic	Flag indicating whether local metadata will be digitally signed.
com.iknowbase.spring.security.saml.metadataProvider.signingAlgorithm	Algorithm used for signing of digital signatures of this entity. Only used for metadata signatures. Only valid for local entities.
com.iknowbase.spring.security.saml.metadataProvic	Signing key used for signing messages or verifying signatures of this entity.
com.iknowbase.spring.security.saml.metadataProvider.sslHostnameVerification	Hostname verification used for verification of SSL connections, e.g. for ArtifactResolution.
com.iknowbase.spring.security.saml.metadataProvic	Security profile used for SSL/TLS connections of the local entity. metaiop or pkix.
com.iknowbase.spring.security.saml.metadataProvider.sslKey	Key used to authenticate this entity 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.metadataProvic	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.autoConnect	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.metadataProvic	If no linked account could be found, redirect the user to this URL.
com.iknowbase.spring.security.saml.metadataProvider.iKnowBaseUsernameAttribute	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.metadataProvic	iKnowBase default login form: Specify css class for the login button.
com.iknowbase.spring.security.saml.metadataProvider.iknowbase.defaultLoginFormShowName	iKnowBase default login form: Specify name that will appear on the login button.
com.iknowbase.spring.security.saml.metadataProvic	EntityID, as specified in the provider's metadata XML.
com.iknowbase.spring.security.saml.metadataProvider.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.metadataProvic	If enabled, verifies metadata signature using pkix signature trust engine.
com.iknowbase.spring.security.saml.metadataProvider.type	Type of provider. SP = Service Provider. IdP = Identity Provider
com.iknowbase.spring.security.saml.metadataProvic	File system path to XML metadata for this provider.
com.iknowbase.spring.security.saml.metadataProvider.hostedSpName	Name of SP default service provider for this application. Must match Entity ID of the service provider.
com.iknowbase.spring.security.saml.metadataProvic	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.

Social authentication provider

Property name	Description
com.iknowbase.spring.security.social.enabled	Enable (true) or disable (false) module.
com.iknowbase.spring.security.social.socialSignupU	Signup URL (absolute or relative) to redirect the user to if the social authentication attempt did not match an iKnowBase user account.
com.iknowbase.spring.security.social.socialProvidersEnabled	Comma separated list of providers. Used when rendering login and activation page. Defaults to all enabled providers.
com.iknowbase.spring.security.social.encryptionPas	Password used when encrypting social authentication tokens before they are persisted to database. Required if you enable social authentication.
com.iknowbase.spring.security.social.encryptionSalt	Salt, which must be an even number of 8 or more hex characters. Used when encrypting social authentication tokens before they are persisted to database. Required if you enable social authentication.

Google specific configuration:

Property name	Description
com.iknowbase.spring.security.social.google.enabled	Enable (true) or disable (false) module.
com.iknowbase.spring.security.social.google.clientId	Client ID registered with the social provider.
com.iknowbase.spring.security.social.google.clientSecret	Client Secret registered with the social provider
com.iknowbase.spring.security.social.google.scope	OAuth scope for requesting permissions with the social provider. Default will enable authentication.

Twitter specific configuration:

Property name	Description
com.iknowbase.spring.security.social.twitter.enabled	Enable (true) or disable (false) module.
com.iknowbase.spring.security.social.twitter.clientId	API Key registered with the social provider.
com.iknowbase.spring.security.social.twitter.clientSecret	API Secret registered with the social provider
com.iknowbase.spring.security.social.twitter.scope	OAuth scope for requesting permissions with the social provider. Default will enable authentication.

Facebook specific configuration:

Property name	Description
com.iknowbase.spring.security.social.facebook.enabled	Enable (true) or disable (false) module.
com.iknowbase.spring.security.social.facebook.clien	App ID registered with the social provider.
com.iknowbase.spring.security.social.facebook.clientSecret	App Secret registered with the social provider
com.iknowbase.spring.security.social.facebook.scop	OAuth scope for requesting permissions with the social provider. Default will enable authentication.

LinkedIn specific configuration:

Property name	Description
com.iknowbase.spring.security.social.linkedin.enabled	Enable (true) or disable (false) module.
com.iknowbase.spring.security.social.linkedin.clientI	API Key registered with the social provider.
com.iknowbase.spring.security.social.linkedin.clientSecret	Secret Key registered with the social provider
com.iknowbase.spring.security.social.linkedin.scope	OAuth scope for requesting permissions with the social provider. Default will enable authentication.

