The LDAP Directory Schema

a guide for the newcomer

Dr. Giovanni Baruzzi

giovanni.baruzzi@syntlogo.de

The LDAP Directory Schema
AGENDA

syntlogo

Why do we need a good schema? 1

- **A new schema can be made in few hours but will live years.**
  - The Schema will be used by many people.
  - For years. You will not be able to change it, only additions are possible.
- **You better think twice before releasing your design.**
- **A clean, minimalistic, functional Schema has an inner Aesthetic**

Why do we need a good schema? 2

syntlogo

- **If you are an infrastructure designer: preserve the ability to integrate new applications:**
  - Don‘t modify standard classes
  - Add your classes with your own prefix and OID
- **If you are an application designer, ask so few as possible to an directroy you have to integrate to.**
  - Make no assumption about existing attributes: your user may have modified them.
  - Mark your classes and attritbutes with your own prefix and OID

From White Pages to Access Control

- **The focus has shifted from the Telephone Book to the access control**
- **The new LDAP will support possibly a portal, with thousand of users**
  - Identification
  - Authentication
  - Authorisation
- **Those needs are mostly not covered by taday‘s standard.**

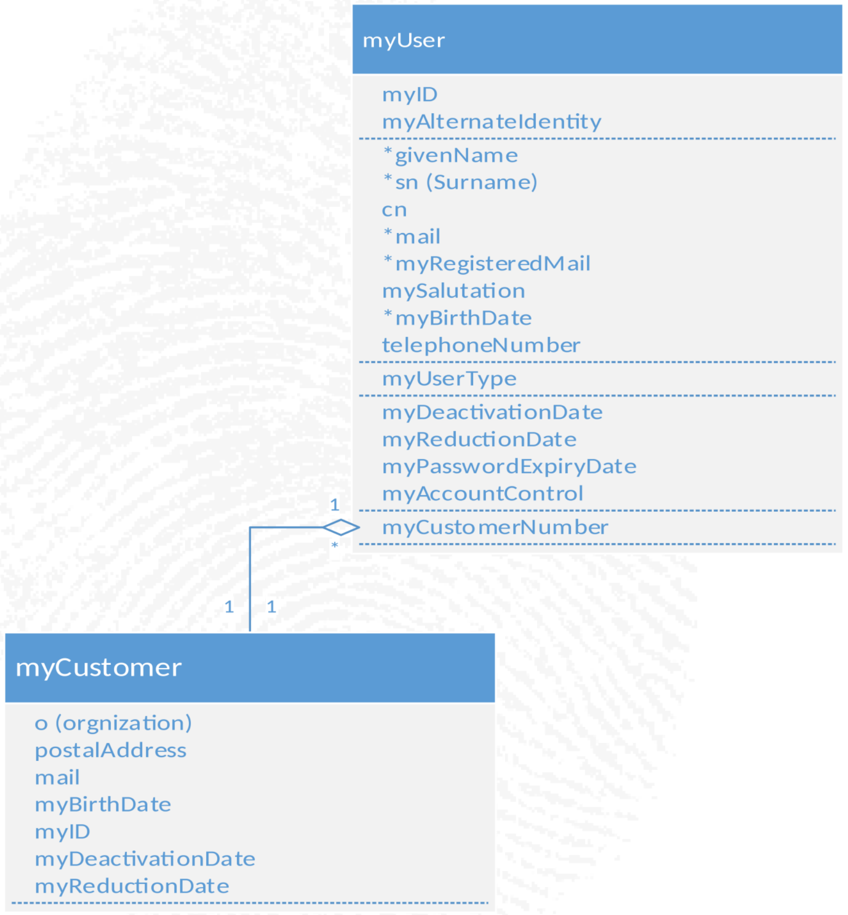
The Design Process 1 Overview

- **Gather the available information**
  - Consider only reliable, consistent, complete Information
- **Build a data model**
- **Organize your information in attributes and classes**
- **Try to implement your needs with standards**
- **For the rest, implement your custom Schema.**

Gather the available Information

- **Consider only reliable, consistent, complete Information (data Quality)**
- **Identify the authoritative source of an information**
  - Between two sources, choose the most next to the origin.
- **Agree on a fixed coding stype/form (Telephone Number, Building Identifier, Division Name and so on.)**
  - 2nd Floor, Room Numner 156 can be coded in sooo many different ways.

