Table of Contents

LdapContrib

Authentication, WikiName mapping and authorization with LDAP.

Introduction

This add-on offers a UserMappingManager, a PasswordManager and a LoginManager for TWiki, using LDAP to provide:

- An automatic generation of WikiNames using LDAP data.
- The possibility to use LDAP groups in access control lists.
- Authentication, by using usernames and passwords stored in LDAP.
    - This is optional, another option is to use your web server for authenticating users, for example with SSO.

It stores permanent and temporary data in a local *cache*, a BerkeleyDB v1 database.

You can choose to fill the cache with all relevant user records from LDAP when first building it (`{PreCache} = 'all'`), or to populate them into the cache as they hit the wiki. The cache can be refreshed at certain intervals, either automatically which defaults to once per day, or by using scheduled job on your web server. You can also refresh the cache via the browser, by appending `?refreshldap=on` to the URL, which is done by clicking the button below:

Refresh Cache

Tip: You can add this button on any page by adding

```
%INCLUDE{"%SYSTEMWEB%.LdapContrib" section="refreshbutton"}%
```

to it.

If you are using TWikiUserMapping today, you can also choose to store preserve these mappings into the database when it is built for the first time.

Also, have a look at these plugins which makes use of the LdapContrib package:

- TWiki:Plugins/LdapNgPlugin: a rewrite of the LdapPlugin by Gerard Hickey as authentication, user management and other LDAP applications were brought together in LdapContrib (this package)
    - In this plug-in, you also have the option to make WikiUsers interactive when hovering over them in a topic, which looks up their LDAP identity in the LDAPContrib cache, before asking and presenting more information from LDAP in a info box.
- TWiki:Plugins/LdapContribAdminPlugin: tools for administering the LdapContrib database. Gives you an overview over the stored users and groups, as well as their status. Show's statistics, and let's you delete users, as well as setting a new WikiName for them.

Getting ready - Information needed to use LdapContrib

Before you can further configure the LDAP connection you will have to answer a set of basic questions about your LDAP server. These are:

- What's the host name (or IP address) of the LDAP server (e.g. ldap.my.domain.com)?
- What port does it listen to LDAP requests (e.g. 389)?

- Do you have a kind of "proxy" user that the wiki can use to perform the initial connection? You need its DN and credentials. Advice: don't use the LDAP admin account, you only need a simple user that has read access to all of the directory (or the relevant parts); it does not need any write access.
- What is the "base dn" of the directory (e.g. `dc=my,dc=domain,dc=com`)?
- What is the common root/branch for all users? For example, Are they all found under `ou=people,dc=my,dc=domain,dc=com` or are they are scattered all over the place?
- What defines a user record in LDAP? (Can for example be defined by the object class for these records (e.g. `objectClass=organizationalPerson`))
- Which attribute of a user record should be used to log in (must be unique)?
- Which attribute(s) of a user record do you want to use to construct a WikiName (used to display them online, pointing to their homepage)?
- Does your LDAP Server use SASL to authenticate connections? If so which authentication mechanism does it use (EXTERNAL, DIGEST-MD5, ...)?

If you want to use LDAP groups in TWiki ACLs (Access control lists):

- What is the common root/branch where all groups are defined (e.g. ou=group,dc=my,dc=domain,dc=com)?
- What object class do group records have (e.g. `objectClass=group`)?
- What's the name attribute of a group?
- Which attribute in a group record defines its members (e.g. member or memberUid)? Note, that if the member attribute of a group is a DN you need to enable "member indirection" (see #Membership).

Collect the answers to these questions either yourself using your favorite LDAP browser, or ask your friendly LDAP admin.

Configuration

The LdapContrib package is configured using a set of variables that need to be added to the `lib/LocalSite.cfg` configuration file. Use the configure tool (at least once) after you installed this package. Have a look at your `lib/LocalSite.cfg` file afterwards. You might also make your changes therein directly to accommodate your installation to your specific LDAP installation and user accounting. See the documentation within the `configure` tool for an explanation of the various options.

Available configuration parameters.

See `lib/TWiki/Contrib/LdapContrib/Config.spec` for all possible configuration parameters which a short explanation of what they do.

Enable LdapContrib

The mapping between LDAP users and TWikiUsers are done by the LdapUserMapping class, which has to be registered as TWiki's UserMappingManager:

```
$TWiki::cfg{UserMappingManager} = 'TWiki::Users::LdapUserMapping';
```

Authentication (log-in)

Whether you are using LDAP directly, or by web server configuration (for example to use Apache with SSO), enable the LdapPasswdUser class as TWiki's PasswordManager:

```
$TWiki::cfg{PasswordManager} = 'TWiki::Users::LdapPasswdUser';
```

The `LdapPasswdUser` provides an interface between users,passwords and emails used by other parts of LdapContrib, as well as the TWiki engine.

Option A: Using LDAP

Set the `TWiki::LoginManager::TemplateLogin` as the LoginManager:

```
$TWiki::cfg{LoginManager} = 'TWiki::LoginManager::TemplateLogin';
```

It will provide a authentication form, and will authenticate with the help of the `TWiki::Users::LdapPasswdUser` class.

There is a further option to fallback to the normal authentication mechanism by defining a secondary password manager. This allows you to create native wiki accounts, e.g. a WikiAdmin account and authenticate him without LDAP. Use the following setting to fallback to a htpasswd based authentication.

```
$TWiki::cfg{Ldap}{SecondaryPasswordManager} = 'TWiki::Users::HtPasswdUser';
```

So whenever authentication against LDAP fails, this second password manager will be used.

