univention
be open.

Tales from integrating OpenLDAP with the modern world

How we solve real-world challenges in Univention Nubus using LDAP

Florian Best
Univention GmbH
LDAPCon 2026

Everything from this talk in one place:

UNIVENTION PORTAL
Frontend
Server
Consumer
OIDC
SAML
KEYCLOAK
Identity Provider
OIDC
SAML
LDAP
AUTHZ
MANAGEMENT UI
OPENLDAP
Identity Store
Primary
Proxy
Replica
USER
GROUP
DEVICE
DIRECTORY MANAGER
UDM
REST
SAMBA / AD
Guardian
PROVISIONING
Create
Update
Remove
UCR
Notifier
Listener
Application Dock
Groupware / Mail
File-Sharing
WiFi-Access
Ingress
Application Gateway
Load Balancing
TLS Termination
Routing
NUBUS FOR UCS
NUBUS FOR KUBERNETES

UNIVENTION PORTAL
Frontend
Server
Consumer
OIDC
SAML
MANAGEMENT UI
OPENLDAP
Identity Store
Primary
Proxy
Replica
USER
GROUP
DEVICE
DIRECTORY MANAGER
UDM
REST
SAMBA / AD
PROVISIONING

Univention Nubus

Identity & Access Management

- **Two deployment platforms**
  - **UCS** - on-premises
    - replication via **Univention Directory Listener / Notifier**
  - **Kubernetes / Cloud**
    - replication via **LDAP syncrepl**
- **Extensible with applications**
  - Groupware & Mail - e.g. **Open-Xchange**
  - File Sharing - e.g. **nextCloud**
  - Network Access - e.g. **RADIUS**
- **Modern provisioning stack**
  - event-driven provisioning based on **NATS**
- **Delegated Administration**
  - policy-based authorization with **Guardian**

Identity Services

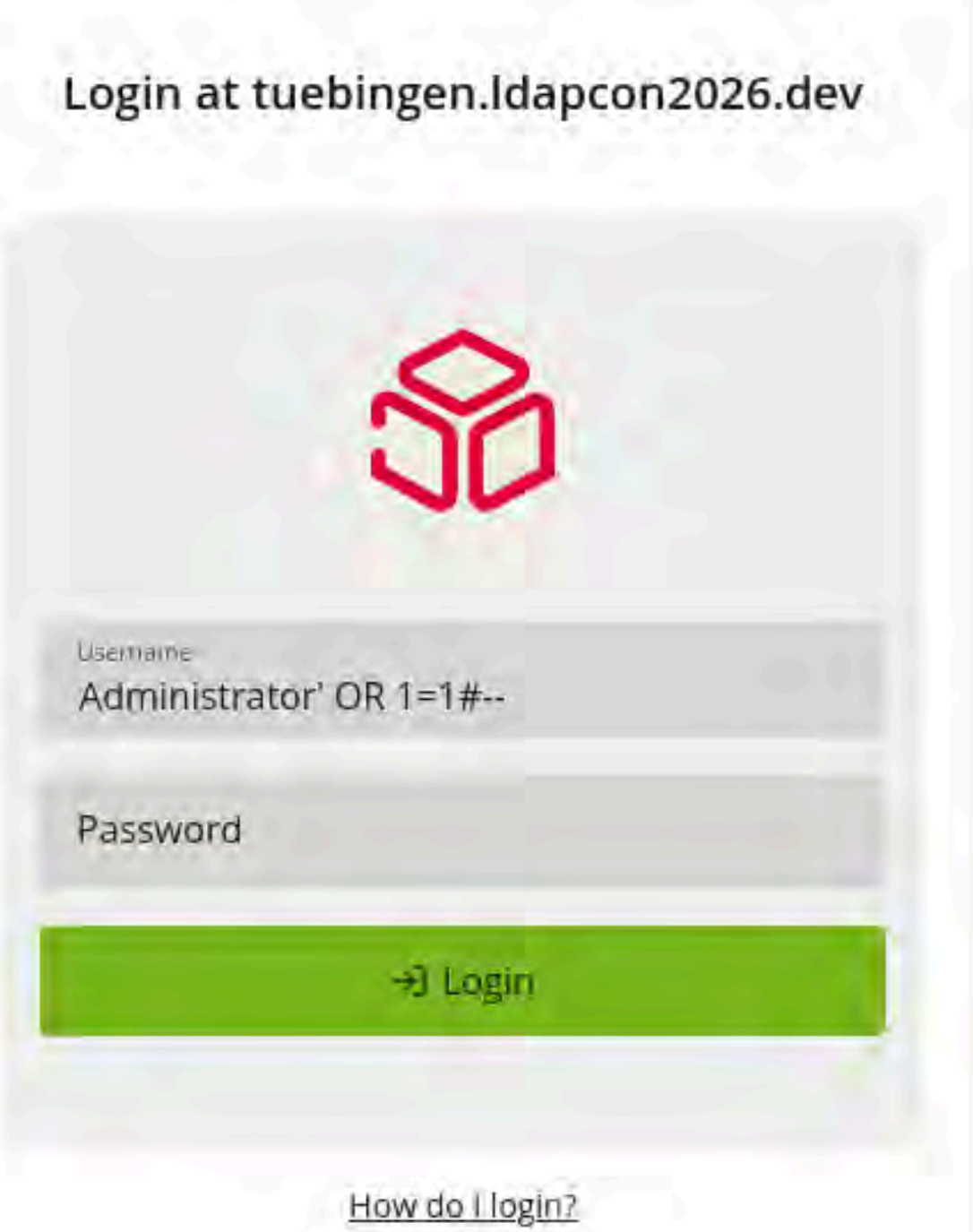
- **OpenLDAP** as the central directory
- **Keycloak** as Identity Provider & Single Sign-On
  - OpenID Connect & SAML
  - centralized authentication & sessions
  - LDAP identity federation
- **Windows & Active Directory integration**
  - **Samba 4** for Windows domain services
  - **AD-Connector** for synchronization with Microsoft Active Directory
  - **AD-Takeover** for migration from Active Directory

OpenLDAP rocks!

- OpenLDAP has been at the core of Univention and our product since 2002
  - It's been the reliable base for all that time. Thank you!
- LMDB is rock solid!
  - Since it was introduced as the Key-Value store backend,
    we don't have robustness or raw performance problems anymore!
- We are also using Samba 4 Active Directory functionality since 2011
  - Also with LMDB as backend, so that large environments work there, too.

Login required!