Microsoft Live / Microsoft account specific configuration:

Property name	Description
com.iknowbase.spring.security.social.live.enabled	Enable (true) or disable (false) module.
com.iknowbase.spring.security.social.live.clientId	API Key registered with the social provider.
com.iknowbase.spring.security.social.live.clientSecret	Secret Key registered with the social provider
com.iknowbase.spring.security.social.live.scope	OAuth scope for requesting permissions with the social provider. Default will enable authentication.

Secure Token authentication

Note: The iKnowBase Secure Token Engine configuration is described in SecureTokenEngine

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

Anonymous / Public authentication provider

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

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.executionTimeout	Execution time user. Should normally not be changed.

Switch User

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

User Account Activation

Property name	Value
com.iknowbase.spring.security.activation.showActivationSuccessURL	Activation success URL (absolute or relative) to redirect the user to if a valid activation token is presented.
com.iknowbase.spring.security.activation.activationF	Failure URL (absolute or relative) to redirect the user to if the user presented an invalid activation token.
com.iknowbase.spring.security.activation.ikbProvider	Enable users to use activation token to set the user's password in the iKnowBase User Repository.
com.iknowbase.spring.security.activation.ikbProvide	Override default password encryption algorithm when setting password.

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.en	Enable (true) or disable (false) processing of token type LOGIN.

15. 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/).

Installation

To install, do the following:

- a)Download and unzip the latest supported version of SOLR. e.g from https://lucene.apache.org/solr/downloads.html
- b)You should now have a top level structure like

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

- c) 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
```

- d) Create a new core based on a basic config set defined under `solr-<version>/server/solr`

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

- e) Copy files from the iknowbase-zip file to the new core

```
$ cd solr-<version>/server/solr/iknowbase
$ cd solr-<version>/server/solr/iknowbase/conf
$ cp ../../../iknowbase/conf/* .
```

- f) edit the file solrcore.properties.

```
jdbcUrl=jdbc:oracle:thin:@//<hostname>:<port>/<sid>
dbUsername=<username>
dbPassword=<password>
```

*g) edit the file `solr-<version>/server/solr/iknowbase/conf/solrconfig.xml`. Change the entry values to identify your db-connection and iKnowBase schema owner.

```
$ vi solr-<version>/server/solr/iknowbase/conf/solrcore.properties
```

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`

*h) Start solr and make a verification

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

Verify it starts without any errors and access `<hostname>:8389` from a browser. Make sure the core iKnowBase is available.

Upgrade an existing SOLR instance

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

- Run step a), b), c) and d) as descriped in chapter Installation.

- e) Copy files from the former solr release to the new core

```
$ cd solr-<version>/server/solr/iknowbase
$ cp <former solr release>/solr/<core>/core.properties core.properties
$ cd solr-<version>/server/solr/iknowbase/conf
$ cp <former solr release>/solr/<core>/conf/solrconfig.xml .
$ cp <former solr release>/solr/<core>/conf/schema.xml .
$ cp <former solr release>/solr/<core>/conf/solrcore.properties .
$ cp <former solr release>/solr/<core>/conf/<additional files customized in
 the former solr release> .
```

- f) edit the file solrcore.properties.

```
$ vi solr-<version>/server/solr/iknowbase/conf/solrcore.properties
```

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`

- h) stop the former solr release
- g) Start the new solr version and make a verification

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

Verify it starts without any errors and access <hostname>:8389 from a browser. Make sure the core iKnowBase is available.

- h) reindex all documents. New versions of SOLR recommend a full reindex. This can be done from iKnowBase Studio (either from the event config or solr Configuration)

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 chkconfig tool.

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.

Configuration

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

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 handled by a secure token engine, which is itself configured in two steps. First, the secure key is stored in the 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); then the Solr plugin must be informed about this qualifier, so that it can load the same value.

Configure the security component by filling in proper values for the following properties in solrcore.properties:

Property name	Description
jdbcUrl	This points to the database where iKnowBase is installed, eg. jdbc:oracle:thin:@// hostname.example.com:portnumber/service_name.
dbUsername	The database-user where iKnowBase is installed.
dbPassword	The password to the iKnowBase database user.
instanceQualifier	The instance_qualifier used to look up the proper secure token engine configuration from installation_properties in the iKnowBase database.

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.

Configure the iKnowBase applications

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

16. iKnowBase web server

NOTE: This chapter assumes the iKnowBase database repository has been created, as outlined in *Quick Installation and upgrade overview*.

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.

Preparations

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

Configure the iKnowBase instance

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

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

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 production.properties webServer
```

