

Definition of a metadata format and DIT structure for a Directory Schema Registry

TERENA Project Directory Schema Registry, Deliverable C

Peter Gietz, Bernhard Winkler, DAASI International Ltd

peter.gietz@daasi.de, bernhard.winkler@daasi.de

Version 0.92, 31.5.2003

1. Status of this document

This is deliverable C of the TERENA project Directory Schema Registry which is co-funded by TERENA, JISC (Joint Information Systems Committee, UK), REDIRIS (Spanish National Research Network), CESNET (Czech National Research Network), POZNAN SUPERCOMPUTING (Poznan Supercomputing and Networking Center, Poland) and DAASI International and performed by DAASI International. Together with four other deliverables [RegIntro], [RegPolicy], [RegArchitecture], [RegBusiness], and the bibliography [RegBib] it forms the documentation of this project.

This document describes all schema needed for storing data in the Directory Schema Registry (DSR). This version 0.92 reflects the status in May 2003 and is the first public version. A final version 1 will be published after all other project documents have been completed.

In some parts of this text, some details not relevant for the currently planned registry are discussed which points to a possible future broadening of the functionality of the registry after the end of this project. A version 2 of this document is planned for the time after the end of the project. For this future version comments and additions to the current document are welcomed. Please send them to the email address of the author.

Table of contents

1.	Status of this document.....	1
2.	Introduction.....	1
3.	Requirements.....	2
4.	Bibliographical references in LDAP.....	2
5.	Metadata specified by the IETF schema WG.....	20
6.	Additional schema for the DSR.....	43
7.	DIT Structure for the Directory Schema Registry.....	61
8.	Pointer to References.....	63
A.	MIME Directory Type Registrations.....	63
B.	Detailed Table of Contents.....	64

2. Introduction

[RegIntro] has discussed prior approaches for solving the schema registry problem. It was shown that there does not exist an LDAP schema registry that fulfils all requirements for a system useful for humans and applications. One prerequisite for such a system that can provide LDAP clients with schema information is that the schema has to be stored in an LDAP directory. This was discussed in section 5.1.1. of [RegIntro]. Methods for how to do that are defined in the core protocol itself [X.501]

and [LDAPModels] as well as in [LDAPSubentry]. As described in [RegIntrod] section 5.1.2., an alternative mechanism was defined in [RFC 1804] that has the advantage of being usable, not only for applications, but also for humans. The schema approach in this document follows these lines, but also provides additional features.

Another presupposition is the mandatory documentation for an inclusion into the registry. A not well documented schema can hardly be reused. Thus metadata about the specification documents will have to be defined in this schema as well. [DC], being the most recognised standard for metadata on text-like objects, is the basis for the respective schema in this registry.

The main part of this document starts with a list of the requirements for the schema of the schema registry in chapter 3. Chapter 4 deals with the problems of using [DC] in LDAP and the proposed interim solution needed for the registry. Chapter 5 discusses the schema as proposed by the IETF schema WG, namely schema for metadata [SchemaMeta] and schema for storing LDAP schema elements [RFC 2927]. Chapter 6 specifies additional schema needed for the Directory Schema Registry. Chapter 7 deals with the design of the DIT structure of the schema registry.

All references can be found in [RegBib]. A number of acronyms are being used throughout the project documentation. Please view chapter 3 of [RegIntrod] for short explanations of these acronyms. Appendix A of [RegIntrod] consists of an alphabetical list of these acronyms which provides the reader with pointers to the places of that document, where they are discussed.

3. Requirements

Following requirements have to be fulfilled by the metadata schema and the DSR implementation:

- LDAP programs must be able to retrieve schema information via LDAP. The respective specifications in [LDAPmodels] and [LDAPSubentry] have to be followed.
- The submission format as specified in [RFC 2927] has to be supported and reflected in the schema of the DSR.
- The metadata set, as defined in [SchemaMeta], has to be reflected in the schema for the DSR.
- The metadata on the specification documents must be modelled in accordance to Dublin Core metadata set.
- [RFC 2629] presents a technique for using XML as a source format for documents in the Internet-Drafts and RFC series of IETF. The metadata on the specification documents should be able to map the structures for the front matter defined in [RFC 2629].
- Additional schema for the storage of additional schema management information has to be specified.

4. Bibliographical references in LDAP

Although there had been other proposals for storing bibliographical references in LDAP than using Dublin Core, namely [Klasen] and other metadata formats, namely [RFC 1807] and [RFC 2629], Dublin Core seems to be the most acknowledged metadata format and thus this project will use schema on the lines of [LDAPDC]. As a requirement, the bibliographic entries have to map all fields specified as front matter in [RFC 2629].

The following section describes an LDAP schema for DC. Section 4.2. describes how to include additional author information without a qualifier representation. Section 4.3. specifies how the proposed schema can be used to store the front matter information specified in [RFC 2629].

4.1. The Dublin Core Metadata set and its LDAP representation

Currently 16 DC elements are specified as the Dublin Core Metadata Element Set (DCMES) by the Dublin Core Metadata Initiative in [DCcurrent]. This section provides the LDAP representation of these elements, building on the approach of [DCLDAP]. Quotes in this chapter are from [DCcurrent], if not stated otherwise.

In this section the following rules apply:

- The term "specification document" refers to a document with technical specifications. In respect to the DSR it means a document with the specification of a schema.
- The sentences starting with "In the case of a specification document" specify the usage of the DC attributes in the DSR.
- The term "this specification" means the specification described by the DC metadata.

4.1.1. The issues about DC Qualifiers

As described in section 3.3.5. of [RegIntro], the DC metadata format includes the possibility to enhance the DC elements by so-called qualifiers that either refine an element or provide encoding information. DC was intended to provide complete interoperability and it is not expected that every implementation will know about every qualifier used in the different DC communities. Thus it is a requirement that implementations may ignore qualifiers. In the words of [DCQual]:

"Inevitably, there will be situations where an agent or client will encounter DCMES descriptions that use unfamiliar qualifiers developed by implementors for specialized local or domain-specific needs. The useful interpretation of such a DCMES description will depend on the ability of an application to ignore the unknown qualifiers and fall back on the broader meaning of the element in its unqualified form. The guiding principle for the qualification of Dublin Core elements, colloquially known as the Dumb-Down Principle, is that a client should be able to ignore any qualifier and use the description as if it were unqualified. While this may result in some loss of specificity, the remaining element value (without the qualifier) should continue to be generally correct and useful for discovery."

As discussed by [Lagoze], the use of qualifiers that enhance the information rather than refine it is a violation of the above mentioned "Dumb-Down Principle". The most common instance of such a violation is to add information about contact data or affiliation of persons (creators of a resource, publishers or contributors), e.g., DC.Creator.Affiliation. Searches that are not aware of this qualifier, interpret such an element as DC.Creator; a name of an organisation may thus be misinterpreted as the name of a creator. Therefore such information should not be stored in qualifiers. The proposal for storing additional information of persons below thus has no implications for storing qualifier information in LDAP which is still an unsolved issue.

[DCLDAP] makes a proposal for storing DC Qualifiers described as following:

"a simple but sub-optimal approach be taken - with any qualifying information being placed at the beginning of the value part of the attribute/value pair, delimited using round brackets, and with any additional qualifiers following using comma separation."

Such implicit semantics without any syntax specification are not recommended. Better alternatives for storing qualifiers in LDAP are:

- attribute description options as defined in [LDAPModels] and as used in [RFC 2596] to add the language of the attribute value; this seems to be a good method for mapping the encoding type of qualifiers
- attribute inheritance as defined in [LDAPModels]; this requires the definition of an attribute subtype for every qualifier and could well be used for qualifiers of the refinement type.

- special syntax for qualifier information, similar to the solution proposed in [DCLDAP]

It is not the purpose of this document to solve the issue of storing qualifiers in LDAP. It needs a separate document that discusses all possibilities and handles all aspects and implications. Such a document is not within the scope of this project.

4.1.2. DC.Title

This element is used for naming a resource.

"A name given to the resource. Typically, a Title will be a name by which the resource is formally known."

In the case of a specification document, it is used for its title.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.1
  NAME 'dcTitle'
  DESC 'A name given to the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Sometimes a resource can have more than one title, e.g., title and subtitle. Thus this attribute is multivalued.

4.1.3. DC.Creator

This element is used for the

"entity primarily responsible for making the content of the resource. Examples of a Creator include a person, an organisation, or a service. Typically, the name of a Creator should be used to indicate the entity."

In the case of a specification document, it is used for the name of its authors.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.2
  NAME 'dcCreator'
  DESC 'An entity primarily responsible for making the content
    of the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

If more information about the author of a specification document is to be stored, this information should be stored in a separate person entry which is specified below in section 4.2. In this case an alternative attribute for dcCreator is used that contains a DN pointer to such an entry. The LDAP representation of this alternative attribute of dcCreator is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.3
  NAME 'dcCreatorPointer'
  DESC 'A DN pointer to an entry with information about the entity
    primarily responsible for making the content of the
    resource'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

Only one of the two attributes dcCreator and dcCreatorPointer is to be used.

4.1.4. DC.Subject

This element is used for describing the topic of the content of the resource via keywords.

"The topic of the content of the resource. Typically, a Subject will be expressed as keywords, key phrases or classification codes that describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme."

In the case of a specification document, it can be used for describing the topic of the specification. No classification scheme is used in the first phase of the DSR.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.4
  NAME 'dcSubject'
  DESC 'The topic of the content of the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.1.5. DC.Description

This element is used to describe the resource.

"An account of the content of the resource. Description may include but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content."

In the case of a specification document, it is used for the abstract of the document.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.5
  NAME 'dcDescription'
  DESC 'An account of the content of the resource'
  SUP description )
```

For a definition of the attribute description see section 4.2.14.

To provide the possibility of having more than one description, this is a multi value attribute.

4.1.6. DC.Publisher

"An entity responsible for making the resource available. Examples of a Publisher include a person, an organisation, or a service. Typically, the name of a Publisher should be used to indicate the entity."

In the case of a specification document, it is used for the name of the standardisation board.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.6
  NAME 'dcPublisher'
  DESC 'An entity responsible for making the resource available'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

If more information about the publisher of a specification document is to be stored, this information can be stored in a separate person entry. In this case an alternative attribute of dcPublisher is used that contains a DN pointer to such an entry. The LDAP representation of this alternative attribute of dcPublisher is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.7
  NAME 'dcPublisherPointer'
  DESC 'A DN pointer to an entry with information about the entity
  responsible for making the resource available'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

Only one of the two attributes dcPublisher and dcPublisherPointer is to be used.

4.1.7. DC.Contributor

"An entity responsible for making contributions to the content of the resource. Examples of a Contributor include a person, an organisation, or a service. Typically, the name of a Contributor should be used to indicate the entity."

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.8
  NAME 'dcContributor'
  DESC 'An entity responsible for making contributions to
        the content of the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

If more information about a contributor of a specification document is to be stored, this information can be stored in a separate person entry. In this case an alternative attribute of dcContributor is used that contains a DN pointer to such an entry. The LDAP representation of this alternative attribute of dcContributor is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.9
  NAME 'dcContributorPointer'
  DESC 'A DN pointer to an entry with information about the entity
        responsible for making contributions to the content of the
        resource'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

Only one of the two attributes dcContributor and dcContributorPointer is to be used.

In the case of a specification document, dcContributorPointer is used to point to an entry that contains names and affiliation of the contributors mentioned in an acknowledgement section.

4.1.8. DC.Date

"A date associated with an event in the life cycle of the resource. Typically, Date will be associated with the creation or availability of the resource. Recommended best practice for encoding the date value is defined in a profile of ISO 8601 [W3CDTF] and follows the YYYY-MM-DD format."

In the case of a specification document, it is used for the publication date of the document. No format is prescribed.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.10
  NAME 'dcDate'
  DESC 'A date associated with an event in the life cycle
        of the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.1.9. DC.Type

"The nature or genre of the content of the resource. Type includes terms describing general categories, functions, genres, or aggregation levels for content. Recommended best practice is to select a value from a controlled vocabulary (for example, the DCMI Type Vocabulary [DCMITYPE]). To describe the physical or digital manifestation of the resource, use the Format element."

[DCMITYPE] describes itself as:

"The DCMI Type Vocabulary provides a general, cross-domain list of approved terms that may be used as values for the Resource Type element to identify the genre of a resource."

It currently specifies the following types: Collection, Dataset, Event, Image, InteractiveResource, Service, Software, Sound, Text, and PhysicalObject.

In the case of a specification document, the DC.Type "Text" is used throughout.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.11
  NAME 'dcType'
  DESC 'The nature or genre of the content of the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.1.10. DC.Format

"The physical or digital manifestation of the resource. Typically, Format may include the media-type or dimensions of the resource. Format may be used to determine the software, hardware or other equipment needed to display or operate the resource. Examples of dimensions include size and duration. Recommended best practice is to select a value from a controlled vocabulary (for example, the list of Internet Media Types [MIME] defining computer media formats)."

"[MIME]" is a reference to an outdated list of MIME media types at: <http://www.isi.edu/in-notes/iana/assignments/media-types/media-types> which should be replaced by the official IANA MIME media type registry at <http://www.iana.org/assignments/media-types/index.html>.

In the case of a specification document, the following MIME types could be used: text/plain, text/html, text/xml, application/postscript, application/pdf, application/rtf, application/xml.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.12
  NAME 'dcFormat'
  DESC 'The physical or digital manifestation of the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.1.11. DC.Identifier

"An unambiguous reference to the resource within a given context. Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Example formal identification systems include the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI) and the International Standard Book Number (ISBN)."

In the case of a specification document, it should be the URL to a copy of the document. The URL should point to the web site of the organisation which published the specification. Alternative URLs, pointing to the same version of the specification stored on another web server, may be added.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.13
  NAME 'dcIdentifier'
  DESC 'An unambiguous reference to the resource within a
  given context'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.1.12. DC.Source

"A reference to a resource from which the present resource is derived. The present resource may be derived from the Source resource in whole or in part. Recommended best practice is to reference the resource by means of a string or number conforming to a formal identification system."

This element can be used, e.g., to reference to an original paper version of a document from which a digital representation, e.g. in HTML, is derived.

In the case of a specification document, this element may be used as a document name (not to be mixed with document title) that exists independently from the format of an instance of the specification document, e.g., the document name "rfc822" versus file names "rfc822.txt" or "rfc822.html".

Relations to other specifications should rather be stored in the DC.Relation element (see below).

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.14
  NAME 'dcSource'
  DESC 'A Reference to a resource from which the present
        resource is derived'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.1.13. DC.Language

"A language of the intellectual content of the resource. Recommended best practice is to use RFC 3066 [RFC 3066] which, in conjunction with ISO 639 [ISO639], defines two- and three-letter primary language tags with optional subtags. Examples include "en" or "eng" for English, "akk" for Akkadian, and "en-GB" for English used in the United Kingdom."

"[ISO639]" refers to a three letter language tag registration site of the Library of congress at <http://www.loc.gov/standards/iso639-2/>. The specification for three letter language tags is [ISO 639-2].

In the case of a specification document, this element should be used with the tags defined in [RFC 3066]. The expected tag for most specifications is "en".

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.15
  NAME 'dcLanguage'
  DESC 'A language of the intellectual content of the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.1.14. DC.Relation

"A reference to a related resource. Recommended best practice is to reference the resource by means of a string or number conforming to a formal identification system."

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.16
  NAME 'dcRelation'
  DESC 'A reference to a related resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

In the case of a specification document, this element should be used with the following tags:

- "obsoletes", if this specification replaces the specification pointed to. [RFC 2223] defines "obsoletes" as:

"This document contains either revised information, or else all of the same information plus some new information, however extensive or brief that new information is; i.e., this document can be used alone, without reference to the older document."

- "obsoleted-by", if this specification is replaced by the specification pointed to. The same definition of "obsoletes" applies accordingly.
- "updates", if this specification enhances the specification pointed to. [RFC 2223] defines the term "updates" as:

"to be used as a reference from a new item that cannot be used alone (i.e., one that supplements a previous document), to refer to the previous document. The newer publication is a part that will supplement or be added on to the existing document; e.g., an addendum, or separate, extra information that is to be added to the original document."
- "updated-by", if this specification is updated by the specification pointed to. The same definition of "updates" applies accordingly.
- "inherits", if this specification is dependent on another specification pointed to because the schema elements specified inherit qualities from a schema element specified in that document.

Ideally, the DSR will use the LDAP attribute description tagging option method for storing these tags. Since tags other than the language tags specified in [RFC 2596] are not implemented in the LDAP-Server used by the DSR (OpenLDAP), it will either have to be implemented by the project or an alternative method will be used, namely by specifying subtypes of the attribute `dcRelation`:

```

attributetype ( 1.3.6.1.4.1.10126.1.7.3.17
  NAME 'dcObsoletes'
  SUP dcRelation'

attributetype ( 1.3.6.1.4.1.10126.1.7.3.18
  NAME 'dcObsoletedBy'
  SUP dcRelation'

attributetype ( 1.3.6.1.4.1.10126.1.7.3.19
  NAME 'dcUpdates'
  SUP dcRelation'

attributetype ( 1.3.6.1.4.1.10126.1.7.3.20
  NAME 'dcUpdatedBy'
  SUP dcRelation'

attributetype ( 1.3.6.1.4.1.10126.1.7.3.21
  NAME 'dcInherits'
  SUP dcRelation'

```

The specification of subtypes is one possible solution for the DC.qualifier problem (see discussion in section 4.1.1.). At this stage of the project no general solution will be provided.

4.1.15. DC.Coverage

"The extent or scope of the content of the resource. Coverage will typically include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and that, where appropriate, named places or time periods be used in preference to numeric identifiers such as sets of coordinates or date ranges."

In the case of a specification document, this element should not be used.

The LDAP representation is:

```

attributetype ( 1.3.6.1.4.1.10126.1.7.3.22
  NAME 'dcCoverage'
  DESC 'The extent or scope of the content of the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

```

4.1.16. DC.Rights

"Information about rights held in and over the resource. Typically, a Rights element will contain a rights management statement for the resource or reference a service providing such information. Rights information often encompasses Intellectual Property Rights (IPR), Copyright, and various Property Rights. If the Rights element is absent, no assumptions can be made about the status of these and other rights with respect to the resource."

In the case of a specification document, this element must be filled and it should reflect the copyright status of the specification. If the specification is dependent on a patent, this should also be noted. A minimal rights statement for, e.g., an RFC could be "copyright (C) The Internet Society (1999)". If no rights at all are claimed, the element should contain "no copyright claims".

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.23
  NAME 'dcRights'
  DESC 'Information about rights held in and over the resource'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.1.17. DC.Audience

"A class of entity for whom the resource is intended or useful. A class of entity may be determined by the creator or the publisher or by a third party."

In the case of a specification document, this element should not be used.

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.10126.1.7.3.24
  NAME 'dcAudience'
  DESC 'A class of entity for whom the resource is intended
  or useful '
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
```

4.1.18. Object class dcContainer

The LDAP representation of an object described with DC metadata is:

```
objectclass ( 1.3.6.1.4.1.10126.1.7.4.1
  NAME 'dcContainerObject'
  DESC 'object containing the Dublin Core attributes'
  SUP top
  AUXILIARY
  MAY ( dcTitle $ dcCreator $ dcCreatorPointer $ dcSubject $
  dcDescription $ dcPublisher $ dcPublisherPointer $
  dcContributor $ dcContributorPointer $ dcDate $ dcType $
  dcFormat $ dcIdentifier $ dcSource $ dcLanguage $
  dcRelation $ dcCoverage $ dcRights $ dcAudience ) )
```

In the case of a specification document, dcTitle, dcCreator, dcSubject, dcDescription, dcDate, dcFormat, dcIdentifier and dcRights should be considered as MUST attributes in contrast to their above specification as MAY attributes. This and the prescribed use of dcCreatorPointer is specified by the following DIT Content Rule which will be used in the DSR:

```
( 1.3.6.1.4.1.10126.1.7.4.1
  NAME 'dcSpecificationWithAuthorInfoContentRule'
  DESC 'Profile for the use of object class dcContainer to
  describe a specification document with dcCreator pointing
  to an entry with additional information on the author'
  MUST ( dcTitle $ dcCreatorPointer $ dcSubject $ dcDescription $
  dcDate $ dcFormat $ dcIdentifier $ dcRights )
  MAY ( dcPublisher $ dcContributorPointer $ dcType $ dcLanguage $
  dcRelation )
```

```

    NOT ( dcCreator $ dcPublisherPointer $
          dcContributor $ dcSource $ dcCoverage $ dcAudience )
)

```

4.2. Additional schema for person information

In cases in which more information about a person responsible for creation, publishing or contribution to a specification document is to be stored, alternative attributes for DC.Creator, DC.Publisher, and DC.Contributor have been specified that point to an entry that contains all such information. This entry is specified in this section.

The person information entry has to include the object class inetOrgPerson as defined in [RFC 2798]. Following elements from that object class or its superior object classes (person and organizationalPerson) are to be used:

4.2.1. Common name

The attribute commonName (cn) is to be used for the full name of the person described by the entry.

This attribute which is a MUST attribute of the object class person is specified in [LDAPSchema] as follows:

"This is the X.520 [X.520] commonName Attribute Type which contains a name of an object. If the object corresponds to a person, it is typically the person's full name."

The LDAP representation is:

```

attributetype ( 2.5.4.3
    NAME 'cn'
    SUP name )

```

The attribute type name from which cn is derived is specified in [LDAPSchema] as following:

```

attributetype ( 2.5.4.41 NAME 'name'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )

```

Recommended practice is to include two forms of the name:

- 1.) the exact form of the name as written in the specification.
- 2.) a general accepted normalised version of the name as specified, e.g., by librarians.

4.2.2. surName

The attribute type surName (sn) which is specified as MUST attribute of the object class person is used to store the surname of a person.

[LDAPSchema]:

"This is the X.500 surname attribute, which contains the family name of a person."

The LDAP representation is:

```

attributetype ( 2.5.4.4
    NAME 'sn'
    SUP name )

```

4.2.3. initials

The attribute type initials which is a MAY attribute of inetOrgPerson is used for storing the initials of a person.

[LDAPSchema]:

"The initials attribute contains the initials of some or all of an individuals names, but not the surname(s)."

The LDAP representation is:

```
attributetype ( 2.5.4.43
  NAME 'initials'
  SUP name )
```

4.2.4. organizationName

The attribute type organizationName (o), which is specified as MAY attribute of the inetOrgPerson object class, is used to store the name of the organisation the person is affiliated to.

[LDAPSchema]:

"This is the X.520 [X.520] organizationName Attribute Type, which contains the name of an organization."

The LDAP representation is:

```
attributetype ( 2.5.4.10
  NAME 'o'
  SUP name )
```

4.2.5. street

The attribute type street, which is specified as MAY attribute of the organizationalPerson object class, is used for storing the street information of the postal address of the person.

[LDAPSchema]:

"This is the X.520 [X.520] streetAddress attribute, which contains the physical address of the object to which the entry corresponds, such as an address for package delivery."

The LDAP representation is:

```
attributetype ( 2.5.4.9
  NAME 'street'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{128} )
```

4.2.6. locationName

The attribute type locationName (l), which is specified as MAY attribute of the organizationalPerson object class, is used to store the city name as part of the postal address of the person.

[LDAPSchema]:

"This is the X.520 [X.520] localityName Attribute Type, which contains the name of a locality or place, such as a city, county or other geographic region."

The LDAP representation is:

```
attributetype ( 2.5.4.7
  NAME 'l'
  SUP name )
```

4.2.7. stateOrProvinceName

The attribute type stateOrProvinceName (st), which is specified as MAY attribute of the organizationalPerson object class, is used for the name of a state or province as part of the postal address of the person.

[LDAPSchema]:

"This is the X.520 [X.520] stateOrProvinceName attribute, which contains the full name of a state or province."

The LDAP representation is:

```
attributetype ( 2.5.4.8
  NAME 'st'
  SUP name )
```

4.2.8. postalCode

The attribute type postalCode, which is specified as MAY attribute of the organizationalPerson object class, is used for storing the postal code as part of the postal address of the person.

[LDAPSchema]:

"This attribute contains a code used by a Postal Service to identify a postal service zone, such as the southern quadrant of a city."

The LDAP representation is:

```
attributetype ( 2.5.4.17
  NAME 'postalCode'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{40} )
```

4.2.9. countryName

The attribute type countryName (c) is used for storing the country name as part of the postal address of the person. This attribute type is not specified as an attribute of an object class describing a person, such as the object classes person, organizationalPerson, or inetOrgPerson. As long as this is the case, it has to be included in an auxiliary object class.

[LDAPSchema]:

"This is the X.520 [X.520] countryName Attribute Type, which contains a two-letter ISO 3166 [ISO3166]country code."

The LDAP representation is:

```
attributetype ( 2.5.4.6
  NAME 'c'
  SUP name
  SINGLE-VALUE )
```

4.2.10. telephoneNumber

The attribute type telephoneNumber, which is specified as MAY attribute of the organizationalPerson object class, is used to store the telephone number of the person.

[LDAPSchema]:

"A value of this Attribute Type is a telephone number complying with ITU Recommendation E.123 [E.123]."

The LDAP representation is:

```
attributetype ( 2.5.4.20
  NAME 'telephoneNumber'
  EQUALITY telephoneNumberMatch
  SUBSTR telephoneNumberSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.50{32} )
```

4.2.11. mail

The attribute type mail, which is specified as MAY attribute of the inetOrgPerson object class, is used for the email address of the person.

[RFC 2798] uses the semantics of the rfc822MailBox attribute as specified in [RFC 1274] as follows:

"The RFC822 Mailbox attribute type specifies an electronic mailbox attribute following the syntax specified in RFC 822."

The LDAP representation is:

```
attributetype ( 0.9.2342.19200300.100.1.3
  NAME 'mail'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )
```

4.2.12. labeledURI

The attribute type labeledURI, which is defined in [RFC 2079] and specified as MAY attribute of the inetOrgPerson object class, is used for the URL of the person.

[RFC 2079]:

"Values placed in the attribute should consist of a URI (at the present time, a URL) optionally followed by one or more space characters and a label. [...]The label is used to describe the resource to which the URI points, and is intended as a friendly name fit for human consumption."

The LDAP representation is:

```
attributetype ( 1.3.6.1.4.1.250.1.57
  NAME 'labeledURI'
  DESCRIPTION 'Uniform Resource Identifier with optional label'
  EQUALITY caseExactMatch
  SUBSTR caseExactSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.2.13. preferredLanguage

[RFC 2798]:

"Used to indicate an individual's preferred written or spoken language. [...] Values for this attribute type MUST conform to the definition of the Accept-Language header field defined in [RFC2068] with one exception: the sequence "Accept-Language" ":" should be omitted. This is a single valued attribute type."

The LDAP representation is:

```
( 2.16.840.1.113730.3.1.39
  NAME 'preferredLanguage'
  DESC 'preferred written or spoken language for a person'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
```

4.2.14. description

[RFC 2256]:

"This attribute contains a human-readable description of the object."

The LDAP representation is:

```
( 2.5.4.13
  NAME 'description'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024} )
```

4.2.15. dcPersonObject object class

An entry that contains additional information about the author of a specification document will be stored with the help of the inetOrgPerson object class as specified in [RFC 2798]. To include additional

information not specified in [RFC 2798], namely the countryName attribute, following additional auxiliary object class has to be added to this entry:

```
objectclass ( 1.3.6.1.4.1.10126.1.7.4.2
  NAME 'dcPersonObject'
  DESC 'object containing additional information about an
        dcCreator, dcPublisher or dcContributor that cannot be
        provided by the inetOrgPerson
        object class'
  SUP top
  AUXILIARY
  MAY c )
```

The following DIT Content Rule specifies a profile for the usage of inetOrgPerson in the context of author information in the DSR:

```
( 2.16.840.1.113730.3.2.2
  NAME 'dcCreatorInfoContentRule'
  DESC 'Profile for the use of object class inetOrgPerson to
        describe an author of a specification document'
  AUX dcPersonObject
  MUST ( cn $ sn )
  MAY ( c $ initials $ o $ street $ l $ st $ postalCode $
        telephoneNumber $ mail $ labeledURI $ displayName $
        givenName $ preferredLanguage )
  NOT ( audio $ businessCategory $ carLicense $ departmentNumber $
        employeeNumber $ employeeType $ homePhone $ homePostalAddress $
        jpegPhoto $ manager $ mobile $ pager $ photo $ roomNumber $
        secretary $ uid $ userCertificate $ x500uniqueIdentifier $
        userSMIMECertificate $ userPKCS12 )
)
```

4.3. The Front matter elements of [RFC 2629]

IETF RFCs and Internet Drafts (see [RegIntrod] section 3.5.1.) will be the main source for the schema registry in its initial phase. Thus it is a requirement that all relevant elements of these texts can be mapped by the LDAP schema for the schema registry.

[RFC 2629] describes a simple XML Document Type Definition (DTD) that can handle the simple formatting requirements of RFC-like documents and provides meaningful markup of descriptive qualities. It defines a front matter section that contains the bibliographical metadata which will be described in this section, together with a specification how the single elements can be represented in LDAP. Thus an automatic inclusion of the respective metadata from an RFC, formatted according to [RFC 2629], can be implemented.

Other sections specified in [RFC 2629] are discussed in section 4.4.

Dublin Core (see [RegIntrod] 3.3.5.) is the general accepted metadata standard for text resources. The attributes specified in section 4.1. will be used for storing the bibliographical information specified in [RFC 2629]. The mapping for this is specified in this section.

All quotes in this section are from [RFC 2629], if not stated otherwise.

Following XML basics are necessary for understanding this front matter:

- the construct start tag (beginning with "<"), text in between, and end tag (beginning with "</") is called element, e.g., "<example>example text</example>".
- elements can be nested, but must not overlap, e.g., "<outer> text <inner> text </inner> </outer>".
- An element can be enriched with so called attributes in the start tag that contain additional information, e.g., "<example name='value'>".

The front matter is specified as follows:

```

<?xml version="1.0"?>
<!DOCTYPE rfc SYSTEM "rfc2629.dtd">
<rfc>
  <front>
    <title ...>
    <author ...>
    <author ...>
    <date ...>
    <area ...>
    <workgroup ...>
    <keyword ...>
    <keyword ...>
    <abstract ...>
    <note ...>
  </front>
  ...
</rfc>

```

4.3.1. The title Element

"The "title" element identifies the title of the document. [...] if the title is more than 42 characters, then an abbreviation should also be provided, e.g.,

```

<title abbrev="Much Ado about Nothing">
  The IETF's Discussion on "Source Format of RFC Documents"
</title>

```

The title element can be represented by using the dcTitle attribute. The abbreviation can be stored in an additional dcTitle attribute.

4.3.2. The author Element

"Each "author" element identifies a document author. Since a document may have more than one author, more than one "author" element may be present. If the author is a person, then three attributes must be present in the "<author>" tag, "initials", "surname", and "fullname", e.g.,

```

<author initials="M.T." surname="Rose"
  fullname="Marshall T. Rose">

```

The "author" element itself consists of an "organization" element, and, an optional "address" element.

The "organization" element is similar to the "title" element, in that an abbreviation may be paired with a long organization name using the "abbrev" attribute, e.g.,

```

<organization abbrev="ISI">
  USC/Information Sciences Institute
</organization>

```

The "address" element consists of an optional "postal" element, an optional "phone" element, an optional "facsimile" element, an optional "email" element, and, an optional "uri" element.

The "postal" element contains one or more "street" elements, followed by any combination of "city", "region" (state or province), "code" (zipcode or postal code), and "country" elements, e.g.,

```

<postal>
  <street>660 York Street</street>
  <street>M/S 40</street>
  <city>San Francisco</city> <region>CA</region>
  <code>94110</code>
  <country>US</country>
</postal>

```

[...] the value of the "country" element should be a two-letter code from ISO 3166. The "phone", "facsimile", "email", and "uri" elements are simple, e.g.,

```

<phone>+1 415 695 3975</phone>
<email>mrose@not.invisible.net</email>
<uri>http://invisible.net/</uri>"

```

The structure of an author element is thus (with "XX" representing the actual values):

```

<author initials="XX" surname="XX" fullname="XX">
  <organization abbrev="XX"> XX </organisation>
  <address>
    <postal>
      <street>XX</street>
      <city>XX</city>
      <region>XX</region>
      <code>XX</code>
      <country>XX</country>
    </postal>
    <phone>XX</phone>
    <email>XX</email>
    <uri>XX</uri>
  </address>
</author>

```

This can only be represented in LDAP with a separate entry. The standard for person objects inetOrgPerson object class as specified in [RFC 2798] is used for modelling this separate entry with addition of the dcPersonObject auxiliary object class (see above section 4.2.15.). Following attributes of [RFC 2798] are used to map the described elements:

- commonName (cn) for the contents of the attribute fullname of the element <author>
- surName (sn) for the contents of the attribute surname of the element <author>
- initials for the contents of the attribute initials of the element <author>
- organizationName (o) for the text of the element <organization> and for the value of the attribute abbrev
- street for the text of the element <street>
- locationName (l) for the text of the element <city>
- stateOrProvinceName (st) for the text of the element <region>
- postalCode for the text of the element <code>
- countryName (c) for the text of the element <country>
- telephoneNumber for the text of the element <phone>
- mail for the text of the element <email>
- labeledURI for the text of the element <uri>

dcCreatorPointer is used for pointing to such a person entry.

4.3.3. The date Element

"The "date" element identifies the publication date of the document. It consists of a month and a year, e.g.,

```
<date month="February" year="1999" />
```

The "date" element also has an optional day attribute"

dcDate is used for storing the text of the element "date" in the following format: <month> <year>.

4.3.4. The area Element

"The "area" elements identify a general category for the document (e.g., one of "Applications", "General", "Internet", "Management", "Operations", "Routing", "Security", "Transport", or "User")"

dcPublisher is used for storing the text of the element "area". The name of the area has to be prefixed by "IETF area".

4.3.5. The workgroup Element

"The "workgroup" elements identify the IETF working groups that produced the document".

dcPublisher is also used for storing the text of the element "workgroup". The name of the working group has to be followed by "Working Group".

4.3.6. The keyword Element

"The "keyword" elements identify useful search terms".

dcSubject is used for storing the text of the element "keyword".

4.3.7. The abstract Element

"A document may have an "abstract" element, which contains one or more "t" elements [...]. In general, only a single "t" element is present, e.g.,

```
<abstract>
  <t>This memo presents a technique for using XML
  (Extensible Markup Language) as a source format
  for documents in the Internet-Drafts (I-Ds) and
  Request for Comments (RFC) series.</t>
</abstract>
```

The "t" element is defined as containing "any number and combination of paragraphs, lists, and figures".

dcDescription is used for storing the text of the element "abstract".

4.3.8. The note Element

"A document may have one or more "note" elements, each of which contains one or more "t" elements [...]. There is a mandatory "title" attribute. In general, the "note" element contains text from the IESG, e.g.,

```
<note title="IESG Note">
  <t>The IESG has something to say.</t>
</note>
```

dcDescription may be used for the text of the element "note". It then should start with the value of the attribute title.

4.3.9. The copyright status

The copyright statement is to be stored in the "ipr" attribute of the "rfc" element which surrounds the whole text (see above section 4.3.).

Three different copyright statements are defined:

- full2026: "indicating that the document is in full conformance with all the provisions of Section 10 of [RFC 2026]"
- noDerivativeWorks2026: "indicating that the document is in full conformance with all the provisions of Section 10 of [RFC 2026] except that the right to produce derivative works is not granted"

- none: "indicating that the document is NOT offered in accordance with Section 10 of RFC 2026, and the author does not provide the IETF with any rights other than to publish as an Internet-Draft."

dcRights, as specified in section 4.1.16., may be used for the values of the attribute "ipr" of the "rfc" element.

4.3.10. The memo's status and table of contents

"Note that text relating to the memo's status, copyright notice, or table of contents is not included in the document's markup -- this is automatically inserted by an XML application when it produces either a text or HTML version of the document.

For storing the memo's status and the table of contents, dcDescription may be used with the first line containing "Status of this document" and "Table of contents" respectively.

4.3.11. The document name

"Finally, if the Internet-Draft is being submitted to an automated process, then the "docName" attribute should be present in the "<rfc>" tag at the beginning of the file. The value of this attribute contains the document (not file) name associated with this Internet-Draft"

Thus rather, e.g., "draft-mrose-writing-rfcs-01" than "draft-mrose-writing-rfcs-01.txt".

dcSource as specified in section 4.1.12. may be used values of the attribute docName of the "rfc" element.

4.4. Other sections specified in [RFC 2629]

Besides the front section, [RFC 2629] also specifies a middle and a back section.

For middle element following elements are specified:

- section element that is used to automatically number the sections of an RFC
- t element that may contain any number and combination of paragraphs, lists, and figures
- list element that contains t elements for items in a list
- figure element for "ASCII artwork"
- xref element used to cross-reference sections, figures, and references
- eref element used to reference external documents
- iref element used to add information to an index
- vspace element to provide formatting guidance to the XML application

For back element following elements are specified:

- references element that contains the document's bibliography
- reference element that contains single references
- section element for appendices

4.4.1. the figure element

To completely store the structured information in LDAP, schema for all these elements of the middle and back element should be provided. In the scope of this document of these only the figure element is of more interest, because schema definitions would appear within this element.

"The "figure" element groups an optional "preamble" element, an "artwork" element, and an optional "postamble" element together. The "figure" element also has an optional "anchor" attribute

that is used for cross-referencing with the "xref" element [...]. There is also an optional "title" attribute that identifies the title of the figure.

The "preamble" and "postamble" elements, if present, are simply text. If a cross-reference is needed to a section, figure, or reference, the "xref" element [...] is used; similarly, if an external-reference is needed, the "eref" element [...] is used. Indexing of text is provided by the "iref" element [...].

The "artwork" element, which must be present, contains "ASCII artwork". Unlike text contained in the "t", "preamble", or "postamble" elements, both horizontal and vertical white space is significant in the "artwork" element."

Thus the structure is:

```
<figure>
  <preamble>
</preamble>
  <artwork>
</artwork>
  <postamble>
</postamble>
</figure>
```

The artwork element can be put into the following XML construct to tell the XML application to not interpret any character within this structure:

```
<![CDATA[  ]]>
```

With this structure only the string "]" is not allowed to be included in the artwork.

"Finally, the "artwork" element has two optional attributes: "name" and "type". The former is used to suggest a filename to use when storing the content of the "artwork" element, whilst the latter contains a suggestive data-typing for the content."

Schema could easily be automatically detected, if the value of the attribute type would be "LDAP schema".

With the definition of additional sub elements for the figure element, additional metadata as those specified in the following chapter could be stored together with the schema definitions.

5. Metadata specified by the IETF schema WG

As mentioned in [RegPolicy], [SchemaMeta] specifies a MIME directory profile for content transfer and encoding of metadata elements used for cataloguing schema listings in a directory schema listing service. This schema was defined as requirement for the DSR.

5.1. MIME types for schema metadata and their LDAP representation

In the following, all metadata defined in [SchemaMeta] as text/directory MIME types will be described as well as an LDAP representation specified for each of them. All quotations in this section are from [SchemaMeta], if not stated otherwise. Although being literal quotes, they had been reformatted in tables and the references had been adapted to the references in [RegBib].

5.1.1. listingName

"Type purpose:	To represent a globally unique identifier for the listing name
Type encoding:	8bit

Type valuetype:	text, with the following syntax (specified using the BNF in [RFC 822]): name = base-component "." sequence "." version base-component = "base" / oid oid = oid-component *("." oid-component) oid-component = 1*DIGIT DIGIT = <any ASCII decimal digit (0x30 - 0x39)> sequence = NZDIGIT *DIGIT NZDIGIT = <any DIGIT except "0" (0x30)> version = sequence
Type special notes:	This type MUST be single-valued. A language parameter MUST NOT be used with this type. For published listings, a value of this type is an OID constructed by the primary listing repository operator based on a root OID administered by that operator, a listing sequence number generated by that operator, a listing version number assigned by that operator, and a file type indicator. For listing requests, a place holder for the root OID is used in place of the actual base OID administered by the primary listing repository operator. This place holder is simply the text string "base" as indicated in the <base-component> rule above."

"The 'listingName' type value MUST be created by the primary listing repository operator."

For LDAPschema, a value of this type is an OID constructed by the schema registry operator based on following root OID:

1.3.6.1.4.1.10126.9.1

followed by a schema sequence number generated by that operator, a listing version number assigned by that operator, and a file type indicator, as specified in [SchemaFile] and described in section 5.3.1. of [RegIntro].

For the registration requests, the string "base" may be used as a place-holder for the root OID in place of the actual base OID administered by the schema registry operator. The string "to be assigned" would be more self-explanatory. The reason for including this type as mandatory for requests (see below sections 5.1.24. and 5.1.25.) seems to be only to keep the identical requirements for request and listing.

The LDAP representation for listingName is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.1
  NAME 'srListingName'
  DESC 'globally unique identifier for the schema listing name '
  EQUALITY objectIdentifierMatch
  SUBSTR objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
  SINGLE-VALUE )

```

5.1.2. listingTitle

"Type purpose:	To represent a real world title of a listed schema unit or schema pak.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC 822]): utf8-text = 1*<any character encoded according to [RFC 2279]>
Type special notes:	This type MAY be multi-valued. A language parameter MUST be used with this type.

	A value of this type MAY contain local or native version numbers or other version indicators for listed schema. Such schema version information MUST be treated as opaque by implementors."
--	---

Note: Throughout this document all references in [SchemaMeta] to "RFC 2044" have been replaced by [RFC 2279] which obsoletes RFC 2044.

The term "schema unit" is defined as:

"a related or grouped set of object attributes that form a discrete unit within the context of a schema for a particular protocol; examples include an LDAP object class".

For the DSR, this definition has to be reworded to make it include all kinds of schema elements, besides an objectclass with its attributes. Thus the following definition is proposed here:

A single schema element of the LDAP protocol, examples include an attribute type, object class, attribute syntax, etc.

The term "schema pak" is defined as:

"a related or grouped set of schema units that collectively specify a schema associated with a particular protocol; an example of a schema pak is the set of LDAP object classes specified in [RFC 2256]."

Listing title does not necessarily have to be the same as the title of the specification document. If the schema consists of only one object class, a good schema title would be the name of the object class. A good title for the single schema element listings would be the name of the schema element.

The LDAP representation for listingTitle is:

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.2
  NAME 'srListingTitle'
  DESC 'a real world title of a listed schema or schema element'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The language parameter is represented by the language tag option as specified in [RFC 2596].

5.1.3. listingUse

"Type purpose:	To represent a statement of intended use for a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]): utf8-text
Type special notes:	This type MAY be multi-valued. A language parameter MUST be used with this type. A value of this type is an in-line text description of the intended use of a listing and MAY include embedded CRLF characters."

