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

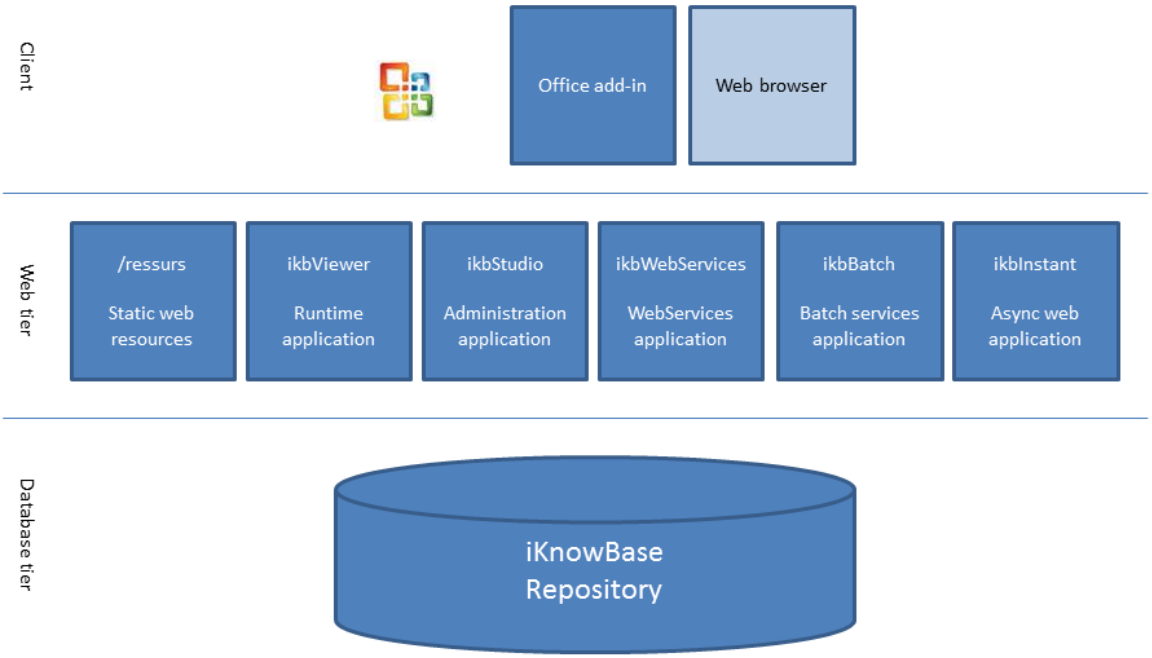
This book is conceptually structure into three parts:

- Part one (comprised of the chapters Installation toplogies, Installation overview, Upgrade overview and Configuration overview) gives an overview of the installation. For experience users, the installation and upgrade overview chapters will also contain 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



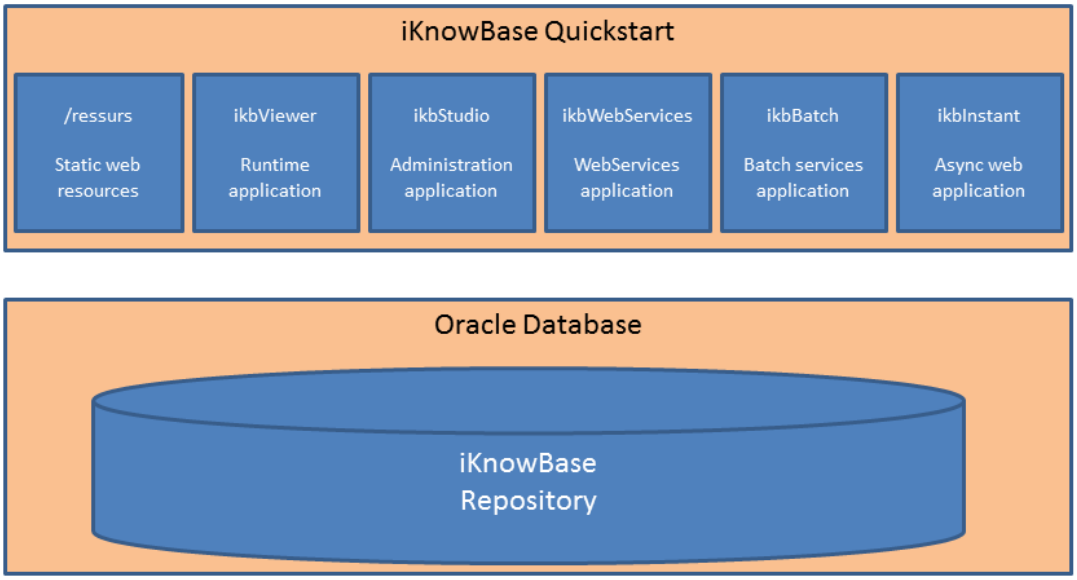
The diagram shows the various components of an iKnowBase installation:

- The database (“iKnowBase repository”) contains configuration, metadata and content. This is deployed to an Oracle Database.
- The resource directory (“/ressurs”) contain static web resources, such as scripts, css-files and images. This is normally deployed in a web server or an application server.
- The ikbViewer application (“/ikbViewer”) is the main run-time web application, responsible for serving pages and requests. This is deployed in a java servlet container.
- The ikbStudio application (“/ikbStudio”) is the main administrative web application, where iKnowBase-applications are developed and maintained. This is deployed in a java servlet container.
- The ikbWebServices application (“/ikbWebServices”) is the optional WebServices-server. When installed, it is deployed in a java servlet container.
- The ikbBatch application (“/ikbBatch”) is the optional batch processing server. When installed, it is deployed in a java servlet container.
- The ikbInstant application (“/ikbInstant”) is the optional real time asynchronous messaging server for web clients. When installed, it is deployed in a java servlet container.
- The Office add-in is an add-in to the Microsoft Office applications, enabling easy editing of Office documents stored in iKnowBase.

Sample topologies

iKnowBase Quickstart

This is the simplest and easiest way to set up installations. In that scenario, all the web tier components are preconfigured, and only the database needs to be set up separately.



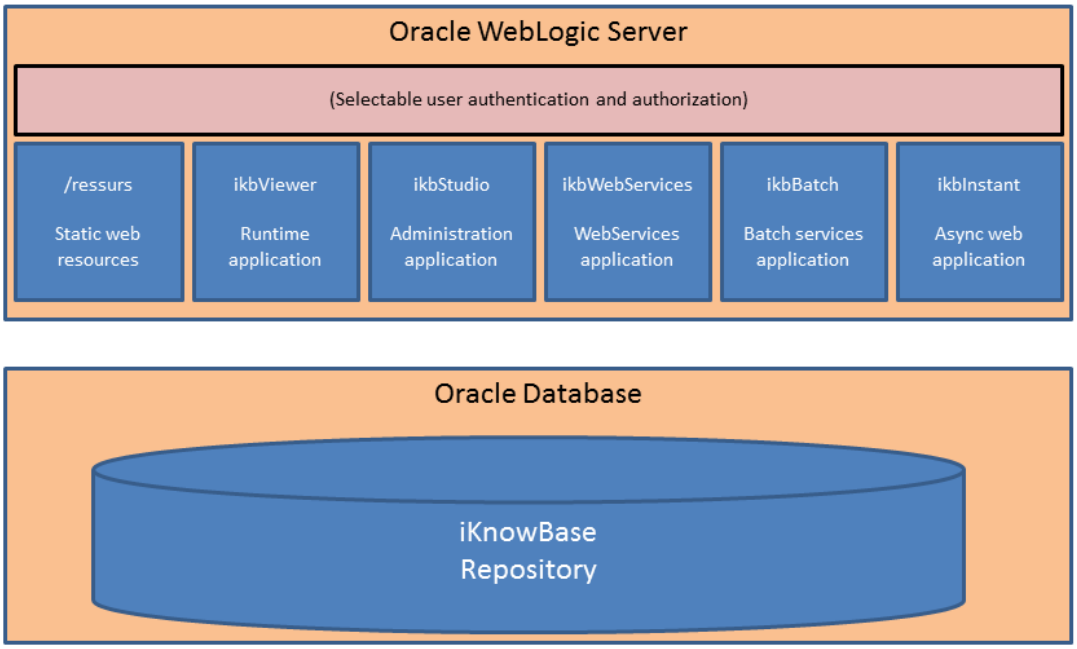
This diagram has really only two parts:

- iKnowBase Quickstart contains an embedded web server, hosting both static content (/ressurs) and JEE-applications
- Oracle Database hosts the iKnowBase repository

In this scenario, user authentication is handled by the embedded web server, looking up users and passwords from the iKnowBase repository itself.

