PANDUIT™ infrastructure for a connected world

SYNAPSENSE a PANDUIT company

Technical Reference 103-SS

Published: **5/3/18**

Web Console Active Directory Configuration

Applies to:	SynapSense Active Directory
Objective:	Add Web Console users to the Active Directory.

Description

By default, the SynapSense® Web Console server is configured only for local authentication. This means users must be configured in the local database to be able to access SynapSense™ Web Console server. In a typical Microsoft Windows based enterprise, network access is only available to users configured in a centralized user database called the Active Directory. This also allows limiting access based on user roles and responsibilities.

Performing the Procedure

Single LDAP Server Configuration

Make all of the configuration changes described in this document in: *<JBOSS_HOME>\standalone\configuration\standalone.xml* file, section *<subsystem xmlns="urn:jboss:domain:security:1.1">*, subsection *security-domain* named *"SynapServer"*

This is only an example. The actual configuration is dependent upon your Active Directory/LDAP Server scheme. The following sample configuration was tested with one particular Active Directory Server. Here is a brief description of it followed by an example:

1. The first two login modules (Remoting and Database) have the same configuration as the non-LDAP scheme. The only difference is that the Database module flag must be set to 'sufficient', not 'required'.

   ```
   <login-module code="Database" flag="sufficient">
   ```

2. com.synapsense.security.LdapExtLoginModule is an extension of JBoss LdapExtLoginModule. Please read the documentation for these modules before you begin so you have a better understanding of the attributes and their meanings. Documentation for the SynapSense module is provided in the Login Modules Documentation section.
   a. In this example, the protocol used is LDAP. Configuration changes for the LDAPS protocol are described in the next section.

b. BindDN has the format “<domain_name>\{$username}”. This specifies how to form a full domain user name. Change the <domain_name> to the actual domain and leave {$username} as a placeholder unless you want to make Active Directory connections with preconfigured “admin” user credentials instead of the current user’s credentials. In the latter case you need to replace the bindCredential {$password} placeholder with the “admin” password as well.

   Note: The password can be stored in a cyphered vault instead of as plain text; however, this configuration is out of scope for this document.

c. Attributes baseFilter, roleFilter, roleAttributeIsDN, roleNameAttributeID and parseRoleNameFromDN should be the same if the target LDAP Server is Microsoft Active Directory.
d. baseCtxDN, rolesCtxDN and getAttributes will be specific to the target LDAP Server scheme.
e. getAttributes specifies a list of additional attributes to pull from the user’s LDAP record. These will be used in the UpdateSynapUserLoginModule, see the next item.

3. com.synapsense.security.UpdateSynapUserLoginModule is used to create or update users authenticated through LDAP in the SynapSense database; essentially, it maps user attributes retrieved from LDAP to user fields in the SynapSense API. Its *userAttributes* will be different in every case; here are some hints:
   a. The number of mappings in *userAttributes* should be equal to the number of parameters in LdapExtLoginModule’s *getAttributes*. You may end up with more or fewer mappings than in this example depending on how much information is available in the user’s record on the target LDAP Server.
   b. The left side of every mapping shows the name of the setter in the ES API User class. The full set is provided in the User Class Fields section.
   c. The right side of every mapping shows one of the names from the LdapExtLoginModule’s *getAttributes*.
4. The RunAs module is used to allow UpdateSynapUserLoginModule to act as an authenticated user on its own. JBoss 7 uses ES API for user updates. It is not a direct JDBC database link as it was in JBoss 4.
   a. roleName attribute must have a “root_permission” value; don’t touch it.
   b. principaLName may have an arbitrary value. This string will appear in the Web Console Activity Log for users authenticated through LDAP and created in the ES database by UpdateSynapUserLoginModule.
5. Restart the Environment Server.
6. Log into Web Console using the local Environment Server administrator account and create User Groups corresponding to the roles in the target LDAP Server, see Mapping LDAP Roles to Environment Server Roles for details.

7. While enabling Active Directory integration for version SS 7.2 and above, update the WebConsole Security configuration as such due to new security features:
   a. Password Expiry Interval is set to 0 so that SS does not process aged passwords.

b. Max Failed Login Attempts is set to 0 so that SS does not block users with excessive login failures since this is the responsibility of Active Directory.

