



Directories and directory implementation

Introduction to directory implementation and the art of attributes

Miroslav Milinovic, University Computing Centre, University of Zagreb

Jasmina Hodzic, Center for Information Technology Services, University of Oslo



Directories

- What is a directory
- Building directories
- Building blocks
- What a directory can be used for



Directory concept

- A directory is a specialized, structured database optimized for read access
 - Directories are usually updated (written) less frequently than they are accessed (read/searched)
- **L**ightweight **D**irectory **A**ccess **P**rotocol is used to access data in directories (a “simplified” version of X.500)
- Directories may be
 - Centralized (one location for data, common root)
 - Distributed (data located in different instances)



Directories

- A directory has:
 - Entries (objects)
 - Which consist of attributes
 - And are referred to and identified by distinguished names
 - A schema (rules that determine structure and contents)
 - objectClass (determines what attributes are required and allowed)
 - Operational attributes (time of creation for an entry etc)
- Directories are usually hierarchically structured to represent the organizational (or indeed any other) frame/boundary



distinguishedName

- Distinguished name for an entry consists of the name of the entry and the names of all the objects hierarchically places above the entry (ordered from bottom to top)
- Example:
 - Top: dn: dc=uiio,dc=no
 - Department: dn: dc=DEP,dc=uiio,dc=no
 - Student: dn: dc=student,dc=DEP,dc=uiio,dc=no
- Distinguished names are used for optimization of read/search



Schema

- Defines the type of entries allowed, their attribute structure and syntax of attributes
- Provides the necessary objectClasses (the ones your directory needs in order to do its job)
- Helps you maintain data integrity
- In short, the schema of a directory determines what kind of data you can store and implicitly what the directory may be used for
- Any number of compliant schema may coexist within a directory



objectClass

- Specifies the set of attributes that describe an object
- Is defined by type, inheritance and attributes
- Can be:
 - Structural
 - Abstract
 - Auxiliary
- You can add your own object classes to represent entries you need
 - But you must follow some rules



Structural objectClasses

- Most common type of objectClass
- Define base contents of an entry
- Every entry must belong to one structural class
- Represents a real world object
- Examples:
 - objectClass: person
 - objectClass: org



Abstract objectClasses

- Used as templates for structural classes
- Define attributes common to a set of structural classes
- Subclasses inherit the common attributes
- Example:
 - objectClass: top



Auxiliary objectClasses

- Defines additional attributes an entry belonging to a particular structural class may have
- An entry may belong to multiple auxiliary object classes
- The core attributes are inherited from the structural classes
- Example:
 - objectClass: posixUser



Inheritance

- All object classes inherit the root class (“top”)
- The inheritance depends on the sequence of class definitions
- Can only inherit from classes that precede it
- Example, person entry in LDAP:
 - objectClass: top
 - objectClass: person
 - objectClass: organisationalPerson
 - ObjectClass: inetOrgPerson