Oracle WebLogic Server 12c (simple scenario)

This is a simple scenario based on WebLogic Server 12c. Here, the WebLogic server handles the HTTP traffic, servers static content and hosts JEE-applications:



This diagram has really only two parts:

- Oracle WebLogic hosting both static content (/ressurs) and JEE-applications
- Oracle Database hosts the iKnowBase repository

User authentication is handled by a pluggable authentication provider; users and passwords can be stored in many different forms and systems. Most used are Oracle Internet Directory, Microsoft Active Directory, a third party LDAP directory or the user information in the iKnowBase repository.

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 the software onto your application server
- Use the quickstart program to install or upgrade the database repository
- If you don’t want to use the quickstart web server, install the new or updated web applications into the proper application server

Download and install the iKnowBase software

Directory structure

We strongly 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
/opt/iknowbase/distributions	Collection of all distribution files, packed
/opt/iknowbase/iknowbase-6.4	Version specific directory for the unpacked software (old version)
/opt/iknowbase/iknowbase-@app.version	Version specific directory for the unpacked software (current version)
/opt/iknowbase/development	Directory for a given instance (example: “development”)
/opt/iknowbase/development/custom-resources	“development” instance customer specific static files (.js, .css, ...)
/opt/iknowbase/production	Directory for a given instance (example: “production”)
/opt/iknowbase/production/custom-resources	“production” instance customer specific static files (.js, .css, ...)

Download and install the software

Using the recommended directory structure above, install the iKnowBase software into the proper location.

```
mkdir /opt/iknowbase

mkdir /opt/iknowbase/distributions

mkdir /opt/iknowbase/development
mkdir /opt/iknowbase/development/custom-resources

mkdir /opt/iknowbase/production
mkdir /opt/iknowbase/production/custom-resources

cd /opt/iknowbase/distributions
# Copy iknowbase-6.6.zip to this directory (/opt/iknowbase/distributions)
cd /opt/iknowbase
```

```
unzip /opt/iknowbase/distributions/iknowbase-6.6.zip
# Note: the .zip-file contains top level directory "iknowbase-6.6"
```

For simplicity, and to avoid accidentally using the wrong version of the quickstart program, we also recommend creating a “quickstart.sh” script that forwards to the proper version:

```
# For development environment
cd /opt/iknowbase/development
# Run the following lines including line starting with "EOF" all in one
 comman
cat > quickstart.sh << 'EOF'
#!/bin/bash
../iknowbase-6.6/quickstart.sh $*
EOF

chmod +x quickstart.sh


# For production environment
cd /opt/iknowbase/production
# Run the following lines including line starting with "EOF" all in one
 command
cat > quickstart.sh << 'EOF'
#!/bin/bash
../iknowbase-6.6/quickstart.sh $*
EOF

chmod +x quickstart.sh
```

Configure the repository-specific property file

You will typically have multiple iKnowBase repositories in a single database, 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.

For each repository, create a property file with all the required settings for connecting to the database and running the embedded web server. The name can be generic (“iknowbase.properties”) or environment specific (“production.properties”), but that is entirely up to you. A sample is provided in etc/SAMPLE.properties. Copy the sample, edit the new file and set proper values for db.URL, db.sysPassword, db.ikbUser and db.ikbPassword.

- db.URL is the jdbc url to the database on the form "jdbc:oracle:thin:`//hostname:port/instance-name", it could typically be something like "jdbc:oracle:thin:`//localhost:1521/ORCL".
- 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.

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 = iknowbase
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:

```
cd /opt/iknowbase/production

./quickstart.sh production.properties createUser
./quickstart.sh production.properties uploadFile ../iknowbase-6.6/etc/
IKB_MASTER_66.dmp
./quickstart.sh production.properties importFile IKB_MASTER_66.dmp
 IKB_MASTER_66

# Optionally download and display import log
./quickstart.sh production.properties downloadFile IKB_MASTER_66.log .
cat IKB_MASTER_66.log

# Run upgrade scripts on the newly created installation
./quickstart.sh production.properties upgradeAll
```

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

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

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

Install and run web applications

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

Note that it is often advisable to configure the quickstart embedded web server even when you will be installing into a full application server, as this is useful for testing and troubleshooting.

Continue with the application server specific chapter:

- *iKnowBase Quickstart*
- *Oracle WebLogic*
- *Oracle GlassFish*

4. Configuration

This part of the book contains information on installation configurations that may be relevant.

Configuration concepts

Configuration of an iKnowBase installation is done through setting named *configuration properties* to specific values. Changes to configuration objects are static, and will require a restart of the application server. The values used by a running application are shown in various locations in the management console, e.g. at /ikb$console.

Property sources

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)

Often, the database table is most convenient, but you may also choose to use any of the available property source. Either way, you can and should verify that the correct values are set, using the ikb $console application as shown above.

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 ”/ikbViewer")
getSystemProperty(name)	Value of system property

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

Qualifier	Description

`hostname == 'tango'`	Used by any application running on a host named “tango”
`contextPath == '/ikbInstant'`	Used by the instant application
`directory == '/opt/iknowbase/sso'`	Used by applications running from the “/opt/iknowbase/sso”-directory
`true`	Used by any application, anytime
`hostname == 'tango' && contextPath == '/ikbBatch'`	Used by the batch application, when running on a host named “tango”
`hostname == 'tango' && contextPath matches '/ikb.*'`	Used by all “/ikb” prefixed applications, when running on a host named “tango” (/ikbBatch, /ikbStudio, ...)

Note that prior to iKnowBase 6.6, the qualifier had a different syntax which did not cater for advanced usage scenarios such as logical expressions. For backward compatibility, this syntax is still supported; however, this will be removed in the future, so we **strongly** recommend using the new expression style instead:

Qualifier	Description
`hostname=tango`	Used by applications running on a host named “tango”
`contextPath=/ikbInstant`	Used by the instant application
`user.dir=/opt/iknowbase/sso`	Used by applications running from the “/opt/iknowbase/sso”-directory
`*`	Used by any application, anytime

Configuring the ikbViewer application

This section describes the main configuration objects for the ikbViewer application.

ContentServerConfiguration

The Content Server is the unit responsible for serving file content, such as word documents or PDFs.

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.

PageEngineConfiguration

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

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. “searchEngineName” refers to the logical name given to the Solr Configuration.

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.

CacheManagerConfiguration

The iKnowBase system provides a content cache, based on Ehcache (http://www.ehcache.org). While Ehcache is used for many caching needs internally in iKnowBase, the content cache is special in that it is in fact a replicated cache: Values stored in the content cache is replicated to all other nodes in a cluster. For more information on Ehcache replication and the meaning of the configuration variables, see http://ehcache.org/documentation/distributed_caching_with_rmi.html.

Property name	Description
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.

SecureTokenEngineConfiguration

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

1. On-demand loading in an Oracle Portal-based Dimension Navigator.
2. Content loading when using the Office add-in.
3. Client authentication integration between ikbViewer and ikbInstant.

This token is generated using the HMAC_SHA1 message authentication algorithm, based on a secret key along with user and time information.

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. Please use a long phrase with multiple words, numbers and special characters.

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

ActivitiProcessEngineConfiguration

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.

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.

BpelProcessServicesConfiguration

When iKnowBase Process Services is installed, the ikbViewer runtime application(s) will need to know how to connect to the iKnowBase Process Services engine (ikbProcessServices), and how to log on to the underlying Oracle BPEL engine.

