SMASER

TRIP LDAP Authentication

White Paper

End User License Agreement

All rights to this software, its documentation and logotypes of the TRIP product family and software (altogether “Software”) supplied by Smaser AG (Smaser) are exclusively owned by Smaser.

The transfer of this Software, solutions or parts thereof requires the prior written agreement of Smaser. Furthermore, the customer has the right to use licensed Software and / or process solutions supplied by Smaser to the extent specified in his contract with Smaser.

The free-to-use non-commercial version doesn’t require a prior written agreement with Smaser but such customers, organizations and/or third parties agree by using the software and / or solution of Smaser to be strongly obliged to keep all rights to this software, documentation and logotypes of the TRIP product family absolutely uninfringed and protected.

Table of Contents

Introduction

About this Document

This document describes how to use a directory service via the Lightweight Directory Access Protocol (LDAP) to authenticate TRIP user logins.

The intended audience that this document targets are systems administrators, TRIP database administrators, and any person involved in the technical aspects of user management in TRIP and applications based on it.

LDAP Prerequisites

In contrast to previous TRIP versions, version 8 does not bundle LDAP libraries, but instead relies on the ones available on the system. If LDAP is not used, no LDAP libraries need to be installed.

Prerequisites for LDAP on Linux/UNIX

On Linux/UNIX platforms TRIP's LDAP integration use OpenLDAP. The OpenLDAP libraries must therefore be installed on the system before LDAP use can be configured. The package to install is typically named "openldap" (RHEL, CentOS and others). You may also want to install the "openldap-clients" package that provides tools that can be used from the command line, e.g. for diagnostic purposes.

The packages you need for full LDAP functionality in TRIP are (for RHEL/CentOS):

- openldap
- openldap-devel
- openldap-compat
- openldap-clients
- openssl-perl

Prerequisites for LDAP on Windows

On Windows, the LDAP functionality built-in into Windows is used. No special prerequisites for LDAP use on Windows therefore exist.

Standard TRIP Authentication Mechanism

Users and Groups

A TRIP installation stores data in collections referred to as databases. To gain access to a database, TRIP requires the authentication of a valid user with granted access to the database. The profiles for such users are stored in a central place in the TRIP installation and can be used with all local databases.

TRIP users can be members of several groups. A group is used as a means to structure users per privilege level, and is a more convenient way to grant database access when there are many user profiles on the system. Access granted to a group will automatically apply to the users who are members of that group. A user will get the combined access privileges from all groups the user is member of.

There is a default group called PUBLIC, which all users are members of. If database access is granted to PUBLIC, all users who log in to TRIP will have access to that database.

Granular Database Access

In the simplest form, read (and possibly write) access is given to a user or group for the entire database. If read access is given in this manner data in all fields in all records of the database is available. Similarly, if write access is given in this manner, all fields in all records in the database can be modified.

Database access can also be granted on field level. Such access privileges specify per field if the user or group will have read or write access. A field for which access is not specified either in this manner or indirectly for the database as a whole will not be accessible by the user.

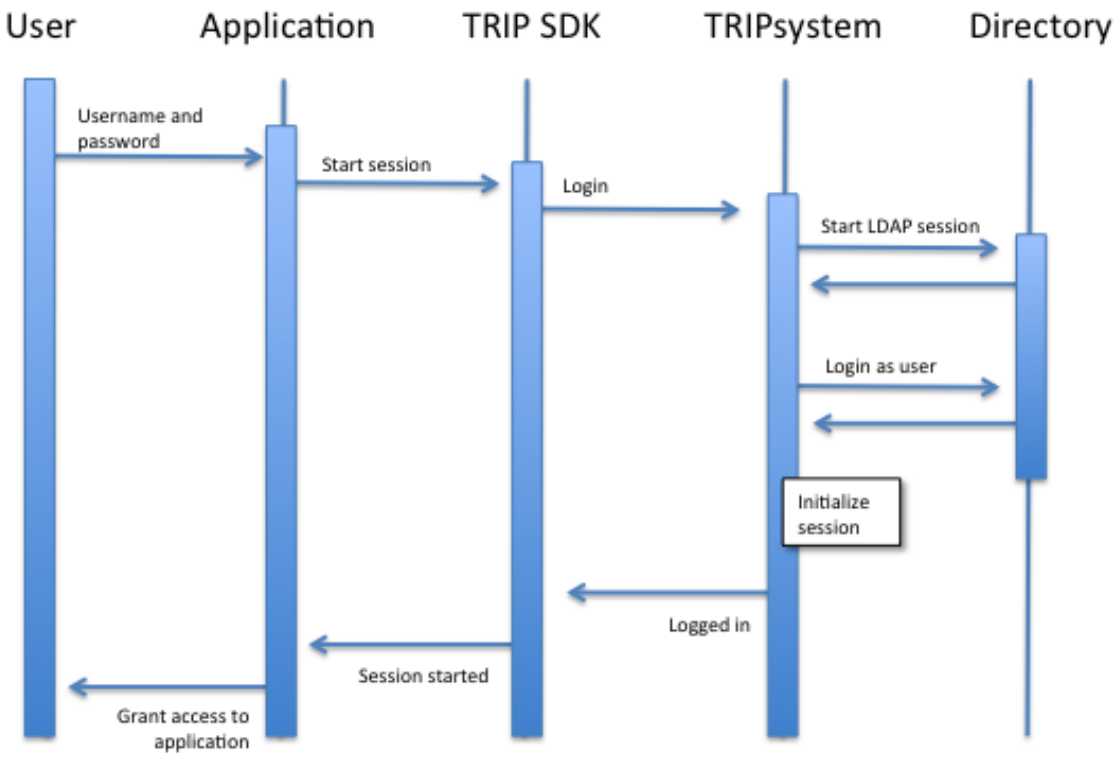
Access rights can also be granted on a record level, which requires the use of a so-called read or write “scope”. Such a scope is a search query that defines the set of records that will be read or write accessible. Records that are not included by the scope query will not be accessible by the user.

LDAP Authentication Flow

General Workflow

LDAP authentication of users can be done for all users except for SYSTEM, which always uses TRIP’s internal authentication mechanism.

The following sequence diagram shows a successful login via LDAP.



When a user logs in to TRIP via an application, he or she provides the name of a user and that user’s password. When LDAP authentication is enabled, TRIPsystem will open an

LDAP session to the directory service and authenticate the user.

Note that the LDAP integration is internal to TRIPsystem. No other parts of TRIP have any dependencies on LDAP, nor are they aware of LDAP even if it is used. LDAP use is thus totally transparent to the SDK and applications using the SDK.

All applications use one of the SDK components to connect to TRIPsystem. These include the TRIPtoolkit (server-side C API), TRIPjtk and TRIPjxp (Java APIs), TRIPclient (Windows-only C and COM API), and TRIPnxp (.NET API).

The above diagram also applies to TRIPclassic and TRIPmanager, that are considered applications in this context.

