SGCWEBSOCKETS · TECHNICAL DOCUMENT

LDAP Client

TsgcLDAPClient: LDAP v3 and Active Directory client to authenticate users, search the directory and read group membership.

eSeGeCe

WWW.ESEGECE.COM

Overview

Let your users sign in with their Windows domain password. Validate credentials against Active Directory, OpenLDAP or any LDAP v3 directory over an encrypted connection, and read the groups that decide what they can do.

The **TsgcLDAPClient** component is an LDAP v3 client (RFC 4511, RFC 4513) designed to authenticate users against **Active Directory**, OpenLDAP or any other LDAP directory, search the directory and read the groups of a user. It uses Indy for the transport and the OpenSSL IO handler for TLS.

Only one operation runs at a time, every public method is serialized, so a single instance can be shared by several threads.

At a glance

COMPONENT CLASS `TsgcLDAPClient`	STANDARDS / SPEC **LDAP v3, RFC 4511 and RFC 4513**
TRANSPORTS **TCP, TLS (LDAPS and StartTLS)**	PLATFORMS **Windows, macOS, Linux, iOS, Android**
FRAMEWORKS **VCL, FireMonkey**	EDITION **Enterprise (also sgcAuth pack)**

Features

- **LDAPS and StartTLS.** `Security` selects `ldapsecLDAPS` (implicit TLS on port 636) or `ldapsecStartTLS` (upgrade on port 389). If the server refuses StartTLS the connection is closed, the client never falls back to clear text. `TLSOptions` controls certificate verification.
- **Four sign-in modes.** `AuthenticationMode` turns the typed name into a bind name: `ldapamUPN` (user@ `UPNSuffix`), `ldapamDownLevel` (`Domain`\user), `ldapamSearchThenBind` (service bind plus `UserSearchFilter`) or `ldapamDN`.

- **Nested groups.** `GetUserGroups` returns the direct `memberOf` values, or with `aNested` every group reached through other groups, using the Active Directory rule LDAP_MATCHING_RULE_IN_CHAIN.
- **Paged search.** `Search` uses Simple Paged Results automatically (`PageSize` , 500 by default), honours `SizeLimit` and `TimeLimit` , and returns the entries and referrals in a `TsgcLDAPEntries` list.
- **Safe binds.** `Bind` refuses a DN with an empty password without contacting the server, closing the unauthenticated bind hole of RFC 4513. `WhoAmI` , `LastResultCode` and `LastErrorMessage` tell you exactly who is bound and why a call failed.
- **One instance, many threads.** Every public method is serialized, so a single `TsgcLDAPClient` can serve the login requests of a multi-threaded HTTP or WebSocket server.

Technical specification

Standards & specs	RFC 4511 · RFC 4513 · RFC 2696 · RFC 4532
Component class	`TsgcLDAPClient` (unit `sgcAuth_LDAP_Client`)
Frameworks	VCL, FireMonkey
Platforms	Windows, macOS, Linux, iOS, Android
Edition	sgcWebSockets Enterprise, also sold in the sgcAuth pack

Main properties

The published and public properties used to configure and drive the component.

`AuthenticationMode`	How Authenticate turns the user name into a bind name: UPN, DownLevel, SearchThenBind or DN.
`BaseDN`	DN where the user and group searches start.
`BindDN`	DN of the service account used by SearchThenBind and restored after every Authenticate.
`ConnectTimeout`	Maximum time in milliseconds to open the TCP connection.
`Domain`	NetBIOS domain name prepended to user names without a backslash in the DownLevel mode.
`Host`	Host name or IP address of the LDAP server or domain controller.
`LastErrorMessage`	Diagnostic message of the last operation.
`LastMatchedDN`	Matched DN returned by the server with the last result.
`LastResultCode`	Result code of the last operation, 0 is success.
`PageSize`	Number of entries requested per page with the Simple Paged Results control, 0 disables paging.
`Password`	Password of the service account BindDN.

`Port`	TCP port of the LDAP server.
`ReadTimeout`	Maximum time in milliseconds to wait for a complete response from the server.
`Security`	Transport security: none, LDAPS (implicit TLS) or StartTLS.
`SizeLimit`	Maximum number of entries the server returns for a search, 0 means no limit.
`TimeLimit`	Maximum time in seconds the server spends on a search, 0 means no limit.
`TLSOptions`	TLS configuration: certificate verification, CA file, client certificate, TLS version and OpenSSL options.
`UPNSuffix`	Suffix appended to user names without @ in the UPN mode.
`UserSearchFilter`	Filter used to find the entry of a user, %s is replaced by the escaped user name.
`Version`	Read-only string exposing the sgcWebSockets library version.