If you want to use SSO, see the section below.

Option B: Using the web server (For example: Apache+SSO)

If you are using apache (for example SSO), set the LdapApacheLogin class as the LoginManager

```
$TWiki::cfg{LoginManager} = 'TWiki::LoginManager::LdapApacheLogin';
```

LdapApacheLogin as TWiki's login manager will handle the user coming back from SSO after it has been authenticated.

Note; It could very well be that LdapApacheLogin can work with other web servers like *nginx*. The only requirement is that the user identity is set in the environment variable REMOTE_USER.

Authorization (access control)

To enable LDAP groups in authorization, meaning that they can be used in ACLs (access control lists), enable the `MapGroups` setting:

```
$TWiki::cfg{Ldap}{MapGroups} = 1;
```

If you want to only support LDAP groups in ACLs, enable the WikiGroupsBackoff flag, which disables the use of WikiGroups. Note that if you do choose to still support WikiGroups, the LDAP group will take precedence in case of a name clash.

```
$TWiki::cfg{Ldap}{WikiGroupsBackoff} = 1;
```

LdapContrib database admin panel

Please download TWiki:Plugins/LdapContribAdminPlugin if you want a admin panel for your LdapContrib database, which provides a GUI to:

- Search for users
- Search for groups
- View statistics of your LdapContrib database

- Search for ignored users and groups
- Change WikiName for your users
- Delete users

Users in the LdapContrib database

Automatic WikiName generation

WikiNames are generated using data from the LDAP server. The following parameters in `configure` defines how the WikiName should be built:

- `WikiNameAttributes`: a comma separated list the LDAP attributes used to generate WikiNames.
- `NormalizeWikiName`: boolean flag; if true (1), a couple of extra operations are performed on the WikiName before storing, i.e. removing illegal characters.
- `WikiNameAliases`: a comma separated key=value list of WikiNames to be mapped to another WikiName (DEPCRETATED: use `RewriteWikiNames` instead).
- `RewriteWikiNames`: a list of rewrite rules; see below

For examples using the `RewriteWikiNames` parameter, see Examples 2 and 3 below.

Example 1 - Concatenating and normalizing

Given the setting

```
$TWiki::cfg{Ldap}{WikiNameAttributes} = 'givenName,sn';
$TWiki::cfg{Ldap}{NormalizeWikiNames} = 1;
```

The `givenName` and `sn` (surname) LDAP attributes will be fetched and concatenated to form a proper WikiName, so that "givenName=Hans-Peter,sn=Schw@rze" will result in the WikiName "!HansPeterSchwrze". Notice that `NormalizeWikiNames` has stripped off the '@' character from the final WikiNames. This is handy when email addresses defines a LDAP unique user.

Example 2 - Use of RewriteWikiNames

Given the WikiName is derived from the mail attribute, then use the following rule to strip off domain parts from the wiki name:

```
$TWiki::cfg{Ldap}{RewriteWikiNames} = {
  '^(.*)@.*$' => '$1'
};
```

This can be further refined to prevent name clashes by adding back the domain part:

```
$TWiki::cfg{Ldap}{RewriteWikiNames} = {
  '^(.*)@(.*?)\..*\..*$' => '$1_$2'
};
```

So given your company uses email addresses like john.doe@germany.mycompany.com and john.doe@spain.mycompany.com it will generate the WikiNames JohnDoeGermany and JohnDoeSpain from it (given NormalizeWikiNames is switched on too).

Example 3 - Use of RewriteWikiNames

If you struggle with many LDAP logins clashing on the same automatically genereated WikiName, a trick is to add additional LDAP attributes to the `WikiNameAttributes`, then to strip it off with a rewrite rule:

Example:

```
$TWiki::cfg{Ldap}{WikiNameAttributes} = 'givenName, sn, sAMAccountName';
$TWiki::cfg{Ldap}{RewriteWikiNames} = {
  '^(.*) [^ ]+?\-adm$' => '$1 Admin',
  '^(.*) (?!\-adm$)[^ ]+?$' => '$1',
}
```

This might look wild at first but does the following:

- generate a temporary WikiName concatenating the attributes according to the `WikiNameAttributes` setting, e.g. "John-Doe Smith jds", where jds is the sAMAccountName value
- let's say there's also a second record for John-Doe Smith that he uses to log in
- with admin rights using his `jds-adm` login.
- rule 1 of the `RewriteWikiNames` will match the `jds-adm` and will generate a nice John-Doe Smith Admin, whereas
- rule 2 will only match those records that don't have the `-adm` suffic at their sAMAccountName.
- note that both rules only copy over the first part of the string captured in brackets (.*) over to the result leaving out the trailing sAMAccountName part.
- finally the string is wikified to make it a proper CamelCase word by all non-alphabetic characters between the parts of the name

WikiName clashes

Depending on the choice of `WikiNameAttributes` and `RewriteWikiNames` rules your LDAP records will be mapped to a proper WikiName. These have to be unique to represent the identity of the person granted access to TWiki. However, while you LDAP records are unique due to their DNs (distinguished names), it is quite common that two independent records result in the same WikiName. That's a so called name clash. You are strongly encouraged to control the way WikiNames are generated to keep the number of name clashes as low as possible. Use appropriate rewrite rues as described above.