Successful LDAP Login

If the LDAP authentication succeeds, as illustrated in the above sequence diagram, the outcome depends on if the user is also registered in TRIP or not, and if guest login is allowed.

If the user is registered in TRIP, the session will be run with that TRIP user's credentials. The password stored in TRIP for this user will not be used, since the password was already verified by the directory service.

If the user is not registered in TRIP, a guest session will be initialized by using the BUILTIN_GUEST user profile. If guest login is disabled (by setting the logical name TDBS_DISALLOW_GUEST to True), the login will instead fail and a TRIP session will not be started.

Failed LDAP Login

If the directory service is not available, or the user is not registered in the directory service or if the password does not match, the login attempt will fail and a TRIP session will not be started.

Using both LDAP and TRIP user profiles

If LDAP authentication is performed, all users except SYSTEM must be authenticated via the configured LDAP directory to be allowed to log in to TRIP. To support a combination of user profiles, some in LDAP and some only in TRIP, set the logical name TDBS_LDAP_FALLBACK to True in the privileged section of tdbs.conf:

```
TDBS_LDAP_FALLBACK=True
```

This causes TRIP to attempt to authenticate the user with its own user database if the LDAP directory reports that the user is not known.

Encrypted Sessions on Linux/UNIX

Enabling Encryption

By default, TRIP connects to the directory server using a simple, unencrypted LDAP connection. If there are concerns about the password security of such connections, an encrypted TLS or SSL connection can be used instead.

TLS is a standard cryptographic protocol used extensively on the Internet to secure communications for banking, commercial transactions, etc.

To enable TRIP to use a TLS connection for LDAP, the following configuration entry is required in the [Privileged] section of the TRIPsystem configuration file tdbs.conf:

```
TDBS_LDAP_MECHANISM=TLS
```

The TLS encryption protocol uses the same port as the unencrypted one (typically 389), but will convert the connection to an encrypted one as part of the encryption handshake procedure.

If the directory server does not expose an unencrypted port (e.g. for security reasons), the SSL cryptographic protocol can be used instead.

```
TDBS_LDAP_MECHANISM=SSL
```

This will direct TRIP to use an SSL encrypted communications channel, which defaults to port 636.

Certificates

Certificates are small data files that digitally bind a cryptographic key to an organization's details. A certificate can be likened to a driver's license or an identity card, and in the same fashion, a certificate identifies both the holder and the issuer. An identity card can be trusted if the issuer is known, and a certificate is trusted in a similar manner. The data that describes the issuer of a certificate is also a certificate. This issuing certificate must be made known to TRIP in order for it to validate the encrypted connection.

Validation requires that the issuing (certificate authority) certificate is known.

If a specific CA certificate is known to be used, this can be set using the tdbs.conf privileged section configuration entry `TDBS_LDAP_CACERT`. This takes as value the fully qualified path to the CA certificate to use to validate the LDAP server's certificate.

If several CA certificates are known to be used, the tdbs.conf privileged section configuration entry `TDBS_LDAP_CERT_DIR` can be used. Its value must point to a directory containing CA certificates in Base64-encoded PEM format. The files each contain one CA certificate.

The certificate files in the `TDBS_LDAP_CERT_DIR` directory are looked up by their CA subject name hash values. These hash values must be symbolic links referring to the matching certificates. Use the `c_rehash` tool as provided with the `openssl-perl` package (RHEL/CentOS) to create links with the appropriate hash value for each certificate.

To skip validation and trust any server certificate, set the following in the privileged section of the tdbs.conf file:

```
TDBS_LDAP_CERT_TRUSTALL=Y
```

NOTE: Previous TRIP Linux/UNIX versions used the Mozilla LDAP SDK to connect to directory services. That integration required TDBS_LDAP_SSL_CERT_DB to be set in tdbs.conf. That configuration setting is no longer used and can be removed.

Refer to OpenLDAP documentation for more details on certificate file formatting and naming.

Encrypted Sessions on Windows

Enabling Encryption

By default, TRIP connects to the directory server using a simple, unencrypted LDAP connection. If there are concerns about the password security of such connections, an encrypted TLS or SSL connection can be used instead.

TLS is a standard cryptographic protocol used extensively on the Internet to secure communications for banking, commercial transactions, etc.

To enable TRIP to use a TLS connection for LDAP, the following configuration entry is required in the [Privileged] section of the TRIPsystem configuration file tdbs.conf:

```
TDBS_LDAP_MECHANISM=TLS
```

The TLS encryption protocol uses the same port as the unencrypted one (typically 389), but will convert the connection to an encrypted one as part of the encryption handshake procedure.

If the directory server does not expose an unencrypted port (e.g. for security reasons), the SSL cryptographic protocol can be used instead.

```
TDBS_LDAP_MECHANISM=SSL
```

This will direct TRIP to use an SSL encrypted communications channel, which defaults to port 636.

Certificates and the Trusted Root Certificate Store

On Windows, the native LDAP API is used to connect to directory services. The certificate database used for validating server certificates is the Windows certificate store of the local machine. In order for an encrypted connection to work, the issuer of the certificate in use by the LDAP server must be found in the Trusted Root Certification Authorities store.

To begin with, you must configure certificate path validation. The steps involved in doing this are:

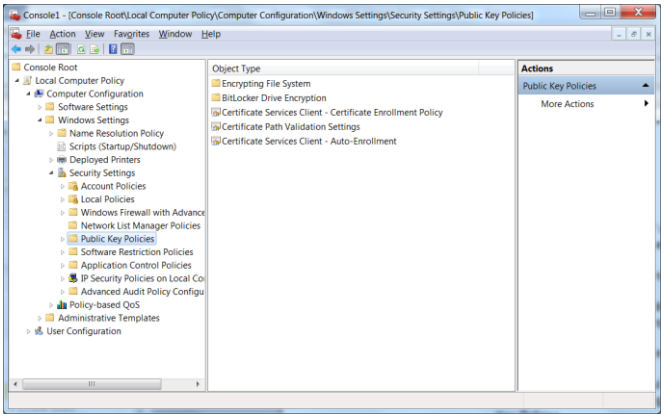
1. Logged on as a user with Administrator privileges, start the mmc.exe program (e.g. under Windows 7 or Server 2008 R2, click the Start button, and type **mmc** in the search box, then press ENTER).

2. Select **Add/Remove Snap-in** from the **File** menu.

3. Select **Group Policy Object Editor** from the **Available snap-ins** list, click **Add**, and select the **Local Computer**, and then click Finish.

4. Click OK to close the dialog window.

5. In the console tree, go to Local Computer Policy, Computer Configuration, Windows Settings, Security Settings, and then click Public Key Policies.