The LDAP representation for listingUse is

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.3
  NAME 'srListingUse'
```

```
DESC 'text description of the intended use of a schema or
      schema element'
SUP description )
```

For a definition of the attribute description see section 4.2.14.

The language parameter is represented by the language tag option as specified in [RFC 2596].

5.1.4. specFile

"Type purpose:	To represent a file name in the schema listing repository for a schema unit content constructed using an appropriate profile of [RFC 2425].
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC 822]): <pre>fname = <a file name as specified in [SchemaFile]> ; all [SchemaFile] <type> values except ; "meta-unit" and "0" MAY be used to ; construct <fname> values</pre>
Type special notes:	When used in schema unit listings and schema unit listing requests, this type MUST be single-valued. When used in schema pak listings and schema pak listing requests, this type MUST be multi-valued. A language parameter MUST NOT be used with this type."

[SchemaFile] specifies the syntax of file names as follows:

"All file names for listing meta data and listing content MUST comply with the following BNF [RFC822] grammar:

```
file-name = sequence "." listversion "." type
sequence = ("0" / "current") / NZDIGIT 0*<DIGIT>
           ; initialized to one (1) for first schema listing
           ; increments by one (1) for each successive schema
           ; listing name
type = "meta-unit" /           ; these <type> values are defined
      "ldap" /                 ; for the initial release of the
      "pak-ldap" /             ; schema listing service
      "whois++" /
      "pak-whois++" /         ; other <type> values may be defined
      "rwhois" /               ; according to community needs in
      "pak-rwhois" /           ; the future
      "whois" /
      "pak-whois"             ; this document will be updated or
                              ; obsoleted when additional <type>
                              ; values are defined

listversion = 1*<DIGIT>
NZDIGIT = <any DIGIT except "0" (0x30)>
DIGIT = <any ASCII decimal digit (0x30 - 0x39)> "
```

[SchemaFile] specifies as an alternative filename a numeric only form:

"Filenames MAY be specified as an OID by prepending the OID value used as a root for the service filename and swapping alphabetic tokens for their numeric equivalent according to the following table:

Token	Number
current	0
meta-unit	0

```

ldap          1
pak-ldap     2
whoispp      3
pak-whoispp  4
rwhois       5
pak-rwhois   6
whois        7
pak-whois    8"

```

For the use in the DSR in it's first operational phase this proposed OID version of the filename will be implemented, whereby only 0 for "meta-unit", 1 for "ldap" and 2 for "pak-ldap" will be used. Other tokens and their numeric equivalents may be introduced in later phases.

The LDAP representation for specFile in schema unit listings is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.4
  NAME 'srSchemaUnitSpecFile'
  DESC 'object identifier as filename for a single schema unit
        listing'
  EQUALITY objectIdentifierMatch
  SUBSTR objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
  SINGLE-VALUE )

```

The LDAP representation for specFile in schema pak listings is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.5
  NAME 'srSchemaUnitSpecFiles'
  DESC 'object identifier as filename for multiple schema unit
        listings'
  EQUALITY objectIdentifierMatch
  SUBSTR objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )

```

5.1.5. relatedTo

"Type purpose:	To represent an indication of a relationship of a published listing or listing request with another published listing or listing request.
Type encoding:	8bit
Type valuetype:	<p>text, with the following syntax (specified using the BNF in [RFC 822]):</p> <pre> related = md-filename *SPACE "\$" *SPACE related-option md-filename = <a metadata file name as specified in [SchemaFile]> related-option = "obsoletes" / "obsoleted-by" / "updates" / "inherits" / vendor-option vendor-option = ("x-" / "X-") vendor-name "-" vendor-specific-relationship vendor-name = 1*TOKEN vendor-specific-relationship = 1*TOKEN TOKEN = <any CHAR except specials, SPACE, CRLF, CTL, and "-"> CHAR = <any ASCII character (0x00 - 0x7f)> specials = "(" / ")" / "<" / ">" / "@" ; MUST be in quoted- / "," / ";" / ":" / "\" / "<" ; string, to use / "." / "[" / "]" ; within a word <"> = <an ASCII quote mark (0x22)> SPACE = <ASCII SP, space (0x20)> CRLF = CR LF </pre>

	CR = <ASCII CR, carriage return (0x0d)> LF = <ASCII LF, line feed (0x0a)> CTL = <any ASCII control character (0x00 - 0x1f) and DEL (0x7f)>
Type special notes:	This type MAY be multi-valued. If a listing is related to another listing, this type is REQUIRED, otherwise the use of this type is OPTIONAL. A language parameter MUST NOT be used with this type. This type is used to indicate relationships between published listings and listing requests as well as between one or more listing requests being submitted for review in parallel. Examples of such relationships include deprecation, revision, inheritance, and those specific to a particular vendor."

"The 'relatedTo' type value MUST be provided by the schema writer as a part of a listing request if the listing proposed in the request has a relationship to published listings and/or other listing requests being reviewed."

The LDAP representation for relatedTo is the following:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.6
  NAME 'srRelatedToListing'
  DESC 'object identifier as filename pointing to a related
        schema listing'
  EQUALITY objectIdentifierMatch
  SUBSTR objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )

```

The md-filename part of the above BNF definitions is stored in this attribute. The related-option part is represented by the same method as the one specified for dcRelation options above in section 4.1.14. Thus following sub-attributes are defined:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.7
  NAME 'srObsoletes'
  SUP srRelatedToListing

```

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.8
  NAME 'srObsoletedBy'
  SUP srRelatedToListing

```

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.9
  NAME 'srUpdates'
  SUP srRelatedToListing

```

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.10
  NAME 'srUpdatedBy'
  SUP srRelatedToListing

```

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.11
  NAME 'srInherits'
  SUP srRelatedToListing

```

The vendor specific relationship is not supported by the DSR.

5.1.6. contactName

"Type purpose:	To represent the name of the contact person, organization, or role for a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]): utf8-text = 1*<any character encoded according to [RFC 2279]>
Type special notes:	This type MUST be single-valued. A language parameter MUST NOT be used with this type."

The term "contact person" is defined as:

"the name of the individual who holds the authority to update a listing and who should be contacted if questions or concerns arise related to a listing or listing request"

The LDAP representation for contactName is the following:

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.13
  NAME 'srContactName'
  DESC 'single valued name'
  SUP name
  SINGLE-VALUE )
```

This attribute, as well as all other attributes about the contact person, will be stored in a single person entry. The schema and schema pak listing entries thus have to include a pointer to that person entry. The LDAP representation of that pointer is the following:

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.14
  NAME 'srContactPersonPointer'
  DESC 'pointer to the entry about the contact person,
  organization, or role for a listing'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
  SINGLE-VALUE )
```

5.1.7. contactLanguage

"Type purpose:	To represent a language understood by the contact person, organization, or role for a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]): c-lang = <a language tag as defined in [RFC 1766]>
Type special notes:	This type MAY be multi-valued. A language parameter MUST NOT be used with this type."

Since this specifies a multi valued attribute, another attribute type than the single valued preferredLanguage attribute type (see above section 4.2.13.) has to be used.

The LDAP representation of contactLanguage thus is:

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.15
  NAME 'srContactLanguage'
  DESC 'language understood by the author or contact person,
  organization, or role for a listing or a specification
```

```

document'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

```

5.1.8. contactEmail

"Type purpose:	To represent the electronic mail address of the contact person, organization, or role for a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]): <pre> c-email = local-part "@" domain-part domain-part = sub-domain *("." sub-domain) sub-domain = 1*<any CHAR, except specials, SPACE, CRLF, and CTL> CHAR = <any ASCII character (0x00 - 0x7f)> specials = "(" / ")" / "<" / ">" / "@" ; MUST be in quoted- / "," / ";" / ":" / "\" / "<" ; string, to use / "." / "[" / "]" ; within a word <"> = <an ASCII quote mark (0x22)> SPACE = <ASCII SP, space (0x20)> CRLF = CR LF CR = <ASCII CR, carriage return (0x0d)> LF = <ASCII LF, line feed (0x0a)> CTL = <any ASCII control character (0x00 - 0x1f) and DEL (0x7f)> </pre>
Type special notes:	This type MUST be single-valued. A language parameter MUST NOT be used with this type."

Since this specifies a single valued attribute, another attribute type than the multi valued mail attribute type (see above section 4.2.11.) has to be used.

The LDAP representation of contactEmail thus is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.16
  NAME 'srContactEmail'
  DESC 'electronic mail address of a person, organization, or
  role'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256}
  SINGLE-VALUE )

```

5.1.9. contactPhone

"Type purpose:	To represent the voice telephone number of the contact person, organization, or role for a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]): <pre> c-phone = 1*<any CHAR, except CTL, CRLF> ; MUST use full international form (e.g., +1 908 582 2409) </pre>

	; as specified in [E.123] CHAR = <any ASCII character (0x00 - 0x7f)> CRLF = CR LF CR = <ASCII CR, carriage return (0x0d)> LF = <ASCII LF, line feed (0x0a)> CTL = <any ASCII control character (0x00 - 0x1f) and DEL (0x7f)>
Type special notes:	This type MUST be single-valued. A language parameter MUST NOT be used with this type."

Since this specifies a single valued attribute, another attribute type than the multi valued telephoneNumber attribute type (see above section 4.2.10.) has to be used.

The LDAP representation of contactTelephone thus is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.17
  NAME 'srContactPhone'
  DESC 'voice telephone number of a person, organization, or role'
  EQUALITY telephoneNumberMatch
  SUBSTR telephoneNumberSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.50{32}
  SINGLE-VALUE )

```

5.1.10. contactAddress

"Type purpose:	To represent the postal address of the contact person, organization, or role for a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]): c-addr = postal-string *5(*SPACE "\$" *SPACE postal-string) postal-string = 1*<any character, except "\$", encoded according to [RFC2279]> SPACE = <ASCII SP, space (0x20)>
Type special notes:	This type MUST be single-valued. A language parameter MUST NOT be used with this type."

The LDAP representation for contactAddress is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.18
  NAME 'srContactAddress'
  DESC 'postal address of a person, organization, or role'
  EQUALITY caseIgnoreListMatch
  SUBSTR caseIgnoreListSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.41
  SINGLE-VALUE )

```

5.1.11. authName

"Type purpose:	To represent the name of the listing authority contact for a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]):

	utf8-text = 1*<any character encoded according to [RFC 2279]>
Type special notes:	This type MUST be single-valued. A language parameter MUST NOT be used with this type. The value of this type MAY be identical to the value of the 'contactName' type defined above "

The term "listing authority contact" is defined as:

"the name of the individual who holds authority to replace a contact person; can be either the contact person for a listing or an alternate contact within the organization to which the contact person belongs (this allows one person organizations to list schema)"

The LDAP representation for authName is the attribute type srContactName as described in section 5.1.6.

This attribute, as well as all other attributes about the listing authority contact, will be stored in a single person entry. The schema and schema pak listing entries thus have to include a pointer to that person entry. The LDAP representation of that pointer is the following:

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.19
  NAME 'srAuthContactPointer'
  DESC 'pointer to the entry about the listing authority contact
  person, organization, or role'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
  SINGLE-VALUE )
```

5.1.12. authLanguage

"Type purpose:	To represent a language understood by the listing authority contact for a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]): lac-lang = <a language tag defined in [RFC 1766]>
Type special notes:	This type MAY be multi-valued. A language parameter MUST NOT be used with this type."

The LDAP representation of authLanguage is the attribute type srContactLanguage as specified in section 5.1.7.

5.1.13. authEmail

"Type purpose:	To represent the electronic mail address of the listing authority contact for a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]): c-email = local-part "@" domain-part domain-part = sub-domain *("." sub-domain)

	<pre> sub-domain = 1*<any CHAR, except specials, SPACE, CRLF, and CTL> CHAR = <any ASCII character (0x00 - 0x7f)> specials = "(" / ")" / "<" / ">" / "@" ; MUST be in quoted- / ", " / ";" / ":" / "\" / "<" ; string, to use / "." / "[" / "]" ; within a word "<" = <an ASCII quote mark (0x22)> SPACE = <ASCII SP, space (0x20)> CRLF = CR LF CR = <ASCII CR, carriage return (0x0d)> LF = <ASCII LF, line feed (0x0a)> CTL = <any ASCII control character (0x00 - 0x1f) and DEL (0x7f)> </pre>
Type special notes:	<p>This type MUST be single-valued.</p> <p>A language parameter MUST NOT be used with this type.</p> <p>The value of this type MAY be identical to the value of the 'contactEmail' type defined above. "</p>

The LDAP representation of authEmail is the attribute type srContactEmail as specified in section 5.1.8.

5.1.14. authPhone

"Type purpose:	To represent the voice telephone number of the listing authority contact for a listing.
Type encoding:	8bit
Type valuetype:	<pre> text, with the following syntax (specified using the BNF in [RFC822]): c-phone = 1*<any CHAR, except CTL, CRLF> ; MUST use full international form (e.g., +1 908 582 2409) ; as specified in [E.123] CHAR = <any ASCII character (0x00 - 0x7f)> CRLF = CR LF CR = <ASCII CR, carriage return (0x0d)> LF = <ASCII LF, line feed (0x0a)> CTL = <any ASCII control character (0x00 - 0x1f) and DEL (0x7f)> </pre>
Type special notes:	<p>This type MUST be single-valued.</p> <p>A language parameter MUST NOT be used with this type.</p> <p>The value of this type MAY be identical to the value of the 'contactPhone' type defined above. "</p>

The LDAP representation of authPhone is the attribute type srContactPhone as specified in section 5.1.9.

5.1.15. authAddress

"Type purpose:	To represent the postal address of the listing authority contact for a listing.
Type encoding:	8bit

Type valuetype:	text, with the following syntax (specified using the BNF in [RFC822]): <pre>c-addr = postal-string *5(*SPACE "\$" *SPACE postal-string) postal-string = 1*<any character, except "\$", encoded according to [RFC2279]> SPACE = <ASCII SP, space (0x20)></pre>
Type special notes:	This type MUST be single-valued. A language parameter MUST NOT be used with this type. The value of this type MAY be identical to the value of the 'contactAddr' type defined above. The value of this type MAY be identical to the value of the 'contactAddr' type defined above. "

The LDAP representation for authAddress is the attribute type srContactAddress as specified in section 5.1.10.

5.1.16. specURL

"Type purpose:	To represent a URL referring to a single schema unit content file.
Type encoding:	8bit
Type valuetype:	uri, formatted as a URL [RFC 1738].
Type special notes:	This type MAY be multi-valued. A language parameter MUST NOT be used with this type."

Note: the reference in [SchemaMeta] to [RFC 1738] is updated (not obsoleted) by [RFC 2396].

For a definition of the term "schema unit" see section 5.1.2.

Values for the specURL type:

"MUST be provided by the primary schema listing repository operator and MUST NOT be accepted from the schema writer".

The LDAP representation for specURL is:

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.20
  NAME 'srSchemaUnitSpecURL'
  DESC 'URL referring to a single schema unit content file'
  EQUALITY caseExactMatch
  SUBSTR caseExactSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

5.1.17. security

"Type purpose:	To represent a description of security considerations for a single schema unit or schema pak.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC 822]): <pre>utf8-text = 1*<any character encoded according to [RFC 2279]</pre>
Type special	This type MAY be multi-valued if it is used within a schema unit listing metadata

notes:	<p>file.</p> <p>This type MUST have at least two values if present in a schema pak listing file. One of these values MUST be a security considerations description for the schema pak itself. The other value MUST consist of the following text:</p> <p style="padding-left: 40px;">Users of this schema pak listing should read the security type values contained in the metadata file associated with each schema unit content file referenced by a pakMember type value.</p> <p>A language parameter MUST be used with this type.</p> <p>A value of this type is an in-line text description of security considerations and MAY include embedded CRLF characters."</p>
--------	---

The LDAP representation of security type is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.21
  NAME 'srListingSecurityNote'
  DESC 'a description of security considerations for a single
        schema unit or schema pak.'
  SUP description )

```

For a definition of the attribute description see section 4.2.14.

The language parameter is represented by the language tag option as specified in [RFC 2596].

5.1.18. created

"Type purpose:	To represent the date and time at which a listing was published.
Type encoding:	8bit
Type valuetype:	<p>text, with the following syntax (specified using the BNF in [RFC822]):</p> <pre> created = date "T" time "Z" date = 4DIGIT "-" 2DIGIT "-" 2DIGIT ; year-month-day ; e.g., 1997-08-27 time = 2DIGIT ":" 2DIGIT ":" 2DIGIT ; hh:mm:ss ; e.g., 00:00:00 thru 23:59:59 ; MUST be based on GMT DIGIT = <any ASCII decimal digit (0x30 - 0x39)> </pre>
Type special notes:	<p>This type MUST be single-valued.</p> <p>A language parameter MUST NOT be used with this type."</p>

Values for the created type:

"MUST be provided by the primary schema listing repository operator and MUST NOT be accepted from the schema writer".

In the DSR following time stamp will be stored as value for "created":

- if sent via email, the date of the email as stored in the "Date" header
- if sent via FTP, the file date of the file put to the FTP repository

The LDAP representation of "created" is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.22
  NAME 'srListingCreatedTime'

```

```

DESC 'A GMT based timestamp created when a schema pak or schema unit
      was published in the schema registry'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE )

```

5.1.19. moreInfo

"Type purpose:	To represent a labeled reference to external content (not stored in the schema listing repository) related to a listing.
Type encoding:	8bit
Type valuetype:	<p>text, with the following syntax (specified using the BNF in [RFC 822]):</p> <pre> more = uri *SPACE "(" label ")" ; MAY be multi-valued or single-valued uri = <a URI as specified in [RFC 1738]> ; in this case the URI is constrained to ; be a URL as specified in [RFC 1738] label = option [*SPACE "\$" *SPACE checksum] ; only one option is allowed per instance ; of this multi-valued metadata element option = "opaque-schema" / "copyright" / "licensing" / "general" / "image" ; this set of options is intended for use in the initial ; release ; of the schema listing service additional options may be ; defined in other documents ; "opaque-schema" signifies that a file containing ; a [RFC 2425]-based schema unit content not currently ; supported by the listing service or other syntax ; specification for a schema unit is being referenced checksum = <an MD5 checksum [RFC 1321] of the information retrievable via the URL value of <uri>> </pre>
Type special notes:	<p>This type MAY be multi-valued.</p> <p>A language parameter MUST be used with this type.</p> <p>The use of this type is REQUIRED if a schema writer wishes to include references to external content related to a listing. Otherwise, this type MUST NOT be used in forming listing requests or published listings. The rationale for including these external references MAY be related to extensive copyright or right-to-use statements, a requirement external to the schema listing service for vendor branding of a listed schema, or a schema specification of a form not expressible using a [RFC 2425] profile currently supported by the schema listing service."</p>

Note: the reference in [SchemaMeta] to [RFC 1738] is updated (not obsoleted) by [RFC 2396].

