Table of Contents

Ldap / AD setup

NetYCE supports user authentication against Ldap and Microsoft Active Directory (AD) servers. The AD support will circumvent the Kerberos protocol and uses Ldap directly.

NetYCE Ldap/AD is supported by delegating the user **authentication** to Ldap/AD completely, but the user **authorization** is controlled by Ldap/AD and managed within NetYCE. Where Ldap/AD is used to select the appropriate user-group for the user, NetYCE must then configure the group is the appropriate permissions levels and scopes.

The Ldap/AD setup must therefore be able to provide the user's targeted user-group. Then, the name of the Ldap/AD user-group must match the NetYCE user-group. Should the named user-group not exist in NetYCE, a default user-group can be defined, resulting in a default set of authorizations.

NetYCE will not support updating Ldap/AD passwords or support notifications of expired or about-to-expire passwords.

In the case of AD, the setup supports `userAccountControl` user-status validation so that user access can be denied when disabled or the password was expired.

> ***Note**:* This section was re-designed in version 7.1.0 to better support common AD configurations. Notably the processing of "**memberOf**" user- and group lists are now configurable.

Yce_setup table

The Ldap/AD configuration for NetYCE is defined using the 'Yce_setup' table. The Custom data tool can be used to manipulate these settings.

Selecting the profile

Multiple setups can exist in this table and are called 'profiles'. Each NetYCE server can have a unique profile or all can share the same, depending on preference.

The entry `default | yce_server | profile` defines which profile will be assigned to a new NetYCE server once it connects to the database. The default is 'netyce.org'.

Profile	Type	Parameter	Str_value	Description
default	yce_server	profile	netyce.org	A new NetYCE server will use this profile by default

After a new server registers itself, the default profile is assigned and can be modified to use a different login profile if desired.

Profile	Type	Parameter	Str_value	Description
yceone	yce_server	profile	netyce.org	the authentication profile used by this server
ycetwo	yce_server	profile	acme.production	the authentication profile used by this server
ycethree	yce_server	profile	acme.test	the authentication profile used by this server

Login policy

Each profile consists of four sections. The first section use **<profile-name> | login_policy** entries.

The login process and the methods used is controlled using these settings. When users are created using the NetYCE GUI, they are considered '**local**' users. These users have their passwords validated against the locally stored password hash of the user and are assigned the user-group that was configured.

After Ldap/AD is setup and a user logs in into NetYCE, this user will also be added to the 'Users' database table, but using the 'ldap'-type. These users are considered '**ldap**' users. The 'ldap' users will authenticate against Ldap/AD. The assigned user-group is the one extracted using the ldap-schema settings.

Since there are two types of users and two matching distinct login methods, the corresponding login method will be used. If the `login_policy|`**`enable_ldap`** is set to `no`, the Ldap login method will no longer be available.

If the group-name for a local or ldap user does not exist within NetYCE, the group named in `login_policy|`**`default_group`** will be assigned instead. This default group is appropriately named `Default` but might itself not exist. Access will be denied in those cases.

Two additional settings control what happens *after* the initial login method failed: the *other* login method can be attempted when these settings allow that.

When `login_policy|`**`local_retry_ldap`** is set to 'yes', a local user may retry logging in using the ldap method. And when `login_policy|`**`ldap_retry_local`** is set to 'yes', an ldap user may retry logging in using the local method.

Finally `login_policy|`**`local_group_override`** controls whether an ldap user is actually assigned the Ldap/AD retrieved user-group, or the one currently configured for the user.

Type	Parameter	Str_value	Description
login_policy	default_group	Default	Local or Ldap users are assigned this NetYCE user-group when the configured or Ldap user-group does not exist in NetYCE
login_policy	enable_ldap	yes	Permit logging in using the ldap login method
login_policy	local_retry_ldap	no	Permit using the ldap login method for local users that failed their local login attempt
login_policy	ldap_retry_local	no	Permit using the local login method for ldap users that failed their ldap login attempt
login_policy	local_group_override	no	Permit the assignment the local user group of the user when that user authenticated against Ldap. The Ldap group is then ignored.

Ldap admin

entries with `<profile-name>` | **`ldap_admin`** define how NetYCE must identify and authenticate itself with the Ldap server.

Type	Parameter	Str_value	Description
Ldap_admin	use_anonymous	no	Is anonymous admin allowed or does an admin_dn and password apply? Yes or No
Ldap_admin	ldap_admin_dn	cn=admin,dc=netyce,dc=org	The admin DN of Ldap to consult user-admin
Ldap_admin	ldap_admin_pass	secret	In case not anonymous, enter the password in cleartext

Ldap servers

Entries with <profile-name> | **Ldap_server** define which Ldap servers to consult and how failover behaves

Two Ldap/AD servers can be configured for redundancy purposes. The server ldap_server|**ldap_server_pri** is the primary server to be connected when an ldap login request is required. The connection is established over tcp-port ldap_server|**ldap_port_pri**. When ldap_server|**ldap_secure_pri** is set to 'yes' it uses ldap over SSL or '**ldaps**' protocol rather than the cleartext 'ldap' protocol. However, by setting the well-known-ports '389' or '636' these overrule the ldap_secure setting to their respective 'ldap' and 'ldaps' values.