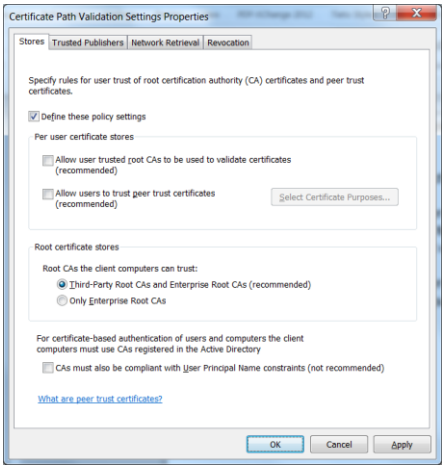


6. Double-click **Certificate Path Validation Settings**, and then click the **Stores** tab.

7. Select the **Define these policy settings** check box.

8. Under **Per user certificate stores**, clear the **Allow user trusted root CAs to be used to validate certificates** and **Allow users to trust peer trust certificates** check boxes.

9. Under **Root certificate stores**, select the root CAs that the client computers can trust, and then click OK to apply the new settings.



With certificate path validation configured, you can now add certificates to the trusted root certification authorities store. The steps involved are:

1. Logged on as a user with Administrator privileges, start the mmc.exe program (e.g. under Windows 7 or Server 2008 R2, click the Start button, and type **mmc** in the search box, then press ENTER).

2. Select **Add/Remove Snap-in** from the **File** menu.

3. Select **Certificates** from the **Available snap-ins** list, click **Add**, and select the **Local Computer**, and then click Finish.

4. Click OK to close the dialog window.

5. Under **This snap-in will always manage certificates for**, click **Computer account**, and then click Next.

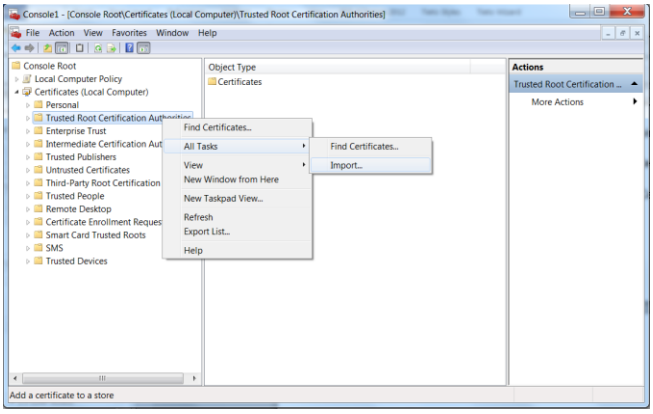
6. Click **Local computer**, and click Finish.

7. Click OK to close the dialog window.

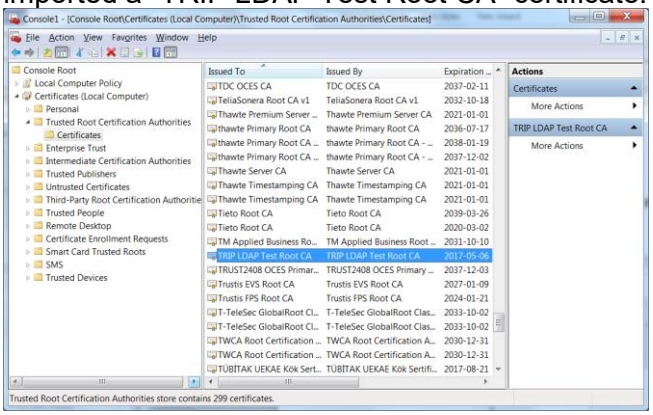
8. In the console tree, expand **Certificates**.

9. Right-click the Trusted Root Certification Authorities store.

10. Click Import to import the certificates and follow the steps in the Certificate Import Wizard.



11. If the import was successful, you should now be able to see your certificate under the Certificates list. In this example, we have imported a “TRIP LDAP Test Root CA” certificate:



For more information on the use and management of root certificates under Windows, please refer to the Microsoft online documentation and knowledge base articles.

Verify Connection

If you first wish to verify that you can reach the LDAP server via a TLS or SSL encrypted connection before you start using it with TRIP, there are a number of third-party LDAP browser clients available. The Softerra LDAP browser is an example of such an application.

Regardless of which client you choose to test the connection with, note that it will most likely not use the same TLS or SSL configuration as described in the previous sections. Despite this, verifying the connection using a third-party tool is still useful and recommended; particularly if you have trouble connecting to the LDAP server from TRIP.

Even if no separate tool for connection verification is available, you can still test the connection from by logging in via TRIPclassic or TRIPmanager. To analyze any problems that may occur, you can temporarily enable LDAP logging. In the privileged section of tdbs.conf, set:

```
TDBS_LDAP_ERRORLOG=TRUE
```

To emit LDAP errors to the `ldap_errors.log` file in the directory indicated by the TDBS_LOG value in tdbs.conf. Alternatively, set:

```
TDBS_LDAP_ERRORLOG=DEBUG
```

To emit additional information that may be valuable for further problem analysis.

LDAP Access

User Account for Browsing

In order for TRIP to be able to query the directory service and perform user authentication against it, an LDAP session must first be started. This means logging in to the directory service using a user identity, which should be reserved for this purpose. This user is registered in thedirectory like any other user. In order for TRIP to be able to use it, it must also be mentioned in the TRIPsystem configuration file tdbs.conf:

```
TDBS_LDAP_USERNAME=cn=Manager,dc=bjensen,dc=com

TDBS_LDAP_PASSWORD=thx1139
```

The username specified for TDBS_LDAP_USERNAME must be a fully qualified Distinguished Name (DN) of a user that has read access to the entire LDAP tree descending from the root node provided by TDBS_LDAP_BASE (described in section 0).

This also means that the file access rights to the tdbs.conf file must be set so that no unprivileged users have read access to it; only the users who actually run TRIP processes on the server (which also includes the daemons/services and the tbserver).

Anonymous Access

If the directory service allows anonymous access, the TDBS_LDAP_USERNAME and TDBS_LDAP_PASSWORD logical names do not have to be used. Instead specify:

```
TDBS_LDAP_ANONYMOUS=True
```

Note that anonymous access does not infer unencrypted access; it only refers to how LDAP clients gain access with the directory server – with or without user credentials. Anonymous access is disabled by default.

LDAP Lookup

When authenticating a user, the user's identity will normally be provided as a Relative Distinguished Name (RDN), rather than as a fully qualified Distinguished Name (DN). In order to turn an RDN to a DN, the TRIPsystem tdbs.conf file must provide values for the logical names TDBS_LDAP_BASE (defining the base of the tree in which users can be

found) and TDBS_LDAP_SEARCH (defining the LDAP search string to use to find users).

For example, if the TRIP user community is collected in a subtree of the LDAP repository with a logical base of ou=tox/o=pharma/c=us, then the base of the search tree should be established as follows:

```
TDBS_LDAP_BASE=ou=tox,o=pharma,c=us
```

To find a user by RDN (for example by the UID or CN that the user presents as their typical login public key), specify an LDAP search string using the %u% substitution string to stand for the user's provided RDN. For example, when using the person structural schema (or some derivation, for example organizationalPerson, or inetOrgPerson) with the uidObject add-on schema the search string would be:

```
TDBS_LDAP_SEARCH=(&(objectclass=person)(uid=%u%))
```

Any occurrence of the "%u%" pattern within the string will be replaced with whatever "username" is provided to TRIP during the login process.

