**Java LDAP SSL authentication**

* [JAVA](https://roufid.com/category/java/)
* 13 APR, 2016
* [10 COMMENTS](https://roufid.com/java-ldap-ssl-authentication/#disqus_thread)

In this article, we will see how to make a secured LDAP authentication using Java.

First thing to do is importing the trust certificate to Java keystore. The default java keystore is named ***cacerts***and is located in***$JAVA\_HOME/jre/lib/security/cacerts***.

***$JAVA\_HOME****is where your JDK is installed (By default : C:/Program Files/Java/jdk1.7.0\_79/).*

**Table of contents**[[hide](https://roufid.com/java-ldap-ssl-authentication/)]

* [1 1- Register your SSL certificate into Java keystore](https://roufid.com/java-ldap-ssl-authentication/#1-_register_your_ssl_certificate_into_java_keystore)
* 2 2- Running your application with SSL
  + [2.1 Writing code in your application](https://roufid.com/java-ldap-ssl-authentication/#writing_code_in_your_application)
* [3 3- LDAP Connection](https://roufid.com/java-ldap-ssl-authentication/#3-_ldap_connection)

1- Register your SSL certificate into Java keystore

Java offers the certificate management utility [keytool](http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/keytool.html" \t "_blank) to handle certificates into your keystore. To import a certificate, you need to specify three arguments :

* **-keystore** : *Absolute path to your keystore. (By default : C:\Program Files\Java\jdk1.7.0\_79\jre\lib\security)*
* **-alias** : *Give a name to your certificate  The given name should not already exist in the keystore*
* **-file** : *Absolute path to the certificate you want to import*

Use the following code to import your certificate into the default java keystore :

|  |  |
| --- | --- |
| 1 | keytool -keystore <PATH\_TO\_JRE>/lib/security/cacerts -import -alias certificate -file <PATH\_TO\_CERTIFICATE>/certificate.cer |

The default keystore password is : **changeit**

2- Running your application with SSL

To allow your application to connect over SSL you can choose one of the two solutions :

* Writing code in your application

Add the following sytem properties before your LDAP authentication

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | //Path to your keystore where you registred the SSL certficate  String keystorePath = "C:/Program Files/Java/jdk1.7.0\_79/jre/lib/security/cacerts";  System.setProperty("javax.net.ssl.keyStore", keystorePath);    // Password of your java keystore. Default value is : changeit  System.setProperty("javax.net.ssl.keyStorePassword", "changeit"); |

* **Add JVM arguments**

Start your application by adding the folowing arguments :

|  |  |
| --- | --- |
| 1 | -Djavax.net.ssl.keyStore="C:/Program Files/Java/jdk1.7.0\_79/jre/lib/security/cacerts" -Djavax.net.ssl.keyStorePassword=changeit |

3- LDAP Connection

Once you added the trusted certificate to Java keystore and started your application with the required arguments, you can use the following code to make a LDAP authentication :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | // Setting the LDAP connection information  Hashtable<String, String> env = new Hashtable<String, String>();  env.put(Context.PROVIDER\_URL, "ldaps://server.local:636");  env.put(Context.INITIAL\_CONTEXT\_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");  env.put(Context.SECURITY\_AUTHENTICATION, "simple");  env.put(Context.SECURITY\_PRINCIPAL, "cn=radouane,ou=people,o=RoufidTutorials,c=fr");  env.put(Context.SECURITY\_CREDENTIALS, "password");    DirContext ctx = null;    try {  // Openning the connection  ctx = new InitialDirContext(env);    // Use your context here...  } catch (NamingException e) {  System.out.println("Problem occurs during context initialization !");  e.printStackTrace();  } |

# How to connect to LDAPS using Java

[Jose Ramon Huerga](https://blogs.mulesoft.com/author/joseramonhuerga/)

March 31, 2021 | 11 mins read

##### **Share post**

LDAP — or lightweight directory access protocol — is a protocol for accessing directory information services where information such as users and groups is stored. Historically, there have been two communication mechanisms to interact with an LDAP server:

* LDAPS (LDAP over SSL) which uses port 636, configured to require SSL. This protocol is old, but is still in use by many organizations.
* LDAP over TLS (StartTLS) using port 389. This protocol is the recommended method for organizations today.

Anypoint Platform provides an [LDAP connector](https://docs.mulesoft.com/ldap-connector/3.3/) that can communicate with servers such as OpenLDAP, Apache Directory, or Microsoft Active Directory. However, this connector supports only LDAP over TLS (StartTLS) so it cannot be used by companies that are still using LDAPS.