To enable redundancy, set ldap_server|**enable_secondary** to 'yes' and configure the corresponding **…**_sec settings conform the above primary values.

To prevent excessive login times should the primary ldap server become unavailable, the secondary server will be used first for new logins for a minimum period. This period is '60' seconds by default.

Type	Parameter	Str_value	Description
Ldap_server	ldap_server_pri	genie.netyce.org	The fqdn or ip of the primary Ldap server
Ldap_server	ldap_port_pri	389	The ip port of the primary Ldap server
Ldap_server	ldap_secure_pri	no	To use secure-ldap 'ldap over SSL'. Well-known ports overrule
Ldap_server	enable_secondary	yes	Is a fallback Ldap server available? Yes or No
Ldap_server	ldap_server_sec	specter.netyce.org	The fqdn or ip of the secondary Ldap server
Ldap_server	ldap_port_sec	389	The ip port of the primary Ldap server
Ldap_server	ldap_secure_pri	no	To use secure-ldap 'ldap over SSL'. Well-known ports overrule
Ldap_server	failback_time	60	The time in seconds to retry the primary Ldap once the fallback is active

Ldap schema

The final part using <profile_name>| **Ldap_schema** entries define the Ldap sources, attributes and relations. Since these are usually specific to the local implementation, they are the hardest to to find the correct values for. The local Ldap admin should be consulted before first attempts.

The Ldap_schema setup parameters can be used to create several authorization sequences to match the needs of the local Ldap/AD organization.

Four base ldap schemas are supported, each illustrated below. They can serve as a starting point for a required setup.

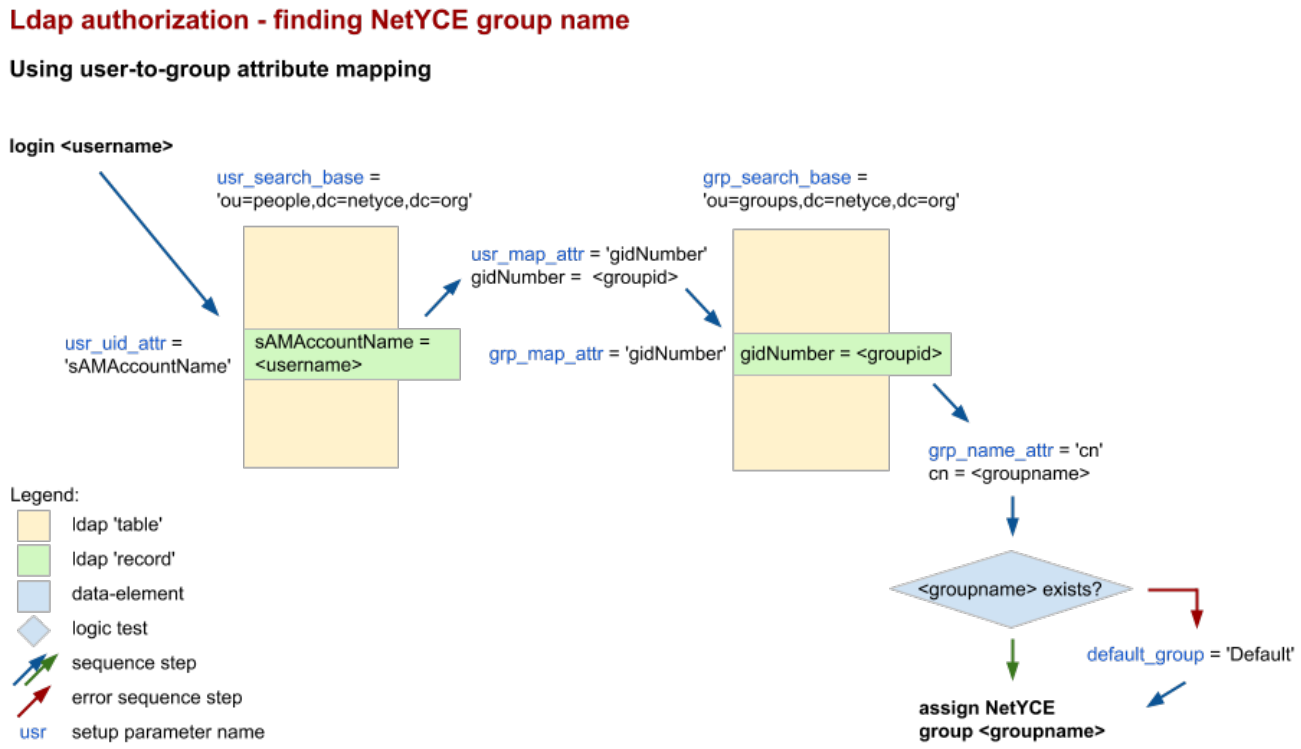
1 - Using user-to-group attribute mapping

This method uses the traditional user to group mapping of native Ldap systems. The user is first located in the user 'table', the value that identifies the group is retrieved form the user 'record' and is then used to find the corresponding group record in the group 'table'.