Note that with Active Directory servers, the user identity attribute is sAMAccountName. This means that if you are using an AD server, you should replace the `(uid=%u%)` part of the expression with `(sAMAccountName=%u%)`.

LDAP User Authentication

Once the user has been found (i.e. their RDN has been dereferenced to a DN) and successfully matched in a directory service record, TRIP will use the provided password to authenticate the user with the directory.

If authentication is successful, the LDAP record for the user will be turned into a TRIP username for use within the CONTROL database. The following variable is used to specify the field from the user record that will provide this mapping, for example in most user-related schemas, this would be the "uid" field:

```
TDBS_LDAP_MATCH=uid
```

Or, for Active Directory servers the “sAMAccountName” field normally contains the user name:

```
TDBS_LDAP_MATCH=sAMAccountName
```

A different field can also be used, also one that is different from what is specified in the TDBS_LDAP_SEARCH expression. This means that it is possible to associate a TRIP user account that is different from the LDAP user account used for authentication. The different TRIP user must be stored in a field (identified by TDBS_LDAP_MATCH) in the same directory service record. The above example, however, uses the same field (uid or sAMAccountName) as is the user’s directory service identity.

How to Authenticate with LDAP in TRIP

LDAP Authentication with TRIP Users

In order to make full use of the database access control mechanisms, the LDAP user will be associated with an existing TRIP user profile. These users normally have the same id in both TRIP and LDAP, but it is possible to set it up to have an LDAP user associated with a differently named TRIP user.

Because TRIP user profiles are still used even though they are authenticated via LDAP, it

means that for a user to get access to TRIP it is not sufficient to have a directory service record, the user must also have a TRIP user profile. When adding a user that must be able to access TRIP, the administrator must effectively register the user twice; once in the directory and once in TRIP. Note that although a password must be supplied also for the TRIP user profile, it isn't used. To avoid this dual administration, automatic LDAP user import can be used. See below for details.

The use of LDAP stops at the authentication of a user. This means that management of TRIP groups and TRIP database access is done using TRIP's own management tools and/or APIs. Such information remains kept internally; none of this uses directory services.

Even though TRIP management tools still have to be used, using LDAP for user authentication brings one major benefit: possibility to enforce password policies, something which is not available in TRIP.

Guest Authentication

If database access requirements for the TRIP applications used are very simple, then using guest authentication might be an option. This means that a single user (BUILTIN_GUEST) will be used for all users authenticated via LDAP but who have no TRIP user profile.

Guest authentication can also be used if the users of the application can be divided into an unprivileged and a privileged group. The privileged users will have their own TRIP user profiles and can get more permissive access rights. The unprivileged users will have no TRIP user profile of their own and will all log in as their directory service user, but be associated with the single TRIP user profile BUILTIN_GUEST, who would in this scenario be given restricted read access privileges only.

Even when guest authentication is enabled, as it is per default, TRIP will always try to associate an existing TRIP user profile with the authenticated LDAP user. This means that it is possible to use regular TRIP user profiles as well as the guest user.

There might be scenarios where it is undesirable to allow the guest user (e.g. for security reasons). To disable guest login, set the following logical name in tdbs.conf:

```
TDBS_DISALLOW_GUEST=True
```

Automatic LDAP user import

If dual administration is too much of a hassle, but guest login imposes too many restrictions (e.g. not possible to have different access rights for different users, user-specific procedures, etc), the third option is to configure TRIP to automatically create a TRIP user profile for a user that has successfully authenticated via LDAP.

This feature is disabled by default. To enable it, set the following logical name in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER=True
```

If the user logging in does not already exist, this setting will cause TRIP to create the user, assigning it the user name identified by the resolving the TDBS_LDAP_MATCH value. This is normally the same as the user ID that is being authenticated with LDAP.

The owner of the user will be assigned the value specified by the logical name TDBS_LDAP_AUTOUSER_UM, or to "SYSTEM" if the name is not specified. The user specified by this symbol must have user manager (UM) privileges. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_UM=TRIPDBA
```

To assign the user to a group in addition to PUBLIC, assign the name of a group owned by the specified user manager to the TDBS_LDAP_AUTOUSER_GROUP logical name. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_GROUP=LDAPUSERS
```

The new TRIP user profile will be assigned a random password, effectively requiring successful authentication via LDAP in order to be able to login to TRIP. To set the TRIP user password to the same one as used to authenticate the user with the LDAP directory, specify the following logical name in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_PWSYNC=True
```

In addition to setting this password at the first login to the same as used in the directory, this logical name will also cause the password for the TRIP profile to be reset to the LDAP user's password during login if it differs. A change in password for the user in the directory will, if this logical name is set, be automatically synchronized to TRIP.

Note that TDBS_LDAP_AUTOUSER_PWSYNC is less effective for subsequent password synchronization when login tickets are in use. This is because when ticket-based authentication is used, password checks are required less often. The password sync will in that case not be performed until the ticket has expired and the user needs to log in explicitly again.

A login procedure (a.k.a. start procedure) can also be assigned to the user. Such a procedure executes automatically every time the user successfully logs in. This is done by specifying the TDBS_LDAP_AUTOUSER_LOGINPROC logical name. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_LOGINPROC=PUBLIC.STARTUP
```

Note that the specified procedure does not have to exist at the time when it is defined for a user. This follows the same semantics as the specification of a start procedure in the TRIP user profile

Special Considerations

Difference in LDAP User ID and TRIP User Names

As described earlier in this document, to get full benefit from the database access control mechanisms in TRIP, a dedicated TRIP user profile for the user is needed in addition to a directory service user.

For any TRIP installation that uses LDAP authentication, the LDAP user ids must match the TRIP user ids, or alternatively the users must have their TRIP user ids registered in their LDAP record.

When migrating an existing TRIP installation to LDAP authentication, the LDAP user id for a person may be different from his/her TRIP user id. In this case, there are two options:

1. Rename TRIP users
2. Assign the TRIP user id to an extra field in the LDAP record for the user.

Unfortunately, there currently is no rename function for TRIP users. This means that in order to rename a TRIP user, these steps must be followed:

1. Create a new TRIP user with an identity that matches the LDAP user.
2. Assign group memberships to the new user to match those of the old user.
3. Assign database access rights to the new user to match those of the old user.
4. If the applications keep additional information about their TRIP users in a database or other location, update (rename) as needed.
5. Delete the old TRIP user. This should be postponed until the TRIP installation is fully configured for LDAP and can go live. This requires, however, that the TRIP license has a high enough registered users limit to allow for this.

As an alternative to the above procedure, it is possible to keep the old TRIP user ids and instead assign the id of a TRIP user to the LDAP record for the user. This eliminates the need for renaming TRIP users, but adds to the responsibilities of those who administer the directory service.

For additional case-specific guidance and recommendations regarding migrating users to LDAP, contact your TRIP distributor. Note that such guidance may not be covered by the regular TRIP support and maintenance contract.

LDAP Timeout

To deal with network congestion or other issues that slows down the LDAP server response time, set the TDBS_LDAP_TIMEOUT logical symbol to the maximum number of milliseconds that TRIP should wait for a response from the LDAP server when searching for the user to authenticate.

For example:

```
TDBS_LDAP_TIMEOUT=3000
```

This sets the timeout to 3 seconds. The default is 10 seconds (i.e. 10000 milliseconds).

The SYSTEM User

The SYSTEM user is not handled by LDAP. It always uses TRIP's internal authentication mechanism.

Max Length of Username and Password

The max lengths of the username and password values are 31 ASCII characters each. This is partially because the need to match the authenticated LDAP user with a TRIP user, and partially because of the same APIs and structures are used toward the applications regardless of whether LDAP authentication is used or not.

If the LDAP user ID or password exceeds 31 characters, login to TRIP is not possible.

Securing tdbs.conf

If TRIP uses a non-anonymous way to access and search the LDAP directory, the name and password for the LDAP browse user will be in plain text in the tdbs.conf file. This means that this file must be secured so that only the users under which the TRIP processes run have read access. The processes are typically associated with the files in the TRIPsystem 'bin' directory (e.g. tbserver, tripd, etc), but may also be server-side applications and tools (e.g. TRIPhighway).

Use Cases for Migrating to LDAP

While these use cases are written with the scenario of migrating an existing TRIP installation to LDAP authentication, the descriptions may also be useful for entirely new TRIP installations.

Encrypted LDAP use with TRIP Users

This scenario has the following characteristics:

- No guest access to TRIP allowed
- No anonymous LDAP access allowed
- A user has the same ID in LDAP and in TRIP
- Encrypted LDAP connection

Benefits and Disadvantages

Benefits

- **Password policies**. By using LDAP authentication, it is possible to enforce password policies, something that is not possible when using the internal TRIP authentication mechanism.
- **More secure**: By using encrypted sessions between TRIP and the directory server in combination with disabled guest user access, a higher level of security enforced when compared to allowing guest user access or connecting in an unencrypted fashion.
- **Database access control**: Database access rights can be assigned to TRIP users and groups in a normal TRIP fashion, which enables full use of TRIP's database access control mechanisms.

Disadvantages

- **Encryption overhead**. Although small, there is an added overhead incurred by connecting to the LDAP server using encryption. If new sessions are established as a result of manual login, this overhead is likely negligible. If sessions are created and destroyed with high frequency, the overhead may be noticeable over time.

- **Parallel user accounts**. The systems administrator has to ensure that all users who need access to TRIP also have a TRIP user account. Using LDAP authentication doesn't mean that TRIP user management can be eliminated; quite the contrary.

Steps

Steps required for the implementation of this scenario:

1. Examine user IDs
    - For each user with a TRIP login, check if the TRIP username is equal to the directory server user ID. Note the IDs that differ.
2. Rename users

For each user with different IDs, rename the users as described in this document under "Automatic LDAP user import

If dual administration is too much of a hassle, but guest login imposes too many restrictions (e.g. not possible to have different access rights for different users, user-specific procedures, etc), the third option is to configure TRIP to automatically create a TRIP user profile for a user that has successfully authenticated via LDAP.

This feature is disabled by default. To enable it, set the following logical name in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER=True
```

If the user logging in does not already exist, this setting will cause TRIP to create the user, assigning it the user name identified by the resolving the TDBS_LDAP_MATCH value. This is normally the same as the user ID that is being authenticated with LDAP.

The owner of the user will be assigned the value specified by the logical name TDBS_LDAP_AUTOUSER_UM, or to "SYSTEM" if the name is not specified. The user specified by this symbol must have user manager (UM) privileges. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_UM=TRIPDBA
```

To assign the user to a group in addition to PUBLIC, assign the name of a group owned by the specified user manager to the TDBS_LDAP_AUTOUSER_GROUP logical name. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_GROUP=LDAPUSERS
```

The new TRIP user profile will be assigned a random password, effectively requiring successful authentication via LDAP in order to be able to login to TRIP. To set the TRIP user password to the same one as used to authenticate the user with the LDAP directory, specify the following logical name in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_PWSYNC=True
```

In addition to setting this password at the first login to the same as used in the directory, this logical name will also cause the password for the TRIP profile to be reset to the LDAP user's password during login if it differs. A change in password for the user in the directory will, if this logical name is set, be automatically synchronized to TRIP.

Note that TDBS_LDAP_AUTOUSER_PWSYNC is less effective for subsequent password synchronization when login tickets are in use. This is because when ticket-based authentication is used, password checks are required less often. The password sync will in that case not be performed until the ticket has expired and the user needs to log in explicitly again.

A login procedure (a.k.a. start procedure) can also be assigned to the user. Such a procedure executes automatically every time the user successfully logs in. This is done by specifying the TDBS_LDAP_AUTOUSER_LOGINPROC logical name. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_LOGINPROC=PUBLIC.STARTUP
```

Note that the specified procedure does not have to exist at the time when it is defined for a user. This follows the same semantics as the specification of a start procedure in the TRIP user profile

- Special Considerations".
- Note that the TRIP license need room for the additional registered TRIP users in order to make it possible to safely establish their new identities.

3. Create a LDAP browse user
   - Create a user in the directory service that TRIPsystem will use to search for the record of the user to authenticate. This user needs read access to the tree in which users are stored.
   - Make the user and its password known to TRIPsystem by editing the tdbs.conf file (see section 0 in this document).
   - Modify the file access settings in the file system for the tdbs.conf file to ensure that only privileged users have access to it and that the password, which is saved here in plain text, is not exposed.
4. Configure Encryption
   - Make sure that the issuer for the certificate used by the LDAP server is known to TRIPsystem. Refer to the sections on encrypted sessions in this document for details.
5. Disable guest and anonymous use by entering the following in tdbs.conf under the Privileged section:
   - TDBS_DISALLOW_GUEST=True
   - TDBS_LDAP_ANONYMOUS=False
6. Determine how to search for LDAP users.
   - Determine the values of the logical names TDBS_LDAP_BASE and TDBS_LDAP_SEARCH in the tdbs.conf file. See section 0 and 0 in this document for details.

7. When ready to go live with LDAP authentication on this server, edit tdbs.conf:

   - TDBS_AUTH_PROVIDER=LDAP

Important Aspects

You cannot go live with LDAP authentication until all users have been transferred to LDAP. If you do, users unknown to the directory server will not be able to log in.

Before migrating your production TRIP server to LDAP, you should test migrate using a test or backup server with a copy of your production system. If no database designs, groups, clusters, access rights or other control objects meanwhile are modified in the production server, it is possible to just copy the CONTROL database files and LDAP-specific tdbs.conf changes back to the production server in order to enable LDAP there without having to do the same job again.

SSL certificates have an expiration date. This has little or no impact on TRIP unless the certificate used by the LDAP server has expired, in which case no LDAP login is possible. Only SYSTEM can login in such a case.