"The moreInfo type value is OPTIONAL, but MUST be provided by the schema writer. if this metadata element is to be included in a published listing."

Concerning the use of a labeled URI including a MD5 hash, there might be some issues for the operation of the registry. Although the hash can be used to prove that the information pointed to has been unchanged, it cannot be used to reconstruct the original content, if it had been changed. To make sure that the information given at the URL pointed to still is relevant for the schema, the registry operator will constantly have to check whether information has changed and if so evaluate whether the

information is still relevant. Such a process does not scale. Therefore the DSR in its first operational phase will not use the MD5-feature and instead store a local copy of the information on a DSR server, linking to that copy.

Another advantage of this alternative is that changes of information contributing to the clarification and better understanding of the schema might more often be posted via the comment mechanisms and thus can be provided to the user in a better way.

If the MD5 mechanism will nevertheless be implemented in the DSR at some stage, the policy document will have to be revised in this respect. Especially issues like a complete change of the copyright status (e.g., from GPL to "the usage of this schema in applications XYZ costs plenty of dollars"). This problem could be solved by specifying a new type for copyright statement, which could be represented by the LDAP attribute type `dcRights`, specified above section 4.1.16. Considerations about all this may be influenced by the specifications in [RegBusiness].

The LDAP representation for `moreInfo` as prescribed in [SchemaMeta] is :

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.23
  NAME 'srMD5moreInfoURI'
  DESC 'Uniform Resource Identifier with mandatory label
        including a MD5-Hash of the information pointed to'
  EQUALITY caseExactMatch
  SUBSTR caseExactSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The LDAP representation for `moreInfo` as used in the first phase of the DSR is :

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.24
  NAME 'srMoreInfoURI'
  DESC 'Uniform Resource Identifier with mandatory label'
  EQUALITY caseExactMatch
  SUBSTR caseExactSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The language parameter is represented by the language tag option as specified in [RFC 2596].

5.1.20. caveat

"Type purpose:	To represent a caveat explaining that content obtained by following external references to information not stored in the schema listing repository is outside of the control of the repository.
Type encoding:	8bit
Type valuetype:	text, consisting of the following in-line text value: Information obtained by following external content references expressed using the <code>moreInfo</code> type are outside of the control of the schema listing service operators. Users of this information should be aware that it is possible for this information to change after the referencing listing has been published.
Type special notes:	This type MAY be multi-valued. A language parameter MUST be used with this type. The use of this type is REQUIRED if a schema writer wishes to include references to external content related to a listing. Otherwise, this type MUST NOT be used in forming listing requests or published listings."

The LDAP representation of `caveat` is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.25
  NAME 'srMoreInfoCaveat'
  DESC 'caveat explaining that content obtained by following
        external references to information not stored in the schema
        listing repository is outside of the control of the
        repository '
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

```

The language parameter is represented by the language tag option as specified in [RFC 2596].

This type is not used in the first operational phase of the DSR (see above section 5.1.19.).

5.1.21. listingComments

"Type purpose:	To represent comments which will be attached to a listing.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC 822]): utf8-text = 1*<any character encoded according to [RFC 2279]>
Type special notes:	This type MAY be multi-valued. A language parameter MUST be used with this type. The use of this type is REQUIRED if during review of a listing request, the primary listing repository operator is asked by the reviewers to include particular comments or generic caveats with a listing prior to publication. Values of this type are in-line text comments or generic caveats associated with a schema listing and MAY include embedded CRLF characters. "

As specified in [SchemaProc] and mandated in [RegPolicy] section 6.6.1., such listingComments can also be included after the publication of the schema listing according to a formal procedure, including a review on the mailing list.

The LDAP representation of listingComment for formal comments is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.26
  NAME 'srListingComment'
  DESC 'comments attached to a listing by a formal process'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

```

The language parameter is represented by the language tag option as specified in [RFC 2596].

5.1.22. schemaPak

"Type purpose:	To represent a reference to a schema pak listing of which a schema unit content file is a member.
Type encoding:	8bit
Type valuetype:	text, with the following syntax (specified using the BNF in [RFC 822]): pak-ref = uri *SPACE "(" label ")" ; MAY be multi-valued or ; single-valued uri = <a URI as specified in [RFC 2396]> ; the URI is constrained ; to be a URL as ; specified in [RFC 1738] ; and corresponds to a ; pak listing file

	label = "ldap" / "whoispp" / "rwhois" / "whois"
Type special notes:	<p>Only one <label> value is allowed across all instances of this metadata element within a single schema unit metadata file.</p> <p>The set of <label> values specified above is intended for use in the initial release of the schema listing service. Additional values may be defined in other documents; this document will be updated to reflect additions to the supported set.</p> <p>This element MUST ONLY be used in schema unit listing metadata files."</p>

Note: In the above BNF, the reference to RFC 1630 which is not an IETF standard was replaced by [RFC 2396] which also updates (not obsoletes) [RFC 1738].

"The schemaPak type value **MAY** be provided by either the primary schema listing repository operator or the schema writer when required."

Since in the first operational phase of the DSR only LDAP schema will be stored, there is no need to store the label.

The LDAP representation of schemaPak is:

```

attributetype ( 1.3.6.1.4.1.10126.1.8.3.27
  NAME 'srSchemaPakURI'
  DESC 'URL with optional label pointing to the schemaPak
  listing to which the described schema element listing belongs'
  EQUALITY caseExactMatch
  SUBSTR caseExactSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )

```

5.1.23. pakMember

"Type purpose:	To represent a reference to a single schema unit content file associated with a schema pak listing.
Type encoding:	8bit
Type valuetype:	<p>text, with the following syntax (specified using the BNF in [RFC 822]):</p> <pre> member-ref = uri *SPACE "(" label ")" ; MAY be multi-valued or ; single-valued uri = <a URI as specified in [RFC 2396]> ; the URI is constrained ; to be a URL as ; specified in [RFC 1738] ; and refers to a schema ; unit content file label = "ldap" / "whoispp" / "rwhois" / "whois" </pre>
Type special notes:	<p>A schema pak MUST consist of more than one schema unit. Therefore, this element MUST be multi-valued</p> <p>A schema pak listing MUST only contain member references for a single protocol. Therefore, only one <label> value is allowed per schema pak listing</p> <p>This element MUST ONLY be used in schema pak listings.</p> <p>The value of a <uri> MUST NOT refer to another schema pak listing."</p>

Note: In the above BNF, the reference to RFC 1630 which is not an IETF standard was replaced by [RFC 2396] which also updates (not obsoletes) [RFC 1738].

Since in the first operational phase of the DSR only LDAP schema will be stored, there is no need to store the label.

The LDAP representation of schemaPak is:

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.28
  NAME 'srPakMemberURI'
  DESC 'Uniform Resource Identifier with optional label
        pointing to a single schema unit content file
        associated with a schema pak listing '
  EQUALITY caseExactMatch
  SUBSTR caseExactSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

5.1.24. Object class srSchemaUnitMetadataObject

[SchemaMeta] makes a general remark that all multivalued types must have at least one value:

"Each MIME Directory Type Registration [...] includes a specification of whether or not a particular type is constrained to be single-valued or permitted to be multi-valued. Types that are permitted to be multi-valued MUST have at least one value, unless otherwise noted in the 'Type special notes' component of a type definition."

One could therefore deduct that all multi-valued attributes must be MUST attributes in the respective object class. On the other hand, some types are clearly marked as optional in the sense that there are cases in which they are not needed.

Another passage specifies who is responsible for which data:

"The following types MUST be included by schema writers in schema unit listing requests: listingName, listingTitle, listingUse, specFile, contactLanguage, contactName, contactEmail, contactPhone, contactAddress, authLanguage, authName, authEmail, authPhone, authAddress, and security.

[...]

The 'listingName' type value MUST be created by the primary listing repository operator.

The 'relatedTo' type value MUST be provided by the schema writer as a part of a listing request if the listing proposed in the request has a relationship to published listings and/or other listing requests being reviewed.

Values for the following types MUST be provided by the primary schema listing repository operator and MUST NOT be accepted from the schema writer: specURL, created, listingComments, and pakMember."

This leads to following definition of an LDAP object class for storing schema unit metadata:

```
objectclass ( 1.3.6.1.4.1.10126.1.8.4.1
  NAME 'srSchemaUnitMetadataObject'
  DESC 'object containing metadata on a schema unit according
        to the IETF schema WG specifications'
  SUP top
  AUXILIARY
  MUST ( srListingName $ srListingTitle $ srListingUse $
        srSchemaUnitSpecFile $ srContactPersonPointer $
        srAuthContactPointer $ srListingSecurityNote $
        srSchemaUnitSpecURL $ srListingCreatedTime $
        srSchemaPakURI )
  MAY ( srRelatedToListing $ srMoreInfoURI $ srListingComment ) )
```

Note: This definition will have to be changed, if the MD5-Mechanism discussed in section 5.1.19. is to be supported. In this case srMoreInfoURI has to be replaced by srMD5moreInfoURI and srMoreInfoCaveat has to be added.

5.1.25. Object class srSchemaPakMetadataObject

"The following types MUST be included by schema writers in schema pak listing requests: listingName, listingTitle, listingUse, specFile, contactLanguage, contactName, contactEmail, contactPhone, contactAddress, authLanguage, authName, authEmail, authPhone, authAddress, and security."

Following LDAP object class is used for representing the metadata of a schema pak:

```
objectclass ( 1.3.6.1.4.1.10126.1.8.4.2
  NAME 'srSchemaPakMetadataObject'
  DESC 'object containing metadata on a schema according
    to the IETF schema WG specifications'
  SUP top
  AUXILIARY
  MUST ( srListingName $ srListingTitle $ srListingUse $
    srSchemaUnitSpecFiles $ srContactPersonPointer $
    srAuthContactPointer $ srListingSecurityNote $
    srSchemaUnitSpecURL $ srListingCreatedTime $ srPakMemberURI )
  MAY ( srRelatedToListing $ srMoreInfoURI $ srListingComment ) )
```

Note: This definition will have to be changed, if the MD5-Mechanism discussed in section 5.1.19. is to be supported. In this case srMoreInfoURI has to be replaced by srMD5moreInfoURI and srMoreInfoCaveat has to be added.

5.1.26. Object class srSchemaPerson

The entries that are pointed to by the attributes srContactPersonPointer and srAuthContactPointer contain the contact data of the contact person and the authority contact for a listing.

Such an entry is modelled with the following object class:

```
objectclass ( 1.3.6.1.4.1.10126.1.8.4.3
  NAME 'srSchemaPerson'
  DESC 'object containing metadata on a schema according
    to the IETF schema WG specifications'
  SUP top
  STRUCTURAL
  MUST ( srContactName $ srContactLanguage $ srContactEmail $
    srContactPhone $ srContactAddress )
  MAY ( description $ title ) )
```

5.2. MIME types for LDAP schema elements and their LDAP representation

[RFC 2927] specifies text/directory MIME types for transferring LDAP schema definitions. In the following, these MIME types will be described and an LDAP representation of these specified. All quotations in this section are from [RFC 2927], if not stated otherwise. Although being literal quotes, they had been reformatted in tables and the references had been adapted to the references in [RegBib].

Although there are ldap attributes defined in [RFC 2252] that have the proper syntax for storing these LDAP schema elements, new attributes with the same syntax are defined in this section for the DSR, since it needs normal user attributes that can easily be retrieved by LDAP programs and the attributes defined in [RFC 2252] are specified as operational attributes.

The Object Classes that include the Attribute Types specified in this section are specified in section 6.3.

5.2.1. ldapSchemas

"Type purpose:	To represent the LDAPv3 attribute "ldapSchemas", defined in section A.1.
Type encoding:	8bit

Type valuetype:	text, encoded according to the BNF of section A.2
Type special notes:	Each value of this type specifies the contents of an LDAP schema definition. A definition of each object class, attribute, matching rule, matching rule use and syntax referenced in a value of ldapSchemas MUST either be defined in one of the schemas referenced in the "IMPORTS" field, or present in the content containing this value."

Section A.1. specifies the LDAP attribute ldapSchemas as follows:

```
" ( 1.3.6.1.4.1.1466.101.120.17
  NAME 'ldapSchemas'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.56
  USAGE directoryOperation )
```

Each value of the ldapSchemas attribute defines one schema. Its syntax, given in section A.2, contains the elements needed for an LDAPv3 server to correctly process operations which use the definitions from this syntax."

Section A.2. specifies the LDAP schema definition syntax as follows:

```
" ( 1.3.6.1.4.1.1466.115.121.1.56
  DESC 'LDAP Schema Definition' )
```

Values in this syntax are represented according to the following BNF:

```
LdapSchema = "(" whsp
numericoid whsp
[ "NAME" qdescrs ]
[ "OBSOLETE" whsp ]
[ "IMPORTS" oids ]
[ "CLASSES" oids ]
[ "ATTRIBUTES" oids ]
[ "MATCHING-RULES" oids ]
[ "SYNTAXES" oids ]
whsp ")"
```

The numericoid field uniquely identifies the schema. A new OID should be assigned if any of the fields of the schema change. It is not possible to import definitions from a schema until an OID has been assigned to it.

The "NAME" field contains optional human-readable labels for the schema.

The "OBSOLETE" field is present if the schema is obsolete.

The "IMPORTS" field lists the OIDs of other schemas which are to be incorporated by reference into this schema. It is an error to have an attribute type or object class defined in a schema with the same name but a different OID as an attribute type or object class in an imported schema. It is also an error to import from two schema definitions in which there are attribute types or object classes with the same names but different OIDs.

The "CLASSES" field lists the OIDs of object classes defined in this schema. A schema need not contain any object class definitions. A schema must not contain two object class definitions of the same name but with different OIDs.