UPCOMING EVENT

Imagine, integrate, and innovate at the #1 integration event

[Register for free](https://connect.mulesoft.com/events/connect/apac?utm_source=msdc&utm_medium=referral&utm_campaign=blog-page-2021-inline&iesrc=ctr)

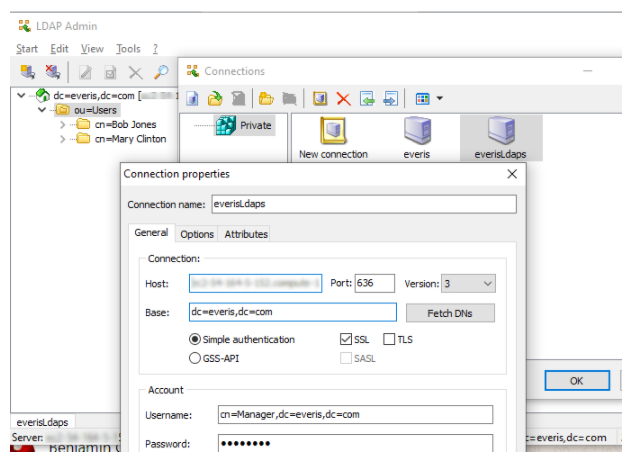
To avoid that limitation of the LDAP connector, I have just written a Mule flow that can connect to an LDAPS server by using a custom Java method.

## ****How to configure an LDAPS server****

To try the code, you will need an LDAP server listening on port 636 using SSL. Installing and configuring an OpenLDAP server is out of the scope of this article, but you may find a step-by-step guide [here](https://www.thegeekstuff.com/2015/01/openldap-linux/). Once you have an LDAP server up and running on port 389 you will need to change its configuration to listen on port 636 by following [these steps](https://www.itzgeek.com/how-tos/linux/centos-how-tos/configure-openldap-with-ssl-on-centos-7-rhel-7.html). To configure SSL you will need a certificate, you can either use a certificate signed by a Certificate Authority (CA) or a self-signed certificate.

To check if your LDAP server is running properly, use [LDAP Admin](http://www.ldapadmin.org/), which is a handy GUI that you can install on your laptop.

In this screenshot, I have configured an LDAP server that contains two users (Bob Jones and Mary Clinton) and that it is listening on port 636 using SSL:

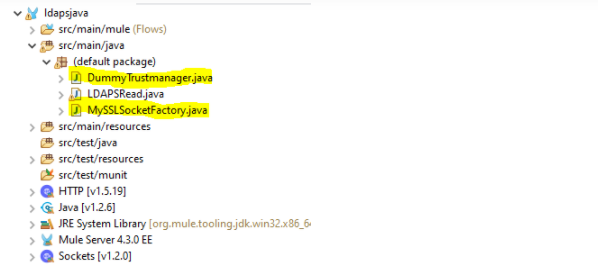


## ****Writing Java code to connect to LDAPS****

To configure your LDAP server to use SSL you have used a certificate. In case it is a self-signed certificate, Mule will get an error during the handshaking phase of the SSL communication unless you do one of these two actions:

* Download the certificate and import it on the trust store of the JVM.
* Create a dummy trust manager and a custom SSL Socket Factory.

In this example, we are going to use the second choice. To do that, it is needed to create two Java classes in the default package:



The first Java class is a dummy trust manager; it just contains these three methods:

public class DummyTrustmanager implements X509TrustManager {

    public void checkClientTrusted(X509Certificate[] xcs, String string) throws CertificateException {

        // do nothing

    }

    public void checkServerTrusted(X509Certificate[] xcs, String string) throws CertificateException {

        // do nothing

    }

    public X509Certificate[] getAcceptedIssuers() {

        return new java.security.cert.X509Certificate[0];

    }

}

The second Java class is a custom SSL Socket Factory; this is an excerpt of the code of that class:

public class MySSLSocketFactory extends SSLSocketFactory {

    private SSLSocketFactory socketFactory;

    public MySSLSocketFactory() {

        try {

            SSLContext ctx = SSLContext.getInstance("TLS");

            ctx.init(null, new TrustManager[] { new DummyTrustmanager() }, new SecureRandom());

            socketFactory = ctx.getSocketFactory();

        } catch (Exception ex) {

            ex.printStackTrace(System.err);

            /\* handle exception \*/

        }

    }

}

To interact with the LDAP server I have created the following Java static method that it is able to search for one user in the directory and retrieve its phone number:

public class LDAPSRead {

  public static String getUserInfo(String url, String principal, String credentials, String search, String name) {

    String[] attrIDs = { "sn", "cn", "telephoneNumber" };

    String phone = "111";

      SearchControls ctls = new SearchControls();

      ctls.setReturningAttributes(attrIDs);

      ctls.setSearchScope(SearchControls.SUBTREE\_SCOPE);

      String filter = "(cn="+name+")";