The issuing certificate can also expire. This is the certificate that must be known to TRIP as described in section **Fel! Hittar inte referenskälla.**. If a standalone certificate database is used such that it will not be automatically updated when updating the operating system, or if the issuing certificate is a company internal one, the systems administrator must ensure to add renewed issuing certificates to the certificate database as they become available. Failure to do this may make it impossible for TRIP to establish encrypted LDAP connections, which in turn will make it impossible to log in to TRIP for other users than SYSTEM.

Configuration Example

Below is an example tdbs.conf configuration for this scenario.

```
TDBS_AUTH_PROVIDER=LDAP

TDBS_LDAP_SERVER=ad.example.com

TDBS_LDAP_MECHANISM=TLS

TDBS_LDAP_ANONYMOUS=False

TDBS_LDAP_USERNAME=cn=Manager,dc=bjensen,dc=com

TDBS_LDAP_PASSWORD=thx1139

TDBS_LDAP_BASE=ou=tox,o=pharma,c=us

TDBS_LDAP_SEARCH=(&(objectclass=person)(uid=%u%))

TDBS_DISALLOW_GUEST=True

TDBS_LDAP_MATCH=uid
```

For a UNIX/Linux TRIPsystem installation, also specify the CA certificate or CA certificate directory using either `TDBS_LDAP_CERT_DIR` or `TDBS_LDAP_CACERT` in the privileged section of the tdbs.conf file (see section “Encrypted Sessions on Linux/UNIX” for details)

On Windows, remember to ensure that the trusted root certificate authorities store is up to date (see section “Encrypted Sessions on Windows” for details).

Users with Different IDs in LDAP and in TRIP

This scenario has the following characteristics:

- No guest access to TRIP allowed
- No anonymous LDAP access allowed
- A user is likely to have an ID different in LDAP and in TRIP
- Encrypted LDAP connection

This scenario is almost identical to the scenario described in section 0, except for that the user IDs in TRIP and in the directory server are different for a large number of users.

Benefits and Disadvantages

Benefits, in addition to the ones specified in under the “Encrypted LDAP use with TRIP Users” use case:

- **No user rename needed**: If there are many TRIP users with differing IDs, and the users have group memberships and/or assigned database access rights, the renaming process may be lengthy. By configuring the directory with the TRIP user names instead, the renaming process can be skipped, saving some time.

Disadvantages, in addition to the ones specified in under the “Encrypted LDAP use with TRIP Users” use case:

- **Extra user info in the directory**. The user will effectively have to have two ID fields in the directory. One regular (e.g. the one that the user logs on to the Windows domain with) and one TRIP-specific that TRIP uses after successful authentication.

Steps

Steps required for the implementation of this scenario:

1. Update directory with TRIP user names.
    - For each user with a TRIP login, add the TRIP user name to the user’s directory record under a field created for this purpose.
2. Create a LDAP browse user
    - Create a user in the directory service that TRIPsystem will use to search for the record of the user to authenticate. This user needs read access to the tree in which users are stored.
    - Make the user and its password known to TRIPsystem by editing the tdbs.conf file (see section “User Account for Browsing” in this document).

- Modify the file access settings in the file system for the tdbs.conf file to ensure that only privileged users have access to it and that the password, which is saved here in plain text, is not exposed.

3. Configure Encryption

   - Make sure that the issuer for the certificate used by the LDAP server is known to TRIPsystem. See section 0 in this document for details.

4. Disable guest and anonymous use by entering the following in tdbs.conf under the Privileged section:

   - TDBS_DISALLOW_GUEST=True
   - TDBS_LDAP_ANONYMOUS=False

5. Determine how to search for LDAP users.

   - Determine the values of the logical names TDBS_LDAP_BASE and TDBS_LDAP_SEARCH in the tdbs.conf file. See section 0 and 0 in this document for details.
   - Set the TDBS_LDAP_MATCH variable to the name of the field holding the TRIP user id (see step 1).

6. When ready to go live with LDAP authentication on this server, edit tdbs.conf:

   - TDBS_AUTH_PROVIDER=LDAP

Important Aspects

See section “Important Aspects“ under “Encrypted LDAP use with TRIP Users”.

Configuration Example

Below is an example tdbs.conf configuration for this scenario.

```
TDBS_AUTH_PROVIDER=LDAP
TDBS_LDAP_SERVER=ad.example.com
TDBS_LDAP_MECHANISM=TLS
TDBS_LDAP_ANONYMOUS=False
TDBS_LDAP_USERNAME=cn=Manager,dc=bjensen,dc=com
TDBS_LDAP_PASSWORD=thx1139
TDBS_LDAP_BASE=ou=tox,o=pharma,c=us
TDBS_LDAP_SEARCH=(&(objectclass=person)(uid=%u%))
TDBS_DISALLOW_GUEST=True
TDBS_LDAP_MATCH=tripuid
```

For a UNIX/Linux TRIPsystem installation, also specify the CA certificate or CA certificate directory using either `TDBS_LDAP_CERT_DIR` or `TDBS_LDAP_CACERT` in the privileged section of the tdbs.conf file (see section “Encrypted Sessions on Linux/UNIX” for details)

On Windows, remember to ensure that the trusted root certificate authorities store is up to date (see section “Encrypted Sessions on Windows” for details).

Guest Users Allowed

This scenario has the following characteristics:

- Guest access to TRIP is allowed
- No anonymous LDAP access allowed
- Most users do not have a TRIP user profile
- A small set of privileged users have TRIP user profiles
- Encrypted LDAP connection

This use case describes a scenario that allows guest user access to TRIP, i.e. users authenticated via LDAP who have no dedicated TRIP user profile.

Benefits and Disadvantages

Benefits

- **Password policies**. By using LDAP authentication, it is possible to enforce password policies, something that is not possible when using the internal TRIP authentication mechanism.
- **More secure**: By using encrypted sessions between TRIP and the directory server in combination with disabled guest user access, a higher level of security enforced when compared to allowing guest user access or connecting in an unencrypted fashion.
- **Minimum of parallel user accounts**. Only users with special privileges in TRIP (e.g. database administrators) will need dedicated TRIP user profiles. Regular users do not.

Disadvantages

- **Encryption overhead**. Although small, there is an added overhead incurred by connecting to the LDAP server using encryption. If new sessions are established as a result of manual login, this overhead is likely negligible. If sessions are created and destroyed with high frequency, the overhead may be noticeable over time.
- **Reduced database access control**: Database access rights must be assigned to the BUILTIN_GUEST user or the PUBLIC group, but no distinction can be made between user types; all will have the same rights to the databases to which access is granted.

Steps

Steps required for the implementation of this scenario:

1. Examine user IDs
    - For each privileged TRIP user (e.g. database administrator user), check if the TRIP username is equal to the directory server user ID. Note the IDs that differ.
2. Rename users

For each privileged user with different IDs, rename the users as described in this document under “Automatic LDAP user import

If dual administration is too much of a hassle, but guest login imposes too many restrictions (e.g. not possible to have different access rights for different users, user-specific procedures, etc), the third option is to configure TRIP to automatically create a TRIP user profile for a user that has successfully authenticated via LDAP.