Property name	Description
com.iknowbase.ProcessServices.WorkflowService.initialContextFactory	Name of the Factory class to use for the InitialContextFactory when connecting to the Process Services Server
com.iknowbase.ProcessServices.WorkflowService.p	URL used to connect to the Process Services Server
com.iknowbase.ProcessServices.WorkflowService.securityPrincipal	Username used for connecting to the Process Services server. This username is used by the application server itself, and is often not a real user.
com.iknowbase.ProcessServices.WorkflowService.s	Password for the given user
com.iknowbase.ProcessServices.BPELWorkflowServ	The domain id for the BPEL-domain. The default value of “domain” is often correct.
com.iknowbase.ProcessServices.BPELWorkflowSer	The realm used by Process Services when connecting on to the BPEL server.
com.iknowbase.ProcessServices.BPELWorkflowServ	The username used by Process Services when connecting to the BPEL-server
com.iknowbase.ProcessServices.BPELWorkflowSer	The password used by Process Services when connecting to the BPEL-server.

The values for WorkflowService.initialContextFactory and WorkflowService.providerURL vary between application servers. Typical values are as follows:

Application server	Property	Format of typical value
Oracle WebLogic Server	WorkflowService.initialContextFactory	weblogic.jndi.WLInitialContextFactory
	WorkflowService.providerURL	t3://localhost:8001

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) for submitting username and password.
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.rememberMeCo	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.rememberMeCo	Set the cookie domain used for RememberMe tokens. Defaults is to send the cookie to the full host name that issued it.

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 QuickStart.
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.realmName	The realm name to use for basic challenge.
com.iknowbase.spring.security.header.unauthorized	HTTP response status code for authentication challenge.
com.iknowbase.spring.security.header.unauthorizedResponseHeaders	Response headers 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.serverName	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 QuickStart.
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.requestHeadValue	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.sendAdditionalUnauthorizedHeaders	Also send a HTTP Basic authentication challenge.
com.iknowbase.spring.security.spnego.realmName	The realm name to use for basic challenge.
com.iknowbase.spring.security.spnego.fallbackRedirectEnabled	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.userDnPattern	DN pattern for users within the base dn given in server url.
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.

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.socialProviders.enabled	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.

Secure Token authentication

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

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

Anonymous / Public authentication provider

Property name	Description
com.iknowbase.spring.security.authentication.ikbpubSharedKey	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.executionName	Execution 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/disable 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.

Configuring the ikbBatch application

This section describes the main configuration objects for the ikbBatch application.

FileConverterConfiguration

The ikbBatch application contains a file converter server, which can be configured to listen for file conversion requests. In order to work, you must install Oracle Outside In on the server, and then set the outside in location as a configuration property.

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.dequeTimeoutS	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.

BatchPageEngineConfiguration

The ikbBatch application contains a page engine server, which can be configured to listen for page rendering requests.

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.

ContentIndexerConfiguration

The ikbBatch application contains a content indexer server, which listens for indexing requests and forwards them to the appropriate search engine for actual indexing. “searchEngineName” refers to the logical name given to the Solr Configuration.

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.dequeueTimeout	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.
com.iknowbase.searchEngine.<searchEngineNam	Type of index server. Currently “SOLR” is the only supported value.
com.iknowbase.searchEngine.<searchEngineName>.indexURL	URL to index 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>.maxOperationTimeout	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.

EmailReaderConfiguration

The ikbBatch application contains an email reader client, which can load email from external mail servers. This can be enabled or disabled through configuration.

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

Configuring the ikbStudio application

There are no configuration items specific to the ikbStudio application.

Configuring the ikbWebdav application

This section describes the main configuration objects for the ikbWebdav application.

FtpServiceConfiguration

The ikbWebdav application contains an embedded FTP-server that lets you use FTP against all documents exposed through the Webdav protocol. This is enabled by setting the following properties:

Property name	Description
com.iknowbase.webdav.ftp.listenPort	Port number to listen for FTP-requests on.

Configuring the ikbWebServices application

This section describes the main configuration objects for the ikbWebServices application.

WebServicesConfiguration

Property name	Description

com.iknowbase.ws.mtomEnabled	Toggles whether MTOM is enabled for the SOAP endpoints.

WebServicesSecurityConfiguration

Property name	Description
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

Configuring the ikbInstant application

This section describes the main configuration objects for the ikbInstant application.

InstantServerConfiguration

The Instant Server is the contact point for Web Clients as well as managing all topics, clients, users and message delivery.

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

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.

Sample script

This is a sample script, showing how to set installation properties:

```
truncate table ikb_installation_properties;
insert into ikb_installation_properties (property_name, property_value)
select 'com.iknowbase.page.PageEngine.isContentCacheEnabled', 'true' from
 dual UNION
select 'com.iknowbase.ProcessServices.WorkflowService.InitialContextFactory',
 'weblogic.jndi.WLInitialContextFactory' from dual UNION
select 'com.iknowbase.ProcessServices.WorkflowService.ProviderURL',
 'opmn:ormi://localhost:6003:home/ikbProcessServices' from dual UNION
select 'com.iknowbase.ProcessServices.WorkflowService.SecurityPrincipal',
 'orcladmin' from dual UNION
select 'com.iknowbase.ProcessServices.WorkflowService.SecurityCredentials',
 'mypassword' from dual;
```

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 installation specific database package must be created.
- The database schema must be populated with startup data. These data can be loaded from an existing installation (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:

```
$ ./quickstart.sh iknowbase.properties createUser
$ ./quickstart.sh iknowbase.properties uploadFile IKB_MASTER_66.dmp
$ ./quickstart.sh iknowbase.properties importFile IKB_MASTER_66.dmp
 IKB_MASTER_66
$ # Optionally download and display import log
$ ./quickstart.sh iknowbase.properties downloadFile IKB_MASTER_66.log .
$ cat IKB_MASTER_66.log
```

Prepare the database schema

Installing the database schema is most easily done using the Quickstart program. Assuming that you have created a file “iknowbase.properties” with the proper information, use the following command:

```
$ quickstart.sh iknowbase.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 quickstart 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 quickstart 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 Quickstart program is the preferred and supported mechanism.
Using the Quickstart program has two steps: First you upload the startup data, and then you import them:

```
$ ./quickstart.sh iknowbase.properties uploadFile IKB_MASTER_66.dmp
$ ./quickstart.sh iknowbase.properties importFile IKB_MASTER_66.dmp
 IKB_MASTER_66
```

The first command will upload the file IKB_MASTER_66.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_66.dmp from the Oracle Directory into the iKnowBase schema. The Quickstart 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_66.

If something fails during import, Oracle Datapump will store log messages in a log file. Use the following command to download the logfile to your local directory:

```
$ ./quickstart.sh iknowbase.properties downloadFile IKB_MASTER_66.log .
```

When using Quickstart, the name of the logfile is always the same as the name of the datafile, with a .log-suffix. Note that this file is created in the Oracle Directory where the dump file exists, so you may also view the logfile on the server without downloading it first.

Note that it is often useful to store a copy of the logfile even when there are no appearent failures.

Import startup data based on database link

Using the Quickstart program, a creation of a schema can be based on another schema reachable over a database link. First you need a global database link to the source schema and then you import it:

```
$ create global database link @myDbLink@ connect to @sourceSchema@ identified
 by @password@ using '@SID@';
$ ./quickstart.sh iknowbase.properties importSchemaFromLink IKB_MASTER_66
 @app.dblink@
```

The first command will create a database link to the source schema. It must be created by a user with system privileges and must not contain any special characters.
The second command will copy the source schema into the new iKnowBase schema. The Quickstart program will in fact use Oracle Datapump to perform both export and import. For the datapump import to succeed, the name of the source database user must be specified and also the database link name where the source schema can be found.

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 source code 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

```
$ quickstart.sh iknowbase.properties exportSource scripts.zip
$ quickstart.sh iknowbase.properties configureUser
$ quickstart.sh iknowbase.properties upgradeAll
$ quickstart.sh iknowbase.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 Quickstart has an easy-to-use mechanism:

```
$ quickstart.sh iknowbase.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:

```
$ quickstart.sh iknowbase.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 Quickstart program's upgradeAll feature:

```
$ quickstart.sh iknowbase.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:

```
$ quickstart.sh iknowbase.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:

```
$ quickstart.sh iknowbase.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 Quickstart command 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

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:

```
$ quickstart.sh IKB_PROD.properties exportFile IKB_PROD.dmp
$ quickstart.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.