To configure it, the **`usr_search_base`** and the **`usr_uid_attr`** are used to locate the user name in ldap. Its record is then retrieved and the value from the attribute named in **`usr_map_attr`** is extracted. This value must identify the group record. The group record is located using the **`grp_search_base`** and the condition where the extracted value matches the attribute in **`grp_map_attr`**.

From that record the group name is retrieved using **`grp_name_attr`**and checked against existing group names in NetYCE. If needed, the default group name form **`grp_name_attr`** is used. The group defines the authorization (permissions) for the user.

Before the group resolving is started, the user authentication takes place. The username and password are used to retrieve the user 'record'. If that fails, access using the ldap login method is denied. This authentication step is common to all authorization methods.



Ldap_schema parameter	Note
usr_search_base	required
usr_uid_attr	required
usr_map_attr	required
usr_list_attr	-empty-

Ldap_schema parameter	Note
usr_list_pattern	-empty-
grp_search_base	required
grp_name_attr	required
grp_map_attr_	required
grp_list_attr	-empty-
grp_list_pattern	-empty-

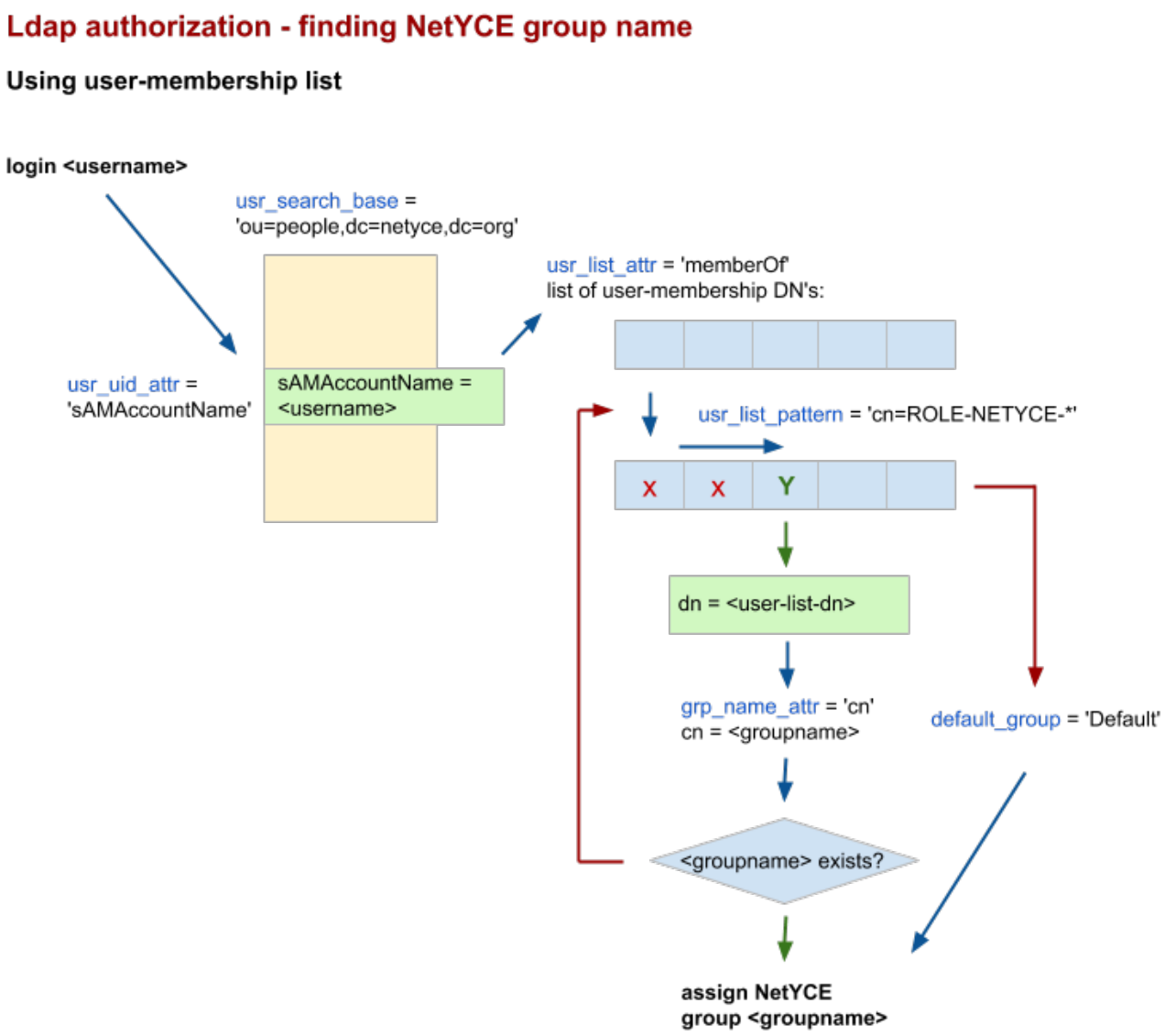
2 - Using user-membership list

Instead of creating a mapping to the group table to find the group record, this method uses a list of group memberships directly in the user record. To indicate that a list is retrieved, the **`usr_list_attr`** parameter is used, often set to “memberOf”. This list **must** consist of an array of ldap 'DN' entries (DN = distinguishedName).