Main methods

The public methods exposed by the component.

`Authenticate`	Validates the credentials of a user according to AuthenticationMode and returns the DN of the user.
`Bind`	Simple bind with a DN and a password, returns False when the server refuses it.
`Connect`	Opens the connection and runs the TLS handshake or StartTLS configured in Security.
`Connected`	Returns True when the TCP connection is open.
`Disconnect`	Sends an Unbind request and closes the connection.
`GetUserGroups`	Returns the DNs of the groups of a user, optionally including the nested groups.
`Search`	Searches the directory and returns the entries, reading all the pages when PageSize is greater than 0.
`WhoAmI`	Runs the WhoAmI extended operation (RFC 4532) and returns the identity of the connection.

Events

The events fired by the component.

`OnConnect`	Fired when the connection, including the TLS handshake, is established.
`OnDisconnect`	Fired when an established connection is closed.
`OnError`	Fired with the message of every error raised by a public method.

Quick Start

Point Host at a domain controller, choose LDAPS or StartTLS, set the service account in BindDN / Password, then call Authenticate with what the user typed.

> **About this scenario.** Connect, authenticate, read the groups. The same code ships in the demo `Demos\26.Authentication\05.LDAP_ActiveDirectory` .

Delphi (VCL / FireMonkey)

```
uses
  sgcAuth_LDAP_Classes, sgcAuth_LDAP_Client;

var
  LDAP: TsgcLDAPClient;
  vUserDN: string;
  oGroups: TStringList;
begin
  LDAP := TsgcLDAPClient.Create(nil);
  LDAP.Host := 'dc01.corp.example.com';
  LDAP.Security := ldapsecLDAPS; // implicit TLS, port 636
  LDAP.BindDN := 'CN=svc-login,OU=Service,DC=corp,DC=example,DC=com';
  LDAP.Password := 'service-password';
  LDAP.BaseDN := 'DC=corp,DC=example,DC=com';
  // find the user with UserSearchFilter, then bind as that DN
  LDAP.AuthenticationMode := ldapamSearchThenBind;
  LDAP.Connect;

  if LDAP.Authenticate(edtUser.Text, edtPassword.Text, vUserDN) then
  begin
    oGroups := TStringList.Create;
    try
      // True: nested groups through LDAP_MATCHING_RULE_IN_CHAIN
      LDAP.GetUserGroups(vUserDN, oGroups, True);
    finally
      oGroups.Free;
    end;
  end
  else
    ShowMessage(LDAP.LastErrorMessage);
end;
```

C++ Builder

```
// uses: sgcAuth_LDAP_Classes, sgcAuth_LDAP_Client
TsgcLDAPClient *LDAP = new TsgcLDAPClient(this);
LDAP->Host = "dc01.corp.example.com";
LDAP->Security = ldapsecLDAPS;
LDAP->BindDN = "CN=svc-login,OU=Service,DC=corp,DC=example,DC=com";
LDAP->Password = "service-password";
LDAP->BaseDN = "DC=corp,DC=example,DC=com";
LDAP->AuthenticationMode = ldapamSearchThenBind;
LDAP->Connect();

String UserDN;
if (LDAP->Authenticate(edtUser->Text, edtPassword->Text, UserDN))
{
  TStringList *Groups = new TStringList();
  LDAP->GetUserGroups(UserDN, Groups, true);
  delete Groups;
}
```

Common scenarios

The following topics come from the online help. Each one explains a part of the component and shows the Delphi code that drives it.

1 · Security modes

The **Security** property selects how the connection is protected:

- `ldapsecNone` : plain LDAP on port 389. Passwords travel in clear text, use it only on a trusted network or for tests.
- `ldapsecLDAPS` : implicit TLS, the TLS handshake starts as soon as the TCP connection is open. Selecting it changes **Port** from 389 to 636.
- `ldapsecStartTLS` : the client connects to port 389 and upgrades the connection with the StartTLS extended operation (RFC 4511 section 4.14) before anything else is sent. If the server refuses StartTLS, **Connect** raises an exception and the connection is closed, the client never continues in clear text.

The TLS settings are in **TLSOptions**. **TLSOptions.VerifyCertificate** is **True** by default: the certificate chain of the server is verified by OpenSSL against the CA certificates of **TLSOptions.RootCertFile**, and with the sgcWebSockets Indy library the certificate must also match **Host**. For Active Directory set **RootCertFile** to a PEM file with the root certificate of your enterprise CA. When the handshake fails, the exception message includes the OpenSSL verification error, for example "unable to get local issuer certificate". Only the OpenSSL IO handler is supported.