Start a web-browser, and navigate to http://YOUR_SERVER_NAME:8080/ressurs/iknowbase (to see documentation), http://YOUR_SERVER_NAME:8080/index.html (for the viewer application), or http://YOUR_SERVER_NAME:8080/ikbStudio (for the Development Studio). The first two should open immediately, while the second one will require login, which requires configuration as shown below.

Deploy the applications

Using the iKnowBase web server, application deployments is specified in the same property file as all other configuration.

Default deployment

The iKnowBase program contains a default configuration that automatically deploys the applications: /ressurs, /. If this is what you want, you don't have to make any changes or configure any deployments.

Specify applications to deploy

If you want to customize the deployment, start by specifying which applications to deploy. The names are only used internally to the property file, but pick names that are easy to understand:

```
# Web server configuration
web.apps=webRessurs,webIkbWebApp
```

NOTE that webIkbWebApp is by default deployed to context root / and MUST be listed as the last application in the web.apps property.

Next, add two entries for each application. First add an entry for the application **path**, which is where the application will be mounted. Next add an entry for the application **WAR-file**, which is where the application itself is located (can be a war file or a directory containing war contents / exploded war). These entries will be the name of the application, followed by ".path" and ".war" respectively:

```
# Web server configuration
web.apps=webRessurs,webIkbWebApp
```

```
# Web application definitions
webRessurs.path=/ressurs
webRessurs.war=iknowbase-resources-7.0.4.war
webIkbWebApp.path=/
webIkbWebApp.war=iknowbase-webapp-7.0.4.war
```

Add custom applications

Often, you will want to add custom deployment, such as a your own resources (images, scripts, etc). This is done the exact same way, but note that the file-property is here used to point to a directory:

```
# Web server configuration
web.apps=webRessurs,webCustomResource,webIkbWebApp

# Web application definitions
webRessurs.path=/ressurs
webRessurs.war=iknowbase-resources-7.0.4.war
webIkbWebApp.path=/
webIkbWebApp.war=iknowbase-webapp-7.0.4.war
webCustomResource.path=/custom-resources
webCustomResource.war=/opt/iknowbase/production/custom-resources
```

Customizing the url mount point

As you can see in the examples, you can change the url mount point at will. Note that the iKnowBase web server will visit the mount points in the order they appear in the `web.apps` property, and that having the wrong order is important. Look at the partial example below:

```
web.apps=root,ressurs
root.path=/
ressurs.path=/ressurs
```

If a client asks for “/ressurs”, this will be captured by the application “root”, and served from there. Therefore, remember to set the longest paths to the beginning of the “web.apps” definition.

Defining virtual hosts

It will sometimes be required to have multiple applications deployed towards different hosts, for example to mount WebDAV at root of “webdav” and “webdav.example.com” and Viewer at the root of all other servers. This can be handled by defined virtual hosts for the required applications:

```
web.apps=webIkbWebApp,webIkbWebdav
webIkbWebApp.path=/
webIkbWebApp.war=iknowbase-viewer-webapp-7.0.4.war
webCustomResource.path=/custom-resources
webCustomResource.war=/opt/iknowbase/production/custom-resources
webCustomResource.vhosts=custom-resource.example.org, custom-resource
```

In the example below, when a client asks for “http://custom-resource/” or “http://custom-resource.example.org”, the request will be served by the webCustomResource application. Root requests to all other hosts (such as http://intranet.example.org or http://testserver.example.org) are handled by the iknowbase-webapp application, which has no particular host name affinity.