In real world you will most probably run into a name clash that you can't possibly resolve. In that case LdapContrib will *enumerate* all JohnSmiths as they are found, calling them JohnSmith, JohnSmith1, JohnSmith2, etc. Each of these maps back to a unique DN in your LDAP directory of course.

When LdapContrib is generating its cache for the first time, the actual mapping is pretty arbitrary, given there are no additional means to distinguish the names. From there on WikiNames are kept, *even when you reconfigure LdapContrib itself*. If you however choose to remove the cache file, or you choose to refresh the cache using "?refreshldap=force", the cache is rebuilt not necessarily keeping these mappings.

Note doing that deleting the cache file, or force refreshing the cache later in the life time of your wiki might accidentally swap the mapping of LDAP records to WikiNames in case they clash. So be very cautious when doing that.

Case Insensitive login

Login can be made case-sensitive by enabling this setting:

```
$TWiki::cfg{Ldap}{CaseSensitiveLogin} = 1;
```

With this setting enabled, logins `jdoe` and `jDoe` are considered as two unique login names and will get two different automatically created WikiNames.

Migrating from TWikiUserMapping to LdapUserMapping

LdapContrib allows you to migrate the rules set if you are changing from TWikiUserMapping to LdapUserMapping.

If `{PreserveTWikiUserMapping}` is set to true in `config`, the users from Main.TWikiUsers will be taken into consideration *when building the first database* (and never again).

```
$TWiki::cfg{Ldap}{PreserveTWikiUserMapping} = 1;
```

This also will include the login names from Main.TWikiUsers which are deleted from LDAP If `{PreserveWikiNames}` is set.

```
$TWiki::cfg{Ldap}{PreserveWikiNames} = 1;
```

If:

- You run LdapContrib for the first time (the LdapContrib database is not created yet).
- The Main/TWikiUsers.txt topic is present with WikiName to login name mappings.
- If `{PreserveTWikiUserMapping}` is true

Then:

1. The Main.TWikiUsers.txt topic will be iterated over the first time LdapContrib is run, meaning the first time user visits the TWiki with LdapContrib enabled.
    - Rules with invalid WikiNames (determined by =TWiki::Func::isValidWikiWord) will be skipped
    - Rules with invalid login names (determined WikiNames (determined by {Exclude},{NormalizeLoginNames} and {LoginPattern} configure flags) will be skipped.
    - Rules with WikiNames that have been seen before in the list will be skipped.
    - Rules with login names that have been seen before in the list will be skipped.
    - Rules with invalid dates will be skipped.
2. All users will be retrieved from LDAP, but not stored immediately.
3. All the users from Main/TWikiUsers.txt (including users deleted from LDAP depending on the `{PreserveWikiNames}` flag) is added, then the other users from LDAP.

Groups in the LdapContrib database

Normalization and rewrite rules

Similar to the WikiName of a user, group names can be normalized using

```
$TWiki::cfg{Ldap}{NormalizeGroupNames} = 1;
```

When multiple groups in your LDAP directory clash on the same group name you might actually wish to merge these groups as used online in your wiki. This is done using the `MergeGroups` flag. When disabled, clashing groups are reported as a warning in the server log files when the LDAP cache is refreshed. To merge:

```
$TWiki::cfg{Ldap}{MergeGroups} = 1;
```

Group names can be rewritten using a set of rewrite rules. This is useful when the names as stored in your LDAP directory don't satisfy your criteria for being displayed online.

A group rewrite rule is specified using

```
$TWiki::cfg{Ldap}{RewriteGroups} = {
  'pattern1' => 'substitution1',
  'pattern2' => 'substitution2',
  ...
};
```

Each rule consists of a pattern that will be substituted in the group name as specified. The substitute can contain variables `$1`, `$2`, ... , `$5` to insert the first, second, ..., fifth bracket pair in the key pattern. (see perl manual for regular expressions).

Example:

Let's say we have the LDAP group "coffeebreak_users"

```
$TWiki::cfg{Ldap}{RewriteGroups} = {
  '(.*)_users' => '$1'
};
```

With this rule, this group will now be stored as "coffeebreak" in the LdapContrib cache.

Where to look for LDAP group members

LDAP servers follow different schemata to define group memberships. The memberships is either stored by the unique user ids directly, or by the full DN (distinguished name) of the user entries. If the latter is the case, `MemberIndiraction` has to be turned on so LdapContrib looks up the unique user id behind a DN instead of using it directly:

```
$TWiki::cfg{Ldap}{MemberIndirection} = 1;
```

The reverse relation, where the *user records* hold membership information (for example using a `memberOf` attribute) is maintained by some LDAP servers automatically. Those that encode membership this way *only* are not supported by the LdapContrib yet.

Furthermore, user objects may have one *primary* group attribute. This is a simple value that stores the id of a default group that user is member of. This attribute is defined by specifying the `PrimaryGroupAttribute` setting:

```
$TWiki::cfg{Ldap}{PrimaryGroupAttribute} = <LDAPAttribute>;
```

LdapContrib reads membership information as they are stored in the group objects, and may map the member object indirectly to the login name. In addition any "primary group" setting stored in the user objects is consulted as well.

Storing group memberships

Group memberships is only stored when refreshing the cache. When a new user hits the wiki after logging in to your SSO solution, a unique WikiName is computed and stored, but not the groups the user is a member of. If groups are enabled, make sure to refresh the cache periodically.

Case Insensitive group

Groups can be made case-sensitive by enabling this setting:

```
$TWiki::cfg{Ldap}{CaseSensitiveGroup} = 1;
```

If the settings is turned off, all the groups will be turned to lowercase before being written to or read from the database.