This list is supposed to indicate the groups the user is a member of, hence the “memberOf” value. To find the appropriate membership within this list, the parameter **`usr_list_pattern`** must be used. The pattern specified will be compared against the DN in the list. This comparison uses wildcards ('*' and '?') and can include patterns for each element in the DN. See the paragraph List patterns on details.

Each DN item in the list is compared against the pattern, and when found matching the corresponding record is retrieved from Ldap. From the record the group name as indicated by **`grp_name_attr`** is checked against the existing NetYCE groups. If it exists the search is done and the group assigned to the user. If it does not exist, the search continues with the next element in the list.

Should none of the DN's match or none of the groups exist, then the default group name form **`default_group`** is used.

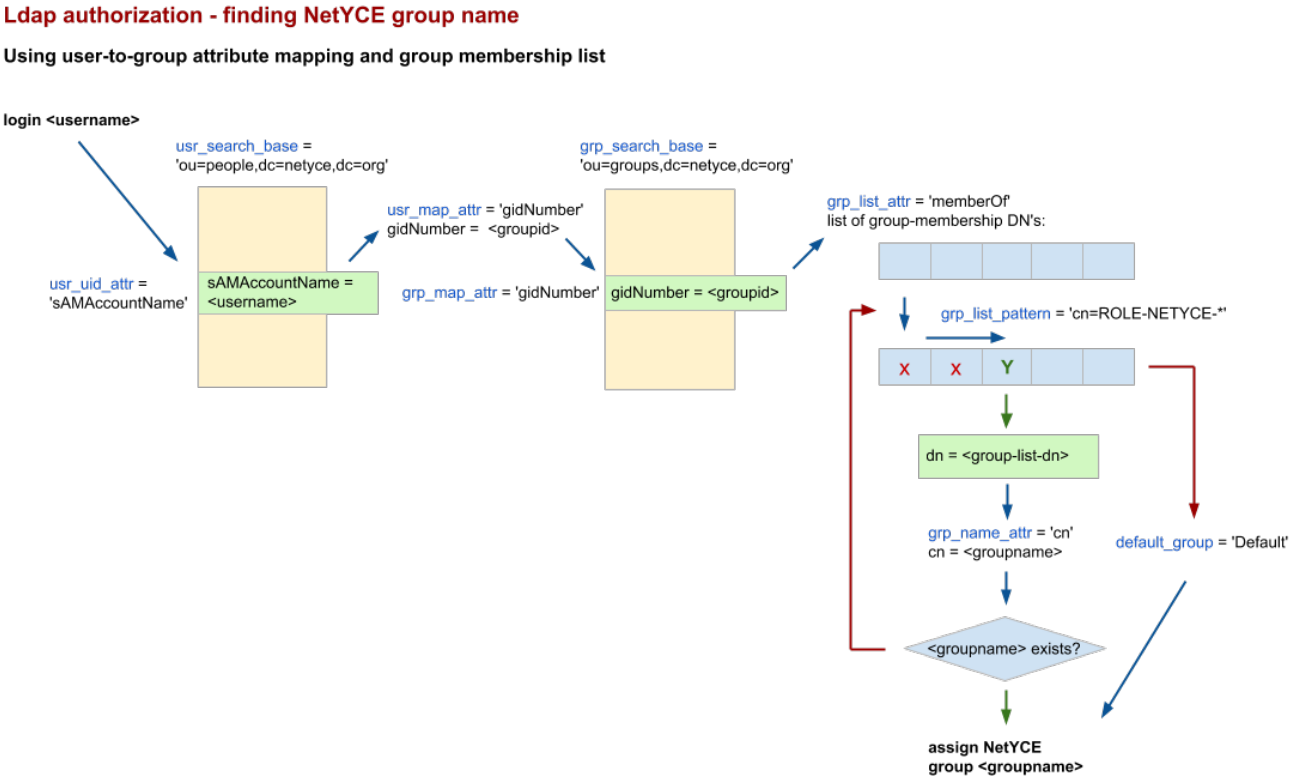


Ldap_schema parameter	Note
usr_search_base	required
usr_uid_attr	required
usr_map_attr	-empty-
usr_list_attr	required
usr_list_pattern	required
grp_search_base	-empty-
grp_name_attr	required
grp_map_attr_	-empty-
grp_list_attr	-empty-
grp_list_pattern	-empty-

3 - Using user-to-group attribute mapping and group membership list

In this setup the user-to-group mapping is used in combination with a membership list for the group. The setup parameters are used similarly to the methods 1 and 2 above, but note that now the **`grp_list_attr`** and **`grp_list_pattern`** are used in order to retrieve the group list from the group

record rater than the users'.



