Generating RDF Models from LDAP Directories

Sebastian Dietzold

University of Leipzig
Institute for Medical Informatics, Statistics and Epidemiology (IMISE)
sebastian.dietzold@imise.uni-leipzig.de

Abstract. Lightweight Directory Access Protocol (LDAP) has gained importance as a database technology for the storage and retrieval of information on network and human resources. This paper describes a method to create RDF models from a directory information tree or an LDAP search query by mapping LDAP schema information into OWL ontologies and LDAP objects into RDF instance triples. Furthermore, it reports on an implementation of this method based on the RDF API for PHP.

Introduction

LDAP was specified by the IETF to provide an easier access to directory services from internet clients via TCP/IP. Since 1997 LDAP v3 servers [1] have found wide distribution for the storage of information about network and human resources. In LDAP directories, all objects consist of pairs of attribute names and values, they are arranged in a directory information tree (DIT) and get a distinguished name (DN), which is build from the relative DN (RDN, simply a selected name-value pair) and the RDN of all parent objects. Every object is an instance of exactly one structural object class and possibly more than one auxiliary object classes. By this instantiation, different attribute types were allowed or enforced. The object classes, attribute types and other metadata are defined in the directory's schema. Various object classes are standardized by the IETF (e.g. `inetOrgPerson` [2]) and target at common tasks for directories such as the usage as a central authorisation component in a network[1].

With respect to the Semantic Web it is important to reuse existing data and schemata in LDAP directories within the Resource Description Framework (RDF, [4]). This means that there has to be a method which allows the conversion of an LDAP schema specification into OWL ontologies. Thus we can map converted classes and properties with all kinds of Semantic Web vocabularies like FOAF and Dublin Core. The resulting OWL ontology must keep the cardinality restrictions and the attribute syntax. It must also allow the reproduction of the DIT.

This paper is divided into three parts. Section 1 will describe the method to generate an OWL ontology from a LDAP schema. On the basis of this, section

[1] For a more complete introduction to LDAP see [3], chapter 3.

2 shows how RDF models are generated from LDAP objects. Finally, an PHP implementation based on the PHP API RAP [5] will be explained briefly in section 3.

1 Conversion of a LDAP directory schema

The schema specification of an LDAP server consists of four parts: the definition of object classes and attribute types, and an index of attibute syntaxes and matching rules implemented on the server. Every class, type, syntax and rule is globally identified by an object identifier (OID[2]) and locally by one or more mnemonic names.

An requirement for the whole process is a small vocabulary which describes base entities of the LDAP world: `object` is the class of all LDAP objects and `hasDITChild` / `hasDITParent` is a pair of (inverse) functional object relations who represent the DIT structure. For a one-way conversion of existing LDAP data into RDF data, only this basic vocabulary is necessary. A more complete LDAP schema ontology will be presented in [7][3].

The next three subsections explain the rules of how to convert a LDAP schema with types and classes to an OWL ontology.

1.1 Attribute syntax

Because the different attribute syntaxes (e.g. the telephone number syntax or the integer syntax) are only named in the schema but not declared, dealing with them can only be achieved by a mapping table in which each syntax OID is put together with the URI of its matching XML Schema datatype. Most of them can simply be matched with `xsd:string`[4], but in some cases we can map to more specific datatypes or we must bear in mind other constraints. These exceptions are:

- The DN and OID syntax must be ignored, because attributes of this type are transformed into object properties.
- The values of any binary syntax attributes should be recoded as the XML Schema datatype `xsd:base64Binary`.
- The Directory String syntax should be `xsd:normalizedString` because no new lines are allowed here.
- Numeric syntaxes must be mapped to an equivalent numeric XML Schema datatype, if possible.

[2] OID are strings of dot-separated numbers which form a hierarchical structure. They were first specified in the ITU-T recommendation X.208 (ASN.1) and are used to identify objects in the broadest sense. An OID can be transformed into an URN by using the prefix `urn:oid:` [6].

[3] The actual version of this vocabulary is available from the persistant URL `http://purl.org/net/ldap` . Within this paper, the namespace `ldap` is assigned to this URL.

[4] Assuming the namespace `xsd` is assigned to `http://www.w3.org/2001/XMLSchema`.

– Special syntaxes like the syntax for telephone numbers can be accurately defined by regular expressions in a non-standard XML Schema document.
– Some syntaxes are suited to be used as URIs too. Optionally telepone numbers can be used with the `tel` URI scheme [8] and `labeledURI` attributes have to be splitted into a resource and its `rdfs:label`.