The "ATTRIBUTES" field lists the OIDs of attribute types defined in this schema. A schema need not contain any object class definitions. A schema must not contain two attribute type definitions of the same name but with different OIDs.

The "MATCHING-RULES" field lists the OIDs of matching rules defined in this schema. A schema need not contain any matching rules.

The "SYNTAXES" field lists the OIDs of syntaxes defined in this schema. A schema need not contain any syntaxes."

The LDAP representation of ldapSchemas in the DSR is the following:

```

attributetype ( 1.3.6.1.4.1.10126.1.11.3.1
  NAME 'srLdapSchemas'
  DESC 'user attribute for storing ldap schema definitions'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.56
  SINGLE-VALUE )

```

The LDAP attribute ldapSchemas as specified in [RFC 2927] (see above) could not be used, since a user attribute is required so it can be processed as a normal attribute. ldapSchemas is on the contrary specified as operational attribute. The syntax specified in [RFC 2927] could be used.

5.2.2. attributeTypes

"Type purpose:	To represent the LDAPv3 attribute "attributeTypes", defined in section 5.1.6 of [RFC 2252]
Type encoding:	8bit
Type valuetype:	text, encoded according to the BNF of "AttributeTypeDescription" given in section 4.2 of [RFC 2252]
Type special notes:	Each value of the type specifies one LDAPv3 attribute type definition. The syntax and matching rules referenced in a value of attributeTypes MUST either be present in this content, or defined in one of the schemas referenced in the "IMPORTS" field of the ldapSchemas line."

[RFC 2252] specifies following Syntax :

```

"AttributeTypeDescription = "(" whsp
  numericoid whsp           ; AttributeType identifier
  [ "NAME" qdescrs ]       ; name used in AttributeType
  [ "DESC" qdstring ]      ; description
  [ "OBSOLETE" whsp ]
  [ "SUP" woid ]           ; derived from this other
                           ; AttributeType
  [ "EQUALITY" woid        ; Matching Rule name
  [ "ORDERING" woid        ; Matching Rule name
  [ "SUBSTR" woid ]        ; Matching Rule name
  [ "SYNTAX" whsp noidlen whsp ] ; see section 4.3
  [ "SINGLE-VALUE" whsp ]   ; default multi-valued
  [ "COLLECTIVE" whsp ]    ; default not collective
  [ "NO-USER-MODIFICATION" whsp ]; default user modifiable
  [ "USAGE" whsp AttributeUsage ]; default userApplications
whsp ")"

```

For up-to-date comments on this structure see [LDAPModels] which also extends this definition with a last element "extensions" which can be used to provide additional information, e.g., X-ORIGIN in:

```

attributeTypes: ( 2.16.840.1.113730.3.1.2013 NAME 'nsAIMid' DESC
  'Netscape defined attribute type' SYNTAX 2.16.840.1.113730.3.7.1 X-
  ORIGIN 'Netscape Directory Server' )

```

The LDAP representation of attributeTypes in the DSR is the following:

```

attributetype ( 1.3.6.1.4.1.10126.1.11.3.2
  NAME 'srLdapAttributeTypes'
  DESC 'user attribute for storing ldap attribute type definitions'

```

```
EQUALITY objectIdentifierFirstComponentMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.3 )
```

5.2.3. matchingRules

"Type purpose:	To represent the LDAPv3 attribute "matchingRules", defined in section 5.1.8 of [RFC 2252]
Type encoding:	8bit
Type valuetype:	text, encoded according to the BNF of "MatchingRuleDescription" given in section 4.5 of [RFC 2252]
Type special notes:	Each value of the type specifies one matching rule definition. The syntax reference in a value of matchingRules MUST either be present in this content, or defined in one of the schemas referenced in the "IMPORTS" field of the ldapSchemas line."

[RFC 2252] specifies following Syntax :

```
"MatchingRuleDescription = "(" whsp
    numericoid whsp ; MatchingRule identifier
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" whsp ]
    "SYNTAX" numericoid
whsp ")"
```

For up-to-date comments on this structure see [LDAPModels] which also extends this definition with a last element "extensions".

The LDAP representation of attributeTypes in the DSR is the following:

```
attributetype ( 1.3.6.1.4.1.10126.1.11.3.3
    NAME 'srLdapMatchingRules'
    DESC 'user attribute for storing ldap matching rules definitions'
    EQUALITY objectIdentifierFirstComponentMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.30 )
```

5.2.4. objectClasses

"Type purpose:	To represent the LDAPv3 attribute "objectClasses", defined in section 5.1.7 of [RFC 2252]
Type encoding:	8bit
Type valuetype:	text, encoded according to the BNF of "ObjectClassDescription" given in section 4.4 of [RFC 2252]
Type special notes:	Each value of the type specifies one LDAPv3 object class definition. The attributes and object classes referenced in a value of objectClasses MUST either be present in this content, or defined in one of the schema referenced in the "IMPORTS" field of the ldapSchemas line."

[RFC 2252] specifies following Syntax :

```
ObjectClassDescription = "(" whsp
    numericoid whsp ; ObjectClass identifier
    [ "NAME" qdescrs ]
```

```

[ "DESC" qdstring ]
[ "OBSOLETE" whsp ]
[ "SUP" oids ]           ; Superior ObjectClasses
[ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ]
                        ; default structural
[ "MUST" oids ]         ; AttributeTypes
[ "MAY" oids ]          ; AttributeTypes
whsp ")"

```

For up-to-date comments on this structure see [LDAPModels] which also extends this definition with a last element "extensions".

The LDAP representation of Object Classes in the DSR is the following:

```

attributetype ( 1.3.6.1.4.1.10126.1.11.3.4
  NAME 'srLdapObjectClasses'
  DESC 'user attribute for storing ldap object class definitions'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.37 )

```

5.2.5. matchingRuleUses

"Type purpose:	To represent the LDAPv3 attribute "matchingRuleUse", defined in section 5.1.9 of [RFC 2252]
Type encoding:	8bit
Type valuetype:	text, encoded according to the BNF of "MatchingRuleUseDescription" given in section 4.5 of [RFC 2252]
Type special notes:	Each value of the type specifies a relationship between a matching rule and attribute types. The matching rule and attribute types referenced in a value of matchingRuleUse MUST either be present in this content, or defined in one of the schemas referenced in the "IMPORTS" statement of the ldapSchemas line."

[RFC 2252] specifies following Syntax :

```

MatchingRuleUseDescription = "(" whsp
  numericoid whsp ; MatchingRule identifier
  [ "NAME" qdescrs ]
  [ "DESC" qdstring ]
  [ "OBSOLETE" ]
  "APPLIES" oids ; AttributeType identifiers
whsp ")"

```

For up-to-date comments on this structure see [LDAPModels] which also extends this definition with a last element "extensions".

The LDAP representation of attributeTypes in the DSR is the following:

```

attributetype ( 1.3.6.1.4.1.10126.1.11.3.5
  NAME 'srLdapMatchingRuleUses'
  DESC 'user attribute for storing ldap matching rule use definitions'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.31 )

```

5.2.6. ldapSyntaxes

"Type purpose:	To represent the LDAPv3 attribute "ldapSyntaxes", defined in section 5.2.4 of [RFC 2252]
----------------	--

	5.3.1 of [RFC 2252]
Type encoding:	8bit
Type valuetype:	text, encoded according to the BNF of "SyntaxDescription" given in section 4.3.3 of [RFC 2252]
Type special notes:	Each value of this type specifies a single LDAPv3 attribute value syntax."

[RFC 2252] specifies following Syntax :

```
SyntaxDescription = "(" whsp
    numericoid whsp
    [ "DESC" qdstring ]
    whsp ")"
```

For up-to-date comments on this structure see [LDAPModels] which also extends this definition with a last element "extensions".

The LDAP representation of attributeTypes in the DSR is the following:

```
attributetype ( 1.3.6.1.4.1.10126.1.11.3.6
    NAME 'srLdapSyntaxes'
    DESC 'user attribute for storing ldap syntax definitions'
    EQUALITY objectIdentifierFirstComponentMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.54 )
```

6. Additional schema for the DSR

In addition to the LDAP schema specified above for storing 1.) bibliographic metadata about the schema specification document (see section 4.1.), 2.) schema listing metadata as defined by the IETF schema WG (see section 5.1.) and 3.) schema for storing schema element definitions as defined by [RFC 2927] (see section 5.2.), more schema is needed for running the DSR. This additional schema is specified in this chapter.

In future versions of the DSR, additional schema elements and other entities that need registration, e.g., LDAP controls and LDAP extended operations (see [RFC 3383]) will be stored and thus need additional schema. Such schema will be specified in future versions of this document.

In the current proposed version of the DSR, following additional schema is needed and thus specified in the following sections:

- schema for additional schema elements not specified in [RFC 2927]
- schema for storing an OID tree
- schema for storing the single parts of schema element definitions
- schema for additional metadata

6.1. Schema for additional schema elements not specified in [RFC 2927]

[RFC 2927] does not specify the LDAP schema elements DIT Structure Rules, DIT Content Rules and Name Forms. This section will specify respective MIME types and LDAP representations for the DSR. The proper formatted MIME Directory Type Registrations can be found in App. A.

The Object Classes that include the Attribute Types specified in this section are specified in section 6.3.

6.1.1. DIT Structure Rules

Type name	DITStructureRules
Type purpose:	To represent the LDAPv3 attribute "dITStructureRules", defined in section 5.4.1 of [RFC 2252]
Type encoding:	8bit
Type valuetype:	text, encoded according to the BNF of "DITStructureRuleDescription" given in section 6.33 of [RFC 2252]
Type special notes:	Each value of this type specifies a single LDAPv3 DIT Structure Rule.
intended usage:	COMMON

[RFC 2252] specifies following Syntax :

```

DITStructureRuleDescription = "(" whsp
    ruleidentifier whsp          ; DITStructureRule identifier
    [ "NAME" qdescrs ]
    [ "DESC" qdstring ]
    [ "OBSOLETE" whsp ]
    "FORM" woid whsp           ; NameForm
    [ "SUP" ruleidentifiers whsp ] ; superior DITStructureRules
    ")"

ruleidentifier = integer

ruleidentifiers = ruleidentifier |
    "(" whsp ruleidentifierlist whsp ")"

ruleidentifierlist = [ ruleidentifier *( ruleidentifier ) ]

```

For up-to-date comments on this structure see [LDAPModels] which also extends this definition with a last element "extensions".

The LDAP representation of attributeTypes in the DSR is the following:

```

attributetype ( 1.3.6.1.4.1.10126.1.11.3.7
    NAME 'srLdapDitStructureRules'
    DESC 'User attribute for storing LDAP DIT Structure Rule
    definitions'
    EQUALITY objectIdentifierFirstComponentMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.17 )

```

6.1.2. DIT Content Rules

Type name	DITContentRules
Type purpose:	To represent the LDAPv3 attribute "dITContentRules", defined in section 5.4.3 of [RFC 2252]
Type encoding:	8bit
Type valuetype:	text, encoded according to the BNF of "DITContentRuleDescription" given in section 6.11 of [RFC 2252]
Type special notes:	Each value of this type specifies a single LDAPv3 DIT Content Rule.
intended usage:	COMMON

[RFC 2252] specifies following Syntax :

```

DITContentRuleDescription = "("
    numericoid ; Structural ObjectClass identifier

```

```

[ "NAME" qdescrs ]
[ "DESC" qdstring ]
[ "OBSOLETE" ]
[ "AUX" oids ] ; Auxiliary ObjectClasses
[ "MUST" oids ] ; AttributeType identifiers
[ "MAY" oids ] ; AttributeType identifiers
[ "NOT" oids ] ; AttributeType identifiers
")"

```

For up-to-date comments on this structure see [LDAPModels] which also extends this definition with a last element "extensions".

The LDAP representation of attributeTypes in the DSR is the following:

```

attributetype ( 1.3.6.1.4.1.10126.1.11.3.8
  NAME 'srLdapDitContentRules'
  DESC 'User attribute for storing LDAP DIT Content Rule
        definitions'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.16 )

```

6.1.3. Name Forms

Type name	NameForms
Type purpose:	To represent the LDAPv3 attribute "ditContentRules", defined in section 5.4.2 of [RFC 2252]
Type encoding:	8bit
Type valuetype:	text, encoded according to the BNF of "NameFormDescription" given in section 6.22 of [RFC 2252]
Type special notes:	Each value of this type specifies a single LDAPv3 Name Form.
intended usage:	COMMON

[RFC 2252] specifies following Syntax :

```

NameFormDescription = "(" whsp
  numericoid whsp ; NameForm identifier
  [ "NAME" qdescrs ]
  [ "DESC" qdstring ]
  [ "OBSOLETE" whsp ]
  "OC" woid ; Structural ObjectClass
  "MUST" oids ; AttributeTypes
  [ "MAY" oids ] ; AttributeTypes
  whsp ")"

```

For up-to-date comments on this structure see [LDAPModels] which also extends this definition with a last element "extensions".

The LDAP representation of attributeTypes in the DSR is the following:

```

attributetype ( 1.3.6.1.4.1.10126.1.11.3.9
  NAME 'srLdapNameForms'
  DESC 'User attribute for storing LDAP Name Form definitions'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.35 )

```

6.2. Schema for storing an OID tree

Most LDAP schema elements have an OID as unique identifier. As discussed in [RegIntrod] and [RegPol], the existing OID registries contain valuable additional information. Thus, pointers to those OID trees should be provided. Instead of putting these pointers in the individual schema element entries discussed below, a separate tree in the DIT of the DSR which is exactly modelled after the OID namespace will be set up that maps relevant OID namespaces. The pointers to the OID registries will be stored in that tree.

Following attributes and Object Classes are needed for storing this information:

6.2.1. srOidComponent

The attribute type srOidComponent is used to store one number of the OID tree. It is used as naming attribute for the single entries of the OID tree.

The LDAP specification of srOidComponent is:

```
attributetype ( 1.3.6.1.4.1.10126.1.12.3.1
  NAME ( 'srOc' 'srOidComponent' )
  DESC 'single component of an oid'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )
```

6.2.2. srNumericOid

The attribute type srNumericOid is used for storing complete OIDs so they can be searchable in the OID tree.

The LDAP specification of srNumericOid is:

```
attributetype ( 1.3.6.1.4.1.10126.1.12.3.2
  NAME ( 'srNumericOid' 'srOid' )
  DESC 'numeric object identifier'
  EQUALITY objectIdentifierMatch
  SUBSTR objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
  SINGLE-VALUE )
```

6.2.3. srOidAuthName

The attribute type srOidAuthName is used for storing the name of the registration authority for a particular OID. The DSR will not aim at finding out all such registration authorities, but will fill in this attribute and the other related attributes (see below) with a best effort approach.

The LDAP specification of srOidAuthName is:

```
attributetype ( 1.3.6.1.4.1.10126.1.12.3.3
  NAME ( 'srOidAuthName'
  DESC 'contact name for an OID registration authority responsible
    for an OID subtree'
  SUP 'name'
```

For a specification of the attribute type name see section 4.2.1.

6.2.4. srExternalLink

The attribute type srExternalLink is used for storing labeled URIs that point to webpages of OID registries that contain additional information about the OID.