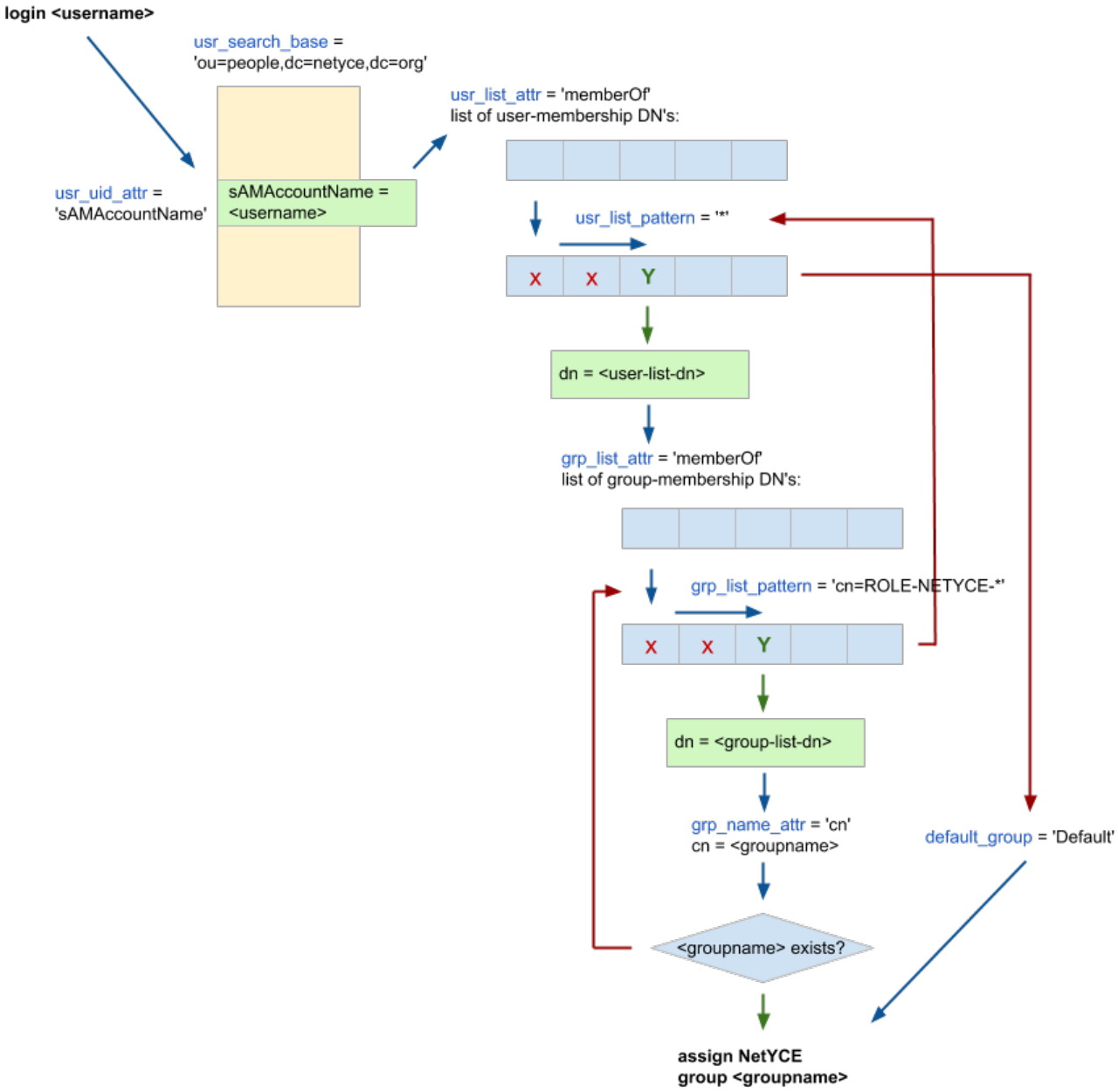
Ldap_schema parameter	Note
usr_search_base	required
usr_uid_attr	required
usr_map_attr	required
usr_list_attr	-empty-
usr_list_pattern	-empty-
grp_search_base	required
grp_name_attr	required
grp_map_attr_	required
grp_list_attr	required
grp_list_pattern	required

4 - Using user-membership and group-membership lists

This setup is using nested user- and group-lists. for each DN in the user membershiplist is the group retrieved and all memberships of the group then tested for a supported group name.