Delphi (VCL / FireMonkey)

```
oLDAP := TsgcLDAPClient.Create(nil);
oLDAP.Host := 'dc01.contoso.com';
oLDAP.Security := ldapsecLDAPS; // Port changes to 636
oLDAP.TLSOptions.VerifyCertificate := True;
oLDAP.TLSOptions.RootCertFile := 'contoso-root-ca.pem';
oLDAP.BaseDN := 'DC=contoso,DC=com';
oLDAP.Connect;
```

2 · Active Directory authentication

Authenticate validates the credentials of a user with a simple bind. **AuthenticationMode** selects how the user name is turned into the bind name. After every call the connection is bound again with **BindDN** and **Password** (the service account, or an anonymous bind when both are empty), so the connection never stays bound as the user.

UPN (`ldapamUPN`): the user signs in with a user principal name such as `john@contoso.com` . When the name has no `@` , **UPNSuffix** is appended.

Delphi (VCL / FireMonkey)

```
oLDAP.AuthenticationMode := ldapamUPN;
oLDAP.UPNSuffix := 'contoso.com';
if oLDAP.Authenticate('john', vPassword, vUserDN) then // binds as john@contoso.com
  ShowMessage('Welcome ' + vUserDN);
```

DownLevel (`ldapamDownLevel`): the classic `DOMAIN\user` logon name. When the name has no `\` , **Domain** is prepended.

Delphi (VCL / FireMonkey)

```
oLDAP.AuthenticationMode := ldapamDownLevel;
oLDAP.Domain := 'CONTOSO';
if oLDAP.Authenticate('john', vPassword, vUserDN) then // binds as CONTOSO\john
  ShowMessage('Welcome ' + vUserDN);
```

SearchThenBind (`ldapamSearchThenBind` , the default): the client binds with the service account (**BindDN**, **Password**), searches **UserSearchFilter** under **BaseDN** and binds with the DN it found. The search must return exactly one entry. This mode works with any directory and lets the user sign in with the `sAMAccountName` , or any attribute you put in the filter.

Delphi (VCL / FireMonkey)

```
oLDAP.AuthenticationMode := ldapamSearchThenBind;
oLDAP.BindDN := 'CN=svc-ldap,OU=Service Accounts,DC=contoso,DC=com';
oLDAP.Password := vServicePassword;
oLDAP.BaseDN := 'DC=contoso,DC=com';
// default: (&(objectCategory=person)(objectClass=user)(sAMAccountName=%s))
if oLDAP.Authenticate('john', vPassword, vUserDN) then
  ShowMessage('Welcome ' + vUserDN);
```

DN (`ldapamDN`): the user name is already the full distinguished name.

Delphi (VCL / FireMonkey)

```
oLDAP.AuthenticationMode := ldapamDN;
if oLDAP.Authenticate('CN=John Smith,OU=Users,DC=contoso,DC=com', vPassword, vUserDN) then
  ShowMessage('Valid credentials');
```

With UPN and DownLevel, when **BaseDN** is set, the DN of the user entry is searched after the bind and returned in `aUserDN`. Otherwise `aUserDN` returns the bind name.

3 · Empty passwords are rejected

RFC 4513 section 5.1.2 defines a bind with a name and an empty password as an *unauthenticated bind*, and many servers answer it with success. A login form which forwards an empty password would accept any user name. **Authenticate** and **Bind** refuse a name with an empty password without contacting the server: they return False with **LastResultCode** = 48 (inappropriateAuthentication). An empty DN with an empty password is still an anonymous **Bind**.

4 · Search with paging

Search returns the entries in a **TsgcLDAPEntries** list. When **PageSize** is greater than 0 (500 by default) the client uses the Simple Paged Results control (RFC 2696) and requests the next pages automatically, so searches return more entries than the page limit of the server (1000 in Active Directory). Continuation references and referrals are collected in **Referrals**, they are never followed.

Delphi (VCL / FireMonkey)

```
oAttributes := TStringList.Create;
oEntries := TsgcLDAPEntries.Create;
try
  oAttributes.Add('cn');
  oAttributes.Add('mail');
  oLDAP.Search('OU=Users,DC=contoso,DC=com', '(&(objectClass=user)(mail=*))',
    ldapscWholeSubtree, oAttributes, oEntries);
  for i := 0 to oEntries.Count - 1 do
  begin
    oAttribute := oEntries[i].Attributes.Find('mail');
    if Assigned(oAttribute) then
      Memo1.Lines.Add(oEntries[i].DN + ': ' + oAttribute.Values.Text);
  end;
finally
  oEntries.Free;
  oAttributes.Free;
end;
```

5 · Groups

GetUserGroups returns the DNs of the groups of a user. Without nesting it reads the `memberOf` attribute of the user entry, which only lists the direct groups. With `aNested` = True it searches under **BaseDN** with the Active Directory matching rule `LDAP_MATCHING_RULE_IN_CHAIN` (`1.2.840.113556.1.4.1941`), which also returns the groups the user belongs to through other groups: `(member:1.2.840.113556.1.4.1941:=<user DN>)` .