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

To switch back to the standard edition version do:

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

Configuring the Activiti BPM Engine tables

In order to use the Activiti BPM Engine with iKnowBase, you will need to install the Activiti tables into the iKnowBase repository. This is done through the quickstart program, which has three discrete commands for configuring the Activiti database tables. Use one of the commands below:

```
$ quickstart <propertyfile> createActivitiSchema
$ quickstart <propertyfile> upgradeActivitiSchema
$ quickstart <propertyfile> dropActivitiSchema
```

Note: Dropping the Activiti schema will drop all process data. Use with care!

6. Java applications

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

- ikbViewer is the core runtime environment, serving web pages to end users
- ikbStudio is the development environment for building new applications
- ikbWebServices provides SOAP Web Services, typically to programs and integration engines
- ikbProcessServicesWS provides SOAP Web Services for integration with Oracle SOA Suite 10g
- ikbBatch provides batch based programs, for things such as document conversion, newsletter generation, etc
- ikbWebDAV provides document access through the WebDAV and FTP protocols
- ikbInstant provides real time asynchronous messaging

Each of the applications are installed in a similar way. Also, there is no conceptual difference between a fresh install and an upgrade, although many application servers have shortcuts to reinstall a new version of an existing application. The install or upgrade process will vary from application server to application server. This chapter describes the generic concepts, while application server specific methods are described in appendixes.

Fresh install

For each of the java applications, you may need the following steps:

- Create a JDBC data source, and expose as a JNDI-reference. This step can often be shared across all applications.
- Set up JAAS-security for the application server (or servlet container), if needed.
- Deploy the application.
- Configure security roles.

Before you start

The iKnowBase applications require an application server or servlet container that supports the Java Enterprise Edition Servlets release 3.0. For many application servers, you will need to configure a separate container for iKnowBase use.

ikbWebDAV requires deployment to context root / and will typically require a separate host (virtual host or separate server depending on application server) for this purpose as ikbViewer is also deployed to /.

Cluster support

The iKnowBase web applications in general support application server clustering and session replication, provided the load balancer is configured with session persistence / sticky sessions. See application server specific documentation for more details.

Exceptions:

- ikbWebdav does NOT support clustering and must NOT be deployed to a cluster containing more than one server instance.
- ikbInstant does NOT support clustering and must NOT be deployed to a cluster containing more than one server instance.
- iknowbase-resources can be deployed to a cluster, but sessions are not in use and availability/ replication is therefore not needed.

Create data source

The iKnowBase web applications will look up the JNDI resource “jdbc/iknowbaseDS”, which needs to be JDBC data source referring to the iKnowBase database repository to use. For some application servers, you will first create a “connection pool” referring to the database, and then a “data source” referring to the data source; other times, you will only create a data source directly referring to the database.

The most important things to keep in mind are these:

- The data source must be called “jdbc/iknowbaseDS”.
- Specify a suitable JDBC data source implementation, typically “oracle.jdbc.pool.OracleDataSource”.
- Specify the username and password for the database login (this is db.ikbUser and db.ikbPassword in the the Quickstart property file).
- Specify the JDBC-url to the database, on the form “jdbc:oracle:thin:@//server:port/instance-name” (this is the same as db.URL in the Quickstart property file).

Deploy the applications

Deploy the applications as required by your application servers, keeping the following in mind:

- The applications are deployed using WAR-file deployment.
- The war files are named "iknowbase-WEBAPPNAME-6.6.war"

Application	Default contextroot	Deployable to cluster	Comment
iKnowBase Batch	/ikbBatch	YES	
iKnowBase Instant	/ikbInstant	NO	
iKnowBase Process Services WS	/ikbProcessServicesWS	YES	
iKnowBase Resources	/ressurs	YES	
iKnowBase Studio	/ikbStudio	YES	
iKnowBase Viewer	/	YES	
iKnowBase Web Services	/ikbWebServices	YES	
iKnowBase Webdav	/	NO	MUST be deployed to /

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

Advanced topics

Deploy with “/ikbViewer” prefix

The iKnowBase Viewer application iknowbase-viewer-webapp-6.6.war has traditionally been deployed to /ikbViewer, but is by default from 6.4 deployed to /. The application will support requests using the old context root /ikbViewer when deployed to /.

The application can still be deployed to /ikbViewer or some other path of the customers choosing, but the context root must then be explicitly changed during deployment.

Background

On a given listening port, the web server should handle all the iKnowBase applications as well as all other custom applications. A typical set of handlers (endpoints) could be this:

- `/ressurs`, which is the resource file directory
- `/ikbStudio`, which is the Development Studio application
- `/ikbWebServices`, which is the WebServices application
- `/ikbBatch`, which is the batch application
- `/`, which is the main runtime application

With the setup above, you can access iKnowBase-pages using “//server/mypagename”.

Deployments before iKnowBase version 6.4 used context root /ikbViewer for the main runtime application, resulting in access to iKnowBase-pages using “//server/ikbViewer/mypagename”

From an iKnowBase perspective, the following requirements must be met if you want to change the context root:

- The main runtime application (the “ikbViewer” application) can be deployed with any endpoint you choose. We typically recommend “/”, but you may use “/ikbViewer” or “/ikb” if you want.

Update domain definition to match new endpoint

The final step is to make iKnowBase generate URLs to the new endpoints. From ikbStudio, edit the domain definition served by the endpoint, and update the relevant URL-prefixes:

- “path to iknowbase application” should be set to "/ikbViewer"
- “path to iknowbase page engine” should be set to "/ikbViewer/page"
- “path to iknowbase content server” should be set to "/ikbViewer/Content"

7. Batch Server

iKnowBase comes with a batch server 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 web application, and is typically installed as /ikbBatch.

Currently, the Batch Server handles two sets of services:

- The EmailReader service fetches email from an email account, and submits to iKnowBase for processing
- The FileConverter service converts various file formats to pdf, html or images.

Note that the Batch server is only available for 64-bit Linux (the “x86_64” architecture).

Installation

The Batch Server is installed the same way as all other iKnowBase web applications. Note that parts of the BatchServer require java version 6.

The EmailReader

Most of the EmailReader conifguration 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. However, there is one configuration you may want to make on the EmailReader batch server.

Enable or disable the EmailReader

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 on the email server side, which might not be built for this kind of activity. Therefore, remember to configure the emailreader as shown in the chapter “Configuration”, so that only one BatchServer instance is active.

The 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
```

Configuring the FileConverter

The FileConverter must be configured as shown in the chapter “Configuration”.

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
```

8. WebDav Server

iKnowBase comes with a WebDav server serving iKnowBase documents to WebDav Clients. The WebDav server is implemented as a java web application, and must be deployed along with the other web applications.

Due to client requirements, the webdav application must be deployed at context root / (the root and default application), and it must therefore also be deployed using a separate hostname (e.g. using webdav.example.com, while other applications are deployed on intranet.example.com).

The Webdav application does not support clustering.

Installation

The WebDav Server is required to be deployed to context root / and will typically require a separate hostname separate from the other deployed applications. See installation guide for how this is accomplished with your application server of choice.

Since authentication to a webdav resource is done using BASIC authentication, SSL is strongly recommended. See the troubleshooting section for accessing WebDav resources without SSL support.

Note: The WebDav Server require java version 6.

The WebDav Server

Enable or disable the WebDav Server

Disable the WebDav server by stopping the Web application.

Configuring the WebDav Server for dimension based WebDav folders

The default setup of the WebDAV server will provide editing access to documents based on document ID, with document editing mostly accessed through the special “Edit in WebDAV” function in the content viewer. However, you may also expose documents in a regular directory structure based on iKnowBase dimension. Using the Metadata manager in Development Studio, attach the dimension type IKB_WEBDAV_FOLDER to the dimensions you would like to be WebDav folders.