Table 1. Mapped syntaxes (selection)

syntax name (OID)	typical attribute	→ XML Schema datatype
JPEG (`mw.28`)	`jpegPhoto`	→ `xsd:base64Binary`
Directory String (`mw.15`)	`cn`, `title`, etc.	→ `xsd:normalizedString`
Postal Address (`mw.41`)	`postalAddress`, etc.	→ `xsd:string`
Integer (`mw.27`)	`uidNumber`, `gidNumber`, etc.	→ `xsd:integer`
Tel. Number (`mw.50`)	`mobile`, `pager`, etc.	→ `xsd:string`

Table 1 shows some LDAP syntaxes[5], the attributes they are used in and the XML Schema datatypes they are mapped to.

1.2 Attribute types

In addition to the syntax, every attribute type schema definition consists of an OID, one or more local names and optionally a description and a single value indicator. An attribute may be the sub-attribute of another attribute, e.g. `commonName` and `title` are sub-attributes of `name`. This simply means that the syntax of the sub-attribute derives from the super-attribute, in particular this does not provide any clue about any specific relation between the sub-attribute and the super-attribute.

Each attribute schema definition will be transformed into OWL according to the following implementation rules:

– Every attribute with the syntax of an OID or a DN is created as an object property while all others become datatype properties[6].
– If there is a super-attribute, the RDFS relation `rdfs:subPropertyOf` should be used.
– Object properties get as range the class `top`[7] and datatype properties the matching XML Schema syntax.
– Local names are used as part of the URI of the new property. The first URI is described explicitly, any other URI and the OID (in URN form) are mapped with `owl:equivalentProperty` to the first one.
– The local name should also be used as the value of a `rdfs:label` property.

[5] Assumimg the OID prefix `wm` is set to `1.3.6.1.4.1.1466.115.121.1` .
[6] In general, the OID syntax is only used in administrative attributes.
[7] According to RFC 2251, the root of any schema class hierarchy has to be `top`.

– If there is a single value indicator in the schema definition, then a cardinality restriction (≤ 1) for the class `top` must be created on this property[8].
– The domain of the property will be generated with the `owl:unionOf` class constructor and the list of all object classes which use this attribute (no need for `owl:unionOf` if there is only one domain class).

An example for a transformed LDAP attribute is presented with the next lines of RDF/XML code. In the sample schema, the single value attribute `countryName` is only used in the object class `country`, therefore its domain is given by `country`.

```
<owl:DatatypeProperty rdf:about="#c">
  <rdfs:label>c</rdfs:label>
  <rdfs:label>countryName</rdfs:label>
  <owl:equivalentProperty rdf:resource="#countryName"/>
  <owl:equivalentProperty rdf:resource="urn:oid:2.5.4.6"/>
  <rdfs:comment>RFC2256: ISO-3166 country ...</rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="#name"/>
  <rdfs:domain rdf:resource="#country"/>
</owl:DatatypeProperty>

<rdf:Description rdf:about="#top">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality>1</owl:maxCardinality>
      <owl:onProperty rdf:resource="#c" />
    </owl:Restriction>
  </rdfs:subClassOf>
</rdf:Description>
```

1.3 Object classes

The schema definition of an object class can contain two different reference types to an attribute. It depends on this type whether an attribute is mandatory in all objects of this class (`MUST`) or whether it is allowed and optional (`MAY`). Furthermore, the definition consists of a local name, an OID and an optional description. All classes are arranged in a subclass hierarchy whose root is the class `top`. Subclasses inherit all attribute conditions of its superclasses[9]. To generate OWL classes from LDAP schema classes, these implementation rules must be followed:

– The local name must be used as part of the URI of the new OWL class. Additionally, the OID (in URN form) is mapped with `owl:equivalentClass` to the class. It is theoretically possible to create an ontology only by using URN-formed OIDs but in fact, many tools do not support URNs so it is more workable to generate a new URL for each class (and property).

[8] Using `top` instead of the domain classes is necessary because the single value restriction is set for the whole LDAP schema and not for the particular domain classes.
[9] In an LDAP schema more than one direct superclass is possible.

– The local name should also be used as the value of a `rdfs:label` property.
– For every mandatory attribute of the object class a new cardinality restriction (= 1) has to be created on the OWL class.
– The new class has to be in `rdfs:subClassOf`-relation with its superclass(es) (in most cases just `top`). `top` must be defined as a subclass of the generic LDAP object class `ldap:object` from [9].