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

Configure SSL

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

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 a 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 `web.honorXForwardedHeaders` in the property file:

```
...
web.honorXForwardedHeaders=true
...
```

Configuring SSL listener in iKnowBase web server

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

```
...
web.ssl.port=8443
web.ssl.keyStorePath=iknowbase.keystore
web.ssl.keyStorePassword=<KEYSTORE_PASSWORD>
web.ssl.certAlias=<Optional certificate alias>
...
```

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>
```

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.

Advanced topics

Specify session cookie domain

By default, the iKnowBase web server 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”.

Note also that specifying a sesision cookie domain means that a single iKnowBase web server can only serve sites belonging to that domain. If you specify a session cookie domain of iknowbase.com means that the same server may no longer serve content from example.com. If you need this scenario, for example to serve http://www.example.com, http://intranett.example.com and http://www.iknowbase.com from the same iKnowBase repository, use two iKnowBase web servers (one for *.example.com and one for *.iknowbase.com) listening at different ports, and use a proxy server (such as Varnish or Apache Traffic Server) to direct requests appropriately.

```
...
web.sessionCookieDomain=example.com
...
```

Specify session cookie name

By default, the iKnowBase web server will use session cookie name `JSESSIONID_<web application name>`, e.g. `JSESSIONID_WEBIKBWEBAPP`. In case of session name collision you may change the application name or just change the session cookie name. Session cookie name can be set using:

```
...
# Change the session cookie to "JSESSIONID_CUSTOM_NAME" for web application
 "webIkbWebApp"
webIkbWebApp.sessionCookieName=JSESSIONID_CUSTOM_NAME
...
```

Specify work directory

During normal execution, the iKnowBase web server needs a directory for storing working files. This directory will be used for e.g. unpacking the web application war files, where each deployed application will get its own subdirectory under the work directory (such as “work/webapps/webIkbWebApp”). The default work directory is “./work” under the current directory, but this can be overridden in the iknowbase property file:

```
...
iknowbase.workDirectory=/vol/workfiles/iknowbase/prod-server
...
```

Note that it is not recommended to use the operating system temporary directory (/tmp) for this purpose. For example, Linux installations typically runs the program “tmpwatch” every day, cleaning up the /tmp directory.

Specify logs directory

The iKnowBase programs writes logfiles during execution, both for command line and web usage. By default, the logfile is written in a subdirectory “logs” under the work directory (i.e. “./work/logs”), but you can override this in the property file:

```
...
```

```
iknowbase.logDirectory=/var/log/iknowbase
...
```

Setting max form size

Max form size that can be submitted to the iKnowBase web server is by default set to 200 000 bytes. This is inteded to help in denial of service attack scenarios where malicious clients send huge amount of data. This limitation does NOT affect file uploads using `multipart/form-data`.

If you need to POST forms larger than 200 000 bytes override the max value per web application in the iknowbase property file:

```
...
[webAppName].formContentSize=200000
# webIkbWebApp.formContentSize=200000
...
```

Troubleshooting

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))))
```

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

See Setting max form size.

Session cookie collision

You may encounter a session cookie collision if you

- Deploy iKnowBase webIkbWebApp to an iKnowBase web server (e.g. host1:443)
- Deploy iKnowBase webIkbWebApp to a separate iKnowBase web server on the same host (same hostname seen by web client) and a different port (e.g. host1:8443)

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.

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.
bc..
"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.

17. Installing on Oracle WebLogic Server

NOTE: This chapter assumes the iKnowBase database repository has been created, as outlined in *Quick Installation and upgrade overview*.

Installation on Oracle WebLogic Server has the following tasks:

- Install the Oracle-software
- Configure WebLogic
- Create a data source and deploy to the target
- Configure web application security
- Deploy the Resource-directory and java web application to the WebLogic Server

Installation and configuration of WebLogic

Non-clustered:

Install a domain containing one or more managed servers.

Clustered:

Install a domain containing

- a cluster
- one or more managed servers supporting the cluster
- The WebDAV and Instant module does not support clustering. This requires an additional virtual host, separate managed server or a cluster containing only one managed server. Do one of
  - Route by vhost: a virtual host exclusively for the non clustered modules backed by one managed server (typically YOURHOSTNAME-nocluster.YOURDOMAINNAME, YOURHOSTNAME-webdav.YOURDOMAINNAME or YOURHOSTNAME-instant.YOURDOMAINNAME)
  - Route by server: a managed server not part of the cluster and route traffic to this instance using a reverse proxy.