The following iKnowBase WebDav metadata are pre configuered “Out-of-the-box”:

- Dimension type “IKB WebDav Folders” external_key = IKB_WEBDAV_ROOT
- Dimension “Image Archive” with dimension types = IKB_WEBDAV_FOLDER
- Attribute “IKB WebDav Folders” external_key = IKB_WEBDAV_ROOT

The attribute collects the dimension types used to identify the dimensions used as WebDav folders. By default only the dimension type “IKB WebDav Folders” is used.

Client configuration

Enabling Basic authentication

By default, the Windows WebDAV clients do not accept Basic authentication. This must be enabled; for Windows 7 the procedure is as follows

- Starting “regedit.exe” (remember to start the application as administrator).
- Find the key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WebClient\Parameters
- Set the value for BasicAuthLevel to 1, or if the key does not exist, create it:
  - Select new from the Edit menu
  - Choose DWORD Value.
  - Enter BasicAuthLevel and exit with ENTER.
  - Select Modify and write 1 in the value field.

The value “1” in the example above will enable Basic authentication on an SSL-enabled site (using https). We do not support the value “2”, which means using Basic authentication over an insecure and non-encrypted communication channel.

Editing an iKnowBase document from Microsoft Office

The most typical use case is to edit an iKnowBase document directly from Microsoft Office. To set this up, do the following:

- Using Development Studio, 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 (Internet Explorer is best, but as of November 2013 Firefox is also supported), click the “Edit using WebDAV” icon

Mounting WebDAV folders in Windows

On Windows you have to start the “Webclient service” to be able to connect to WebDav:

- Start -> Run -> Services.msc
- Start Webclient service. Set the properties so it will start automatically next time you boot Windows.

Connect iKnowBase as a network drive:

- Start the Windows Explorer
- Right click on My Computer
- Select "Connect to Network drive..."
  - Drive: could be any unused drive letter.
  - Folder: ex. http://www.myserver.com:9801/ikbWebdav/webdav.
  - If you get the error message “Service does not exist”, wait one minute and try again.
- Check if the folder “Image Archive” is accessible on the newly connected network drive.

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)

9. 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

The Apache Solr Search Server is included in the iKnowBase distribution as a zip file including both Solr and iKnowBase configuration. To install, do the following:

- Unzip the file to a directory where you want to run the Solr server from. The default Solr core is named “iknowbase”.
- Navigate to the folder `<solr_home>/solr/iknowbase/conf`.
- Copy the file `solrcore.properties.sample` to `solrcore.properties`.
- Change the properties in `solrcore.properties` to define your environment, as shown in the Configuration-chapters below
- By default, description and body content will only be searchable, but not stored in SOLR. Change schema.xml if you want a different behavior.

Note: The iKnowBase search component, which handles security, requires Java version 7.

Starting and stopping

A start/stop script for linux is provided in `<solr-home>/bin`. It can be placed under `/etc/init.d`. Change the SOLR_RUNASUSER to the linux user who should run Apache Solr. 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.

Master-Slave setup

You may configure multiple Solr instances in a Master-Slave configuration, for example indexing is done on one node, while searching is done on another.
Read more about this in the Solr documentation, and configure the following properties in solrcore.properties:

Property name	Description
enable.master	If this is the master, this should be set to true.
enable.slave	If this is a slave, this should be set to true.
masterUrl	If this is a slave, this applies to the master Solr url with the core-name, eg. http://hostname.example.com:8983/solr/iknowbase.
pollInterval	If this is a slave, this applies to the poll interval for the slave to check for updates of the index. Default is set to 60 seconds (00:00:60).

You will have to make this configuration for each node (master and slaves). The master will typically be used for indexing, while slave servers normally handles the search.

Configure the iKnowBase applications

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

10. Instant Server

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 web application and is typically installed as /ikbInstant.

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

Installation

The Instant Server is installed the same way as all other iKnowBase web applications, but make sure you also configure the SecureTokenEngine *InstallationGuide#SecureTokenEngine* to enable web client single sign on between ikbViewer and ikbInstant. An alternative without single sign on between ikbViewer and ikbInstant 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.

The Instant Server

Enable or disable the Instant Server

Disable the Instant server by stopping the Web application.

Configuring the Instant Server

See the *Installation Guide#Configuring the ikbInstant application* for configuration options.

Testing and troubleshooting

For developers, the console available at /ikbInstant/ikb$console provides

- instant > topis: A list of all topics.
- instant > topis > [topicName]: Detailed information about the specific topic, including all connected clients.
- cache > caches: Default features for iKnowBase applications.
- cache > configuration: Default features for iKnowBase applications.
- monitor: Default features for iKnowBase applications.
- logging: Default features for iKnowBase applications.
- services: Default features for iKnowBase applications.
- java: Default features for iKnowBase applications.

11. Web Application Security

Quick install

The default configuration will for iKnowBase Quickstart and Oracle GlassFish provide

- 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.

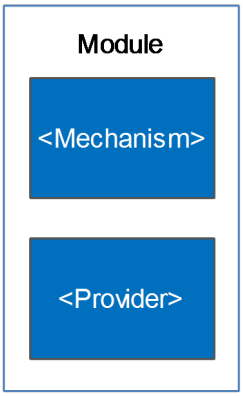
If you need to customize authentication, the basic steps are

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

Overview

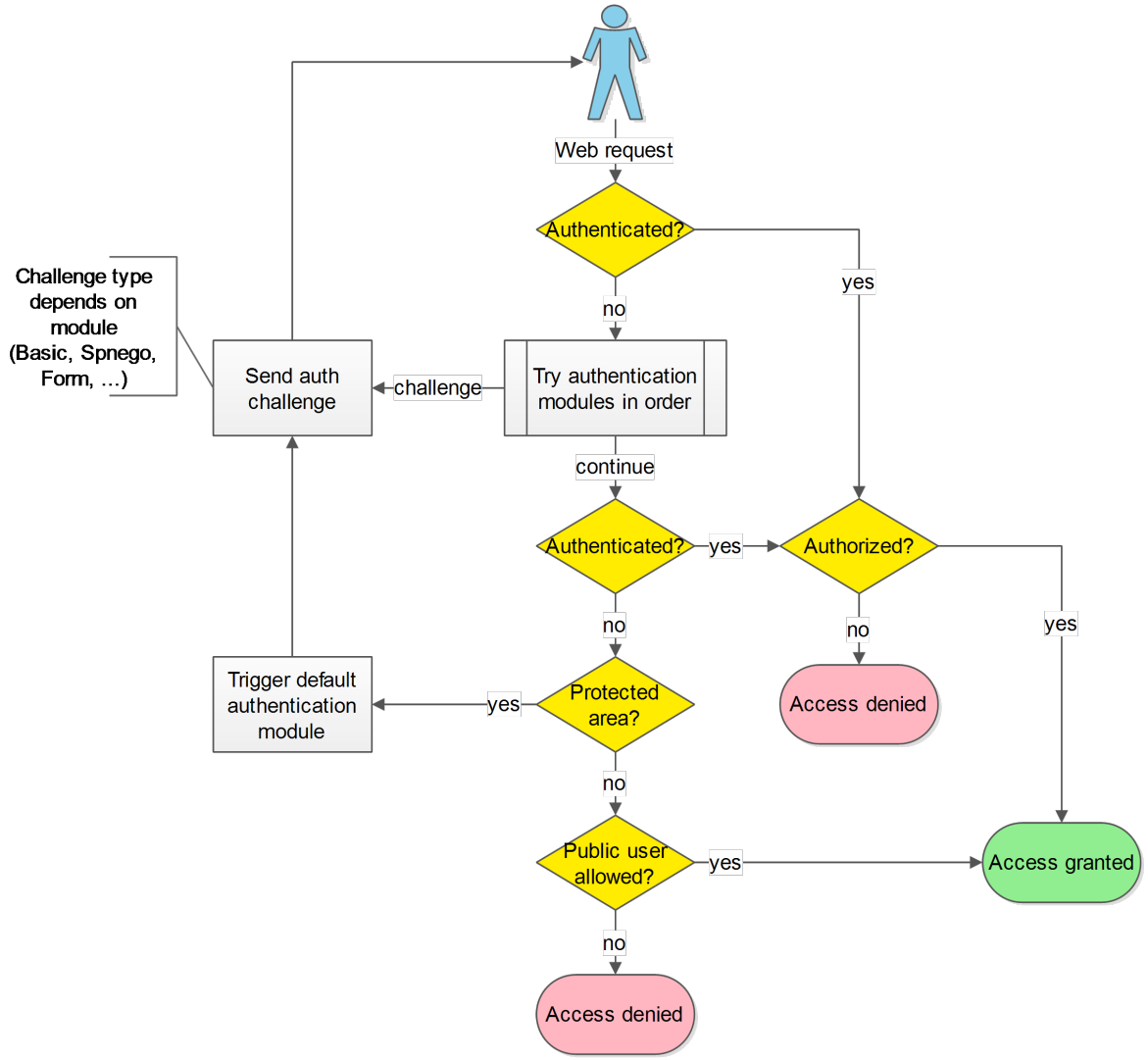
The iKnowBase web application security implementation is based on the Spring Security framework and takes care of user authentication and authorization to specific web application protected areas.