Build a data model



The standards available

- Syntaxes
- Attributes
- Classes
- Class types

Standard syntaxes

Syntax Encodings	Description
Directory String	UTF-8 String. The most used Encoding
Generalized time	Very important encoding for data & time attributes. Allows comparison.
Numeric string	a sequence of digits
Boolean	TRUE/FALSE
Octet String	Binary value
Other Syntaxes	
DN	Distinguished Name.
Boolean	TRUE/FALSE
Telephone Number	very important and in widespread use
Postal Address	a number of strings separated by a dollar “$” sign.
Numeric string	a sequence of digits
IA5 String	a string of ASCII characters

Standard Attributes

Name of the attribute	Description
objectClass	Needed for the every object for the object class definition.
cn	Common name: generic name of the object. For a person's object it may be the complete name of given Name and family name.
sn	Surname. The family name of a person.
c	country: the ISO designation of a country: IT, DE, UK etc.
l	Locality.
o	Organization's name. As this is often used as a container.
ou	Organisational Unit very often used as a container.
description	Free text.
postalAddress	Postal Address is the Address as printed on envelopes: a set of lines. The single lines are separated by a dollar sign.
telephoneNumber	Telephone number, normally coded including the international prefix.
member	Used for the object class "group", it contains only the DN of a member. The attribute is multivalued.
uniqueMember	Multivalued attribute to represent members in a "groupOfUniqueNames".
userPassword	Very special attribute!

Standard Classes

Name of the object class	Description
person	the basic objectClass for person
residentialPerson	an extension for private persons
OrganizationalPerson	extended for persons in an organisation
inetOrgPerson	the most used objectClass to represent an user
groupOfNames	a collection of objects. widely used to assign a right.
groupOfUniqueNames	Sam as above with control about doublettes
organization	An object to represent an organisation, used mostly as a container for other objects.
organizationalUnit	As the objectClass “organization”, in use to give a structure to a directory information tree. Seldom used to describe a division or a real organization.

Standard Class types

Name of the object class	Description
Abstract	a non-existent class, needed for logical completness. The most common ABSTRACT objectclass is top
Structural	can be used to create entries and can be part of a class hiearchy
Auxiliary	may be added into any convenient entry to add possible and mandatory attributes

The LDAP Directory Schema AGENDA

- –Why do we need a good schema?
- –From the White Pages to Access Control
- –The Design Process
- –The available Standards
- –Best Practices
- –The User object
- –The plumbing
- –Implementing the Schema over the time

Best practices 1

- **Remember that a schema is to be read and understood by many people beyond your department and you should not assume that they have direct access to written documentation.**
  - Design a Schema that is intuitive to undertand
  - Spend enough time trying to find the right name for an attribute. Good names are an advantage for everybody.

Best practices 2

- Use a short prefix for the name of your attributes to easily identify them.
- Use an OID number if the project has a scope beyond a single organisation.
- the attribute name is being written in the directory for every instance of the attribute, so length matters.

Bet Practices 3

- Avoid names like: “xdTelephoneNumberPrimaryTieNumber”
- For complex, long and recurring concepts use an abbreviation and capitalize it: instead of “xdFacsimileTelephoneNumberCountryCode” you may use “xdFAXCountryCode”

Best Practices 4

- Avoid mixing languages in the definitions. Instead of “xdDepartmentODIKurzbezeichnung” name the attribute “xdODIDeptShortName”
- When using English to name objects, be grammatically correct: the container for all groups objects cannot be “ou=group” but has to be “ou=groups”.
- Be careful not to be too generic: don’t call an Attribute “myStatus”: use instead “myEmploymentStatus”.
- On the other side, avoid overengineering your names.