Create and deploy data source

From the administrative console (http://localhost:7001/console), create a data source with

- Name: iknowbaseDS
- JNDI Name: jdbc/iknowbaseDS
- Supports Global Transactions: Disable
- Target: The cluster where the applications will be deployed.

Configure web application security

The default web application security mode for iKnowBase deployed to WebLogic is container mode, which means we'll rely on WebLogic for authentication. You may change to the other web application security modes at any time.

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

However, some of WebLogic's authentication modules like SPNEGO and SAML2 require special deployment descriptor and role protection not included in the standard web archive to function properly. Contact support for assistance if you require use of these WebLogic authentication modules. Note that iKnowBase's set of authentication modules includes both SPNEGO and SAML2.

If you are using container mode for authentication, start with adding the orcladmin user to the default realm's user repository (WebLogic internal is default, but may be set to other supported user repositories).

iKnowBase does not require any roles for these users, as authorization will be done based on the mapped user in the iKnowBase User Repository.

Deploy applications

Non-clustered

Deployment example using the WebLogic console:

- Select “deployments” in the domain structure.
- Select “install” in the “Summary of deployments” screen
- Select the relevant web archive.
- Choose to deploy as an application (and not as a library)
- Target applications:
  - A managed server or vhost
    - You may enable all modules on a single deployment, but if you use multiple non clustered managed servers, you must ensure that the WebDAV and Instant modules is enabled on only one manager server.

Clustered

Deployment example using the WebLogic console:

- Select “deployments” in the domain structure.
- Select “install” in the “Summary of deployments” screen
- Select the relevant web archive.
- Choose to deploy as an application (and not as a library)
- Target applications:
  - Deploy iknowbase-webapp to the cluster on / with all required modules except WebDAV and Instant.
  - Deploy iknowbase-webapp with a different name like “iknowbase-webapp-nocluster” to the vhost/ managed server on / with the WebDAV and Instant modules enabled (all the other modules may be disabled).
    - Configure WebDAV to respond to WebDAV specific hostnames (see WebDAV configuration)
    - Instant will respond to the /ikbInstant path.
- If you want, go to the tab “Configuration, General”, and set the context root.
- If you make configuration changes, and need to save a deployment plan, store it in a safe directory.

Clusters and session replication

The iKnowBase web application is configured to replicate HTTP sessions when targeted to an application server cluster. This enhances support for failover and reduces impact for the user during a failover scenario.

The session replication mechanism configured is

```
<session-descriptor>
<persistent-store-type>replicated_if_clustered</persistent-store-type>
</session-descriptor>
```

The persistent-store-type “replicated_if_clustered” requires a homogeneous deployment to the cluster. You cannot target applications with this setting to selected parts of the cluster.

The persistent-store-type can be changed with deployment plans. See weblogic.xml session descriptor element for available options.

Configure user realms (authentication)

See *iKnowBase Installation Guide* > *Web Application Security* for additional authentication options provided by iKnowBase.

The next sections discuss WebLogic Container Mode for authentication.

Using Oracle Internet Directory for authentication

WebLogic allows simple configuration of authentication providers. Configure as appropriate by following this procedure:

- Chose the realm to change
- Under “Providers”, “Authentication”, add the Oracle Internet Directory provider
- Set the control flag as required, normally to “sufficient” to allow users to log in if they exist in this provider
- Add further configuration values as required by your OID-configuration.

Using the iKnowBase user repository for authentication

If you do not require WebLogic container mode for authentication, we recommend switching to iKnowBase’s own authentication modules, see *iKnowBase Installation Guide* > *Web Application Security* for additional explanations.

If you require WebLogic container mode it is possible to authenticate directly against the iKnowBase User Repository, through the custom IKBAuthenticationPlugin supplied as part of iKnowBase.

Overview

When installed, this provider will lookup usernames and passwords from the IKB_USER-table in the iKnowBase database schema, where the passwords are stored in encrypted form (SHA1-hash algorithm). A user will be authenticated if the username matches the one in the database, and the hashed password from the database matches what the user enters.