> `sqli://` *? There must be a better way…*



crudeoauth - OAuth 2.0 authentication at OpenLDAP

- SASL **OAUTHBEARER (RFC 7628)** mechanism
  - authenticate at LDAP server using OAuth 2.0 Access Token
    - e.g. `preferred_username` claim as SASL authentication identity
- usable by any SASL-enabled service (e.g. mail servers)
  - inspired by crudesaml for SAML
  - PAM module available as well
- recently re-written in Rust
  - previous OAuth C library is unmaintained → security concerns
  - Memory management aspects:

    > `C`: *remember to* `free()`
    > `Rust`: *we'd rather not let you double-*`free()`
    > `LMDB`: *don't* `free()` *at all, but* `mmap()`

- free as in bearer, not as in beer! Check it out:

https://github.com/univention/crudeoauth

MAPPING OF AUTHENTICATION IDENTITY:

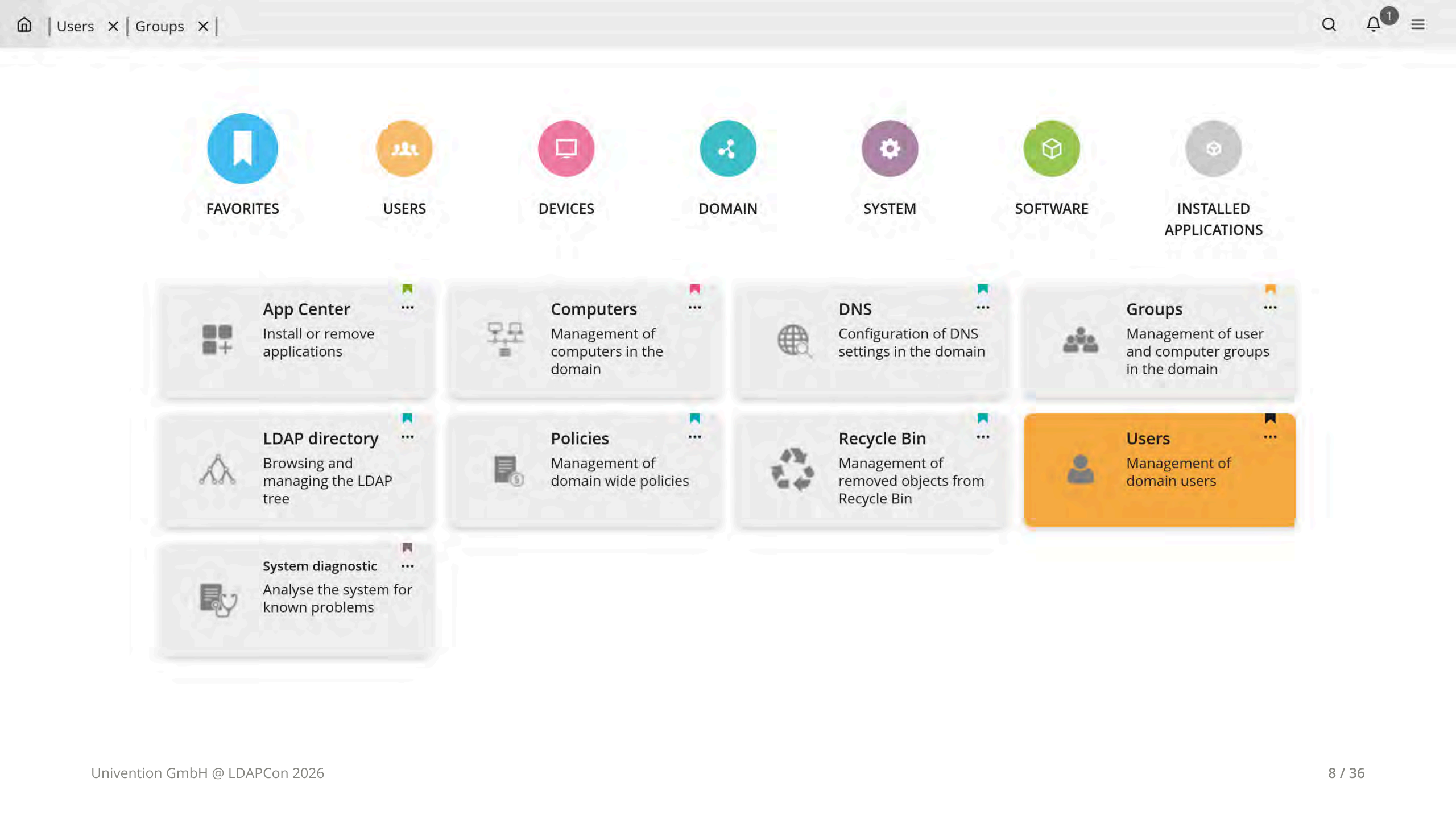
`/etc/ldap/slapd.conf`

```
authz-regexp
    uid=([a-zA-Z0-9]*),cn=oauthbearer,cn=auth
    ldap:///dc=example,dc=org??sub?(&(objectClass=person)(uid=$1))
```

CONFIGURATION OPTIONS:

`/etc/ldap/sasl2/slapd.conf`

```
mech_list: … OAUTHBEARER
oauthbearer_userid: preferred_username
oauthbearer_trusted_jwks0: /path/to/authorization_server_certs.jwks
oauthbearer_trusted_iss0: https://sso.example.org/realms/master
oauthbearer_trusted_aud0: ldaps://example.org/
oauthbearer_trusted_azp0: https:/client.example.org/oidc/
oauthbearer_required_scope0: openid
# oauthbearer_disallowed_username0: root
# oauthbearer_allowed_alg0: RS256
# oauthbearer_allowed_alg1: PS256
# oauthbearer_disallowed_alg0: HS256
# oauthbearer_disallowed_alg1: HS384
# oauthbearer_disallowed_alg2: HS512
# oauthbearer_disallowed_alg3: RS512
```


Users
Groups
1
FAVORITES
USERS
DEVICES
DOMAIN
SYSTEM
SOFTWARE
INSTALLED APPLICATIONS
App Center
Install or remove applications
Computers
Management of computers in the domain
DNS
Configuration of DNS settings in the domain
Groups
Management of user and computer groups in the domain
LDAP directory
Browsing and managing the LDAP tree
Policies
Management of domain wide policies
Recycle Bin
Management of removed objects from Recycle Bin
Users
Management of domain users
System diagnostic
Analyse the system for known problems

Users | Groups

Users

CLOSE

Search users...

+ ADD | EDIT | DELETE | ··· MORE

One user of 241 selected.