Ldap authorization - finding NetYCE group name

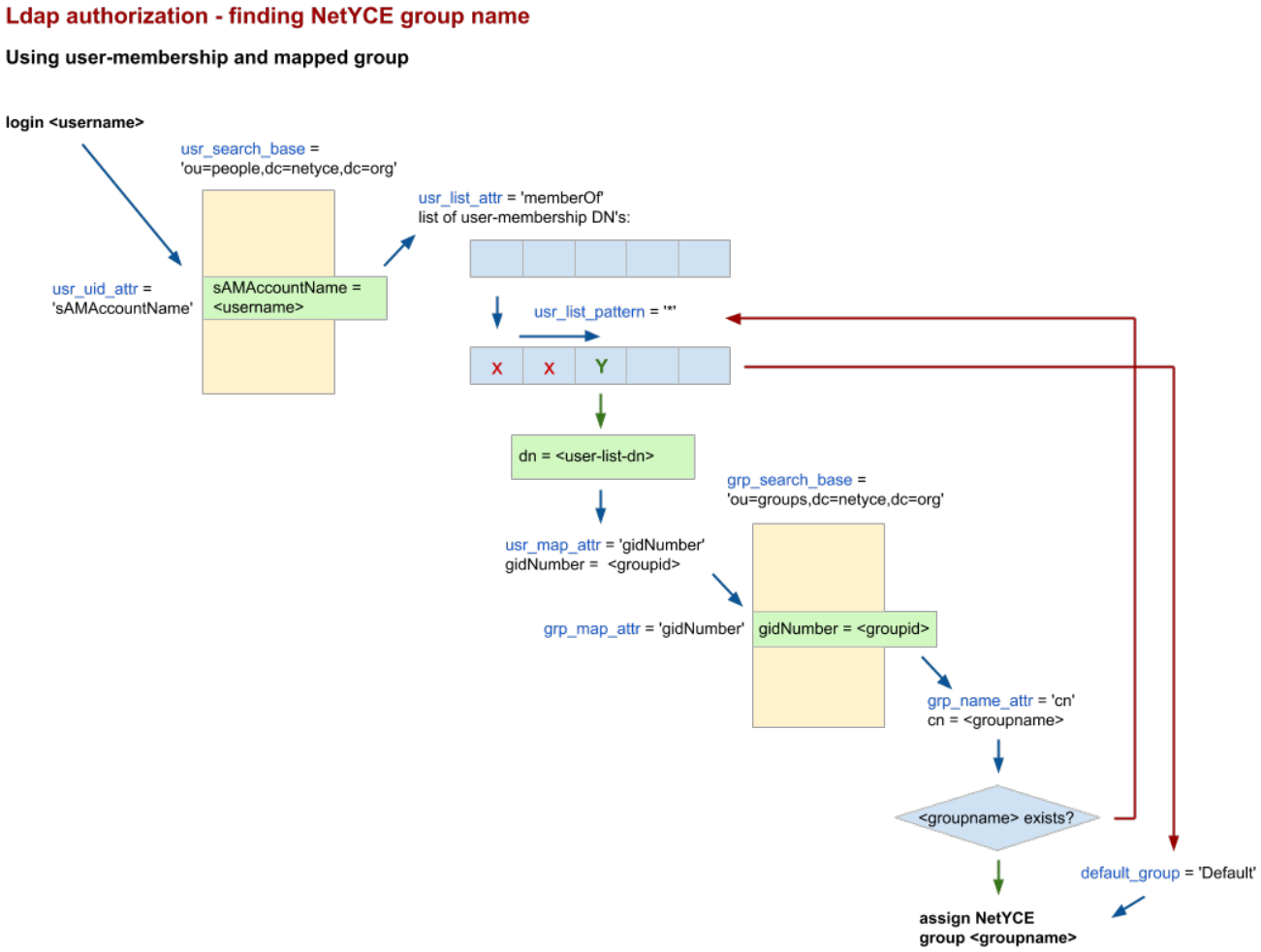
Using nested user-membership and group-membership lists



Ldap_schema parameter	Note
usr_search_base	required
usr_uid_attr	required
usr_map_attr	-empty-
usr_list_attr	required
usr_list_pattern	required
grp_search_base	-empty-
grp_name_attr	required
grp_map_attr_	-empty-
grp_list_attr	required
grp_list_pattern	required