Installation

To install the plugin, perform the following steps:

- Install “iknowbase-weblogic-plugin-7.0.4.jar” on your server, for example in the directory "/app/oracle/Middleware/wlserver_12.1/server/ext"
- Change “bin/setDomainEnv.sh” (or .bat), and add the plugin to the PRE_CLASSPATH.

```
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/ext/
iknowbase-weblogic-plugin-7.0.4.jar"
```

- Edit the weblogic realm definition, and add a new authentication provider of type “CustomDBMSAuthenticator”. Be sure to configure the following properties:
  - The “Control flag” should normally be “SUFFICENT”, to make sure that it is sufficient for the user to be authenticated through this provider only.
  - “Plaintext Passwords Enabled” should be set to false, since all passwords in the iknowbase database are encrypted.
  - “Data Source Name” should be set to a preconfigured data source, which points to the iKnowBase database. Note that this is the *name* of the data source, not the *jndi name*.
  - “Group membership searching” should be set to “limited”.
  - “Plugin class name” must be set to "com.iknowbase.weblogic.IKBAuthenticationPlugin"
- Make sure that there are no other providers installed above this one that are marked as “REQUIRED”. Typically, install this provider on top.
- Make sure that the data source you referred to above is deployed to all servers in the domain, since all servers will be using this new authenticator.

Troubleshooting

By default, the plugin does not write any log information. However, if the java system property “com.iknowbase.weblogic.IKBAuthenticationPlugin.log” is set to the value “true”, the plugin will log operations to standard out, which is normally captured into the server log file (AdminServer.log for the admin server). Enable the system property in the startup script, like this (note that this is two lines only; they are broken into multiple lines here for layout purposes).

```
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/ext/
iknowbase-weblogic-plugin-7.0.4.jar"
EXTRA_JAVA_PROPERTIES="-
Dcom.iknowbase.weblogic.IKBAuthenticationPlugin.log=true"
```

With this, you will see log output matching this:

```
<Sep 25, 2009 7:05:30 PM CEST> <Notice> <Security> <BEA-090082> <Security
 initializing using security realm myrealm.>
IKBAuthenticationPlugin.lookupPassword: username=weblogic
IKBAuthenticationPlugin.lookupPassword: Found password for user=weblogic
IKBAuthenticationPlugin.userExists: Searching for username=weblogic
IKBAuthenticationPlugin.userExists: Search for username=weblogic returns true
IKBAuthenticationPlugin.lookupUserGroups: Searching for username=weblogic
IKBAuthenticationPlugin.lookupUserGroups: Search for username=weblogic
 returns[]
```

Configure SSL

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

Terminating SSL in the application server

The procedures for terminating SSL directly in the application server can be found in the WebLogic documentation.

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 a HTTP header called “WL-Proxy-SSL” with the value "true"
- Configure the WebLogic application server domain to detect WebLogic Plugin headers. From the WebLogic console, select your domain, and then Configuration -> WebApplications. Here, enable the “WebLogic Plugin Enabled” setting.
- Verify this setup by loading /ikb$console/java/request using from a secure connection; the value “Requested URL” should indicate a https-scheme
- For more information, see http://fusionsecurity.blogspot.no/2011/04/ssl-offloading-and-weblogic-server.html.

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

```
<Virtualhost ...>
...
RequestHeader set WL-Proxy-SSL true
...
</Virtualhost>
```

In the WebLogic domain configuration (config.xml), you would find the following snippet:

```
<web-app-container>
  <weblogic-plugin-enabled>true</weblogic-plugin-enabled>
</web-app-container>
```

Terminating SSL using Apache with WebLogic Plugin

If you use the Oracle WebLogic Plugin for Apache, also set this parameter `WLProxySSLPassThrough On` to pass on the SSL information to Oracle WebLogic Server.

Troubleshooting

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))))
```

WARN – BEA-101388 – The ServletContext was passed to the ServletContextListener.contextInitialized method of a ServletContextListener that was neither declared in web.xml or web-fragment.xml, nor annotated with javax.servlet.annotation.WebListener

Related to activation of the iKnowBase Instant module on WebLogic. This warning can be ignored.

iKnowBase tracking reference: IKB-2893

WebServices: java.lang.NoSuchMethodError: oracle.xml.parser.v2.XMLDocument.setSkipNodeNameValidate

WebLogic 12.1.3 JDBC driver conflict. Note that 12c database drivers for JDBC SQL XML support are not required for 12.1.3. iKnowBase 6.6 and 6.7 documented JDBC SQL XML additions required for WLS 12.1.1 and 12.1.2, but that is not a required installation step for WLS 12.1.3.