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.

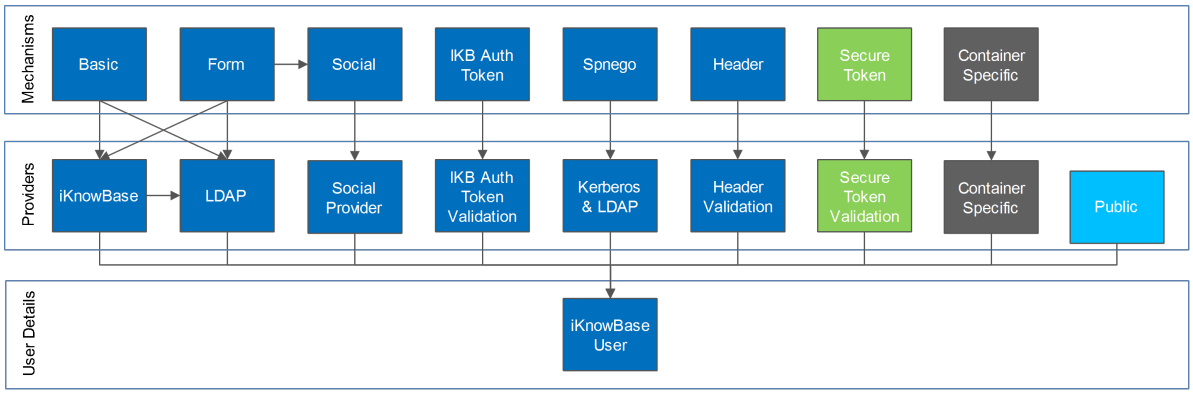
The authentication process follows this flow:



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 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	Applications	Authentication module	Authentication method	Authentication provider	Comment
iKnowBase Quickstart	*	Form	Form	iKnowBase User repository	
iKnowBase Quickstart	Instant, WebDav	Basic	Basic	iKnowBase User repository	Due to client compatibility.
Oracle GlassFish	*	Form	Form	iKnowBase User repository	
Oracle GlassFish	Instant, WebDav	Basic	Basic	iKnowBase User repository	Due to client compatibility.
Oracle WebLogic	*	Container	Container dependant	Container dependant	

Instant and WebDav

iKnowBase Instant requires Basic 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.

iKnowBase WebDav requires Basic or Spnego authentication. Basic is set as the default.

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 path `/ikb$auth/<authentication mechanism>/authenticate`.

As an example, while the default authentication is form based, a client may trigger the HTTP Basic mechanism by accessing `/ikb$auth/basic/authenticate`.

Available authentication modules

Available authentication mechanisms:

Authentication mechanism	Trigger	Can be used as default	Authentication provider	Description
Basic	/ikb$auth/basic/ authenticate	YES	UsernamePassword	Basic authentication for username and password.
Container	/ikb$auth/container/ authenticate	YES (WebLogic only)	Container	Container based authentication

				(depends on security setup in the application server).
Form	/ikb$auth/form/authenticate	YES	UsernamePassword	Form based authentication for username and password.
AutoForm	/ikb$auth/autoform/authenticate	YES	UsernamePassword	Form based authentication using an autogenerated login form (handy as a backup if the normal login form is out of order).
Header	/ikb$auth/header/authenticate	YES	Header validation	Request header based authentication.
Spnego	/ikb$auth/spnego/authenticate	YES	Kerberos realm and LDAP	SPNEGO based authentication for Kerberos single sign on (i.e. Microsoft Windows domain single sign on).
Google	/ikb$auth/google	NO	Social connection	Uses form based module. Will verify a preexisting social connection (created using account activation link).
Twitter	/ikb$auth/twitter	NO	Social connection	Uses form based module. Will verify a preexisting social connection (created using account activation link).
Facebook	/ikb$auth/facebook	NO	Social connection	Uses form based module. Will verify a preexisting social connection (created using account activation link).
LinkedIn	/ikb$auth/linkedin	NO	Social connection	Uses form based module. Will verify a preexisting social connection (created using account activation link).
iKB Secure Token	N/A	NO	Secure token validation	iKnowBase secure token based authentication.
iKB Auth Token	N/A	NO	Auth token validation	iKnowBase Auth token based authentication.

RememberMe	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.

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><ikbViewer contextPath>/ikb$auth/<providerid>`. Example: `https://www.example.com/ikb$auth/google` or `https://www.example.com/ikbViewer/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 social account from one of the installed providers to the newly created iKnowBase account and authenticate using social authentication.

WARNING: Anyone with a valid non expired trusted activation token will be allowed to set password / connect using ANY social 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
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 ALL	/ikb$auth	2
iKnowBase ALL	/ikb$console	3
iKnowBase Batch	Default	3
iKnowBase Studio	Default	3
iKnowBase Viewer	Default (NO ACL)	1
iKnowBase Viewer	ACL with public user	1
iKnowBase Viewer	ACL without public user	2
iKnowBase Viewer	/private	2
iKnowBase WebDAV	Default	2
iKnowBase WebServices	Default	2 and TRUSTED

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 application (Viewer, Studio, Batch, ..) and they all have an `/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
./quickstart.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 iKnowBase Quickstart and Oracle GlassFish, 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:

- Only possible for ikbViewer
- Form action must be [contextPathForIkbViewer]/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"

As the web applications do not share authenticated sessions, 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 ikbViewer is deployed to /)
- /ikbViewer/j_spring_security_check (if ikbViewer is deployed to /ikbViewer)
- /ikbStudio/j_spring_security_check
- /ikbBatch/j_spring_security_check
- ...

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 iKnowBase QuickStarts 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.userDnPattern.1	CN={0},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 quickstart application allows multiple user bases. The application uses the following logic when looking for these values:

- Always look for `com.iknowbase.spring.security.ldap.userDnPattern.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={0},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.userDnPattern.1	CN={0},CN=Users

This a more complex configuration, with three userbases

Property name	Value
com.iknowbase.spring.security.ldap.userDnPattern.1	CN={0},CN=Employees
com.iknowbase.spring.security.ldap.userDnPattern.2	CN={0},CN=Premium Members
com.iknowbase.spring.security.ldap.userDnPattern.3	CN={0},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	myserver-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.remoteAddrExp	^10\..*

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

Property name	Value
com.iknowbase.spring.security.spnego.realRemoteAddrExp	^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.requestHeaderExp	`.*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.sendAdditionalHttpBasicChallenge	true
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.