Precaching users and groups

LdapContrib can be set to *precache* users and groups, meaning updating all user and group entries at certain intervals to ensure that the database is correct. Three `{Precache}` modes may be set; 'all', 'existing' and 'off'.

By default everyone can refresh the cache by appending ?refreshldap=on and ?refreshldap=force to their TWiki URL. A database refresh may take up to several minutes. If you want to limit the possibility to refresh the LdapContrib database to the command line, set the `{CLIOnlyRefresh}` flag in `configure`

```
$TWiki::cfg{Ldap}{CLIOnlyRefresh} = 1;
```

Please note that the `{CLIOnlyRefresh}` is not applied the first time you run LdapContrib; When the database is non-existent, a database refresh given that `{PreCache}` is not set to `off`.

User precaching

When doing a database refresh:

- *all* stores a WikiName for every possible LDAP login name. If there is 50 000 users in LDAP, 50 000 WikiName to login name mappings will be stored in the LdapContrib database.
- *existing* updates the existing users in the LdapContrib database. If there are 1000 users in the LdapContrib database, and 50 000 on the LDAP server, only the 1000 will be updated.
- *off* turns off precaching.

Both 'all' and 'existing' updates groups, by including the memberships for the users which is in the database at the time of the group refresh.

all downloads and computes a WikiName for **all** relevant LDAP records when the database is refreshed. Example: If the database is refreshed for the first time, and you have 10000 eligible users stored in LDAP, 10000 WikiNames will be stored in the LdapContrib database. Subsequent database refreshes will add all user records that have appeared since (new users), and update the existing ones.

existing downloads all user records from LDAP, but only users already stored in the LdapContrib cache if there are changes. For new users to enter the database, they will have to authenticate using for example `TemplateLogin` or `LdapApacheLogin`.

off does not refresh the cache at all.

Group PreCaching

Groups memberships are refreshed if `$TWiki::cfg{Ldap}{Precache}` is set to `all` or `existing`. `off` does nothing.

All relevant groups with members are downloaded from LDAP. Then all the groups are stored in the LdapContrib cache, along with the members which is present in the cache after the user precache process (see above). LDAP users which is not in the cache by now are not stored.

Automatic vs manual database refresh

Use the `{MaxCacheAge}` too choose automatic or manual database refresh.

This setting defaults to 86400 seconds, meaning 24 hours. This updates the database 24 hours, and is triggered by the first request that hits the site when this period has expired.

To remove this burden from the "first visitor in the morning", you choose manual refresh by setting `{MaxCacheAge}` to zero:

```
$TWiki::cfg{Ldap}{MaxCacheAge} = 0;
```

Now, the database has to be refreshed by appending ?refreshldap=on to the request. This can be done by either hitting the "Refresh Cache" button above, or by setting up an appropriate cronjob on the machine running your wiki server.

To trigger an explicit update of the cache on 5 past midnight every day use a cronjob similar to:

```
5 0 * * * cd <wiki-install-path>/bin && ./view refreshldap=on Main/WebHome >/dev/null
```

This will call the engine on the command line and provide the necessary query parameters so that the LdapContrib will run refresh on the database.

Restricting database refresh to command-line

The `{CLIOnlyRefresh}` parameter defines who which can refresh the database. If the parameter is enabled, only a user with access to the command line can do it (CLI). If not, everyone can.

The database is refreshed by passing the URL parameter *?refreshldap=on*.

Backing up the database after a refresh

Configuration flags `{BackupCacheFile}` and `{BackupFileAge}` makes LdapContrib backup it's database after a successful refresh.

- If `{BackupCacheFile}` is true, the main database will be backed up after it's refreshed.
- If `{BackupFileAge}` is set, the database will only be backed up when the newest backup file is older than `{BackupFileAge}` seconds.

If something goes wrong with the database file, for example caused by a bad configuration change, you can swap the latest backup file, `/cache.db.DATE` with your current one (`/cache.db`), allowing your users to continue using TWiki while you troubleshoot the problem.

Simple configuration example

For the sake of this documentation we assume that users accounts in your cache are at least of the type `posixAccount` and optionally also of type `inetOrgPerson` storing email addresses. Moreover users are stored in a subtree `ou=people` and groups are defined in `ou=group`. Here are some example LDAP records:

```
dn: uid=testuser1,ou=people,dc=my,dc=domain,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Test User1
uid: testuser1
uidNumber: 1024
gidNumber: 100
homeDirectory: /home/testuser1
loginShell: /bin/bash
mail: testuser1@my.domain.com
```

```
dn: uid=testuser2,ou=people,dc=my,dc=domain,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Test User2
uid: testuser2
uidNumber: 1024
gidNumber: 100
homeDirectory: /home/testuser2
loginShell: /bin/bash
mail: testuser2@my.domain.com
mail: testuser2@gmx.com

# users, Group, nats.informatik.uni-hamburg.de
dn: cn=users,ou=group,dc=my,dc=domain,dc=com
objectClass: posixGroup
cn: users
gidNumber: 100
memberUid: testuser1
memberUid: testuser2
```

Please have a look at your LDAP manual on how to set up an LDAP server and populate it with user account records. Have a look at the Quick-Start Guide on how to install OpenLdap.