Delphi (VCL / FireMonkey)

```
oGroups := TStringList.Create;
try
  oLDAP.GetUserGroups(vUserDN, oGroups, True);
  vIsAdmin := oGroups.IndexOf('CN=App Admins,OU=Groups,DC=contoso,DC=com') ≥ 0;
finally
  oGroups.Free;
end;
```

6 · Escaping helpers against LDAP injection

Never concatenate user input into a filter or a DN. The unit `sgcAuth_LDAP_Classes` declares the escaping functions:

- `sgcLDAPEscapeFilterValue(const aValue: string): string`: escapes `*`, `(`, `)`, `\` and NUL of a filter assertion value (RFC 4515).
- `sgcLDAPEscapeDNValue(const aValue: string): string`: escapes an attribute value used inside a DN (RFC 4514).

Authenticate and **GetUserGroups** already escape the values they put in their filters. Use the helpers for your own **Search** calls.

Delphi (VCL / FireMonkey)

```
vFilter := '(&(objectClass=user)(mail=' + sgcLDAPEscapeFilterValue(edtMail.Text) + '))';
vDN := 'CN=' + sgcLDAPEscapeDNValue(vCommonName) + ',OU=Users,DC=contoso,DC=com';
```

7 · Error handling

A result returned by the server is stored in **LastResultCode**, **LastErrorMessage** (the diagnostic message of the server, or the RFC 4511 name of the code when the server sent none) and **LastMatchedDN**. `sgcLDAPResultCodeToText` converts a code to its name, for example 49 to `invalidCredentials`.

- **Bind** and **Authenticate** return False when the credentials are refused.
- Other server errors, protocol errors, timeouts and TLS failures raise **EsgcLDAPError**, and the **OnError** event is fired with the message. A server side error keeps the connection open, a transport or protocol error closes it.
- An invalid filter raises **EsgcLDAPFilterError**, whose **Position** property is the position of the error in the filter, before anything is sent.

Active Directory returns code 49 for every refused bind and explains the reason in the diagnostic message with a `data` value, for example `80090308: LdapErr: DSID-0C09044E, comment:`

`AcceptSecurityContext error, data 52e, v4563`. The component returns that text unchanged in **LastErrorMessage**, the most common values are:

- `525`: user not found.
- `52e`: invalid credentials (wrong password).
- `530`: logon not permitted at this time. `531`: logon not permitted at this workstation.
- `532`: password expired. `773`: the user must reset the password.
- `533`: account disabled. `701`: account expired. `775`: account locked out.

Do not show these details to the user who is signing in, a generic "invalid user name or password" message does not reveal which accounts exist. Log them for the administrators.

Delphi (VCL / FireMonkey)

```
if not oLDAP.Authenticate(vUser, vPassword, vUserDN) then
begin
  if (oLDAP.LastResultCode = 49) and (Pos('data 775', oLDAP.LastErrorMessage) > 0) then
    Log('Account locked: ' + vUser)
  else
    Log(Format('LDAP login failed (%d %s): %s', [oLDAP.LastResultCode,
      sgcLDAPResultCodeToText(oLDAP.LastResultCode), oLDAP.LastErrorMessage]));
  ShowMessage('Invalid user name or password');
end;
```

Sources used to build this document

Every external claim links back to a primary source. The online help references are the canonical pages the company maintains for this component.

Primary standard / spec: RFC 4511	datatracker.ietf.org/doc/html/rfc4511
Primary standard / spec: RFC 4513	datatracker.ietf.org/doc/html/rfc4513
Primary standard / spec: RFC 2696	datatracker.ietf.org/doc/html/rfc2696
Primary standard / spec: RFC 4532	datatracker.ietf.org/doc/html/rfc4532
Online help: component page	www.esegece.com/help/sgcWebSockets/ (TsgcLDAPClient)
Delphi demo project (in the sgcWebSockets package)	`Demos\26.Authentication\05.LDAP_ActiveDirectory`
Component page	www.esegece.com/products/websockets/http/ldap-client/
sgcAuth pack	www.esegece.com/products/sgcauth/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly documented surface of the LDAP Client component shipped with sgcWebSockets Enterprise and the sgcAuth pack. For the full property, method and event reference consult the online help linked above.