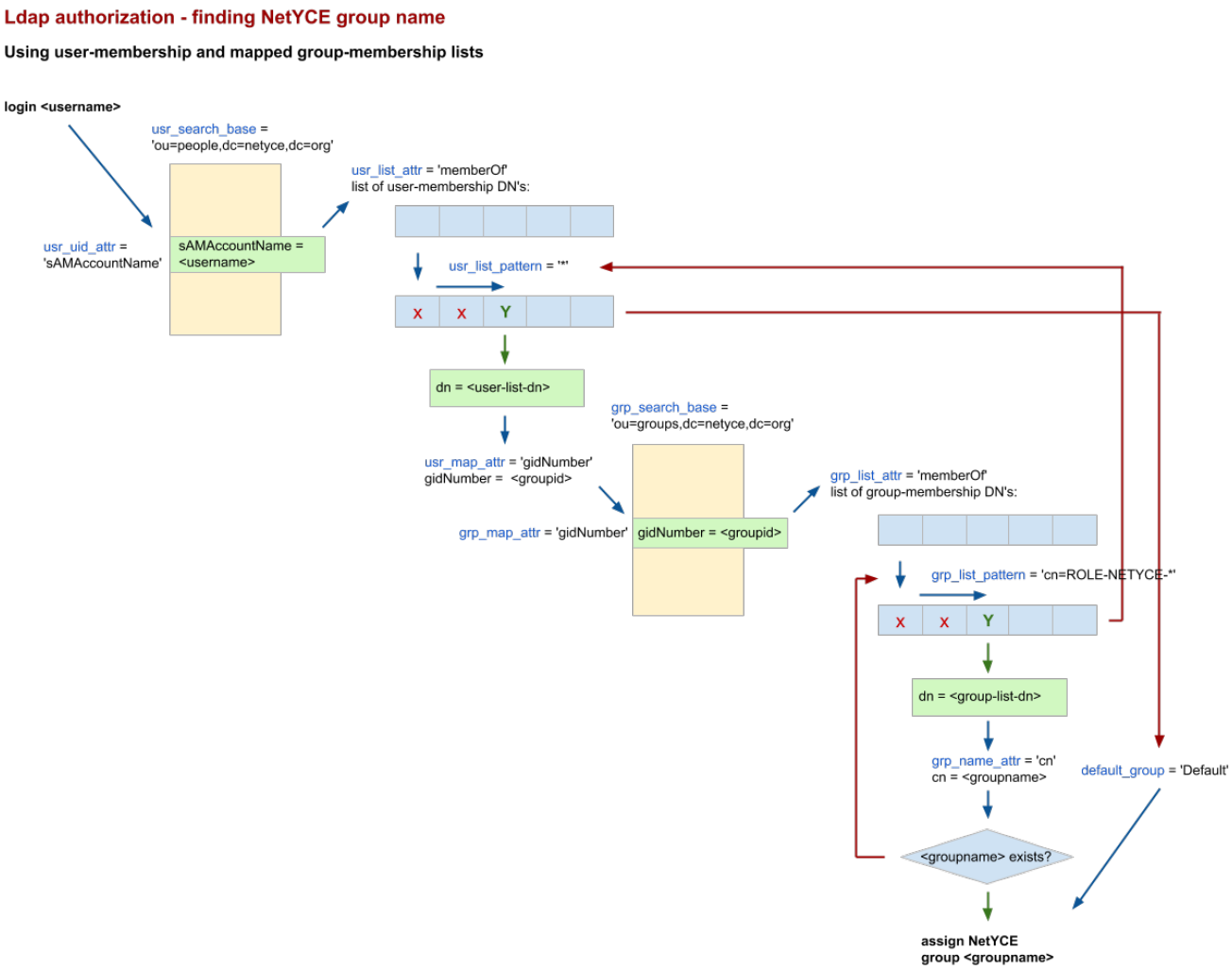
5 - Using user-membership and mapped group

> ***Note**:* This configuration is not (yet) supported. If a use-case demands its implementation, please contact NetYCE.



6 - Using user-membership and mapped group-membership lists

> ***Note**:* This configuration is not (yet) supported. If a use-case demands its implementation, please contact NetYCE.



User attribute mappings

NetYCE users using Ldap authentication and authorization have a user-account created for them in the NetYCE administration. As with local users, ldap users have several attributes associated with them. For ldap users these can be retrieved from the Ldap user record when logging in successfully.

The following setup ldap_schema variables can be defined which will be assigned to the NetYCE user account:

ldap_schema variable	**NetYCE account variable**	**Sample value**
usr_uid_attr	User_id	sAMAccountName
usr_name_attr	Full_name	displayName
usr_mail_attr	user_mail	mailAddress
usr_tel_attr	user_tel	telephoneNumber
usr_mobile_attr	user_mobile	mobileNumber
usr_notes_attr	user_notes	departmentName
usr_status_attr	n/a	userAccountControl

In this list the “usr_uid_attr” is not really a mapping variable since it is the login name of the user and mandatory for the ldap operation. And neither is “usr_status_attr”. It is used to support the userAccountControl status from AD (see AD user account control).

List patterns

The **`usr_list_pattern`** and **`grp_list_pattern`** values are used to filter DistinguishedName (DN) entries from a member list. These DN entries usually have formats like:

```
CN=ROLE-NETYCE-E-department1,OU=subgroup1,OU=group1,DC=netyce,DC=org
CN=ROLE-NETYCE-D-department2,OU=subgroup2,OU=group2,DC=netyce,DC=com
```

To filter the different elements of these DN we need a comparison function with some advanced features. The pattern to match against the DN's is as with the DN's separated by commas and each element will be compared against its corresponding DN element. Only if all elements of the pattern match all elements of the DN will the comparison result in a match.

The pattern however, needs not to have elements for all elements of the DN. If the pattern runs out of elements the comparison is considered a match. Therefore a pattern with only one element will be tested against the first element of the DN and ignore the remaining elements.

The pattern matching is case in-sensitive and supports the wildcard characters '*' and '?' in the value of each element. The identifier part (e.g. CN, OU, DC) of each element can have no wildcards and must be included in the element, unless the entire element is an “*”.