Example

Here is the sample configuration described above. Make sure you change the values highlighted with bold font and check others for correctness in your particular case:

```
<security-domain name="SynapServer" cache-type="default">
  <authentication>
    <login-module code="Remoting" flag="optional">
      <module-option name="password-stacking" value="useFirstPass"/>
    </login-module>
    <login-module code="Database" flag="sufficient">
      <module-option name="unauthenticatedIdentity"
value="anonymous"/>
      <module-option name="dsJndiName" value="java:/SynapDS"/>
      <module-option name="principalsQuery" value="SELECT
user_password as PASSWD FROM tbl_users WHERE user_name=?"/>
      <module-option name="rolesQuery" value="SELECT ifnull(p1.name,
priv.name),'Roles' FROM tbl_users u, user_groups ug, group_privileges
gp, privileges priv left join privileges_parents pp on priv.id =
pp.parent_id left join privileges p1 on (pp.privilege_id=p1.id or
pp.parent_id = p1.id) where u.user_id=ug.usr_id and
ug.grp_id=gp.grp_id and gp.prv_id=priv.id and u.user_name = ? group
by ifnull(p1.name, priv.name)"/>
      <module-option name="hashAlgorithm" value="SHA1"/>
      <module-option name="hashEncoding" value="HEX"/>
      <module-option name="hashUserPassword" value="true"/>
    </login-module>
    <login-module code="com.synapsense.security.LdapExtLoginModule"
flag="required">
      <module-option name="java.naming.provider.url"
value="ldap://hostname:389"/>
      <module-option name="java.naming.security.authentication"
value="simple"/>
      <module-option name="bindDN"
value="<domain_name>\{$username}"/>
      <module-option name="bindCredential" value="{$password}"/>
      <module-option name="baseCtxDN" value="
OU=users,dc=eng,dc=mobilephone,dc=net"/>
      <module-option name="baseFilter" value="(sAMAccountName={0})"/>
      <module-option name="rolesCtxDN"
value="OU=Groups,DC=Synapsense,DC=Int"/>
      <module-option name="roleFilter" value="(member={1})"/>
      <module-option name="roleAttributeIsDN" value="true"/>
      <module-option name="roleNameAttributeID" value="CN"/>
      <module-option name="parseRoleNameFromDN" value="true"/>
```

```
      <module-option name="getAttributes"
value="givenName,sn,mail,telephoneNumber,title"/>
    </login-module>
    <login-module
code="com.synapsense.security.UpdateSynapUserLoginModule"
flag="required">
      <module-option name="userAttributes"
value="FName=givenName,LName=sn,primaryEmail=mail,primaryPhonenumber=
telephoneNumber,title=title"/>
    </login-module>
    <login-module code="RunAs" flag="optional">
      <module-option name="principaLName" value="System"/>
      <module-option name="roleName" value="root_permission"/>
    </login-module>
  </authentication>
</security-domain>
```

Single LDAPS (LDAP over SSL) Server Configuration

The configuration for the LDAPS protocol is mostly the same as for LDAP and only requires the following additional steps:

1. Add the following lines in <JBOSS_HOME>\standalone\configuration\standalone.xml file, system-properties section:

```
<property name="javax.net.ssl.trustStore"
value="<full_path>\server.keystore"/>
<property name="javax.net.ssl.trustStorePassword"
value="<keystore_password>"/>
```

   See **Obtaining Active Directory Certificate** section for details about server.keystore file.

2. Add a new module option anywhere in the com.synapsense.security.LdapExtLoginModule configuration:

```
<module-option name="java.naming.security.protocol" value="ssl"/>
```

3. Change the port and the protocol information in the com.synapsense.security.LdapExtLoginModule configuration:

```
<module-option name="java.naming.provider.url"
value="ldaps://hostname:636"/>
```

Multiple LDAP(S) Servers

It is possible to authenticate users against as many LDAP(S) servers a needed. For every LDAP(S) server, add a corresponding com.synapsense.security.LdapExtLoginModule directly after the previous one. The first one must go directly after the Database login module. The last one must be before com.synapsense.security.UpdateSynapUserLoginModule. Make sure that for every com.synapsense.security.LdapExtLoginModule you set the *flag=optional*, not *required* as for a single LDAP(S) server.