    Hashtable env = new Hashtable();

    env.put(DirContext.INITIAL\_CONTEXT\_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");

    env.put(DirContext.PROVIDER\_URL, url );

    env.put(DirContext.SECURITY\_AUTHENTICATION, "simple");

    env.put(DirContext.SECURITY\_PRINCIPAL, principal);

    env.put(DirContext.SECURITY\_CREDENTIALS, credentials);

    env.put("java.naming.ldap.factory.socket", "MySSLSocketFactory");

    DirContext dirContext = null;

    try {

      dirContext  = new InitialDirContext(env);

      NamingEnumeration e = dirContext.search(search, filter,ctls);

      while (e.hasMore()) {

        SearchResult entry = (SearchResult) e.next();

        System.out.println(entry.getName());

        Attributes attrs = entry.getAttributes();

        Attribute attr = attrs.get("telephoneNumber");

        phone = (String) attr.get();

      }

    }

    catch (Exception e) {

      System.err.println(e);

    }

    return phone;

  }

}

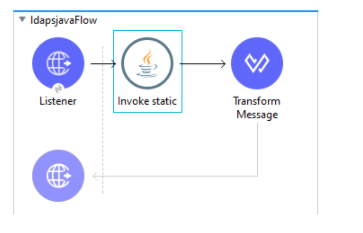
These are the main sections of the method:

* It accepts as parameters the URL of the LDAP server, the principal user and its password, the branch where the users are stored, and the user name.
* It uses the standard Java package javax.naming.directory to connect with the LDAP server.
* It populates a class InitialDirContext with several properties, including the URL of the server, the principal user, its password, and the custom SSL Socket Factory created on previous steps.
* It searches in the LDAP server the user name and retrieves an attribute with the telephone number.

This method is read only; in case you need to modify the content of the directory, you can easily create a second method that — for example — uses the operation dirContext.modifyAttributes() to change an attribute.

## ****Creating a Mule flow to use the custom Java method****

To test the custom Java method, I have created this simple flow that listens on port 8081 and invokes the static Java method using the [Java module](https://docs.mulesoft.com/java-module/1.2/) of Mule:



This is the configuration of the Java Invoke static module. Note, that the parameters are passed using DataWeave:

The properties are stored on a file called settings.properties on the folder src/main/resources:

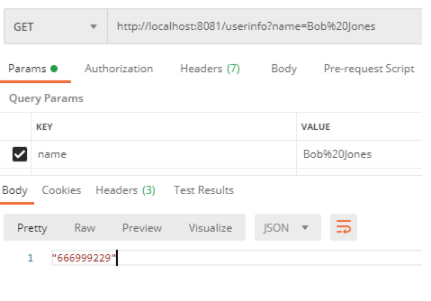
url=ldaps://ldaps.compute-1.amazonaws.com

principal=cn=Manager,dc=company,dc=com

credentials=password

search=dc=company,dc=com

To test the flow, I have used Postman, passing as a query parameter the name of the user to be searched:



As you can see, the flow takes the value of the query param userinfo and passes it to the custom Java method which in turn, returns the telephone number of that user stored in the LDAPS server.

## ****Key takeaways****

If you’re using an LDAPS server and you cannot use the standard LDAP Connector of Mule, you can use this custom Java method to interact with your LDAPS server. If the server has been configured with a self-signed certificate, and you do not want to modify the trust store of the JVM, you can use the dummy trust manager and the custom SSL Socket Factory provided in this article.

Although Mule provides hundreds of connectors, sometimes it is useful to write your own Java method. As you can see in this example, it is easy to execute custom Java code by using the standard Java module of Mule.

If you want to try the code, you can download it from [this repository](https://github.com/jrhuerga/ldaps-java). You can find more information about the Mule 4 LDAP connector [here](https://docs.mulesoft.com/ldap-connector/3.3/).

Jose Ramon Huerga is a MuleSoft Ambassador, you can find him on LinkedIn [here](https://www.linkedin.com/in/jrhuerga/).

### Java LDAP SSL connection example

**package**com.ldap;  
  
**import**java.util.Hashtable;  
  
**import**javax.naming.Context;  
**import**javax.naming.NamingEnumeration;  
**import**javax.naming.NamingException;  
**import**javax.naming.directory.Attribute;  
**import**javax.naming.directory.Attributes;  
**import**javax.naming.directory.BasicAttribute;  
**import**javax.naming.directory.DirContext;  
**import**javax.naming.directory.ModificationItem;  
**import**javax.naming.directory.SearchControls;  
**import**javax.naming.directory.SearchResult;  
**import**javax.naming.ldap.InitialLdapContext;  
**import**javax.naming.ldap.LdapContext;  
  
**public**class LdapExample {  
  
    **public**static final **String**JAVA\_HOME = "C:\\Program Files\\Java\\jdk1.8.0\_131";  
     
    **public**static void main(String[] args)  
    {  
        final **String**ldapAdServer = "host.com:6368";  
        final **String**ldapSearchBase = "dc=bam,dc=edu,dc=hk";  
  
        final **String**ldapUsername = "uid=37b7fe15, dc=bam, dc=edu, dc=hk";  
        final **String**ldapPassword = "bambam";  
         
        final **String**ldapAccountToLookup = "(|(universityid=ddz044071))";  
         