- [ikbViewer context path]/private
- [ikbInstant context path]/private
- [all deployed applications context path]/ikb$console

- /ikbStudio
- /ikbBatch
- /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|.*/ikb\$console|.*/ikb\$runner|/ikbStudio|/
ikbBatch|/ikb\$content|/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.headerNameUserExtractExpr	Usern([^,]+)
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.realRemoteAddrHeader	[IP address to Oracle HTTP Server]

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>.en	true
com.iknowbase.spring.security.social.<provider>.clie	<API key from the application registration>
com.iknowbase.spring.security.social.<provider>.cli	<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 the ikbViewer application, set the instance qualifier to match the context path ("/" or "/ikbViewer") where the iKnowBase Viewer is deployed. See _Installation Guide > Configuration > The ikb_installation_properties table > Qualifier_

I am being logged out from ikbViewer while active in ikbStudio (or vice versa)

Web application sessions are NOT shared and the session will expire if you've been inactive for too long. This is very visible when using Form based authentication without a single sign on system. If your are using a single sign on system or relying on Basic authentication, the logout/login process happens behind the scenes.

You may increase the session timeout, but beware that this will lead to increased memory consumption due to an increase in number of live sessions and an increased security risk. As such you may want to only do this in a development environment.

You may activate RememberMe functionality, which will log you in if the session has expired.

You may also use a different authentication mechanism other than Form based authentication.

'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.

12. iKnowBase Quickstart embedded web server

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

The iKnowBase distribution contains an embedded web server usable for small to medium installations, with very small footprint and very easy installation.

Preparations

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

Configure the quickstart instance

If you've not already done so, set up the quickstart 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 quickstart instance

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

```
cd /opt/iknowbase/production
./quickstart.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 Quickstart embedded web server, application deployments is specified in the same property file as all other configuration.

Default deployment

The Quickstart program contains a default configuration that automatically deploys the most relevant applications: /ressurs, /ikbStudio and /ikbViewer. 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,webIkbStudio,webIkbViewer
```

NOTE that webIkbViewer 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. These entries will be the name of the application, followed by ".path" and ".war" respectively:

```
# Web server configuration
web.apps=webRessurs,webIkbStudio,webIkbViewer

# Web application definitions
webRessurs.path=/ressurs
webRessurs.war=iknowbase-resources-6.6.war
webIkbViewer.path=/
webIkbViewer.war=iknowbase-viewer-webapp-6.6.war
webIkbStudio.path=/ikbStudio
webIkbStudio.war=iknowbase-studio-webapp-6.6.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,webIkbStudio,webIkbViewer

# Web application definitions
webRessurs.path=/ressurs
webRessurs.war=iknowbase-resources-6.6.war
webIkbViewer.path=/
webIkbViewer.war=iknowbase-viewer-webapp-6.6.war
webIkbStudio.path=/ikbStudio
webIkbStudio.war=iknowbase-studio-webapp-6.6.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 Quickstart 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 ikbViewer at the root of all other servers. This can be handled by defined virtual hosts for the required applications:

```
web.apps=webIkbViewer,webIkbWebdav
webIkbViewer.path=/
webIkbViewer.war=iknowbase-viewer-webapp-6.6.war
webIkbWebdav.path=/
webIkbWebdav.war=iknowbase-webdav-webapp-6.6.war
webikbWebdav.vhosts=webdav.example.org, webdav
```

In the example below, when a client asks for "http://webdav/" or "http://webdav.example.org", the request will be served by the iknowbase-webdav application. Root requests to all other hosts (such as http://intranet.example.org or http://testserver.example.org) are handled by the iknowbase-viewer 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 quickstart 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 Quickstart application server, set the property `web.honorXForwardedHeaders` in the property file:

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

Configuring SSL listener in iKnowBase Quickstart

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

```
...
web.ssl.port=8443
web.ssl.keyStorePath=quickstart.keystore
web.ssl.keyStorePassword=<KEYSTORE_PASSWORD>
...
```

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

The following example demonstrates generating the quickstart.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 ./quickstart.keystore -storepass
 <KEYSTORE_PASSWORD> -importkeystore -srckeystore ./www_example_com.pkcs12 -
srcstoretype PKCS12 -srcstorepass <KEYSTORE_PASSWORD>
```

Advanced topics

Specify session cookie domain

By default, the quickstart 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 quickstart 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 quickstart 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 work directory

During normal execution, the quickstart 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/ikbViewer” and “work/ikbBatch”). The default work directory is “./work” under the current directory, but this can be overridden in the quickstart 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 quickstart 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 in iKnowBase Quickstart 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 quickstart property file:

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

```
# webIkbViewer.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.

13. 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
- Deploy the Resource-directory and java web applications to the WebLogic Server

Installation and configuration of WebLogic

Install a domain containing

- a cluster
- one or more managed servers supporting the cluster
- a virtual host exclusively for WebDav backed by one managed server (typically YOURHOSTNAME-webdav.YOURDOMAINNAME)
  - WebDav does NOT support clustering and must be deployed to a single managed server or virtual host targeting a single managed server.
- a virtual host exclusively for Instant backed by one managed server (typically YOURHOSTNAME-instant.YOURDOMAINNAME). Can be the same managed server as used by Webdav.
  - Instant does NOT support clustering and must be deployed to a single managed server or virtual host targeting a single managed server.

WebDav must be deployed to context root / and since ikbViewer also typically is deployed to /, we recommend creating a WebLogic virtual host and target WebDav to this virtual host. A web request to / on the WebDav virtual host will be routed to WebDav, and on all other hosts be routed to the Viewer.

JDBC drivers

iKnowBase requires JDBC for Oracle Database with SQL XML support. This feature is not enabled in WebLogic by default and must be installed by adding xmlparserv2_sans_jaxp_services.jar and xdb6.jar to the application server classpath.

WebLogic is distributed with JDBC drivers and the additional JDBC SQL XML drivers MUST match the database version for the installed JDBC drivers, which MUST be equal to or greater than your installed Oracle Database version.

The jar files can be found on the Oracle web page, or in the product directory of your database (check `${ORACLE_HOME}/jdbc/lib`, `${ORACLE_HOME}/rdbms/jlib` and `${ORACLE_HOME}/xdk/lib`).

For Oracle Database 11g and higher

For Oracle Database 11g and higher, we recommend installing the Oracle Database 12c JDBC driver set with SQL XML support.

If you are upgrading an existing installation, read Oracle's documentation regarding JDBC client upgrade to 12c for WebLogic and backwards compatibility.

The set consists of the following files:

- ojdbc7.jar
  - NOTE JDK6: use ojdbc6 if running with JDK6
  - NOTE Dynamic Monitoring Service (DMS): If you need support for DMS in your application server (iKnowBase does not), use ojdbc7dms.jar

- ucp.jar
- ons.jar
- orai18n.jar
- orai18n-mapping.jar
- oraclepki.jar
- osdt_cert.jar
- osdt_core.jar
- xmlparserv2_sans_jaxp_services.jar
- orai18n-collation.jar
- xdb6.jar

Make the libraries available to WebLogic by uploading them to a directory on the Weblogic server, i.e. `${WL_HOME}/server/lib/db12c/`, and add them to the classpath by editing `setDomainEnv.sh`:

```
# Database 12c client
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
ojdbc7.jar"
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
ucp.jar"
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
ons.jar"
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
orai18n.jar"
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
orai18n-mapping.jar"
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
oraclepki.jar"
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
osdt_cert.jar"
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
osdt_core.jar"
# Database 12c - SQL XML support
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
xmlparserv2_sans_jaxp_services.jar"
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
orai18n-collation.jar"
PRE_CLASSPATH="${PRE_CLASSPATH}${CLASSPATHSEP}${WL_HOME}/server/lib/db12c/
xdb6.jar"
```

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 a user repository (realm)

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.

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

For each of the application archives, deploy 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:
  - All applications except iKnowBase WebDav and iKnowBase Instant should be targeted to the cluster.
  - iKnowBase WebDav should be targeted to the webdav virtual host
  - iKnowBase Instant should be targeted to the instant virtual host
- If you want, go to the tab “Configuration, Ressurs”, 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 applications are configured to replicate HTTP sessions when they are 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.

Note: ikbWebdav does not support clustering and is not configured with persistent-store-type.

Configure user realms (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-6.6.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-6.6.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-6.6.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>
```