Best Practices 5

- **Define VERY WELL what you store in an attribute and HOW this will be stored. Format/conventions must apply throughout the Directory**
  - Consider regular expression matching

- TelephoneNumber examples:
  - 0049 (0) 7031 87 95 52
  - +49 7031 879552
  - +497031879552
  - +49 7031 87 95 52

Best Practices 6

- **Information to include in the directory**
  - Personal Contact Informationlike Name, Telephone Number, Address, email.
  - Descriptive information, such as an employee number, job title, manager or administrator identification.
  - Individual software preferences or software configuration information.
  - Contract or client account details
  - Home contact information

Best Practices 7

syntlogo

- **Information to exclude**
  - large, unstructured objects, such as images or other media.
  - information that is not needed IN THE DIRECTORY
    - it risks being redundant and the cost of maintaining it is oft unnecessary. Leave it at the source. Store the reference.

Best Practices 8

- **How can we keep related Information together?**
- **Let us imagine that we have to store a few postal addresses for an object. An address is made of the following information:**
  - 1) Street,
  - 2) House number
  - 3) Postal code
  - 4) city
  - 5) state
  - 6) country.

Best Practice 9

- **The flat option: define all the attributes that you need and add a number to the definition.**
  myAddress1Street
  myAddress1StreetNumber
  myAddress1PostalCode
  myAddress1City
  myAddress1State
  myAddress1Country
  myAddress2Street
  myAddress2StreetNumber
  myAddress2PostalCode
  myAddress2City
  myAddress2State
  myAddress2Country
  …..and so on….

Best Practice 10

- **The container Option: Define a container object for one Address and append as many of them as needed to the main object.**

Cumbersome and costly, but the best solution if the number of addresses is high.
You may even try innovative solutions if the addresses have oft duplicates: you may store them in a separate subtree and just reference them.

Best Practices 11

- **The structured Attribute Option: use an Attribute and define string inside it with separators (or an XML object). This may not be searchable, but it solves the problem with a minor effort, if you don’t have to share the solution with many partners.**

This is used by the standard attribute “postalAddress”, where a sequence of strings separated by a dollar sign represents the various lines of an address printed on an envelope.

The LDAP Directory Schema
AGENDA

syntlogo

- Why do we need a good schema?
- From the White Pages to Access Control
- The Design Process
- The available Standards
- Best Practices
- The User object
- The plumbing
- Implementing the Schema over the time

The objects of our Directory: User 1

The user is the most important object of all the directories.

- ***Typical information to identify a person:***
  - Name, Family Name
  - Email, address
  - Telephone Number
  - Post address
  - Card ID, Social Security Number, Tax ID

The objects of our Directory: User 2

- ***Relationship with the Organisation***
  - Personal number
  - Customer number
  - Organisational unit
  - Manager
  - Sales representative
  - Organisational role
  - Secretary
  - Cost centre

The objects of our Directory: User 3

- ***Contracts and agreements***
  - Job description
  - Assigned task
  - Assigned Projects
  - Contract number
  - Exemptions
  - Entitlements
  - Allowances

The User Object 1

- Plan to use the inetOrgPerson structural class as base class
- Extend it with an AUXILIARY class myUser

The User Object 2

- **The „Customer Number“**
- this attribute does not exist in the richest standard class the „inetOrgPerson“.
- Name: we are going to use our prefix „my“ and the result is quite direct: myCustomerNumber.
- Syntax: „Directory String“ with the „Case Ignore String“ as matching rule.