Use the following settings for the above example:

```
$TWiki::cfg{Ldap}{Host} = 'ldap.my.domain.com';
$TWiki::cfg{Ldap}{Port} = 389;
$TWiki::cfg{Ldap}{UserBase} = 'ou=people,dc=my,dc=domain,dc=com';
$TWiki::cfg{Ldap}{LoginFilter} = 'objectClass=posixAccount';
$TWiki::cfg{Ldap}{LoginAttribute} = 'uid';
$TWiki::cfg{Ldap}{WikiNameAttributes} = 'cn';
$TWiki::cfg{Ldap}{NormalizeWikiNames} = 1;
$TWiki::cfg{Ldap}{GroupBase} = 'ou=group,dc=my,dc=domain,dc=com';
$TWiki::cfg{Ldap}{GroupFilter} = 'objectClass=posixGroup';
$TWiki::cfg{Ldap}{GroupAttribute} = 'cn';
$TWiki::cfg{Ldap}{MemberAttribute} = 'memberUid';
$TWiki::cfg{Ldap}{MemberIndirection} = 0;
$TWiki::cfg{Ldap}{MapGroups} = 1;
```

Debugging LDAP

Very often either the directory structure of your LDAP server is rather complicated or the settings of LdapContrib are not as intuitive or easy to comprehend. For this reason the first best thing to do is to make sure the TWiki server can communicate with the LDAP server.

You can use the `twiki/tools/ldaptest` utility to test the basic LDAP connection independently from LdapContrib and TWiki: Edit the utility and modify the settings for LDAP server, base DN, bind user and bind DN. Run the utility, optionally with an LDAP search query parameter to explore the records as returned by your LDAP server.

To debug the LdapContrib settings, enable `{Ldap}{Debug}` in configure and analyze the output as generated in the TWiki debug log. Try to examine the output as generated by (a) a normal access to the site versus (b) a `refresh` of the LDAP database. There should be sufficient information in the log files to get to know what exactly happens during this important step.

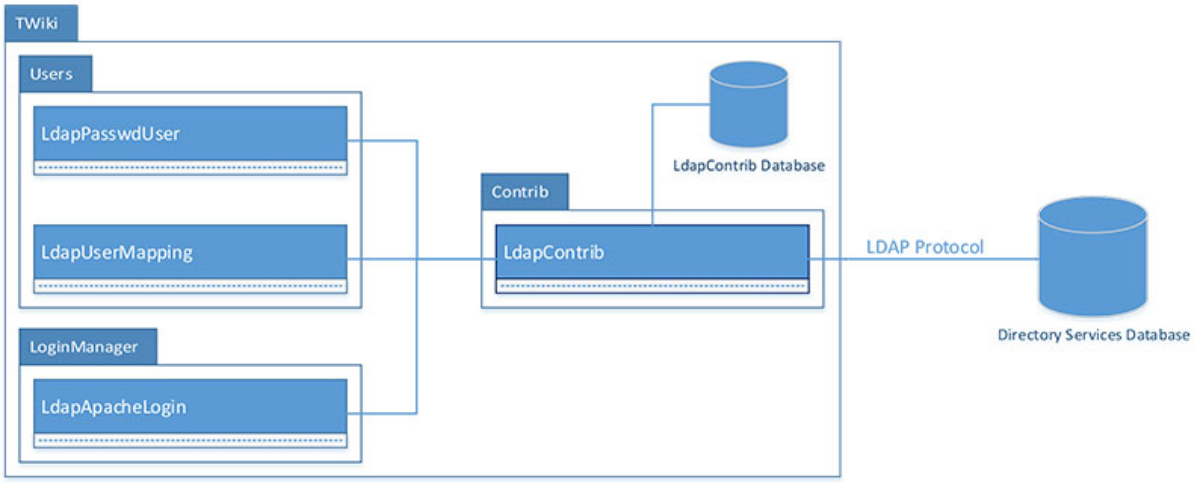
Note that that `{Ldap}{Debug}` will *only* switch on debug notes of LdapContrib, *not* of your web server performing LDAP requests on its own depending on your setup.

The LdapContrib creates `twiki/working/work_areas/LdapContrib/cache.db`, which is a binary database file (BerkeleyDB v1). Use the `twiki/tools/ldapdump` utility to convert the file into plain text if you want to

examine the database.

This can also be achieved using the tool `db_dump` available in various Linux package repositories: `db_dump -p -f cache.txt cache.db`

(Advanced) Implementation documentation



Package diagram for the LdapContrib add-on

As shown in the figure above, LdapContrib implements three TWiki engine classes:

- A LoginManager - TWiki::LoginManager::LdapApacheLogin.
  - The LoginManager should be used if you want to authenticate using the web server. If not, stick with TemplateLogin.
  - If authentication is successful using apache, and the identity returned to the TWiki engine does not exist in the LdapContrib cache, LdapApacheLogin will fetch the user record from LDAP and add it to the cache.
- A UserMappingManager - TWiki::Users::LdapUserMapping.
  - The UserMappingManager is in charge of providing a link between WikiNames, login names and LDAP groups to the TWiki engine.
  - This class allows to use user names and groups stored in an LDAP server inside TWiki in a transparent way. This replaces TWiki's native way to represent users and groups using topics with according LDAP records.
- A PasswordManager - TWiki::Users::LdapPasswdUser.
  - The PasswordManager is in charge of providing a link between LDAP login names, their passwords and emails.
  - It is used by the TWiki engine to get the emails for a specific user.
  - It is also used when you want to use LDAP for authentication.
  - This class does not grant any write access to the ldap server for security reasons. So you need to use your ldap tools to create user accounts.