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))))
```

14. Installing on GlassFish Server

NOTE: Support for GlassFish 3 has been deprecated. Support for GlassFish 4 has been removed. See release notes.

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

Installation on GlassFish Server has the following tasks:

- Install the GlassFish-software
- Create a data source, and deploy to the target
- Deploy the Resource-directory and java web applications to the GlassFish Server
- NOTE: iKnowBase WebDav must be deployed on contextroot / and it is therefore recommended to use a separate server for iKnowBase WebDav.

Installation and configuration of GlassFish itself.

Run a normal installation of GlassFish.

Configure a domain containing

- a cluster
- one or more instances supporting the cluster
- a standalone instance or an additional cluster supported by a single instance for WebDav and Instant
  - WebDav and Instant does NOT support clustering and must be deployed to a single server instance or cluster backed by a single server instance.

Configuring a database data source

To run iKnowBase under GlassFish, you need to add the required jdbc driver, a connection pool and a data source:

First, install the Oracle JDBC Driver:

- iKnowBase requires JDBC for Oracle Database with SQL XML support: The Oracle jdbc-driver (ojdbc7.jar), the XML-database support (xdb6.jar) and the Oracle XML Parser v2 (xmlparserv2_sans_jaxp_services.jar).
- The jar files can be found on the Oracle web page, or in the product directory of your database (check `${ORACLE_HOME}/jdbc/lib`, `${ORACLE_HOME}/rdbms/jlib` and `${ORACLE_HOME}/xdk/lib`).
- Install the driver to the GlassFish server or domain where you want to install iKnowBase, e.g. either `${GLASSFISH_HOME}/glassfish/lib/` or `${GLASSFISH_HOME}/glassfish/domain/domain1/lib/`
- Restart your GlassFish instance.

Next, create a connection pool:

- From the GlassFish console, select “Resources, JDBC, Connection Pools” from the tree.
- Click “New” to create a new pool, and configure it as follows:
- Name = jdbc/iknowbaseDSPool
- ResourceType = javax.sql.ConnectionPoolDataSource
- Vendor=oracle
- Datasource Classname=oracle.jdbc.pool.OracleConnectionPoolDataSource
- Ping=Enabled
- Three properties (only)
  - url=jdbc:oracle:thin:@//dbhost.example.com:1521/dbname.domain
  - user=<database user>
  - password=<database password>

Finally ,create a data source:

- From the GlassFish console, select “Resources, JDBC, JDBC Resources” from the tree.
- Click “New” to create a new data source, and configure it as follows:
  - JNDI Name = jdbc/iknowbaseDS
  - Pool Name = jdbc/iknowbaseDSPool

Configuring cluster single-sign-on-state

The availability service for the cluster and web container must be enabled (enabled by default).

Single-sign-on-state for web container availability should also be enabled to support replicating the authenticated user.

Configuring the HTTP listener for ikbInstant

ikbInstant requires that support for Comet and/or Websockets is enabled on the HTTP listener.

- From the GlassFish console, select “Configurations, <your configuration name>, Network Config, Protocols, http-listener-1 (or your listener name)” from the tree.
- Select the HTTP tab
- Set Comet Support to Enabled

Deploy the applications

For each of the application archives, deploy using the GlassFish console:

- Select “Applications” in the administration console.
- Select "Deploy"
- Select the relevant web archive.

- Set *context root* (deployments using the administration console will not pick up the configured context root)
- Target applications:
  - All applications except iKnowBase WebDav and iKnowBase Instant should be targeted to the main cluster.
  - iKnowBase WebDav and iKnowBase Instant should be targeted to the standalone instance or separate cluster backed by a single instance.
- Availability:
  - All applications except iKnowBase Instant, iKnowBase WebDav and iKnowBase resources should have the “Availability” flag set.

Deploy the /ressurs-directory

The easiest mechanism is to deploy the iknowbase-resources-6.6.war file, which will automatically expose the resources.

If you want to have the /ressurs-directory (or other directories) available from the file system, unzip them to a directory and configure a reverse proxy in front of GlassFish to server this content for requests starting with “/ressurs”. As a standard deployment of ikbViewer and ikbWebdav will use context root /, the GlassFish docroot cannot be used.

Configure Web Application Security

iKnowBase does not support the GlassFish container mode for authentication and you must use one of the available Spring Security authentication modules or extend with your own Spring Security module.

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 the application server

The procedures for terminating SSL directy in the application server can be found in the glassfish 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 “X-Forwarded-Proto” with the value “https”, ref http://en.wikipedia.org/wiki/List_of_HTTP_header_fields
- Configure the glassfish http protocol listeners to use a scheme-mapping that understands this header, ref http://java.net/jira/browse/GLASSFISH-18685
- Verify this setup by loading /ikb$console/java/request using from a secure connection; the value “Requested URL” should indicate a https-scheme
- Note that you will need at least glassfish 3.1.2.2 for this setup to work

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>
```

Use the following example from the glassfish domain.xml:

```
<network-config>
  <protocols>
    <protocol name="http-listener-1">
      <http default-virtual-server="server" max-connections="250" scheme-
mapping="X-Forwarded-Proto">
...
```

Troubleshooting

Using custom passwords on java keystores

If you have changed the passwords on GlassFish’s java truststore (cacerts.jks) and keystore (keystore.jks) during the setup and plan to deploy the iKnowBase Batch Server, the passwords for accessing the keystores must be set using JVM options
`-Djavax.net.ssl.keyStorePassword=<your_new_password>`
`-Djavax.net.ssl.trustStorePassword=<your_new_password>`

The Batch Server will fail to start (`SSLInitializationException: Failure initializing default system SSL context`) if the passwords are not set.

GlassFish server.log: AS-NAMING-00006 and RAR8067

The following log statements in server.log is related to activiti enabled applications

- SEVERE: AS-NAMING-00006: Exception in NamingManagerImpl copyMutableObject: java.lang.IllegalStateException: This web container has not yet been started
- WARNING: RAR8067: Unable to determine pool type for pool [jdbc/iknowbaseDSPool], using default pool type

The issue can occur while reloading / redeploying one of the activiti enabled applications (viewer or batch). The activiti job executor thread is busy looking for asynchronous tasks and might not shut down within the time used for reloading or redeploying the application.

This error results in a non-functioning activiti job executor and asynchronous tasks will not be loaded by this application.

To resolve the issue, restart the application server.

iKnowBase issue reference: IKBR-1124

GlassFish server.log: log4j called after unloading and Class invariant violation

The GlassFish server will report the following error message during startup:

```
[#|<TIMESTAMP>|SEVERE|glassfish3.1.2|
javax.enterprise.system.std.com.sun.enterprise.server.logging|
_ThreadID=1;_ThreadName=Thread-2;|log4j:ERROR log4j called after unloading,
 see http://logging.apache.org/log4j/1.2/faq.html#unload.|#]
[#|<TIMESTAMP>|SEVERE|glassfish3.1.2|
javax.enterprise.system.std.com.sun.enterprise.server.logging|
_ThreadID=1;_ThreadName=Thread-2;|java.lang.IllegalStateException: Class
 invariant violation
    at org.apache.log4j.LogManager.getLoggerRepository(LogManager.java:199)
    at org.apache.log4j.LogManager.getLogger(LogManager.java:228)
    at
 org.slf4j.impl.Log4jLoggerFactory.getLogger(Log4jLoggerFactory.java:66)
    at org.slf4j.LoggerFactory.getLogger(LoggerFactory.java:270)
    at org.slf4j.LoggerFactory.getLogger(LoggerFactory.java:281)
```

Set the JVM option `-Dorg.apache.catalina.loader.WebappClassLoader.ENABLE_CLEAR_REFERENCES=false` to resolve this issue.

Example for disabling `ENABLE_CLEAR_REFERENCES` on Domain Administration Server and a cluster named myCluster:

```
./asadmin create-jvm-options --
 -'Dorg.apache.catalina.loader.WebappClassLoader.ENABLE_CLEAR_REFERENCES=false'
./asadmin create-jvm-options --target myCluster --
 -'Dorg.apache.catalina.loader.WebappClassLoader.ENABLE_CLEAR_REFERENCES=false'
```

Log4j reference: *http://logging.apache.org/log4j/1.2/faq.html#unload*
iKnowBase issue reference: IKB-2604

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))))
```