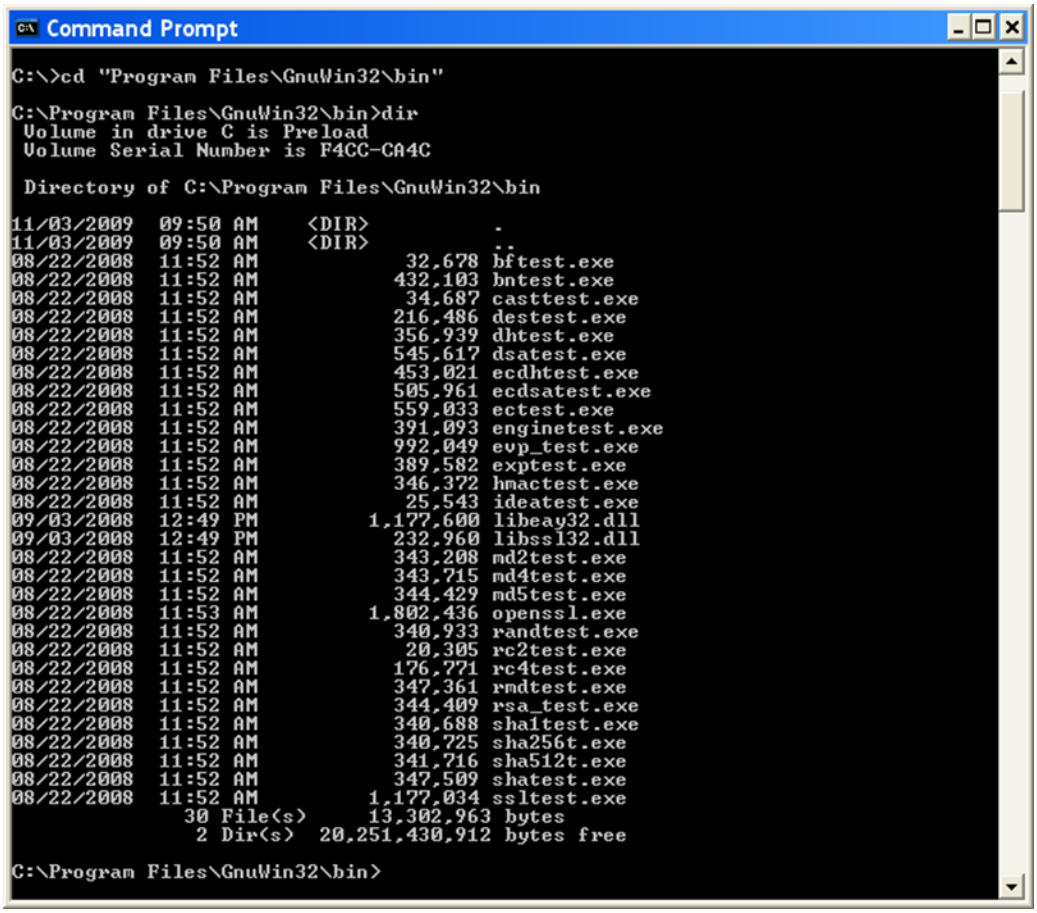
Mapping LDAP Roles to Environment Server Roles

Starting with the SynapSense version 6.4, release, the Environment Server uses a much more flexible User Group Management mechanism that allows mapping LDAP Server roles into Environment Server groups without any mapping files. Just login to Web Console using any Environment Server account with administrator privileges (local Environment Server administrator for the first time you configure LDAP) and create User Groups with names matching the roles from LDAP Server; then assign privileges as required.

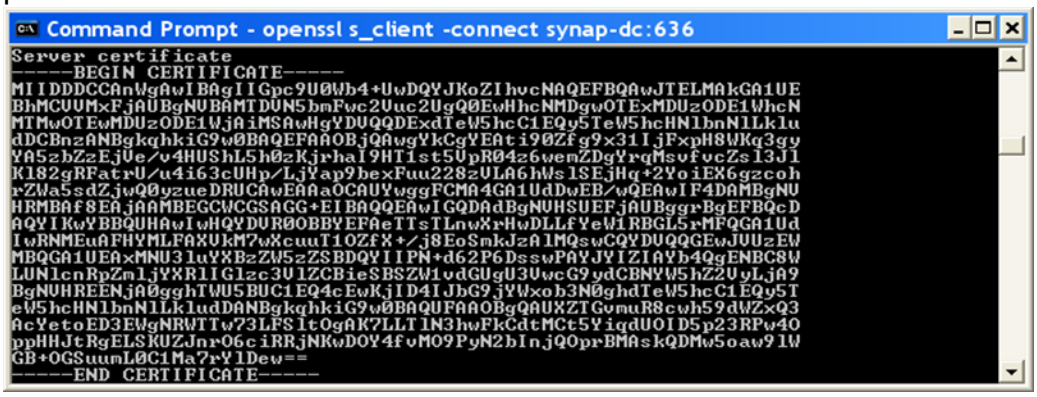
Obtaining the Active Directory Certificate

For every LDAPS server in the configuration, you need to obtain and store its certificate to be able to make SSL connections. The certificate can be requested from the LDAP server administrator or retrieved with an openssl tool using the following steps:

1. Install the latest openssl.exe from **http://gnuwin32.sourceforge.net/packages/openssl.htm** on the host running the SynapSense server.
2. Open a command line interface (DOS) on the server where SynapSense is installed. Change directory to C:\Program Files\GnuWin32\bin (Default install directory for openssl).