This feature is disabled by default. To enable it, set the following logical name in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER=True
```

If the user logging in does not already exist, this setting will cause TRIP to create the user, assigning it the user name identified by the resolving the TDBS_LDAP_MATCH value. This is normally the same as the user ID that is being authenticated with LDAP.

The owner of the user will be assigned the value specified by the logical name TDBS_LDAP_AUTOUSER_UM, or to “SYSTEM” if the name is not specified. The user specified by this symbol must have user manager (UM) privileges. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_UM=TRIPDBA
```

To assign the user to a group in addition to PUBLIC, assign the name of a group owned by the specified user manager to the TDBS_LDAP_AUTOUSER_GROUP logical name. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_GROUP=LDAPUSERS
```

The new TRIP user profile will be assigned a random password, effectively requiring successful authentication via LDAP in order to be able to login to TRIP. To set the TRIP user password to the same one as used to authenticate the user with the LDAP directory, specify the following logical name in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_PWSYNC=True
```

In addition to setting this password at the first login to the same as used in the directory, this logical name will also cause the password for the TRIP profile to be reset to the LDAP user’s password during login if it differs. A change in password for the user in the directory will, if this logical name is set, be automatically synchronized to TRIP.

Note that TDBS_LDAP_AUTOUSER_PWSYNC is less effective for subsequent password synchronization when login tickets are in use. This is because when ticket-based authentication is used, password checks are required less often. The password sync will in that case not be performed until the ticket has expired and the user needs to log in

explicitly again.

A login procedure (a.k.a. start procedure) can also be assigned to the user. Such a procedure executes automatically every time the user successfully logs in. This is done by specifying the TDBS_LDAP_AUTOUSER_LOGINPROC logical name. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_LOGINPROC=PUBLIC.STARTUP
```

Note that the specified procedure does not have to exist at the time when it is defined for a user. This follows the same semantics as the specification of a start procedure in the TRIP user profile

- Special Considerations".
- Note that the TRIP license need room for the additional registered TRIP users in order to make it possible to safely establish their new identities.

3. Create a LDAP browse user
   - Create a user in the directory service that TRIPsystem will use to search for the record of the user to authenticate. This user needs read access to the tree in which users are stored.
   - Make the user and its password known to TRIPsystem by editing the tdbs.conf file (see section "User Account for Browsing" in this document).
   - Modify the file access settings in the file system for the tdbs.conf file to ensure that only privileged users have access to it and that the password, which is saved here in plain text, is not exposed.
4. Configure Encryption
   - Make sure that the issuer for the certificate used by the LDAP server is known to TRIPsystem. See section 0 in this document for details.
5. Enable the guest user and disable anonymous LDAP use by entering the following in tdbs.conf under the Privileged section:
   - TDBS_DISALLOW_GUEST=False
   - TDBS_LDAP_ANONYMOUS=False
6. Determine how to search for LDAP users.
   - Determine the values of the logical names TDBS_LDAP_BASE and TDBS_LDAP_SEARCH in the tdbs.conf file. See section "LDAP Lookup" and "LDAP User Authentication" in this document for details.

7. When ready to go live with LDAP authentication on this server, edit tdbs.conf:

    - TDBS_AUTH_PROVIDER=LDAP

Important Aspects

By allowing users to log in as guests, you reduce the requirements on systems administration somewhat. However, this has a negative impact on how database access rights can be assigned. Because all users who log in as guest will share the same TRIP user profile (BUILTIN_GUEST), all users will by necessity share the same access rights. This is perfectly OK - as long users do not need different access to different databases and/or different fields in the databases. If users nevertheless are grouped such that different groups must have different levels of access to the data, either a different configuration should be considered, or an application-controlled mechanism must be enforced. Note that by restricting access using application logic is less secure from the database point of view.

For other important aspects that also relate to this use case, see section 0.

Configuration Example

Below is an example tdbs.conf configuration for this scenario.

```
TDBS_AUTH_PROVIDER=LDAP

TDBS_LDAP_SERVER=ad.example.com

TDBS_LDAP_MECHANISM=TLS

TDBS_LDAP_ANONYMOUS=False

TDBS_LDAP_USERNAME=cn=Manager,dc=bjensen,dc=com

TDBS_LDAP_PASSWORD=thx1139

TDBS_LDAP_BASE=ou=tox,o=pharma,c=us

TDBS_LDAP_SEARCH=(&(objectclass=person)(uid=%u%))

TDBS_DISALLOW_GUEST=FALSE

TDBS_LDAP_MATCH=uid
```

For a UNIX/Linux TRIPsystem installation, also specify the CA certificate or CA certificate directory using either `TDBS_LDAP_CERT_DIR` or `TDBS_LDAP_CACERT` in the privileged section of the tdbs.conf file (see section “Encrypted Sessions on Linux/UNIX” for details)

On Windows, remember to ensure that the trusted root certificate authorities store is up to date (see section “Encrypted Sessions on Windows” for details).

Simplest Possible

This scenario has the following characteristics:

- Guest access to TRIP is allowed
- Anonymous LDAP access used
- Most users do not have a TRIP user profile

- A small set of privileged users have TRIP user profiles
- Unencrypted LDAP connection

This use case exemplifies the simplest possible way to configure LDAP authentication in TRIP. Please note the disadvantages below for this particular setup before considering it.

Benefits and Disadvantages

Benefits

- **Password policies**. By using LDAP authentication, it is possible to enforce password policies, something that is not possible when using the internal TRIP authentication mechanism.
- **Minimum of parallel user accounts**. Only users with special privileges in TRIP (e.g. database administrators) will need dedicated TRIP user profiles. Regular users do not.

Disadvantages

- **Less secure**: This configuration will cause the user password (among other things) be sent from TRIP to the directory server using an unencrypted connection. If interception of this network traffic is a concern, then this scenario might not be suitable.
- **Reduced database access control**: Database access rights must be assigned to the BUILTIN_GUEST user or the PUBLIC group, but no distinction can be made between user types; all will have the same rights to the databases to which access is granted

Steps

Steps required for the implementation of this scenario:

1. Examine user IDs
    - For each privileged TRIP user (e.g. database administrator user), check if the TRIP username is equal to the directory server user ID. Note the IDs that differ.
2. Rename users

For each privileged user with different IDs, rename the users as described in this document t under “Automatic LDAP user import

If dual administration is too much of a hassle, but guest login imposes too many restrictions (e.g. not possible to have different access rights for different users, user-specific procedures, etc), the third option is to configure TRIP to automatically create a TRIP user profile for a user that has successfully authenticated via LDAP.

This feature is disabled by default. To enable it, set the following logical name in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER=True
```

If the user logging in does not already exist, this setting will cause TRIP to create the user, assigning it the user name identified by the resolving the TDBS_LDAP_MATCH value. This is normally the same as the user ID that is being authenticated with LDAP.