Name	Path
ute	ldapcon2026.dev:/tenants/People/Berlin
uwe	ldapcon2026.dev:/tenants/People/Bremen
veit	ldapcon2026.dev:/tenants/People/Bremen
vera	ldapcon2026.dev:/tenants/People/Tübingen
willi	ldapcon2026.dev:/tenants/People/Tübingen
wine	ldapcon2026.dev:/tenants/People/Berlin
xavier	ldapcon2026.dev:/tenants/People/Bremen
xena	ldapcon2026.dev:/tenants/People/Berlin
yvonne	ldapcon2026.dev:/tenants/People/Tübingen
zanna	ldapcon2026.dev:/tenants/People/Tübingen
zoltan	ldapcon2026.dev:/tenants/People/Berlin

Users > wine | Groups

Users > wine

CUSTOMIZE THIS PAGE | SAVE | BACK

Type: *User*
Position: *ldapcon2026.dev:/tenants/People/Tübingen*

- General
- Groups
- Account
- Contact
- Options
- OX App Suite
- Advanced settings
- Policies

UPLOAD PROFILE IMAGE

REMOVE

Basic settings

User account

Title

First name: Wilhelmine

Last name *: Winkelmann-Wüstig

User name *: wine

Description: Wilhelmine Winkelmann-Wüstig - Sekreta ...

Password

Password (retype)

☐ Override password history

☐ Override password check

Primary e-mail address (mailbox)

mail@example.org

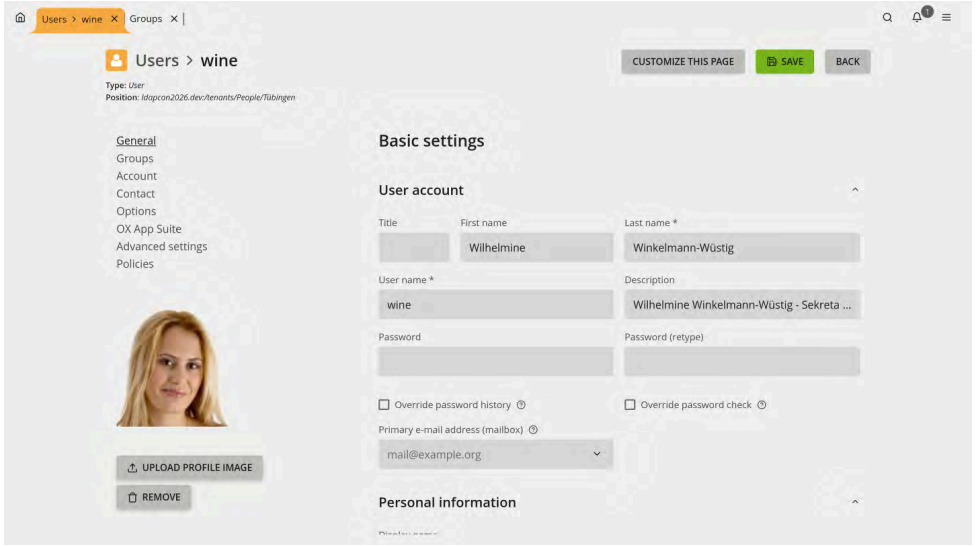
Personal information

Display name

Univention Directory Manager (UDM)

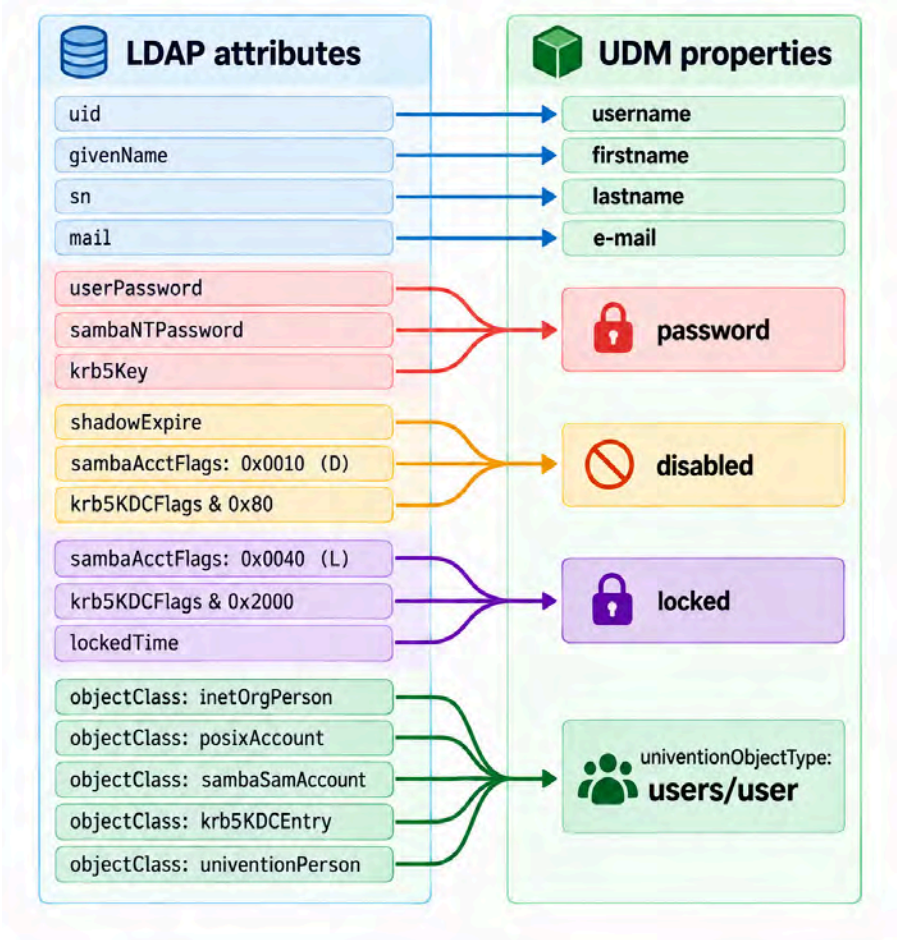
> *make it as easy as possible to administrate the directory*

1. Web user interface - for interactive administration
2. CLI - for scripting and command-line administration
   - `udm "users/user" modify --dn "$dn" --set city="Tübingen"`
3. HTTP REST API - for automation & integration



- Library written in Python
  - a bit like an **Object Relational Mapper** (ORM) for LDAP
- enriches **LDAP metadata**: human readable labels, descriptions, tooltips, help texts, internationalization
- enforces application-level **consistency**
  - referential integrity
    - non-DN relations, e.g. computer ↔ DNS via `aRecord`
  - locking & uniqueness
  - revert on failure (no LDAP transaction support yet)
  - stricter value sanitization (> DIT content rules)
- **extensible** by apps and customers
  - behavior (schema, hooks, types) for specific use-cases
  - views (widgets, layouting)