The User Object 3

- **The “delegate Administrator” information**
  - First Choice: a Boolean attribute.
  - Second Choiche: a small string with a list of values: “01” is the delegated administrator.
    - you just use the following search (on more lines only for typographic clarity):
      “(&(objectclass=myUser)
      (myCustomerNumber=123456)
      (myUserType=01))”

The User Object 4

- **myld**

To identify the object across many different applications and systems

- a unique code and use it as a RDN and call it „myID“.
- This attribute is not designed to be seen or processed by human being,
- a few hexadecimal digits.
- being a unique identifier of the object, it is better that the attribute would be defined as „single Value“.

The User Object 5

- **Some standard attributes**
- givenName,sn, cn, mail, telephoneNumber
- Why not? They are perfectly defined in purpose and gives us a reason to use a standard class.
  Let's imagine for a while that a standard application needs to access the directory to accomplish its duties: for example a white pages search: having the most standard attributes filled is a guarantee that the application could still fulfil its scope.

The User Object 6

- **Other custom attributes**
  - myRegisteredMail stores the Mail address verified at the registration.
  - mySalutation is very requested in many countries like Germany.
  - MyBirthDate is important as a further check of identity, especially in countries where people can change family name (Germany).

The User Object 7

- **Life Cycle Control**
  - myDeactivationDate
  - myReductionDate

Store the date when an account has been disabled (myDeactivationDate) and after six months to reduce it to a minimal form (myReductionDate), enough to use it in Auditing. After 10 years you may safely delete the object.

- Please note that the syntax is „generalizedTime“, a way to store year, month day, hours and minutes in reverse form („201511031230Z“) and appending the zone information. This Syntax allows us to compare dates.

The User Object 8

- **Password Control**
  - myPasswordExpiryDate
  - myAccountControl

store some flags into a „myAccountControl, The numeric value is converted in binary where the single bit has a specific meaning.

use the bit meaning used in ActiveDirectory

It is sometimes difficult to know when your password would expire. And every server does it in a different way. A good way to avoid the problem its to store the expiry date explicitly in a generalized time attribute called „myPasswordExpiryDate“.

The LDAP Directory Schema
AGENDA

syntlogo

- Why do we need a good schema?
- From the White Pages to Access Control
- The Design Process
- The available Standards
- Best Practices
- ➡ The User object
- The plumbing
- Implementing the Schema over the time

The Plumbing: OID 1

- An object identifier (OID) is a numeric string used to identify an object in LDAP. OIDs are used in schema, controls, and extended operations that require unique identification.
- After you have obtained a base OID, you can add branches to it for your organization's object classes and attributes.

The Plumbing: OID 2

- **E.G. the Syntlogo GmbH Enterprise Number to build the OID: 1.3.6.1.4.1.13299.**
  - The attributes will be 1.3.6.1.4.1.13299.1 and the counting on: 1.3.6.1.4.1.13299.1.2, 1.3.6.1.4.1.13299.1.3 and so on.
  - The object classes will receive and OID starting from 1.3.6.1.4.1.13299.2.1 and the counting.
- **To get you your own enterprise number, apply at the IANA: http://pen.iana.org/pen/PenApplication.page**

The Plumbing: Attribute definition 1

```
AttributeTypeDescription = "(" whsp numericoid whsp   AttributeType identifier
 [ "NAME" qdescrs ]                ; name used in AttributeType
 [ "DESC" qdstring ]                ; description
 [ "OBSOLETE" whsp ]
 [ "SUP" oid ]               ; derived from this other AttributeType
 [ "EQUALITY" woid                ; Matching Rule name
 [ "ORDERING" woid                ; Matching Rule name
 [ "SUBSTR" woid ]                ; Matching Rule name
 [ "SYNTAX" whsp noidlen whsp ] ; Syntax OID
 [ "SINGLE-VALUE" whsp ]             ; default multi-valued
 [ "COLLECTIVE" whsp ]               ; default not collective
 [ "NO-USER-MODIFICATION" whsp ] ; default user modifiable
 [ X-ORDERED whsp type ]             ; non-standard - default not X-Ord
["USAGE" whsp AttributeUsage ]     ; default userApp. ext.
 whsp ")"
```