The LDAP representation of such an external link is:

```
attributetype ( 1.3.6.1.4.1.10126.1.12.3.4
  NAME 'srExternalLink'
  DESC 'Uniform Resource Identifier with optional label that
```

```
        points to external information'
    SUP labeledURI )
```

For a definition of the attribute description see section 4.2.12.

6.2.5. srInternalLink

The OID entries contain a link to the entry of the schema element in question.

The LDAP representation of such an internal link is:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.6
    NAME 'srInternalLink'
    DESC 'Uniform Resource Identifier with optional label that
        points to additional information internal to the DSR'
    SUP labeledURI )
```

For a definition of the attribute description see section 4.2.12.

6.2.6. srOidObject object class

The object class srOidObject is used for modelling the single entries of the OID tree.

```
objectclass ( 1.3.6.1.4.1.10126.1.12.4.1
    NAME 'srOidObject'
    DESC 'Information about an ASN.1 Object Identifier'
    SUP top
    STRUCTURAL
    MUST ( srOc $ srNumericOid )
    MAY ( displayName $ description $ labeledURI $ mail $
        postalAddress $ srOidAuthName $ telephoneNumber $
        srInternalLink $ srExternalLink) )
```

Usage of the attributes of this object class is:

- The attribute srOc is used as naming attribute.
- The attribute srNumericOid contains the OID.
- The attribute displayName will not be used in the first phase of the DSR.
- The attribute description may be used to store additional information about the OID in question, that was found relevant by the DSR operator.
- The attribute labeledURI is used for pointing to the respective HTML pages of the two OID registries that describe the OID in question.
- The attributes srOidAuthName, mail, postalAddress, and telephoneNumber are used to store name and contact information about the registration authority responsible for the OID in question. The DSR will not aim at finding out all these data, but will fill in these attributes with a best effort approach.

6.2.7. srOidRegistry object class

The entry inside the DIT of the DSR that functions as root of the OID tree is modelled with the structural object class srOidRegistry.

The OID tree is called registry, since all OIDs of schema elements that are included in the DSR will have to be represented in the OID tree. It does not provide a general registry like the ones discussed in [RegIntro].

The LDAP specification of srOidRegistry is:

```
objectclass ( 1.3.6.1.4.1.10126.1.12.4.2
    NAME 'srOidRegistry'
```

```
DESC 'collection of information on OIDs'
SUP top
STRUCTURAL
MUST cn
MAY ( description $ labeledUri ) )
```

The attributes used in this object class are all specified in [RFC 2798]. Their usage in this entry is the following:

- cn (see section 4.2.1.) is used to name the Entry. The name is "OID Registry".
- description (see section 4.2.14.) is used to give a short description of the OID registry and its functions.
- labeledURI (see section 4.2.12.) is used to provide pointers to the OID registries pointed to in the single oid entries. Thus, two values will be stored in the first phase of the DSR:
 - "http://www.alvestrand.no/objectid Object Identifier Registry of H. Alvestrand"
 - "http://asn1.elibel.tm.fr/en/oid/index.htm Object Identifier Tree of ASN.1 Information site"

6.3. Schema for storing the single parts of schema element definitions

The single schema elements will be stored in separate LDAP entries. Besides the attributes for complete definitions as specified in section 5.2., these entries will include attributes for each single part of these definitions. The schema needed for this is defined in this section.

6.3.1. Schema element names

Every LDAP schema elements except LDAP syntaxes have a NAME part in the syntax of their definition. It is used for storing a so-called short name or descriptor as an alternative to using the numeric OID for specifying an LDAP element, e.g. "country" instead of "2.5.6.2". Even for syntaxes names are being used instead of the OID, e.g., "DN Syntax" for "1.3.6.1.4.1.1466.115.121.1.12".

This is specified in [LDAPModels]:

"The NAME field provides a set of short names (descriptors) which are be used as aliases for the OID."

Some LDAP schema elements have more than one name and one of them may be an acronym. The latter is often used as descriptor for naming attributes, i.e. the value of such attributes are used to form the RDN of an entry, e.g., "countryName" and "c". A list of descriptors that may be used for building the RDN instead of an OID is specified in [LDAPDN], where it says:

"If the AttributeType is in the following table of attribute types associated with LDAP [Schema], then the type name string, a <descr>, from that table is used, otherwise it is encoded as the dotted-decimal encoding, a <numericoid>, of the AttributeType's OBJECT IDENTIFIER. The <descr> and <numericoid> is defined in [LDAPModels].

The type name string is not case sensitive.

String	X.500 AttributeType
CN	commonName (2.5.4.3)
L	localityName (2.5.4.7)
ST	stateOrProvinceName (2.5.4.8)
O	organizationName (2.5.4.10)
OU	organizationalUnitName (2.5.4.11)
C	countryName (2.5.4.6)
STREET	streetAddress (2.5.4.9)

```
DC      domainComponent (0.9.2342.19200300.100.1.25)
UID     uid (0.9.2342.19200300.100.1.1)
```

Note: This table lists the complete set of type name strings which all implementations MUST recognize in DN string representation. As no extension could reasonable require all existing implementations be updated to recognize additional type name strings, this table is not extensible."

[LDAPModels] specifies <descr> and <numericoid> as follows:

```
keystring = leadkeychar *keychar
leadkeychar = ALPHA
keychar = ALPHA / DIGIT / HYPHEN
descr = keystring
```

```
number = DIGIT / ( LDIGIT 1*DIGIT )
numericoid = number *( DOT number )
```

Additional descriptors that should be included into this list are currently discussed. For LDAPv2, an IANA registry for such RDN capable descriptors had been maintained. The idea to revive the IANA registry for these descriptors has also come up again and currently seems to be consensus.

In the DSR, these names and these acronyms are stored in the following two attributes.

All descriptors are stored in `srLdapElementDescriptor`, with the following attribute definition:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.1
  NAME 'srLdapElementDescriptor'
  DESC 'Descriptor of an LDAP element'
  SUP name )
```

For a definition of the attribute name see section 4.2.1.

For supporting a new respective IANA registry, the descriptors that may be used for forming RDNs are additionally (redundantly) stored in `srLdapElementRdnDescriptor`, with the following attribute definition:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.2
  NAME 'srLdapElementRdnDescriptor'
  DESC 'Descriptor of an LDAP element that may be used for
  forming an RDN'
  SUP name )
```

6.3.2. schema element description

Every LDAP schema element has a DESC part in the syntax of its definition which [LDAPModels] defines as:

"The DESC field optionally allows a descriptive string to be provided by the directory administrator and/or implementor. While specifications may suggest a descriptive string, there is no requirement that the suggested (or any) descriptive string be used."

In the DSR the description field is stored in the following attribute:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.3
  NAME 'srLdapElementDescription'
  SUP description )
```

For a definition of the attribute description see section 4.2.14.

6.3.3. Obsolete indicator

Every LDAP schema element has an OBSOLETE part in the syntax of its definition which [LDAPModels] defines as:

"The OBSOLETE field, if present, indicates the element is not active."

In the DSR the obsolete field is stored in the following attribute:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.4
  NAME 'srLdapObsolete'
  DESC 'indicator if an LDAP element is active or not'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE )
```

If the element is still active, the value of this attribute is "FALSE", if not, the value is "TRUE".

6.3.4. Subtype of

The LDAP elements attribute type, object class and DITStructureRules can be inherited from another instance of their kind. This inheritance is indicated by the SUP field in the definition of these elements.

[LDAPModels] on attribute types:

"SUP oid specifies the direct supertype of this type;"

and on object classes:

"SUP <oids> specifies the direct superclasses of this object class;"

and on DIT Structure rules:

"SUP identifies superior rules (by rule id);"

In the DSR the sup field in all three cases is stored in the following attribute, containing a DN pointer to the entry of the superior LDAP element:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.5
  NAME 'srLdapSup'
  DESC 'DN pointer to the superior object class, attribute type or
  DIT content rule from which the current element inherits
  qualities'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
```

6.3.5. Matching rule pointer

The LDAP schema element attribute type has three fields in the syntax of its definition, namely EQUALITY, ORDERING, and SUBSTR which contain OIDs of respective types of matching rules.

[LDAPModels]:

"EQUALITY, ORDERING, SUBSTRING provide the oid of the equality, ordering, and substrings matching rules, respectively;"

In the DSR every matching rule will be stored in a separate entry. Thus, instead of the OID of the matching rule, the DN of that entry will be stored in the attribute type entries. Thus, the following three attribute are used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.6
  NAME 'srLdapEqualityMatchingRulePointer'
  DESC 'DN pointer to an entry containing information about an
  equality matching rule'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.10126.1.13.3.7
  NAME 'SRldapOrderingMatchingRulePointer'
  DESC 'DN pointer to an entry containing information about an
```

```

        ordering matching rule'
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.10126.1.13.3.8
  NAME 'SRldapSubstrMatchingRulePointer'
  DESC 'DN pointer to an entry containing information about an
        substrings matching rule'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE )

```

6.3.6. Syntax pointer

The LDAP schema elements attribute type and matching rule have a SYNTAX field in the syntax of their definition which contains the numeric OID of an LDAP syntax. In attribute type descriptions an optional maximal length of the attribute value can be added to the syntax OID.

[LDAPModels] specifies:

"SYNTAX identifies value syntax by object identifier and may suggest a minimum upper bound;" and gives following details:

"A suggested minimum upper bound on the number of characters in a value with a string-based syntax, or the number of bytes in a value for all other syntaxes, may be indicated by appending this bound count inside of curly braces following the syntax's OBJECT IDENTIFIER in an Attribute Type Description. This bound is not part of the syntax name itself. For instance, "1.3.6.4.1.1466.0{64}" suggests that server implementations should allow a string to be 64 characters long, although they may allow longer strings. Note that a single character of the Directory String syntax may be encoded in more than one octet since UTF-8 is a variable-length encoding."

In the DSR every syntax will be stored in a separate entry. Thus, instead of the OID of the syntax, the DN of that entry will be stored in the attribute type and matching rule entries. Thus, the following attribute is used:

```

attributetype ( 1.3.6.1.4.1.10126.1.13.3.9
  NAME 'SRldapSyntaxPointer'
  DESC 'DN Pointer to an entry containing information about an
        LDAP syntax'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE )

```

For storing the suggested minimum upper bound for an attribute value in the DSR the following attribute is used:

```

attributetype ( 1.3.6.1.4.1.10126.1.13.3.10
  NAME 'SRldapAttributeTypeUpperBound'
  DESC 'maximal length of an attribute type'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )

```

6.3.7. Single value indicator

The LDAP schema element attribute type has a SINGLE-VALUE field in the syntax of its definition which indicates whether the attribute type may contain multiple values or only one value.

In the DSR the single value field is stored in the following attribute:

```

attributetype ( 1.3.6.1.4.1.10126.1.13.3.11
  NAME 'srLdapSingleValue'
  DESC 'indicator if an LDAP attribute type is multi valued or not'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE )

```

If the attribute type is multi valued, the value of this attribute is "FALSE", if not, the value is "TRUE".

6.3.8. Collective attribute indicator

The LDAP schema element attribute type has a COLLECTIVE field in the syntax of its definition which indicates whether the attribute type is used as collective attribute.

[LDAPModels]:

"COLLECTIVE indicates this attribute type is collective [X.501];"

[X.501] specifies:

"A user attribute may be defined to be collective. This indicates that the same attribute values will appear in the entries of an entry collection ..."

"Collective attributes shall be multi-valued"

"An operational attribute shall not be defined to be collective"

The usage of collective attributes within LDAP is specified in [LDAPCollective].

In the DSR the collective field is stored in the following attribute:

```

attributetype ( 1.3.6.1.4.1.10126.1.13.3.12
  NAME 'srLdapCollective'
  DESC 'indicator if an LDAP attribute type is collective or not'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE )

```

If the attribute type is collective, the value of this attribute is "TRUE", if not, the value is "FALSE".

6.3.9. No user modification indicator

The LDAP schema element attribute type has a NO-USER-MODIFICATION field in the syntax of its definition which indicates whether the values of the attribute type may be modified by a normal user.

[LDAPModels]:

"NO-USER-MODIFICATION indicates this attribute type is not user modifiable;"

"NO-USER-MODIFICATION requires an operational usage."

In the DSR the no user modification field is stored in the following attribute:

```

attributetype ( 1.3.6.1.4.1.10126.1.13.3.13
  NAME 'srLdapNoUserModification'
  DESC 'indicator if an LDAP attribute type is modifiable by
  the user or not'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE )

```

If the attribute type is user modifiable, the value of this attribute is "TRUE", if not, the value is "FALSE".

6.3.10. Usage indicator

The LDAP schema element attribute type has a USAGE field in the syntax of its definition which indicates the usage of the attribute type.

[LDAPModels]:

"USAGE indicates the application of this attribute type;"

Following values are allowed for this field:

```
"usage = "userApplications"      / ; user
          "directoryOperation"    / ; directory operational
          "distributedOperation"  / ; DSA-shared operational
          "dSAOperation"         ; DSA-specific operational"
```

In the DSR the usage field is stored in the following attribute:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.14
  NAME 'srLdapAttributeTypeUsage'
  DESC 'usage for an attribute type'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
```

6.3.11. Extensions

Opposed to [RFC 2252], [LDAPModels] allows for extensions in the definition of schema elements.

It is not planned to support this feature in the first version of the DSR.

6.3.12. Object class type

The LDAP schema element object class has a field in the syntax of its definition which indicates the type of the object class.

[LDAPModels]:

"the kind of object class is indicated by one of ABSTRACT, STRUCTURAL, or AUXILIARY, default is STRUCTURAL;"

In the DSR the information about the type of an object class is stored in the following attribute

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.17
  NAME 'srLdapObjectClassType'
  DESC 'Indicates whether an object class is ABSTRACT STRUCTURAL
        or AUXILIARY'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
```

6.3.13. MUST, MAY and NOT attributes

The LDAP schema elements object class, Name Forms and DIT Content Rules have MUST and MAY fields in the syntax of their definition, which point to mandatory and optional attribute types. In addition, DIT Content Rule has a NOT field in the syntax of its definition which points to attributes that are not allowed to be used.

[LDAPModels]:

"MUST and MAY specify the sets of required and allowed attribute types, respectively;"

"MUST, MAY, and NOT specify lists of attribute types which are required, allowed, or precluded, respectively, from appearing in entries subject to this DIT content rule;"

"MUST and MAY specify the sets of required and allowed, respectively, naming attributes for this name form;"

"All attribute types in the required ("MUST") and allowed ("MAY") lists shall be different."

In the DSR every attribute type will be stored in a separate entry. Thus instead of the OID of the attribute type, the DN of that entry will be stored. Thus, the following attributes are used for storing MUST, MAY, and NOT attributetypes

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.18
  NAME 'srLdapMustAttributePointer'
  DESC 'DN Pointer to a required attribute'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )

attributetype ( 1.3.6.1.4.1.10126.1.13.3.19
  NAME 'srLdapMayAttributePointer'
  DESC 'DN Pointer to an optional attribute'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )

attributetype ( 1.3.6.1.4.1.10126.1.13.3.20
  NAME 'srLdapNotAttributePointer'
  DESC 'DN Pointer to an attribute not allowed to be used'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.3.14. Applies Attribute

The LDAP schema element Matching Rule Use has an APPLIES field in the syntax of its definition which points to attribute types to which the rules apply.

[LDAPModels]:

"APPLIES provides a list of attribute types the matching rule applies to;"

In the DSR every attribute type will be stored in a separate entry. Thus, instead of the OID of the attribute type, the DN of that entry will be stored. Thus, the following attribute will be used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.21
  NAME 'srLdapAppliesAttributePointer'
  DESC 'DN Pointer to an attribute, a Matching Rule applies to'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.3.15. Structural Object class pointer

The LDAP schema element Name Form has an OC field in the syntax of its definition which points to a structural object class to which the rule applies.

[LDAPModels]:

"OC identifies the structural object class this rule applies to,"

In the DSR every object class will be stored in a separate entry. Thus, instead of the OID of the object class, the DN of that entry will be stored. Thus, the following attribute will be used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.22
  NAME 'srLdapStructOcPointer'
  DESC 'DN pointer to a required objectclass of a Name Form'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.3.16. Rule identifier

The LDAP schema element DIT Structure Rule has a field in the syntax of its definition which specifies an ID for a rule.

[LDAPModels]:

" <ruleid> is the rule identifier of this DIT structure rule;"

ruleid = number

In the DSR following attribute is used to store the rule ID:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.23
  NAME 'srRuleIdentifier'
  DESC 'rule identifier'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )
```

6.3.17. Name Form Pointer

The LDAP schema element DIT Structure Rule has the field FORM in the syntax of its definition which specifies a Name Form associated with the rule.