UDM user objects: LDAP schema made human



- UDM exposes **friendly property names**
  - `username` instead of `uid`
  - `firstname` / `lastname` instead of LDAP schema names
- UDM unifies multiple LDAP object classes into **one object type**: e.g. `users/user`
- One UDM object combines several LDAP worlds: **LDAP Person**, **POSIX**, **Kerberos**, **Samba/AD**, **Mail**, **Univention extensions**
  - Yes, `inetOrgPerson` has `faxsimileTelephoneNumber` - but still no birth date in 2026
- UDM keeps states and password hashes in sync!
  - **multiple LDAP attributes**:
    - `password`, `disabled`, `locked`
    - patched Heimdal + Samba
    - custom LDAP overlay `shadowbind`, `k5pwd`, `pwd_scheme_kinit`
    - AD/S4-synchronization

Technical Challenges

> `dn1 == dn2` - *what could possibly go wrong?*

Distinguished Names (DN) comparisons

ARE THE FOLLOWING DISTINGUISHED NAMES EQUAL?

- `uid=Dirk T. Tree,cn=users,ou=Tübingen\+1,dc=freeiam,dc=org`
- `UID=Dirk T. Tree,CN=users,OU=Tübingen\+1,DC=freeiam,DC=org`
  → case insensitive attribute names
- `UID = Dirk T. Tree , CN = users , OU = Tübingen\+1 , DC = freeiam , DC = org`
  → trailing/leading whitespace (between names, values, separators)
- `UID = DIRK T. TREE , CN = USERS , OU = TÜBINGEN\+1 , DC = FREEIAM , DC = ORG`
  → if matching rule is case-insensitive
- `UID = DIRK T. TREE , CN = \55\53\65\52\53 , OU = TÜBINGEN\2B1 , DC = FREEIAM,DC=ORG`
  → hex-escaped values (must: separator chars e.g. `+` for multi-valued RDNs)
- `UID = DIRK T. TREE , commonName = \55\53\65\52\53 , OU = TÜBINGEN\2B1 , DC = FREEIAM…`
  → attribute alias name
- `UID = DIRK T. TREE , 2.5.4.3 = \55\53\65\52\53 , OU = TÜBINGEN\2B1 , DC = FREEIAM…`
  → numeric OID of attribute
- `UID = DIRK T. TREE ,2.5.4.3 = \55\53\65\52\53 , OU = "TÜBINGEN+1", DC = FREEI…`
  → multiple consecutive spaces in values can be stripped but quoted values from LDAP V2 are obsolete
- **YOU CANNOT RELIABLY NORMALIZE A DN WITHOUT THE SCHEMA!** - i.e. on client side. Unlike a URI.

Want to know more?: https://docs.freeiam.org/en/latest/examples/ldap_dn.html

User supplied DNs: to compare or not to compare?

EXAMPLE SCENARIOS:

- **Administrator:** types on CLI: Spelling differs when adding/removing group members
- **Samba4-Connector:** Automatic mapping of DNs
  - especially bad for when multiple values are synchronized: The whole operation aborts

TWO IMPERFECT APPROACHES:

1. Compare in Python: may disagree with OpenLDAP

```
$ udm "users/user" modify --dn "$dn" --remove groups="cn = Domain   users , cn = groups , dc = example , dc = org"
```

> ***WARNING:*** *cannot remove* `"cn = Domain   users , cn = groups , dc = example, dc = org"`
> *from groups, value does not exist*

2. Pass through unchanged: may fail in OpenLDAP

```
$ udm "users/user" modify --dn "$dn" --append groups="cn = domain   users , dc = example , dc = org" --set city="Tübingen"
```

> ***LDAP Error: Type or value exists:*** *value #1 provided more than once*

or:

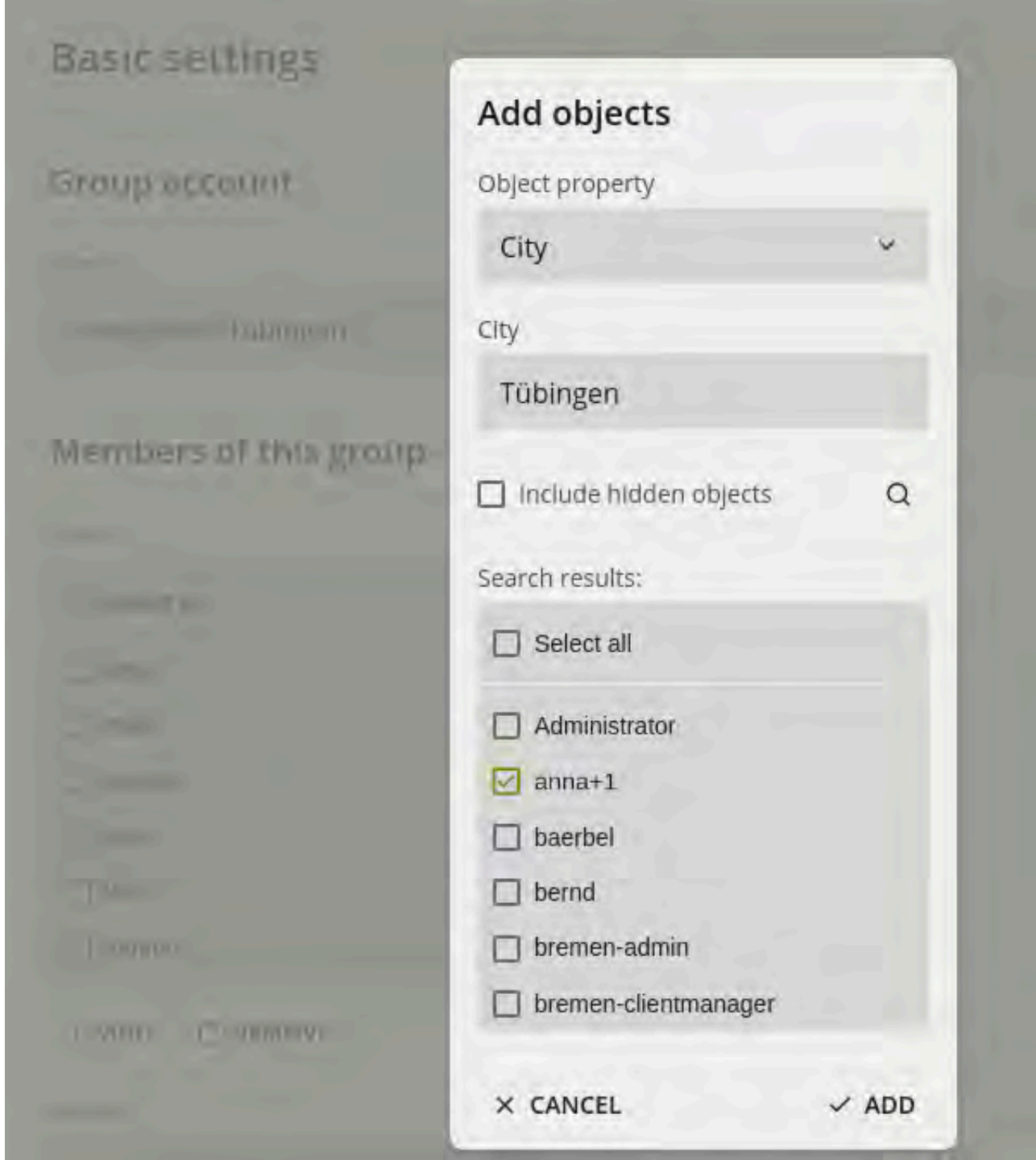
> ***LDAP Error: No such attribute:*** *modify/delete: uniqueMember: no such attribute*