All three classes communicate with a *helper* class `TWiki::Contrib::LdapContrib`. This class is in charge of communicating with the remote LDAP server. It stores permanent and temporary (caching) information in the local LdapContrib database. This database is most often found in the `/working/work_areas/LdapContrib/` folder relative to your TWiki installation.

LdapContrib database / cache storage format

The cache stores a series of key-value pairs in a BerkeleyDB v1 database. The following keys are used:

- WIKINAMES - list of all wikiNames
- LOGINNAMES - list of all loginNames
- GROUPS - list of all groups
- UNKWNUSERS - list of all usernames that could not be found in LDAP (to avoid future LDAP lookups)
- UNKWNGROUPS - list of all group names that could not be found in LDAP (to avoid future LDAP lookups)
- GROUPS::$groupName - list of all loginNames in group groupName (membership)
- GROUP2UNCACHEDMEMBERSDN::$groupName - list of all DNs (when in memberIndirection mode) that could not be resolved to a user or group existing in the cache when $groupName was retreived from LDAP
- EMAIL2U::$emailAddr - stores the loginName of an emailAddr
- U2EMAIL::$loginName - stores the emailAddr of a loginName
- U2W::$loginName - stores the wikiName of a loginName
- W2U::$wikiName - stores the loginName of a wikiName
- DN2U::$dn - stores the loginName of a distinguishedName
- U2DN::$loginName - stores the distinguishedName of a loginName
- U2CREATED:$loginName - stores the create date of the entry
- U2UPDATED:$loginName - stores the update date of the entry (meaning wikiName, dn or emails for the loginName is altered)

Writing a custom class communicating with LdapContrib

In order to create your own class communicationg with the LdapContrib database or LDAP using the `TWiki::Contrib::LdapContrib:: class, you will have to connect to the TWiki engine and create a object of =LdapContrib`.

```
#!/usr/bin/perl

use strict;
use warnings;

BEGIN {
  require '/twikipath/bin/setlib.cfg';
  push ( @INC, "/twikipath/lib/" );
  push ( @INC, "/twikipath/lib/CPAN/lib/" );
}

use TWiki;
use TWiki::Contrib::LdapContrib;

my $twiki = new TWiki('admin');
my $session = $TWiki::Plugins::SESSION;
my $ldap = TWiki::Contrib::LdapContrib::getLdapContrib($session);
```

Then, you can read and write to the LdapContrib database, as well query the LDAP server for information

Example 1: Reading from the LdapContrib database.

```
$ldap->getCacheTie('read'); # Tying read lock
my $data = $ldap->{data};

# Print all WikiNames in the database
print $data->{WIKINAMES};
```

```
$ldap->untieCache(); # Release lock
```

Example 2: writing to the LdapContrib database.

```
$ldap->getCacheTie('write'); # Tying exclusive lock
my $data = $ldap->{data};

# Changing WikiName for login 'jdoe' in the LDAP database
my %wikiNames = map { $_ => 1 } split /\s*,\s*/, $data->{WIKINAMES};
my $oldWikiName = $data->{U2W::jdoe};
my $newWikiName = 'JohnRobertDoe';
$wikiNames{$newWikiName} = 1;

delete $data->{'W2U::$oldWikiName'};              # Deleting old WikiName from W2U
delete $wikiNames{$oldWikiName};                  # Deleting old WikiName from WIKINAMES
$data->{'W2U::$newWikiName'} = 'jdoe';            # Setting new WikiName in W2U
$data->{'U2W::jdoe'} = $newWikiName;              # Setting new WikiName in U2W
$data->{'WIKINAMES'} = join ',', keys %wikiNames  # Updating WIKINAMES list

$ldap->untieCache(); # release lock
```

Example 3: Doing a search to the LDAP server

```
my $result = $ldap->search(filter=>'mail=*@gmx*');
my $errorMsg = $ldap->getError();
my @entries = $result->sorted('sn');
my $entry = $result->entry(0);
my $value = $entry->get_value('cn');
my @emails = $entry->get_value('mail');
```

The ldapdbtest tool

`ldapdbtest` is a tool to test the reliability of the LdapContrib database.

Since the database is very simple, without the constraints of a relational database like MySQL, more responsibility is left to the user.

The tool were made to control your logic when you want to enhance LdapContrib.

If it gives an error, it means that some logic writing to the database is wrong, and has to be investigated by a developer:

```
# cd /twiki/bin/
# perl ../tools/ldapdbtest

Test 1: Checking that each WikiUser maps to only one LoginName:
  12536 WikiUsers counted ...
  No duplicates found.

Test 2: Checking that each LoginName maps to only one WikiUser:
  12536 LoginNames counted ...
  No duplicates found.

Test 3: Given U2W::$loginName = $wikiName, and W2U::$wikiName = $reverseLoginName, check that $lo
  12536 U2W and 12536 W2U entries tested , No errors found.

Test 4: Given U2DN::$loginName = $dn, and DN2U::$dn = $reverseLoginName, check that $loginName eq
  4 U2DN and 4 DN2U entries tested , No errors found.

Test 5: Checking that each LoginName in LOGINNAMES has an U2W entry, and vice versa
  12536 LoginNames counted from LOGINNAMES, 12536 LoginNames counted from U2W::$LoginName
```

```
  No errors found.

Test 6: Checking that each WikiName in WIKINAMES has an W2U entry, and vice versa
  12536 WikiNames counted from WIKINAMES, 12536 WikiNames counted from W2U::$WikiName
  No errors found.

Test 7: Checking that each group in GROUPS has an GROUP entry, and vice versa
  2 groups counted from GROUPS, 2 groups counted from GROUPS::$group
  No errors found.

Test 8: If we find a $loginName in U2CREATED, U2UPDATED or U2EMAILS, control that the $loginName
  login names found in 12536 U2CREATED, 1 U2UPDATED and 2 U2EMAIL entries. Every login name was f

Test 9: Test that each $timestamp found in U2CREATED::$user = $timestamp and U2UPDATE::$user = $t
  12536 U2CREATED and 1 U2UPDATED entries tested , No errors found.

Test 10: Testing for unknown keys in cache:
  Valid keys (16): WIKINAMES LOGINNAMES GROUPS UNKWNUSERS UNKWNGROUPS GROUPS GROUP2UNCACHEDMEMBER
  37628 keys tested, No errors found.
```