Some examples of patterns that can be used with these DN examples:

```
sample DN's:
  CN=ROLE-NETYCE-E-department1,OU=subgroup1,OU=group1,DC=netyce,DC=org
  CN=ROLE-NETYCE-D-department2,OU=subgroup2,OU=group2,DC=netyce,DC=com
sample patterns:
  CN=ROLE-NETYCE-D-*
  CN=ROLE-NETYCE-?-dep*
  CN=ROLE-NETYCE-*,OU=*group*
  CN=ROLE-NETYCE-*,*,*,DC=netyce,DC=com
```

If there is a need to match against more than one possible value for an element these can be included in the pattern element by separating them with a '|'. Many of these multiple values may be included for each element of the pattern. When comparing using multiple values, each value is evaluated from left to right and the first value match of an element will stop the evaluation for that element, effectively performing an 'OR' operation on the elements values.

Please note that the syntax of each element is `<identifier>=<value>[|<value>]`. The “CN”, or identifier, part must have an = followed by one or more values separated by an `|`.

```
sample multiple-value patterns:
  CN=ROLE-NETYCE-E-*|ROLE-NETYCE-D-*
  CN=ROLE-NETYCE-E-*|ROLE-NETYCE-D-*,*,*,*,DC=org|com
  *,*,*,*,DC=org|com
wrong are:
  CN=ROLE-NETYCE-E-*|CN=ROLE-NETYCE-D-*
  C*=ROLE-*
  *,*,*,*,DC=org|DC=com
  ,DC=org|com
```

An empty pattern will be equivalent to the 'pass-all' pattern:

```
pass-all pattern:
  *
```

Active Directory support

Microsoft Active Directory (AD) is LDAP based and can be accessed using either Kerberos and Ldap. NetYCE uses Ldap due to its more generic and low-level operation.

multi-Forest deployments

NetYCE support of AD is limited to a single “Forest” using redundant servers. The redundant servers therefore are exposing the same AD domain tree and must be considered backups of each other. In a multi-forest environment, each AD server would expose different domain trees, requiring distinct ldap schemas.

In cases where a multi-forest environment is to be supported by NetYCE, a workaround is available in the form of dedicated NetYCE front-end servers. Since the ldap schema is part of a login-profile and profiles are selected on a per server name basis, users belonging to one AD forest can be assigned a dedicated server while users belonging to another AD forest must connect to their corresponding server.

Once logged-in, and the permissions granted, the users will experience the same NetYCE database, networking and jobs environment.

User account control

Users logged-in using ldap will not be able to change their passwords. But they might be denied access to NetYCE using the AD “userAccountControl”.

This account control is enabled by setting **`usr_status_attr`** to “userAccountControl”. Since the AD implementation involves a bitmask where each bit represents a status, an appropriate mapping of these bitmasks must be used. NetYCE currently recognizes the following masks (using hex notation) and their statuses.

```
 0x02     => "disabled"
 0x08     => "no_homedir"
 0x010    => "lockout"
 0x022    => "passwd_not_required"
 0x040    => "passwd_cant_change"
 0x080    => "encrypt_passwd_allowed"
 0x0100   => "temp_dupl_account"
 # 0x0200 => "normal_account"
 0x40000  => "smartcard_required"
 0x800000 => "passwd_expired"
```

If the user account control is validated true for any of these masks, access will be denied with the indicated reason. Obviously the 0x0200 representing normal access should be skipped. If additional masks are to be implemented, please contact NetYCE.

ldapsearch examples on AD

query user 'test'

```
ldapsearch -LLL -H ldap://192.168.56.101:389 -b 'dc=netyce,dc=org' -D
'administrator@netyce.org' -W '(sAMAccountName=test)'
```

query whether user 'test' is a member of the operators group:

```
ldapsearch -LLL -H ldap://192.168.56.101:389 -b 'dc=netyce,dc=org' -D
'administrator@netyce.org' -W
'(&(objectClass=user)(sAMAccountName=test)(memberof=CN=operators,CN=Users,DC
=netyce,DC=org))'
```

query user 'test' on a ldaps server

```
 ldapsearch -LLL -H ldaps://192.168.56.104:636 -b 'dc=netyce,dc=org' -D
'administrator@netyce.org' -W '(sAMAccountName=test)'
```

Query the groups user 'test' is a member of

```
ldapsearch -LLL -H ldaps://192.168.56.104:636 -b 'dc=netyce,dc=org' -D
'administrator@netyce.org' -W
'(&(objectCategory=group)(member=CN=test,CN=Users,DC=netyce,DC=org))'
```

For this to work add the following to the ldap.conf to disable certificate authentication:

```
HOST 192.168.56.104
PORT 636
TLS_REQCERT ALLOW
```

https://forums.opensuse.org/showthread.php/401522-performing-ldapsearch-over-tls-ssl-against-active-directory#post1908811 explains the steps needed to also authenticate the certificate.

Errors and causes:

```
Ldap access error (80090308: LdapErr: DSID-0C0903C5, comment:
AcceptSecurityContext error, data 52e, v23f0)
```

Something with the bind is not going right, check configuration regarding ldap server, ldap_admin_dn, ldap_admin_pass and if use_anonymous is proper.

```
Ldap group lookup failed (0000208D: NameErr: DSID-03100213, problem 2001
(NO_OBJECT), data 0, best match of: 'DC=netyce,DC=org' )
```

this can happen when you have local_group_override set to no and the group variables are not properly set resulting in no groups to be found for the user who did authenticate properly