Offer possible choices for group memberships in a Web UI

- Group values contains `uniqueMember` from OpenLDAP
- Selection dialog searches `DN` for possible choices:
  - optionally on top: python-ldap-normalized
- `uniqueMember: cn=Anna\2B1` vs `dn: cn=anna+1`
  - string mismatch on selection
- WORST CASE: → silently dropped group membership on saving

2 PROBLEMS

- OpenLDAP syntax-normalizes `uniqueMember` values, but does not schema-normalize them:
  - stored DN may differ in case from the actual entry's DN.
- Python-LDAP normalization differs from OpenLDAP normalization



Towards reliable DN comparison

NON-SOLUTIONS

- ❌ Reimplement schema-based comparison on application level
  - normalization functions for all matching rules - too error prone!
- ❌ Use LDAP `compare()` operation
  - can *only* compare against existing entries
  - need for path traversal of all elements
- ❌ Split modification into multiple LDAP append operations
  - doesn't solve the underlying problem

PRAGMATIC APPROACH

- Expose `libldap` normalization in `python-ldap`:
  - `dn2str(dn, flags=…)`
    `flags=LDAP_DN_FORMAT_LDAPV3`
  - normalizes only syntax!
    - not schema / aliases / syntax / matching-rules
- 🥳 Finally in `python-ldap 3.5.0`
  - Issue #257 requested 2019-01-18
  - PR !466 provided 2022-04-01
  - merged to main: 2025-08-07
  - release date: 2026-10-??
  - in Debian stable: 2029-??-?? 🤡

A SOLUTION

OpenLDAP could provide:

Extended operation for LDAP DN comparison

- implemented as an overlay
- `compare_dn(dn_a, dn_b)`
- `dn_intersection(l_dn_a, l_dn_b)`
- doesn't scale for all use-cases
  - lazy local comparison first
- **To Be Implemented!**

Recycle Bin

> `ldapdelete` … *now undo it.*

Recycle Bin: Implementable with built-in OpenLDAP features?

HUMANS MAKE MISTAKES - AND NEED TO RESTORE DELETED OBJECTS

- **Preserve deleted objects in a inactive form**
  - strip the original object classes; store as `objectClass: extensibleObject`
    - prevents authentication and normal object-class-based searches
  - use UUID as RDN (`univentionObjectIdentifier=43026e47-f1e8-4e85-…`)
    - stable and easy to handle
  - move to a separate database: `cn=recyclebin,cn=internal`
- **Automatic expiry**, e.g. after 90 days
  - use LDAP's `dynamicObject` object class
  - retention controlled by the operational attribute `entryTtl`
- **Preserve references for later restoration**
  - `univentionRecycleBinReference: dn:groups/group:users:uuid:550e8400-e29b-41d4-a716-446655440000`
  - meaning: restore the deleted object's DN into the property `users` of the `groups/group` object identified by `uuid=550e8400-…`
  - UUID based: allows rename of group between remove and restore of user

Recycle Bin: Security-sensitive aspects

- Objects **may be deleted by managed nodes** or OU-specific admins
  - LDAP ACLs allow them to delete certain objects
  - but they **must not be allowed to create or modify recycle-bin entries**
- Objects are **moved** to the recycle bin **by an asynchronous listener**
  - problems: stale caches, aggregated modifications, race conditions
  - **security decisions happen outside** the original LDAP operation
- **Arbitrary writes to the recycle-bin** database could enable privilege escalation
  - e.g. injecting a membership in `Domain Admins` before restoring → **domain takeover**
  - same for: allowing arbitrary writes during restore

PREFERABLE DESIGN

Make it synchronous and server-side

- **LDAP Control on Delete**
  - `moveToRecycleBin: TRUE`, `retentionTime: 90d`
  - authorize using the caller's **delete permission**
  - preserve the entry internally using **root DN privileges**
- **Extended Operation for restore**
  - `restoreFromRecycleBin`
  - recycle-bin ID/DN + optional `targetDN`
  - authorize the operation without granting arbitrary write access to stored entries

Performance

> *OpenLDAP is the world's fastest and most efficient directory software*

Performance: Virtual List View Benchmark

Implement pagination for UDM REST API & Web UIs

LDAP Search Control	Page Size	Result	Runtime	Memory / Notes
				Test environment: 200.000 users; 8 GB RAM; sort by `uid:caseIgnoreOrderingMatch`
Normal search	200.000	Successful	6.125 s	
Virtual List View + Server Side Sorting	50	**Aborted**	**3 - 4 min**	**Out of memory**
Virtual List View + Server Side Sorting	2000	Successful	1m 36.634s	`slapd` memory: 65.7%; all sorted VLV pages retrieved

> *OpenLDAP is the world's fastest and most efficient directory software?*

- **Luckily, just a bug in the *python-ldap* client library!**
  - context ID of the previous page not passed along in the request
  - each page-request started a new search, loading whole result set into RAM
  - we fixed it (`python-ldap` 3.4.8 - PR 618)
    - unnoticed since 11 years

Performance: Virtual List View & Simple Paged Results

OpenLDAP is the world's fastest and most efficient directory software!