The Plumbing: Attribute definition 2

```
attributeTypes: ( 1.3.6.1.4.1.13299.1.1 NAME ( 'myId' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.2 NAME ( 'mySalutation' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.3 NAME ( 'myBirthdate' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.24  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.4 NAME ( 'myRegisteredMail' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.5 NAME ( 'myAlternateIdentity' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
attributeTypes: ( 1.3.6.1.4.1.13299.1.6 NAME ( 'myUserType' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15   )
attributeTypes: ( 1.3.6.1.4.1.13299.1.7 NAME ( 'myDeactivationDate' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.24  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.8 NAME ( 'myReductionDate' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.24  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.9 NAME ( 'myAccountControl' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.6 )
attributeTypes: ( 1.3.6.1.4.1.13299.1.10 NAME ( 'myCustomerNumber' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.15  )
```

The Plumbing: Class definition 1

syntlogo

```
objectClasses: (1.3.6.1.4.1.13299.2.1 NAME
'myUser'
SUP top
AUXILIARY
MAY ( myId $ mySalutation $ myBirthdate $
myRegisteredMail $ myAlternateIdentity $
myUserType $ myDeactivationDate $
myReductionDate $ myAccountControl $
myCustomerNumber )
)
```

The Plumbing: change the Schema 1

```
dn cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( 1.3.6.1.4.1.13299.1.1 NAME ( 'myId' ) SYNTAX 1.3.6.1.4.1.1466.115.121.1.
 15 SINGLE-VALUE  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.2 NAME ( 'mySalutation' ) SYNTAX 1.3.6.1.4.1.1466.11
 5.121.1.15  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.3 NAME ( 'myBirthdate' ) SYNTAX 1.3.6.1.4.1.1466.115
 .121.1.24  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.4 NAME ( 'myRegisteredMail' ) SYNTAX 1.3.6.1.4.1.146
 6.115.121.1.26 SINGLE-VALUE  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.5 NAME ( 'myAlternateIdentity' ) SYNTAX 1.3.6.1.4.1.
1466.115.121.1.15  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.6 NAME ( 'myUserType' ) SYNTAX 1.3.6.1.4.1.1466.115.
 121.1.15   )
attributeTypes: ( 1.3.6.1.4.1.13299.1.7 NAME ( 'myDeactivationDate' ) SYNTAX 1.3.6.1.4.1.1
 466.115.121.1.24  )
-
```

The Plumbing: Changing the Schema 2

```
cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( 1.3.6.1.4.1.13299.1.8 NAME ( 'myReductionDate' ) SYNTAX 1.3.6.1.4.1.1466
 .115.121.1.24  )
attributeTypes: ( 1.3.6.1.4.1.13299.1.9 NAME ( 'myAccountControl' ) SYNTAX 1.3.6.1.4.1.146
 6.115.121.1.6 )
attributeTypes: ( 1.3.6.1.4.1.13299.1.10 NAME ( 'myCustomerNumber' ) SYNTAX 1.3.6.1.4.1.14
 66.115.121.1.15  )
-
add: objectClasses
objectClasses: (1.3.6.1.4.1.13299.2.1 NAME 'myUser' SUP top AUXILIARY MAY ( myId $ mySalut
 ation $ myBirthdate $ myRegisteredMail $ myAlternateIdentity $ myUserType $ myDeactivatio
  nDate $ myReductionDate $ myAccountControl $ myCustomerNumber ))
-
```

Implement over the time

- you can alway add an extra attribute or an object class.

- It is more difficult to remove an obsolete attribute.

The only reasonable action is to add the clause "OBSOLETE" to both the definition.

Keep it as simple as possible, but too simple.
(after A. Einstein)

The LDAP Schema

- **Thank you for your attention**