Installation Instructions

- For an ***automated installation***, run the configure script and follow "Find More Extensions" in the in the ***Extensions*** section.
  - ♦ See the installation supplement on TWiki.org for more information.

- Or, follow these ***manual installation*** steps:
  - ♦ Download the ZIP file from the extension home on twiki.org (see below).
  - ♦ Unzip **`LdapContrib.zip`** in your twiki installation directory.
  - ♦ Set the ownership of the extracted directories and files to the webserver user.
  - ♦ Install the dependencies (if any).

Manifest

File	Description
`data/TWiki/LdapContrib.txt`	Documentation topic.
`lib/TWiki/Contrib/LdapContrib/Config.spec`	Configuration parameters definitions.
`lib/TWiki/Contrib/LdapContrib.pm`	LDAP Helper class for speaking to LDAP and the LdapContrib database.
`lib/TWiki/LoginManager/LdapApacheLogin.pm`	A TWiki LoginManager.
`lib/TWiki/Users/LdapPasswdUser.pm`	A TWiki PassowrdManager.
`lib/TWiki/Users/LdapUserMapping.pm`	A TWiki UserMappingManager.
`tools/ldaptest`	Tool to test your connection to LDAP.
`tools/ldapdump`	Tool to dump the binary LdapContrib database (BerkeleyDB) into a human-readable format.
`tools/ldapdbtest`	Tool that runs a number of tests on the LdapContrib database in order to find inconsistencies.
`pub/TWiki/LdapContrib/LdapContribPackageDiagram.jpg`	Package diagram for the TWiki::Contrib::LdapContrib add-on.

Contrib Info

This work was partly sponsored by

- Spanlink Communications
- Trivadis
- IBM
- Deutsche Flugsicherung
- Testo
- CERN

Original author:	Michael Daum
Copyright:	© 2006-2010 Michael Daum http://michaeldaumconsulting.com © 2007-2014 TWiki:TWiki.TWikiContributor © 2013-2014 CERN
License:	GPL (GNU General Public License)
Plugin Version:	2014-08-03

Show Change History Hide Change History

2014-08-03:	TWikibug:Item7537: Fixed bug where LASTUPDATED is not getting checked properly -- TWiki:Main.TerjeAndersen
2014-06-06:	TWikibug:Item7509: LdapContrib 2014-05-21 short comings -- TWiki:Main.HideyoImazu
2014-05-21:	TWikibug:Item7498: LdapContrib May 2014 Update, including database locks, performance tweaks, not purging deleted users from LDAP +++ -- TWiki:Main.TerjeAndersen
2013-09-03:	TWikibug:Item7332: Added configure flag for backing up the LDAP cache file when it is updated -- TWiki:Main.TerjeAndersen
2013-08-27:	TWikibug:Item7331: Added support for taking TWikiUserMapping into consideration when migrating to LdapUserMapping -- TWiki:Main.TerjeAndersen
2012-08-07:	TWikibug:Item6910: Support transliteration of Croatian charaters -- TWiki:Main.GoranJakovljevic
2012-06-21:	TWikibug:Item6891: Support GSSAPI mechanism for SASL authentication -- TWiki:Main.HideyoImazu
2012-06-07:	TWikibug:Item6878: Add {Ldap}{CaseSensitiveLogin} setting and logic -- TWiki:Main.PeterThoeny
2012-05-05:	TWikibug:Item6837: Add new ldapdump utility; fix broken refresh button; better error and debug reporting; fix {Ldap}{Exclude} list; doc improvements -- TWiki:Main.PeterThoeny
2011-05-07:	TWikibug:Item6701: Doc improvements -- TWiki:Main.PeterThoeny
16 Dec 2010:	implemented new `expand` feature of `GROUPINFO` found in newer TWikis
02 Dec 2010:	fixed glitch where IDs weren't extracted from the cache db
01 Dec 2010:	added workaround to make the TWiki::Func API usable early enough in the processing chain; fixed the way name clashes are resolved to stabilize choices once made; added `refreshldap=force` feature to override previous decisions resolving name clashes; deprecated `WikiNameAliases` in favor of `RewriteWikiNames`
22 Nov 2010:	fixed removing realm from login again; improved clash report message
16 Nov 2010:	added tool to debug ldap server connection and structure
09 Nov 2010:	removed every lowercasing of login names; added docu and better defaults for `RewriteWikiNames`
17 Sep 2010:	fixed build script; removed hardcoded normalization that stripped off domain parts from login and wiki names; fixed utf8 transcoding for modern perls; be more careful when translation of WikiNameAttributes to proper WikiNames checking existing homepages; implemented `groupAllowsChange()` for modern TWikis
17 Nov 2009:	incremental cache updates for *big* ldap directories and inner groups feature (Cyril Bousquet, IBM); added scope parameter searching for users and groups; added rewrite rules for wiki names; added name clash resolution for wiki names; fixed race condition in cache update leading to file corruption; cleanup of internal API to properly deal with