Adding objectClasses

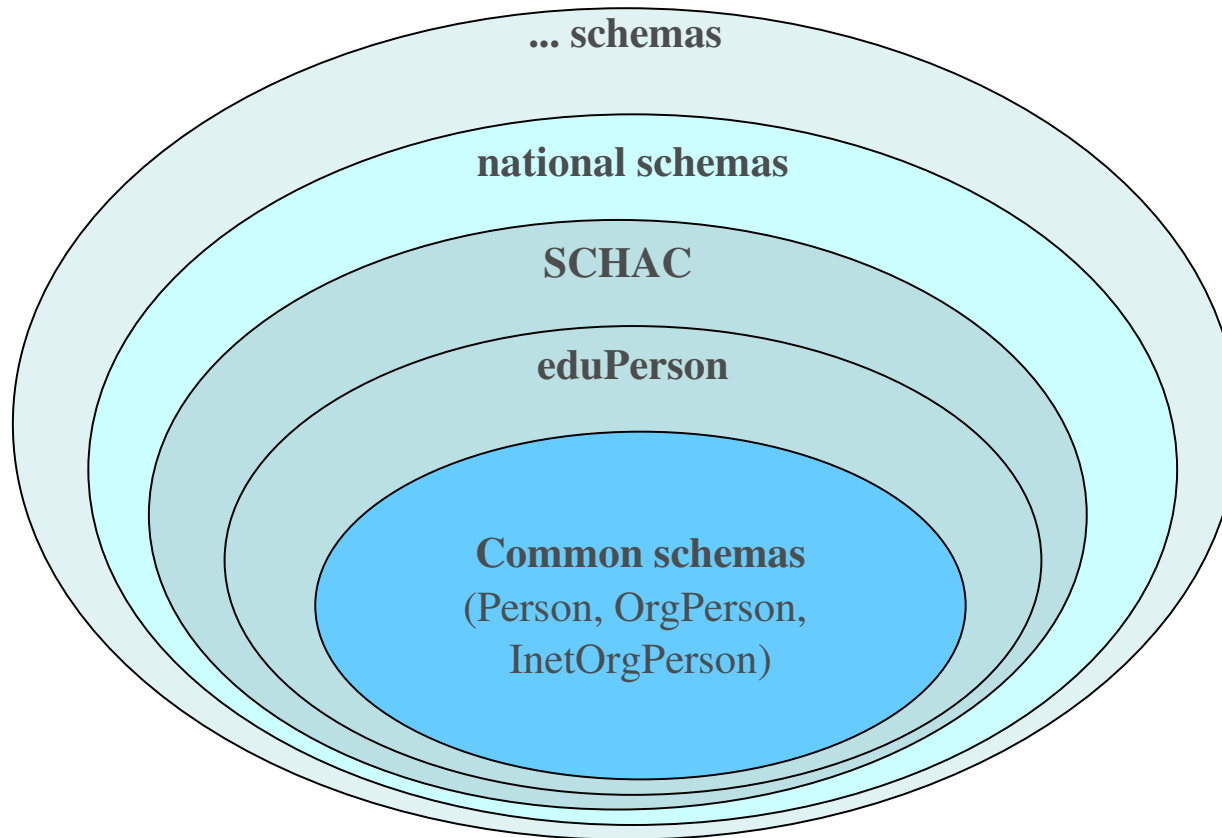
- Check and double-check if you really need to add a new object class (there are quite a few available out there)
- Get an OID assignment for your institution from IANA
- Create new objectClasses for new attributes
- DO NOT make up or reuse an OID
- DO NOT modify a standard objectClass
- DO NOT populate standard attributes in non-standard ways



Directory schema

- schema defines the attribute structure, syntax and semantics
- the tricky part:
 - value space (vocabularies used)
 - semantics
 - having common attribute syntax and vocabulary is not enough
- what does “faculty” mean?
 - ... at the institutional level?
 - ... at the national level?
 - ... at the international level?

Schema layers





The art of attributes

- Always use the original schema's:
 - syntax, vocabulary, semantics
- If in doubt introduce new schema/attribute but do not change or misuse original specification
- Adding attributes or values is easy (?)
- Modification is hard



An example: EduPersonAffiliation

EduPerson Affiliation value	An interpretation (UK fed, technical recomm. draft)
Student	Undergraduate or postgraduate student
Faculty	Teaching staff
Staff	All staff
Employee	Other than staff or faculty (e.g. contractor)
Member	All above
Affiliate	Relationship short of full member
Alum	Graduated



More examples ...

- Entitlement (**eduPersonEntitlement**)
 - value is URI / URN
 - example:
urn:mace:washington.edu:confocalMicroscope
- Scope concept
 - **eduPersonScopedAffiliation**
 - Syntax: affiliation@domain
 - Example: faculty@cs.berkeley.edu



Mapping schemas/attributes

- (inter)institutional schema harmonisation – as a must/need (“killer apps”?)
- “schema onion” (common → eduPerson → SCHAC → ...schema)
- adding/changing attributes/values
- readable values vs. codes vs. opaque values
- redundancy vs. interoperability (fedPersonAffiliation)



A directory at the University of Oslo

- An OpenLDAP-implementation
- Accessed in average 3.6 milion times a day
- Populated from a metadirectory
- LDAP-implementation at UiO consists of several trees (with a common root):
 - organization (core, inetOrgPerson, eduPerson, eduOrg, norEduPerson, norEduOrg schema etc)
 - organization structure, people, groups
 - Serves UiO's whitepages
 - Federation backend



A directory at the University of Oslo

- Other trees
 - System
 - Traditional NIS information (uid, gid etc)
 - user, group and netgroup trees
 - Mail
 - Backend for UiO's e-mail system
 - Contains information about quota limits, spam settings and other relevant e-mail data
 - Provides authentication information for e-mail targets