3. Use the openssl tool to retrieve an active directory certificate. Run openssl s_client -connect <AD hostname>.com:636, where hostname is AD name. Copy the lines between BEGIN CERTIFICATE and END CERTIFICATE (including these two) into notepad and save it as “server.pem” file.



4. Openssl is only required to generate the keystore and is not part of SynapSense LDAP/Active Directory integration. It may therefore be uninstalled once the certificate generates successfully.

When you have the server certificate file, store it in a keystore file. Note: If you have HTTPS enabled in Web Console, you can use the same file for LDAPS SSL certificates; otherwise, you will need to create one. In either case, the following steps are the same:

1. Using the keytool included with the JDK, import saved key into a keystore: keytool -import -alias <AD hostname> -file server.pem -keystore server.keystore. Answer “yes” when the tool asks if you want to trust this certificate.
   Note: Remember to write down the new keystore password if you’re creating the new keystore file.
2. Save the server.keystore file in <*JBOSS_HOME*>*standalone**configuration* folder.
   Note: You can store it in any other folder; this one is just a convenient default.
3. If you need to remove the certificate from the keystore, then execute “%JAVA_HOME%\bin\keytool" -delete -alias <AD hostname> -keystore server.keystore. You may be asked to re-enter the storepass value to complete the delete step.

Repeat the steps from this section for every LDAPS server you want to authenticate user against.

Login Modules Documentation

com.synapsense.security.LdapExtLoginModule

This is a slightly modified version of org.jboss.security.auth.spi.LdapExtLoginModule that allows the use of not only fixed credentials for LDAP binding and searching but those of an active Subject, too. Optionally, this can pull additional attributes from a user's LDAP record and inject them into Subject.

Modified module options:

bindDN, bindCredential – These now support macro expansion. Supported macros are {$username} and {$password}. For example, if you need not fixed but current user credentials and some specific domain to be used for LDAP binding, then set bindDN=mydomain\{$username} and bindCredential={$password}

New module options:

getAttributes – A comma-separated list of attributes to pull from a user record. The record will be located with the same baseCtxDn and baseFilter used for authentication. Optionally, the default is **none** (null string).

attributesPrincipal Name of a principal Group in which to put the attributes. Optionally, the default is **Attributes**.

Other module options are exactly the same as for org.jboss.security.auth.spi.LdapExtLoginModule.

com.synapsense.security.UpdateSynapUserLoginModule

This module does not perform any validation but merely updates user information stored in the SynapSense Database with data from an already authenticated Subject.

In addition, it performs two more steps:

1. If any subject's role has a matching group in ES then it will add that group’s privileges (recursively) to the subject's roles. Effectively, this step maps ES groups/privileges to JBoss/EJB roles.

2. If any attribute mappings are specified and corresponding attributes were provided by previous login modules then it will use them to create/update user fields (first/last name, email, title etc.).

Module options:

userAttributes – A string in form userField=attrName[,...] where userField is User class setter name and attrName is an attribute name.
For example, FName=givenName,title=title will take values of attributes 'givenName' and 'title' and pass them to User.setFName() and User.setTitle() methods. Optionally, the default is none (null string). See User Class Fields for the complete list.

attributesPrincipal – Name of a principal Group from which to take attribute values. Optionally, the default is **Attributes**. Must be the same value as in attributesPrincipal of com.synapsense.security.LdapExtLoginModule.

Note: This module works via ES API; you need to add RunAsLoginModule after (not before) it.

User Class Fields

The following fields can be used as the right side in *userAttributes* mapping of com.synapsense.security.UpdateSynapUserLoginModule:

FName – First Name

LName – Last Name

middleInitial – Middle Name Initials

title – Title

primaryPhonenumber – Primary Phone Number

secondaryPhonenumber – Primary Phone Number

primaryEmail – Primary Email Address

secondaryEmail – Secondary Email Address

smsAddress – SMS Address (an email of SMS gateway service)

description – Description