LDAP Search Control	Page Size	Result	Runtime	Memory / Notes
Normal search	200.000	Successful	6.125 s	
Simple Paged Results	50	Successful	10.394 s	
Simple Paged Results	**2000**	Successful	**6.248 s**	Performance close to normal search - very good!
Virtual List View + Server Side Sorting	50	Successful	36.942 s	`slapd` memory: 22%;
Virtual List View + Server Side Sorting	**2000**	Successful	**10.325 s**	`slapd` memory: 22%; Performance very good

- Virtual List View:
  - ❌ is bound to the connection: incompatible with stateless multiprocessing
  - ❌ risk of DoS: hard to define safe limits for a generic product
  - ❌ how to handle reached limits when availability is critical?
- Simple Paged Results:
  - ❌ doesn't provide total count, or possibility for backwards navigation
  - ❌ doesn't survive slapd restart: next fetch is a complete new search - no client detection possible → endless loop
  - ❌ stability guarantee - concurrent modifications between receiving pages?

Performance: `memberOf` vs `dynlist` (in XXL-environment)

DYNLIST

- dynamically compute **virtual** attributes
- during `read`
- provides functionality *similar* to `memberOf`

PERFORMANCE:

- **12 seconds** for `memberOf` calculation (without nested groups)
- **21 seconds** for `memberOf` calculation (**with nested groups**)

→ For read-optimized databases, it makes sense to compute group membership at write-time instead

MEMBEROF

- **store backwards-references** at objects
  e.g. group memberships at users
- on each `write` operation
- read therefore is pre-calculated

PERFORMANCE:

- **0.248 seconds** to retrieve `memberOf` values
- nested group search via `LDAP_MATCHING_RULE_IN_CHAIN`
  - OpenLDAP overlay prototype
  - in our GitHub repo: univention/openldap PR #1

OpenLDAP ITS 9795 - Remove memberof overlay

- OpenLDAP 2.7 almost dropped the `memberof` overlay module
  - Prevented by our comment

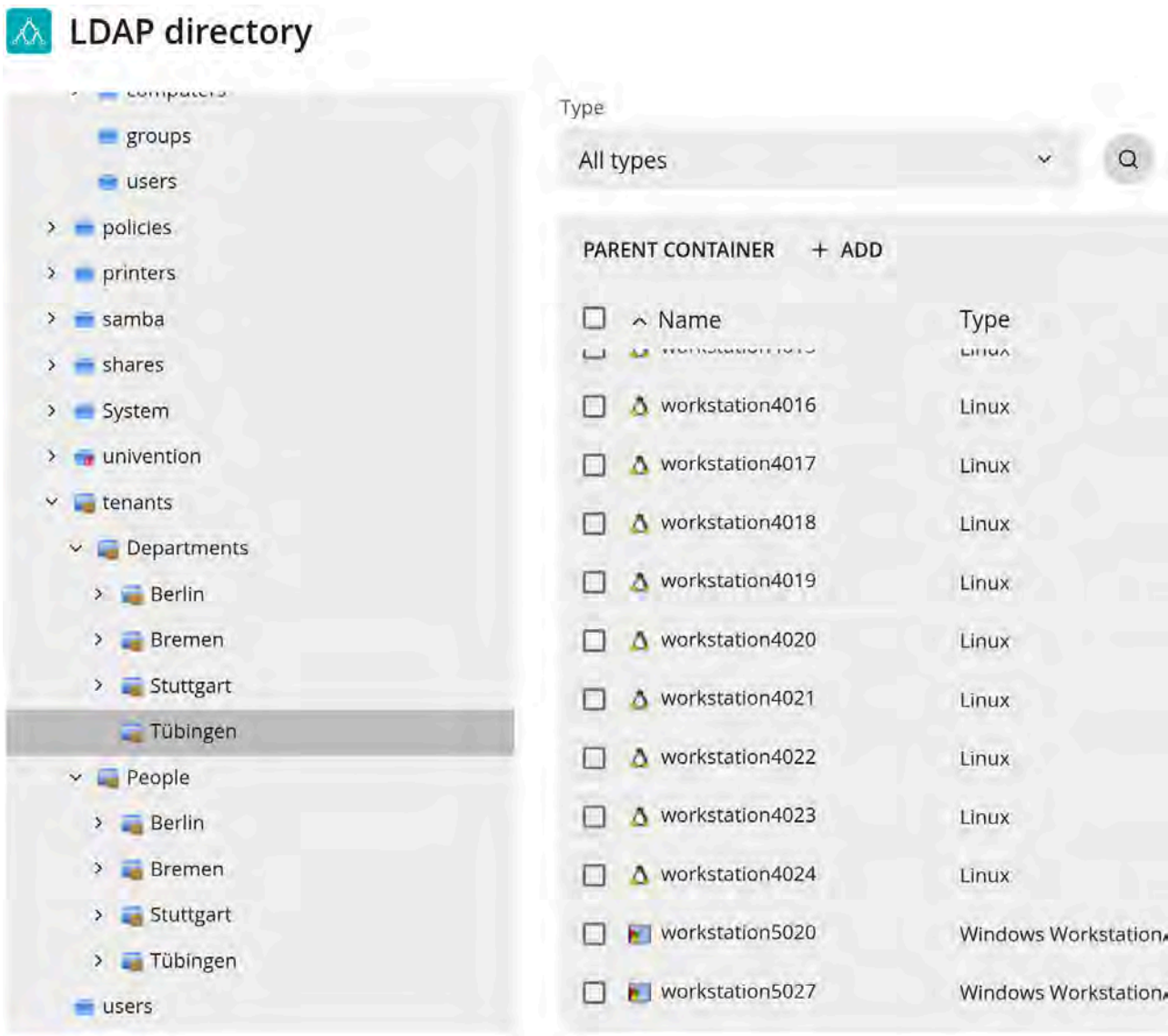
Thank you for keeping memberOf inside of OpenLDAP!