The owner of the user will be assigned the value specified by the logical name TDBS_LDAP_AUTOUSER_UM, or to “SYSTEM” if the name is not specified. The user specified by this symbol must have user manager (UM) privileges. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_UM=TRIPDBA
```

To assign the user to a group in addition to PUBLIC, assign the name of a group owned by the specified user manager to the TDBS_LDAP_AUTOUSER_GROUP logical name. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_GROUP=LDAPUSERS
```

The new TRIP user profile will be assigned a random password, effectively requiring successful authentication via LDAP in order to be able to login to TRIP. To set the TRIP user password to the same one as used to authenticate the user with the LDAP directory, specify the following logical name in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_PWSYNC=True
```

In addition to setting this password at the first login to the same as used in the directory, this logical name will also cause the password for the TRIP profile to be reset to the LDAP user’s password during login if it differs. A change in password for the user in the directory will, if this logical name is set, be automatically synchronized to TRIP.

Note that TDBS_LDAP_AUTOUSER_PWSYNC is less effective for subsequent password synchronization when login tickets are in use. This is because when ticket-based authentication is used, password checks are required less often. The password sync will in that case not be performed until the ticket has expired and the user needs to log in explicitly again.

A login procedure (a.k.a. start procedure) can also be assigned to the user. Such a procedure executes automatically every time the user successfully logs in. This is done by specifying the TDBS_LDAP_AUTOUSER_LOGINPROC logical name. For example, in the privileged section of tdbs.conf:

```
TDBS_LDAP_AUTOUSER_LOGINPROC=PUBLIC.STARTUP
```

Note that the specified procedure does not have to exist at the time when it is defined for a user. This follows the same semantics as the specification of a start procedure in the TRIP user profile

- Special Considerations”.
- Note that the TRIP license need room for the additional registered TRIP users in order to make it possible to safely establish their new identities.

3. Create a LDAP browse user

   - Create a user in the directory service that TRIPsystem will use to search for the record of the user to authenticate. This user needs read access to the tree in which users are stored.

- Make the user and its password known to TRIPsystem by editing the tdbs.conf file (see section “User Account for Browsing” in this document).

- Modify the file access settings in the file system for the tdbs.conf file to ensure that only privileged users have access to it and that the password, which is saved here in plain text, is not exposed.

4. Enable the guest user and anonymous LDAP use by entering the following in tdbs.conf under the Privileged section:

   - TDBS_DISALLOW_GUEST=False

   - TDBS_LDAP_ANONYMOUS=True

5. Determine how to search for LDAP users.

   - Determine the values of the logical names TDBS_LDAP_BASE and TDBS_LDAP_SEARCH in the tdbs.conf file. See section ”LDAP Lookup” and “LDAP User Authentication” in this document for details.

6. When ready to go live with LDAP authentication on this server, edit tdbs.conf:

   - TDBS_AUTH_PROVIDER=LDAP

Important Aspects

Because of the security impacts that this configuration has, it is not recommended for production use or for connection to corporate directory servers. This scenario is best used in a standalone fashion for development purposes, using a separate, small directory server with dummy accounts for which there is no security impact if somebody listens in on the traffic.

Configuration Example

Below is an example tdbs.conf configuration for this scenario.

```
TDBS_AUTH_PROVIDER=LDAP

TDBS_LDAP_SERVER=ad.example.com

TDBS_LDAP_MECHANISM=SIMPLE

TDBS_LDAP_ANONYMOUS=True

TDBS_DISALLOW_GUEST=False

TDBS_LDAP_BASE=ou=tox,o=pharma,c=us

TDBS_LDAP_SEARCH=(&(objectclass=person)(uid=%u%))
```

Troubleshooting

This section provides some hints and suggestions for resolving issues with LDAP configuration and access.

I cannot get LDAP to work with TRIP. How do I find out why?

If the error message from TRIP does not provide enough information, enabling LDAP error or debug logging can provide some very useful hints. Set the following in the privileged section of the tdbs.conf file:

```
TDBS_LDAP_ERRORLOG=DEBUG
```

Or, to just limit it to errors:

```
TDBS_LDAP_ERRORLOG=TRUE
```

The log file `ldap_errors.log` will be written in the directory identified by the TDBS_LOG value in tdbs.conf.

This log file is always appended to and shared among all sessions. This means that once you have obtained the log, you should disable the TDBS_LDAP_ERRORLOG. Also, avoid having this logging enabled on production systems, but if you do feel compelled to have it enabled there anyway, only use the value TRUE so that only error situations are logged.

LDAP connections on my Linux or Solaris are very slow

This is typically caused by a DNS misconfiguration on the TRIP machine. The LDAP server address (as specified in tdbs.conf using the TDBS_LDAP_SERVER) must be possible to resolve via DNS. It is not sufficient to only have the name specified in /etc/hosts.

Make sure that the configured LDAP server is possible to find via DNS lookup (test using dig or nslookup). If dig and nslookup fails to resolve the LDAP server name, add the requisite DNS server to /etc/resolv.conf. Note that this file may be automatically updated by the network manager tool - check what applies to your operating system / Linux distribution before you edit this file manually!

Windows: CA certificate validation in TRIP fails for SSL/TLS

Install CA certificate as a trusted root certificate

Your LDAP server may be using a self-signed certificate for SSL/TLS connections, or the certificate may be issued by a non-public certificate authority (CA) such as a corporate CA. In such circumstances, ensure that the issuing CA certificate is installed as trusted on the Windows TRIP server. See section “Certificates and the Trusted Root Certificate Store” in this document for more information.

CA certificate trust installed, but verification still fails

Enable LDAP debug logging (see related troubleshooting hint above). If the log contains an error with code 800B010F and message "The certificate's CN name does not match the

passed value" (possibly localized to your configured OS language), the cause may be a Windows issue caused by naming conflicts in the certificate versus what the Windows LDAP functionality expects. To resolve this, you will have to adjust the windows registry. Follow these steps:

1. Open the registry editor application
2. Locate registry key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LDAP
3. Create a new REG_DWORD value that is named UseHostnameAsAlias, and set the value to anything other than zero.
4. Reboot

Refer to this Microsoft page for additional information:

https://support.microsoft.com/en-us/help/2275950/an-error-occurs-when-you-try-to-establish-ssl-connections-to-the-nodes

Linux/UNIX: CA certificate validation in TRIP fails for SSL/TLS

Verify that you have set either the TDBS_LDAP_CERT_DIR or TDBS_LDAP_CACERT logical names in the privileged section of the tdbs.conf file.

If you are using TDBS_LDAP_CERT_DIR, verify that:

- The CA certificates are located in the specified directory.
- The certificates are in Base64-encoded PEM format
- There are symbolic links in the directory named after the hash of the certificate subject names. See section “Certificates” on page 8 in this document for more information.

If you are using TDBS_LDAP_CACERT, verify that the value represents an existing file and that the file is a certificate in Base64-encoded PEM format.

If the above checks out but you still have not been able to resolve the matter, you can enable LDAP debug logging to obtain additional hints as to what may be wrong. For information on how to set this up, see troubleshooting hint for this above.