        LdapContext ctx = null;  
        try  
        {  
            Hashtable env = new Hashtable();  
            env.put(Context.SECURITY\_AUTHENTICATION, "simple");  
            if(ldapUsername != null)  
            {  
                env.put(Context.SECURITY\_PRINCIPAL, ldapUsername);  
            }  
            if(ldapPassword != null)  
            {  
                env.put(Context.SECURITY\_CREDENTIALS, ldapPassword);  
            }  
  
            env.put(Context.SECURITY\_PROTOCOL, "ssl");  
            env.put(Context.INITIAL\_CONTEXT\_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");  
             
            if( "ssl".equals(env.get(Context.SECURITY\_PROTOCOL)) )  
            {  
                env.put(Context.PROVIDER\_URL, "ldaps://"+ldapAdServer);  
            }  
            else  
            {  
                env.put(Context.PROVIDER\_URL, "ldap://"+ldapAdServer);  
            }  
  
            System.out.println("java home: "+System.getProperty("java.home"));  
            System.setProperty("javax.net.ssl.keyStore", JAVA\_HOME+"\\jre\\lib\\security\\cacerts");  
            System.setProperty("javax.net.ssl.trustStore", JAVA\_HOME+"\\jre\\lib\\security\\cacerts");  
             
             
            //ensures that objectSID attribute values  
            //will be returned as a byte[] instead of a String  
            env.put("java.naming.ldap.attributes.binary", "objectSID");  
             
            // the following is helpful in debugging errors  
            //env.put("com.sun.jndi.ldap.trace.ber", System.err);  
             
            SearchControls ctls = new SearchControls();  
            ctls.setSearchScope(SearchControls.SUBTREE\_SCOPE);  
            ctls.setReturningAttributes(  
                new String[] {  
                    "ds-cfg-listen-port",  
                    "objectclass"  
                } );  
             
            ctx = new InitialLdapContext( env, null );  
  
            LdapExample ldap = new LdapExample();  
             
            NamingEnumeration list = ldap.findAccount(ctx,  
                    ldapSearchBase, ldapAccountToLookup);  
             
            listResult( list );  
             
             
            // modify attributes  
            ModificationItem[] mods = new ModificationItem[1];  
  
         Attribute mod0 = new BasicAttribute("visible", "yes");  
         //Attribute mod1 = new BasicAttribute("number2", "AAA");  
  
         mods[0] = new ModificationItem(DirContext.REPLACE\_ATTRIBUTE, mod0);  
         //mods[1] = new ModificationItem(DirContext.ADD\_ATTRIBUTE, mod1);  
  
         ctx.modifyAttributes("uid=304f8d81-5bce11e7-80babd7d-ae3f753a,dc=cuhk,dc=edu,dc=hk", mods);  
        }  
        catch(Exception e)  
        {  
            e.printStackTrace();  
        }  
        finally  
        {  
            try  
            {  
                if(ctx!=null)  
                {  
                    ctx.close();  
                }  
            }  
            catch(Exception e)  
            {  
            }  
        }  
        System.out.println("LdapExample exit");  
    }  
     
    **public**NamingEnumeration findAccount(DirContext ctx,  
            **String**ldapSearchBase, **String**searchFilter) throws NamingException {  
  
        SearchControls searchControls = new SearchControls();  
        searchControls.setSearchScope(SearchControls.SUBTREE\_SCOPE);  
  
        NamingEnumeration results = ctx.search(ldapSearchBase, searchFilter, searchControls);  
  
        return results;  
    }  
     
    **public**static void listResult(NamingEnumeration results) throws NamingException  
    {  
        **int**i = 1;  
        SearchResult searchResult = null;  
        while(results.hasMoreElements())  
        {  
            searchResult = (SearchResult) results.nextElement();  
            System.out.println("============ record "+i+" ============");  
            System.out.println( searchResult.getNameInNamespace() );  
            Attributes attrs = searchResult.getAttributes();  
            NamingEnumeration enumer = attrs.getAll();  
            Attribute attrObj;  
            while( enumer.hasMore() )  
            {  
                attrObj = (Attribute)enumer.next();  
                //System.out.println( "attr: "+attrObj.getClass().getName() );  
                //System.out.println( "attr: "+attrObj );  
                 
                System.out.print( "name: "+attrObj.getID()+", " );  
                System.out.println( "value: "+attrObj.get() );  
                //System.out.println( "attr: "+(attrObj instanceof javax.naming.directory.Attribute) );  
                 
                //break;  
            }  
            i++;  
        }  
    }  
}