- **Test environment:**
  - 200.000 users
    - 99 groups / user + 1 XXL group
  - 20.000 groups
    - 99 groups with 200.000 members each

Performance: LDAP ACL configuration for distributed OU structures

`WRITE` RULES HURT READ PERFORMANCE

- Experiments with **multi-tenant OU structures** showed:
  - only a few additional ACLs can severely degrade LDAP performance
  - `write` (= `+wrscx`) **includes** the right to `read`
  - → `read` / `search` requests have to evaluate **write-specific ACLs**
- UCS@school is a real-world stress case:
  - **80 `access to` blocks** (21 from UCS core, 59 from UCS@school)
  - complex `dn.regex` target matching
  - regex capture groups + `.expand`
  - `set` ACLs using `user`, `this` and LDAP URIs



Our LDAP ACLs look like this (page 1)

```
# grant at least read access to school Replica Directory Nodes to operational attributes, if not limited later on
access to attrs="structuralObjectClass,entryUUID,creatorsName,createTimestamp,entryCSN,modifiersName,modifyTimestamp"
        by group/univentionGroup/uniqueMember="cn=DC-Verwaltungsnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" +rscxd break
        by group/univentionGroup/uniqueMember="cn=DC-Edukativnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" +rscxd break
        by group/univentionGroup/uniqueMember="cn=Member-Verwaltungsnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" +rscxd break
        by group/univentionGroup/uniqueMember="cn=Member-Edukativnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" +rscxd break
        by * +0 break

# grant write access to Replica Directory Nodes and Managed Nodes for certain univention app center settings
access to dn.regex="^univentionAppID=([^,]+),cn=([^,]+),cn=apps,cn=univention,{{ LDAP_BASE }}$"
  filter="(objectClass=univentionApp)" attrs=entry,@univentionApp,@univentionObject
        by group/univentionGroup/uniqueMember="cn=DC-Edukativnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by group/univentionGroup/uniqueMember="cn=DC-Verwaltungsnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by group/univentionGroup/uniqueMember="cn=Member-Edukativnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by group/univentionGroup/uniqueMember="cn=Member-Verwaltungsnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by * +0 break

# prevent unauthorized access to user password data

access to filter="(|(objectClass=ucsschoolAdministrator)(&(univentionObjectType=users/user)(!(objectClass=ucsschoolType))))"
  attrs=sambaNTPassword,userPassword,krb5Key,sambaPasswordHistory,pwhistory
        by self +0 break
        by group/univentionGroup/uniqueMember="cn={{ DOMAIN_ADMINS }},cn=groups,{{ LDAP_BASE }}" +0 break
        by set="user/objectClass & ([ucsschoolAdministrator] | [ucsschoolTeacher] | [ucsschoolStaff] | [ucsschoolStudent] | [ucsschoolLegalGuardian])" none
        by * +0 break

access to filter="(|(objectClass=ucsschoolStaff)(objectClass=ucsschoolTeacher))" attrs=sambaNTPassword,userPassword,krb5Key,sambaPasswordHistory,pwhistory
        by self +0 break
        by group/univentionGroup/uniqueMember="cn={{ DOMAIN_ADMINS }},cn=groups,{{ LDAP_BASE }}" +0 break
        by set="user/objectClass & [ucsschoolAdministrator]" +0 break
        by set="user/objectClass & ([ucsschoolTeacher] | [ucsschoolStaff] | [ucsschoolStudent] | [ucsschoolLegalGuardian])" none
        by * +0 break

access to filter="objectClass=ucsschoolStudent" attrs=sambaNTPassword,userPassword,krb5Key,sambaPasswordHistory,pwhistory
        by self +0 break
        by group/univentionGroup/uniqueMember="cn={{ DOMAIN_ADMINS }},cn=groups,{{ LDAP_BASE }}" +0 break
        by set="user/objectClass & ([ucsschoolAdministrator] | [ucsschoolTeacher])" +0 break
        by set="user/objectClass & ([ucsschoolStudent] | [ucsschoolStaff] | [ucsschoolLegalGuardian])" none
        by * +0 break

# Replica Directory Nodes need write access to the members of the group Domain Computers
access to dn.exact="cn=Domain Computers,cn=groups,{{ LDAP_BASE }}" attrs="uniqueMember,memberUid"
        by group/univentionGroup/uniqueMember="cn=DC-Verwaltungsnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by group/univentionGroup/uniqueMember="cn=DC-Edukativnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by * +0 break

# Replica Directory Nodes can read and write policy containers for MS GPOs and msPrintConnectionPolicy objects (part 1)
access to dn.base="cn=policies,cn=system,{{ LDAP_BASE }}" attrs=children
        by group/univentionGroup/uniqueMember="cn=DC-Verwaltungsnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by group/univentionGroup/uniqueMember="cn=DC-Edukativnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by * +0 break

# Replica Directory Nodes can read and write policy containers for MS GPOs and msPrintConnectionPolicy objects (part 2)
access to dn.children="cn=policies,cn=system,{{ LDAP_BASE }}"
  filter="(|(objectClass=msGPOContainer)(objectClass=organizationalRole)(objectClass=msPrintConnectionPolicy))"
  attrs=entry,children,{{ MSGPO_MSPRINT_CONTAINER }}
        by group/univentionGroup/uniqueMember="cn=DC-Verwaltungsnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by group/univentionGroup/uniqueMember="cn=DC-Edukativnetz,cn=ucsschool,cn=groups,{{ LDAP_BASE }}" write
        by * +0 break
```

ACL performance: some constructs are dramatically more expensive

`access to ...`

INVESTIGATION STILL ONGOING!

`<what>`	Relative cost
`dn.children`	1×
`dn.regex`, no captures	~7×
`dn.regex` + captures	~18×

`by ...`

`<who>`	Relative cost	Cache
`group/...`	1×	per operation
`set` : `user`	~23×	✗
`set` : `user` + `this`	~41×	✗
`set` + LDAP URI	~53–56×	✗

A **MUST READ** on `set` ACLs:

https://areq.gitlab.io/posts/2021-05-16-openldap-set-acls/

ANALYSIS OF OUR UCS@SCHOOL ACL STRUCTURE

```
access to
  dn.regex="^uid=...,cn=users,ou=(...),dc=..."
    attrs=entry,<user attributes>

  by <servers of this OU>    write
  by <other actor>         ...
  by *                       +0 break

  # Example <by> clauses:
  by group/univentionGroup/uniqueMember.expand="cn=OU$2-..."
  by set="this/ucsschoolSchool & user/ucsschoolSchool"
  by set.expand="([ldap:///]+user/entryDN+[?...])/ucsschoolSchool"
```

STRUCTURAL PROBLEM IN ACL DESIGN

- Extensible ACL stacks require:
  - continuation of ACL evaluation: `by * +0 break`
  - so that another rule can apply to the target
- EXTENSION PROPOSAL
  - Allow ACLs to be written from the actor's perspective
  - start with `access by` and then describe the targets

 Test-Environment: 4 vCPU; 12 GB RAM; 377k objects

Critical security implications

```
(&(objectClass=person)(|
    (mail=%s)(uid=
      Don't forget Escaping!)(uid=Administrator
    )
))
```

LDAP ACLs Quiz: will this `add` operation succeed?

```
access to dn.base="ou=managed,dc=example,dc=org" attrs=children
    by dn.exact="cn=server1,ou=servers,dc=example,dc=org" write
    by * none stop

# memberserver may only create "device" objects here
access to dn.onelevel="ou=managed,dc=example,dc=org" filter="(objectClass=device)" attrs=entry,@device
    by dn.exact="cn=server1,ou=servers,dc=example,dc=org" write
    by * none stop
```

```
$ ldapadd -D "cn=server1,ou=servers,dc=example,dc=org" -w "secret" <<EOF
```

```
dn: uid=alice,ou=managed,dc=example,dc=org
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
uid: root
cn: root
uidNumber: 0
gidNumber: 0
homeDirectory: /root/
userPassword: {crypt}$6$H.05VD/DuPnyIoNC$yIJwYBZNWU4fkCV8SE0/
EOF
```

enable `add_content_acl`!