	interim caching structures
25 May 2009:	extended transliteration of utf8 chars to polish chars (Bartosz Dziuda)
09 Apr 2009:	added RewriteGroups and MergeGroups feature
08 Apr 2009:	fixing nested ldap groups
02 Mar 2009:	prevent error when called in the middle of the TWiki constructor
27 Feb 2009:	fixing use of uninitialized value in user mapper; optimized check for existing user to respect the exclude configuration setting; renamed LdapPassword to LdapPasswdUser again to please configure; transliteration of umlaut (this seems to happen occasionally during svn ci/co or whatever)
11 Feb 2009:	allow utf8 chars in passwords; all strings from LDAP are tainted, even if it comes from mod_ldap via remote_user; only use LDAP query as a last resort to check if a user exists; prevent explicitly excluded login names to be added to the cache
20 Jan 2009:	fixed getting emails for groups and login names
19 Dec 2008:	fixed group mapping under MapGroup=off
05 Dec 2008:	fixed group mapping
04 Dec 2008:	fixed getting email info
06 Oct 2008:	dropped support for TWiki < 4.2.3; added support TLS encryption (by Wolfgang Karall)
12 Jun 2008:	added workarounds to use LDAP and MailInContrib on TWiki-4.2.0
25 May 2008:	added alias feature, fixed normalization error, fixed cache update issue added login manager for 4.2
05 May 2008:	implemented WikiNamesAliases
14 Feb 2008:	allow to disable cache aging setting MaxCacheAge to zero
01 Feb 2008:	distinguish groups clashing with user names by appending a suffix
30 Jan 2008:	first beta towards TWiki-4.2
07 Jan 2008:	fixed initializing the cache
21 Dec 2007:	added LdapApacheLogin, made updating the cache quasi atomic
22 Nov 2007:	fixed recognition of WikiGroups in a mixed setting
05 Oct 2007:	enabled native user registration using the secondary password manager; added support to change a user's LDAP password from within TWiki; added patch for `TWiki.pm` that backports some of the fixes from TWiki-4.2 to TWiki-4.1.2
05 Sep 2007:	added SASL support, added normalization of login and group names, added secondary password manager
31 Aug 2007:	rewrite of the cache
08 June 2007:	don't use the store object during TWiki's destructor; don't lookup login names of groups
04 June 2007:	don't be case sensitive for login names; fixed several utf8 issues; fixed crash when no groups where found; caching mapping privately; added MaxCacheAge; added support for nested LDAP groups
30 Apr 2007:	fixed return value on illegal lookup calls
24 Apr 2007:	be robust against the lookup-API being called with the wrong parameters; added `Debug` flag; fixed/improved group loading; deprecating `BasePasswd` in favor of `UserBase`; deprecating `BaseGroup` in favor of `GroupBase`
04 Apr 2007:	fixed group mapping on >4.1.2; renamed BasePasswd config parameter to UserBase; renamed BaseGroup config parameter to GroupBase; working around broken `configure` in 4.1.x
12 Jan 2007:	enhanced normalization of WikiNames so that they are proper WikiWords; WikiNames can be constructed from a list of LDAP attributes now
18 Dec 2006:	various performance improvements; fixed usage of `limit` argument; renamed configuration option "WikiNameRemoveWhiteSpace" to "NormalizeWikiName"; support for large databases using paged LDAP search results; new configuration option "Exclude" to exclude standard TWiki user accounts, e.g. RegistrationAgent, from being looked up in LDAP; added support for faster API implementing `isMemberOf`; added

	Config.spec file to integrate the LdapContrib into TWiki's "configure" tool; added support for WikiNames derived from mail attributes
03 Nov 2006:	fixed binding to the server by first searching the full dn instead of assuming a fixed one (issue found by Cederic Weber); added new feature MapGroup to be able to switch off group mapping and have ; login-to-wikiname conversion only
02 Aug 2006:	added a user accounts in memory cache
19 July 2006:	public release
24 May 2006:	API adjustments, improved wikiname generation
28 Apr 2006:	Initial version

Dependencies:

Name	Version	Description
Authen::SASL	>=2.00	Optional
DB_File::Lock	>=0.05	Required
DB_File	>=1.82	Required
Digest::MD5	>=2.36	Required
Net::LDAP	>=0.33	Required
IO::Socket::SSL	>=1.0	Optional
Unicode::MapUTF8	>=1.11	Required
File::Copy	>=2.05	Optional

Contrib Home:	http://TWiki.org/cgi-bin/view/Plugins/LdapContrib
Feedback:	http://TWiki.org/cgi-bin/view/Plugins/LdapContribDev
Appraisal:	http://TWiki.org/cgi-bin/view/Plugins/LdapContribAppraisal

Related Topics: AdminDocumentationCategory, TWikiUserAuthentication

---

This topic: TWiki > LdapContrib
Topic revision: r0 - 2014-06-06 - TWikiContributor

Ideas, requests, problems regarding TWiki? Send feedback
Note: Please contribute updates to this topic on TWiki.org at TWiki:TWiki.LdapContrib