As an example, the next XML listing shows the converted standard object class `country` which has the mandatory attribute `countryName`.

```
<owl:Class rdf:about="#country">
  <rdfs:label>country</rdfs:label>
  <rdfs:subClassOf rdf:resource="&ldap;top" />
  <owl:equivalentClass rdf:resource="urn:oid:2.5.6.2"/>
  <rdfs:comment>RFC2256: a country</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality>1</owl:cardinality>
      <owl:onProperty rdf:resource="#c" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

The resulting OWL schema requires at least OWL DL because of the `owl:unionOf` class constructor.

2 Conversion of a LDAP directory information tree

To convert the directory objects from a DIT into an RDF model, the classes and properties of the previously created OWL ontology and the basic LDAP ontology have to be used. While doing this, these steps have to be followed:

– Construct the URI of the LDAP object according to RFC 1959 [10]. This means that the object gets an URI in the form `ldap://[host]/[URL encoded DN]`, e.g. `ldap://ldap.example.org/c=de`.
– Identify the main structural object class in order to take this as the `rdf:type` of the object. Any auxiliary classes become additional types.
– Convert all attribute values from DN and OID syntax attributes into resources and create that object property triple.
– Convert all other attribute values to literals[10] and create a datatype property triple.
– Use the objects RDN value as a `rdfs:label`.

As a result of this process, a typical LDAP object of the `organization` type gets an RDF representation which looks like this:

[10] Binary values have to be encoded as `xsd:base64Binary`.

```
<organization rdf:about="ldap://example.org/o=IMISE,c=de">
  <rdf:type rdf:resource="#labeledURIObject" />
  <rdfs:label>IMISE</rdfs:label>
  <o>IMISE</o>
  <telephoneNumber>+49 341 97 16100</telephoneNumber>
  <labeledURI>http://www.imise.de/</labeledURI>
  <description>Institute for  ...</description>
  <ldap:hasParent rdf:resource="ldap://example.org/c=de"/>
</organization>
```

With this method it is possible to convert a complete LDAP directory including schema and data. At the same time only parts of a DIT, e.g. a result set of an LDAP query can be converted into and reused in RDF. In the next section, I will briefly explain an PHP implementation that is based on the PHP API RAP.

3 Implementation

As a proof of the concept, I have implemented the method in the server-side script language PHP as a web application using the wrapper functions to the OpenLDAP library and classes for the schema and server handling from the open-source project phpLDAPadmin on the LDAP side as well as the PHP API RAP on the RDF side.

Implementation was rather simple because all the used libraries are mature and well-documented projects. For the test run, I converted the IMISE LDAP Server with over 300 schema items and 2300 DIT objects into RDF. The current version of the `ldap2owl.php` script is free accessable[11].

References

1. Wahl, M., Howes, T.A., Kille, S.: Lightweight Directory Access Protocol (v3). RFC 2251, The Internet Engineering Task Force (IETF) (1997) http://www.ietf.org/rfc/rfc2251.txt.
2. Smith, M.C.: Definition of the inetOrgPerson LDAP Object Class. RFC 2798, The Internet Engineering Task Force (IETF) (2000) http://www.ietf.org/rfc/rfc2798.txt.
3. Howes, T.A., Smith, M.C., Good, G.S.: Understanding and Deploying LDAP Directory Services. Second edn. Macmillan Network Architecture and Development Series. Macmillan Technical Publishing (2003)
4. Lassila, O., Swick, R.R.: Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, World Wide Web Consortium (W3C) (1999) http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.
5. Bizer, C.: RAP (RDF API for PHP). Website (2004) http://www.wiwiss.fu-berlin.de/suhl/bizer/rdfapi/.
6. Mealling, M.: A URN Namespace of Object Identifiers. RFC 3061, The Internet Engineering Task Force (IETF) (2001) http://www.ietf.org/rfc/rfc3061.txt.

[11] `http://purl.org/net/ldap/ldap2owl.php`

7. Dietzold, S.: LDAP and the Semantic Web. in preparation (2005)
8. Schulzrinne, H.: The tel URI for Telephone Numbers. RFC 3966, The Internet Engineering Task Force (IETF) (2004) http://www.ietf.org/rfc/rfc3966.txt.
9. Dietzold, S.: Basic vocabulary to use LDAP data in RDF. OWL ontology (2005) http://purl.org/net/ldap.
10. Howes, T.A., Smith, M.C.: An LDAP URL Format. RFC 1959, The Internet Engineering Task Force (IETF) (1996) http://www.ietf.org/rfc/rfc1959.txt.