`man:slapd.conf(5)`:

> *`add_content_acl` on | off*
> *Controls whether Add operations will perform ACL checks on the content of the entry being added.*
> ***This check is off by default.***

`man:slapd.access(5)`:

> *Also if Add content ACL checking has been configured on the database [...], add (=a) will be required on all of the attributes being added.*

→ Now go, check your config and add some DIT Content Rules!

https://www.openldap.org/faq/data/cache/1473.html

Example LDAP ACL which restricts access to add only the objectClass "inetOrgPerson"

- **Role-based** AUXiliary object classes:
  - `schoolAdministrator`
  - `schoolTeacher`
- You are a teacher
- **Self Service**
  - allows to modify yourself
  - via write `@inetOrgPerson` (all attributes)

VULNERABLE ACL:

```
access to filter="objectClass=person" attrs="entry,@inetOrgPerson"
    by self write stop
    by * +0 break
```

- → allows to raise privileges from `schoolTeacher` to `objectClass=schoolAdministrator`

HOW TO DO IT INSTEAD?

- 3 complex clauses required:
  - (or list all allowed attributes explicitly)

```
access to filter="objectClass=person" attrs="=objectClass" value="inetOrgPerson"
    by self write stop
    by * +0 break

access to filter="objectClass=person" attrs="objectClass"
    by self +0 stop
    by * +0 break

access to filter="objectClass=person" attrs="entry,@inetOrgPerson"
    by self write stop
    by * +0 break
```

- DIT content rules
  - can restrict object class combinations
    - if suitable

Centralizing authorization: Delegative administration

ACL configuration is complex, performance-critical, security-critical

- Different authorization settings for our components
  - OpenLDAP: LDAP ACLs
    - UDM `Properties` vs LDAP `Attributes`
  - Univention Web UI: "UMC operation sets & policies"
  - Portal & UDM REST API: UCR configuration
- Administrators don't want to learn all of these systems

→ We are currently in the process of centralizing authorization

- Administrators can configure policies with one unified concept
- Application services perform the authorization themselves
  - using fully privileged service accounts for LDAP writes
  - New problems for developers 🥳

UDM delegative administration: Inventing our own policy language

- For our policy model, we want to author rules actor-first (`access by`) rather than target-first (`access to`)

```
# Organizational Unit Administrators
access by role="udm:organizational-unit-admin"

  to objecttype="users/user" position.subtree="context=udm:position"
    description="Write user objects in own OU"

    grant actions="search,read,create,modify,rename,remove,move,create-report"
    grant properties="firstname,lastname,username" permission="write"
    grant properties="password,serviceSpecificPassword" permission="writeonly"
    grant properties="roles" permission="none"
    grant properties="*" permission="write"
    grant properties="groups"
      if-values='
         ldapDNList(OBJECT_PROPERTIES.groups)
         .intersects(["cn=Domain Admins,cn=groups,dc=example,dc=org",])
      ' permission="none"
```

Developer pain: Comparing DNs

- Be careful about comparing DNs in security contexts!
  - Deny-lists can easily be tricked if your comparison is broken - and **it definitely is!**
  - remember: `2.5.4.3=Domain Admins,cn=groups,dc=example,dc=org` compares equally

```
access by … to …
  grant properties="groups"
    if-values='
       ldapDNList(OBJECT_PROPERTIES.groups)
       .intersects(["cn=Domain Admins,cn=groups,dc=example,dc=org",])
    ' permission="none"
```

- Only allow the use of Allow-lists:

```
ldapDNList(OBJECT_PROPERTIES.groups).subsetOf([
  "cn=Domain Users,cn=groups,dc=example,dc=org",
  "cn=Tübingen,cn=groups,dc=example,dc=org",
])
```

TL;DR: My personal LDAP extension proposals

BASIC

- Apply `ppolicy` to container subtree, inherited by contained objects, with optional LDAP filter matching
- Alternative ACL authoring order: actor (`by`) before target (`to`)
- `LDAP_MATCHING_RULE_IN_CHAIN` (`1.2.840.113556.1.4.1941`) for nested group search (consider adopting our reference implementation)
- Extended operation for schema-aware LDAP DN comparison
- Connection-independent Virtual List View context-ID
- Multiple transactions per connection (the specs support TXN-ID)
- `Set`-ACL caching
- Tune large multi-valued attributes for speed - especially `uniqueMember`
- Per-attribute metadata for bidirectional synchronization (conflict resolution)
- Extended operation / control for efficient result counting (LDAP equivalent of `SELECT COUNT(*)`)

MORE FUTURISTIC

- Searchable structured values e.g. JSON syntax with matching rules
- Support for non-DN/hierarchical relationships - including referential integrity
- Impersonating the other way round: authz-id should write in the name of the authc-id (or auditable metadata)

SIMPLY ANNOYING

- Overlay module build independent of OpenLDAP build process (exposed headers, loadable shared objects)
- No configuration-file line-length limit
- Attribute-scoped Relax control e.g. only `entryUUID`
- Modernize LDAP schemas - common attributes and appropriate matching/ordering rules
- No direct way to grant writing all attributes of an object class without allowing to add arbitrary AUX `objectClass`es
- IA5 Ordering Matching Rule for ASCII fields
- `python-ldap`: More releases, async, transaction context manager

univention

THANK YOU!

Time for discussion...

(or contact me afterwards)

Florian Best
Open Source Software Engineer
Univention GmbH
best@univention.de
https://github.com/spaceone/

Everything from this talk in one place:

We're hiring!

Are you interested in solving LDAP problems?

Apply as Open Source Software Engineer at Univention GmbH

Send a speculative application or apply at https://www.univention.com/about-us/careers/

be open.