



LDAP Authentication

Revision 1.0

Revision 1.0

Doc Type

Date 02/24/2009

Written by Sebastian Wenzky

Verified by Marco Meschieri Logical Objects

Approved by Marco Meschieri Logical Objects

LogicalDOC LDAP Authentication 1.0

© 2009 Logical Objects snc, via Bonasi 2/A – 41012 Carpi Italy. All rights reserved.
<http://www.logicalobjects.com>

This document is subject to change without notice.

License

This work is licensed under a GNU Free Documentation License 1.2.
<http://www.gnu.org/licenses/fdl-1.2.txt>

Disclaimer

Documentation is provided 'AS IS' and all express or implied conditions, representations and warranties, including any implied warranty of merchantability, fitness for a particular purpose or on-infringement, are disclaimed, except to the extent that such disclaimers are held to be legally invalid.

LogicalDOC LDAP Authentication 1.0

Change Log

Rev.	Date	Changes
1.0	02/24/2009	Document creation

Distribution

Person	Company	Position	Contact

Contents

1 CHAPTER.....	5
TITLE2.....	5
Title3.....	5

1 Introduction

This document (should) not be used for external purposes as the document will only describe the current state of development. Furthermore, the document consists of a first tutorial of LDAP implementation for the **LogicalDOC** Team. But please reconsider: of course, the document is open source, thus can be used for every case you wish for.

At a glance, all features being described regarding the blueprints site has been developed and thus now available through this implementation. Tested directory server where some use cases and some virtual company structures has been built are:

- Apache Directory Server
- Active Directory Server on Windows Server 2003 Professional SP2

Supported authentication mechanism being available with **LogicalDOC** are:

- Simple Authentication (Password will be transmitted to the server)
- DIGEST-MD5 via SASL (Challenge Response)

2 Architecture

The initial questions for developers belongs to used technologies and frameworks. As described and depicted earlier upon the blueprints site, Jsecurity and Acegi does not offer the flexibility and – thus – the ease of use to manage directories efficiently. As **LogicalDOC** is built up with Spring its a wise decision to take cover of this framework, since sophisticated support is available through an appropriated plug-in. This module is called **Spring-LDAP**. Spring-LDAP has the same matter of usage as JDBC-Template or the Hibernate-Template.

Every LDAP-Template has its own authentication source where authentication informations will be managed and stored. LdapTemplate itself connects via JNDI context (with use of authentication source) onto a directory server.

LogicalDOC uses two different authentication sources. At first, the simple authentication is supported by the class *BasicLDAPContextSource*. Digest-MD5 is supported by the extending class named *DigestMD5LdapContextSource*.

Please review the LDAP spring context for more informations regarding configuration options.

Authentication source is an essential aspect of the LDAP synchronisation as well as for the LDAP authentication component. Different kinds of authentication sources differ on how users can be authenticated through a directory server. Here comes the userAuthenticationPattern property in the game (can be reviewed on Simple authentication Apache Directory server LDAP context).

```
<property name="userAuthenticationPatern"  
value="{logonAttribute}={userName},{userBaseEntry}"/>
```

The wild cards *logonAttribute*, *userName* as well as the *userBaseEntry* manages the connection string being transmitted to the directory server. When you review more deeply this attributes, you can find those attrrbutes upon other parts of the LDAP Spring context (excluded *userBaseEntry* which is based on the property *userBase*). To understand this issue, please compare the context sources of both directory servers and you will see it why this approach has been taken in such a way. Active Directory lookups user entries in a more different way.

As described earlier, the authentication source is used to synchronize (scheduled task) the user database as well as for ad-hoc login of users. Before you can use synchronisation(scheduled task) you have to enter a valid username as well password to authenticate (for instance in the night) automatically with a directory server. Thus the attributes *userName* as well as password is in place.

```
<property name="userName" value="swenzky" />  
<property name="password" value="Test1234" />
```

UserName will be taken and stored against the userAuthenticationPattern.

If another wants to logon, the username will be taken and stored as well against this userAuthenticationPattern.

For instance, following output can be obtained on storing several attributes against this userAuthenticationPattern.:

```
uid=swenzky, ou=users, ou=system
```

So, its no rocket science ;-)

2.1 CurrentDN Attribute for non Active Directories

The currentDN attribute is needed on authenticating via scheduled task(we need to find out where the admin user is stored): `ou=users, ou=system`

This is only needed with Apache Directory Server. Active Directory needs this in no case, its fully magically ;)

2.2 User and Group Mapping

A further essential part belongs to the user and group mapping. Every company and institution has its own structure. Therefore, every installation of **LogicalDOC** (with LDAP) must be configured manually (at the first time) to let **LogicalDOC** know on how users and groups can be found and combined together. Therefore the class `LDAPUserGroupContext` is responsible for those things.

Following the attributes will be described:

userIdentifierAttribute

Defines the attribute responsible to unique identifying a user. This could be CN(Common name) on Active Directory or uid within Apache Directory.

logonAttribute

This attribute is essential for validating a userName placed on LoginSite(e.g. Given from LoginForm) against an LDAP entry. And yes, if you reconsider a peace of moment you could ask the question about differences between this attribute and **userIdentifierAttribute**. Please visit the Active Directory authentication source bean to see it. The identifier is CN but not the **logonAttribute**.

userClass

The user class in LDAP responsible storing user informations.

groupClass

Same as userClass but works for groups

groupIdentifierAttribute

Same as userIdentifierAttribute but works for groups.

userBase

list of entries where users can be stored

groupBase

list of entries where groups can be stored

Please reconsider: `userBase` is a part of the `userAuthenticationPattern`.

The newly created **AuthenticationProvider**

What the hell is that? Well, after we are able to gain access to third-/foreignsystems to authenticate and synchronise users, we have to take a new approach on how authentication can be reliant managed. A number of authentication providers are needed to validate a user in a correct way. A chain of authentication providers spawns on the horizon, must be managed at once. Initially such a provider is passed into the core module that will keep in track of all available authentication components. If a user wants to get access to **LogicalDOC**, the `LoginForm` will call this authentication provider and obtains a boolean value whether the login has been accepted of – at least – one authentication provider.

Currently available authentication providers are:

BasicAuthentication

Manages the basic userstore of **LogicalDOC**.

LdapAuthentication

Manages the authentication against a directory server

At least, one authentication must return a **true value** to gain access to **LogicalDOC**. `BasicAuthentication` can be disabled as well, so only authentication via LDAP is possible(whole flexibility can be reached).