[LDAPModels]:

"FORM specifies the name form associated with this DIT structure rule;"

In the DSR every Name Form will be stored in a separate entry. Thus, instead of the OID of the Name Form, the DN of that entry will be stored. Thus, the following attribute will be used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.24
  NAME 'srLdapNameFormPointer'
  DESC 'DN pointer to a Name Form'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.3.18. Auxiliary object class pointer

The LDAP schema element DIT Content Rule has the field AUX in the syntax of its definition which specifies an auxiliary object class associated with the rule.

[LDAPModels]:

"AUX specifies a list of auxiliary object classes which entries subject to this DIT content rule may belong to;"

In the DSR every Object Class will be stored in a separate entry. Thus, instead of the OID of the Object Class, the DN of that entry will be stored. Thus, the following attribute will be used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.25
  NAME 'srLdapAuxOcPointer'
  DESC 'DN pointer to an auxiliary object class'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.3.19. Imports pointer

The construct ldapSchema defined in [RFC 2927] (see section 5.2.1.) has the field IMPORTS in the syntax of its definition which specifies other schemas which are to be incorporated by reference into the defined schema.

In the DSR every schema will be represented in a separate entry. Thus following attribute will be used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.26
  NAME 'srLdapSchemaImports'
  DESC 'pointer to an imported schema'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.3.20. Object Class pointer

The construct `ldapSchema` defined in [RFC 2927] (see section 5.2.1.) has the field `CLASSES` in the syntax of its definition which specifies the Object Classes belonging to this schema.

In the DSR every Object Class will be stored in a separate entry. Thus, instead of the OID of the Object Class, the DN of that entry will be stored. Thus, the following attribute will be used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.27
  NAME 'srLdapOcPointer'
  DESC 'pointer to used objectclasses'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.3.21. Attribute Type pointer

The construct `ldapSchema` defined in [RFC 2927] (see section 5.2.1.) has the field `ATTRIBUTES` in the syntax of its definition which specifies the Attribute Types belonging to this schema.

In the DSR every Attribute Type will be stored in a separate entry. Thus, instead of the OID of the Attribute Type, the DN of that entry will be stored. Thus, the following attribute will be used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.28
  NAME 'srLdapAttributeTypePointer'
  DESC 'pointer to used attributetypes'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.3.22. Matching Rule pointer

The construct `ldapSchema` defined in [RFC 2927] (see section 5.2.1.) has the field `MATCHING-RULES` in the syntax of its definition which specifies the Matching Rules belonging to this schema.

In the DSR every Matching Rule will be stored in a separate entry. Thus, instead of the OID of the Matching Rule, the DN of that entry will be stored. Thus, the following attribute will be used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.29
  NAME 'srLdapMatchingRulePointer'
  DESC 'pointer to used matchingrules'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.3.23. Matching Rule Type

Matching Rules have different types, namely Equality, Substring and Ordering Matching Rules. To specify the type in a Matching Rule entry in the DSR the following attribute Type is used:

```
attributetype ( 1.3.6.1.4.1.10126.1.13.3.30
  NAME 'srLdapMatchingRuleType'
  DESC 'Indicates whether a Matching Rule is of type "equality",
  "ordering" or "substring"'
  EQUALITY caseExactMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
```

6.3.24. Object Class `LdapSchemaObject`

The entry to store information about an LDAP Schema is modelled by the following Object Class:

```
objectclass ( 1.3.6.1.4.1.10126.1.13.4.1
  NAME 'srLdapSchemaObject'
  DESC 'LDAP Schema definition'
  STRUCTURAL
  MUST ( cn $ srLdapSchemas )
  MAY ( srLdapObsolete $ srLdapSyntaxPointer $
  srLdapSchemaDescriptor $ srLdapSchemaImports $
```

```

        srLdapOcPointer $ srLdapAttributeTypePointer $
        srLdapMatchingRulePointer )

```

6.3.25. Object Class LdapElementObject

All attributes common to all LDAP schema elements are defined in the following ABSTRACT Object Class which will be inherited by all other object class that model LDAP schema elements:

```

objectclass ( 1.3.6.1.4.1.10126.1.13.4.2
  NAME 'srLdapElementObject'
  DESC 'toplevel objectclass of each LDAP schema object'
  ABSTRACT
  MAY ( srLdapElementDescriptor $ srLdapElementRdnDescriptor $
        srLdapElementDescription $ srLdapObsolete $ cn ) )

```

6.3.26. Object Class LdapAttributeType

The entry to store information about LDAP Attribute Types is modelled by the following Object Class:

```

objectclass ( 1.3.6.1.4.1.10126.1.13.4.3
  NAME 'srLdapAttributeType'
  DESC 'LDAP Attribute Type definition'
  SUP srLdapElementObject
  STRUCTURAL
  MUST ( srNumericOid $ srLdapAttributeTypes
  MAY ( srLdapSup $
        srLdapEqualityMatchingRulePointer $
        srLdapSubstrMatchingRulePointer $
        srLdapOrderingMatchingRulePointer $
        srLdapSyntaxPointer $
        srLdapAttributeTypeUpperBound $
        srLdapSingleValue $
        srLdapCollective $
        srLdapNoUserModification $
        srLdapAttributeTypeUsage ) )

```

6.3.27. Object Class LdapObjectClass

The entry to store information about LDAP Object Classes is modelled by the following Object Class:

```

objectclass ( 1.3.6.1.4.1.10126.1.13.4.4
  NAME 'srLdapObjectClass'
  DESC 'objectClass definition'
  SUP srLdapElementObject
  STRUCTURAL
  MUST ( srNumericOid $ srLdapObjectClasses )
  MAY ( srLdapSup $ srLdapObjectClassType $
        srLdapMustAttributePointer $
        srLdapMayAttributePointer $
        srLdapObjectClasses ) )

```

6.3.28. Object Class LdapSyntax

The entry to store information about LDAP Syntaxes is modelled by the following Object Class:

```

objectclass ( 1.3.6.1.4.1.10126.1.13.4.5
  NAME 'srLdapSyntax'
  DESC 'syntax definition'
  SUP srLdapElementObject
  STRUCTURAL
  MUST ( srNumericoid $ srLdapSyntaxes )

```

6.3.29. Object Class LdapMatchingRule

The entry to store information about LDAP Matching Rules is modelled by the following Object Class:

```

objectclass ( 1.3.6.1.4.1.10126.1.13.4.6
  NAME 'srLdapMatchingRule'
  DESC 'matchingRule definition'
  SUP srLdapElementObject
  STRUCTURAL
  MUST ( srNumericOid $ srLdapMatchingRules $
        srLdapSyntaxPointer )
  MAY srLdapMatchingRuleType )

```

6.3.30. Object Class LdapMatchingRuleUse

The entry to store information about LDAP Matching Rule Uses is modelled by the following Object Class:

```

objectclass ( 1.3.6.1.4.1.10126.1.13.4.7
  NAME 'srLdapMatchingRuleUse'
  DESC 'matchingRuleUse definition'
  SUP srLdapElementObject
  STRUCTURAL
  MUST ( srNumericOid $ srLdapMatchingRuleUses )
  MAY srLdapAppliesAttributePointer )

```

6.3.31. Object Class LdapNameForm

The entry to store information about LDAP Name Forms is modelled by the following Object Class:

```

objectclass ( 1.3.6.1.4.1.10126.1.13.4.8
  NAME 'srLdapNameForm'
  DESC 'nameForm definition'
  SUP srLdapElementObject
  STRUCTURAL
  MUST ( srNumericOid $ srLdapStructOcPointer $
        srLdapMustAttributePointer $ srLdapNameForms )
  MAY ( srLdapMayAttributePointer ) )

```

6.3.32. Object Class LdapDitStructureRule

The entry to store information about LDAP DIT Structure Rules is modelled by the following Object Class:

```

objectclass ( 1.3.6.1.4.1.10126.1.13.4.9
  NAME 'srLdapDitStructureRule'
  DESC 'DIT Structure Rule definition'
  SUP srLdapElementObject
  STRUCTURAL
  MUST ( srRuleIdentifier $ srLdapNameFormPointer $
        srLdapDitStructureRules)
  MAY SRldapSup )

```

6.3.33. Object Class LdapDitContentRule

The entry to store information about LDAP DIT Content Rules is modelled by the following Object Class:

```

objectclass ( 1.3.6.1.4.1.10126.1.13.4.10
  NAME 'srLdapDITContentRule'
  DESC 'DIT Content Rule definition'
  SUP srLdapElementObject
  STRUCTURAL
  MUST ( srNumericOid $ srLdapDitContentRules )
  MAY ( srLdapAuxOcPointer $ srLdapMustAttributePointer $
        srLdapMayAttributePointer $ srLdapNotAttributePointer ) )

```

6.4. Schema for additional metadata

Besides the metadata defined by the IETF schema WG and described in section 5.1., additional metadata have to be stored for the operation of the DSR. These are specified in this section.

6.4.1. reference pointer

Every schema entry and every schema element entry of the DSR will include a pointer to the specification document, where to find text describing it.

Such a reference pointer will be stored in a labeledURI, in which the label has the following format:

```
label = par ( "$" page )
par = number * ( "." number )
page = number
```

Following Attribute Type will be used to store such reference pointers:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.1
  NAME 'SRreferencePointer'
  DESC 'Uniform Resource Identifier pointing to a specification
        document with mandatory label which specifies paragraph
        number and optionally page number'
  EQUALITY caseExactMatch
  SUBSTR caseExactSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

6.4.2. syntax ok

In the DSR automatic Syntax checks will be performed. In the following attribute the result of this check will be stored:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.2
  NAME 'srSyntaxOK'
  DESC 'the syntaxcheck was successfull'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE )
```

6.4.3. oid ok

In the DSR OID checks will be performed. In the following attribute the result of this check will be stored:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.3
  NAME 'srOidOK'
  DESC 'the OID is correct'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE )
```

6.4.4. informal listing comment pointer

In addition to the formal comment mechanism as described in section 5.1.21., [RegPolicy] specifies an additional informal comment mechanism. Hereby an Internet user may make additional comments. These comments are stored in separate entries below the entry to which the comment refers. The latter entry contains a pointer to such entries, which are described in section 6.5.

The LDAP representation of an informal comment pointer is:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.5
  NAME 'srlistingInformalCommentPointer'
  DESC 'Pointer to entries that contain comments attached to a
        listing by a Web form filled out by an Internet user'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

6.4.5. Link to external information

The DSR contains external links that point to additional information about the schema or schema element, especially the respective web pages of the LDAP Schema Viewer which is described in [RegIntro].

The LDAP representation of such an external link is the attribute `srExternalLink` as described in section 6.2.4.

6.4.6. Objectclass `additionalMetadataObject`

In the DSR, following Object Class is used to store the additional metadata specified in this section:

```
objectclass ( 1.3.6.1.4.1.10126.1.14.4.1
  NAME 'srLdapAdditionalMetadataObject'
  DESC 'additional metadata provided for schema registry entries'
  AUXILIARY
  MAY ( srInformalCommentPointer $ srReferencePointer $
        srSyntaxOK $ srOidOK $ srExternalLink ) )
```

6.5. Schema for informal comments entries

In addition to the formal comment mechanism as described in section 5.1.21., [RegPolicy] specifies an additional informal comment mechanism. These are stored in separate entries, which are described in this section.

6.5.1. informal listing comment number

The attribute used to form the RDN of the entries containing informal comments is `srListingInformalCommentNumber`, which contains an integer. The first comment to an entry has the value 1 in this field, the second 2 and so on.

The LDAP representation of an informal comment is:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.10
  NAME 'srListingInformalCommentNumber'
  DESC 'number of an informal comment'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  SINGLE-VALUE )
```

6.5.2. informal listing comment

The attribute to store informal comments is `srListingInformalComment`.

The LDAP representation of an informal comment is:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.11
  NAME 'srListingInformalComment'
  DESC 'comments attached to a listing by a Web form filled out by
        an Internet user'
  SUP srListingComment
  SINGLE-VALUE )
```

For a definition of the attribute `srListingComment`, see section 5.1.21.

6.5.3. informal listing comment type

The informal comments can be classified in following groups: "implementation", "application", "caveat", "historic", "copyright", "licensing", "general", "other". The attribute that stores the type of the comment can have one of these values.

The LDAP representation of an informal comment type is:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.12
  NAME 'srListingInformalCommentType'
  DESC 'Type of a comment attached to a listing by a Web form
        filled out by an Internet user'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

6.5.4. informal listing comment date

The date when the informal comment was created in the web user interface is stored in the DSR.

The LDAP representation of an informal comment date is:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.13
  NAME 'srListingInformalCommentCreatedTime'
  DESC 'A GMT based timestamp created when an informal comment was
        was created in the schema registry'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
```

6.5.5. informal listing comment author

The name of the author of an informal comment is stored in the DSR.

The LDAP representation of an informal comment author is:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.14
  NAME 'srListingInformalCommentAuthor'
  DESC 'Name of an author of an informal comment'
  SUP name
  SINGLE-VALUE )
```

For the definition of the attribute type name see 4.2.1.

6.5.6. informal listing comment author email

The email address of the author of an informal comment is stored in the DSR.

The LDAP representation of an informal comment author email is:

```
attributetype ( 1.3.6.1.4.1.10126.1.14.3.15
  NAME 'srListingInformalCommentAuthorMail'
  DESC 'Name of an author of an informal comment'
  SUP mail
  SINGLE-VALUE )
```

For the definition of the attribute type name see 4.2.11.

6.5.7. Objectclass informalCommentObject

In the DSR, following Object Class is used to store the informal comment as specified in this section:

```
objectclass ( 1.3.6.1.4.1.10126.1.14.4.2
  NAME 'srLdapInformalCommentObject'
  DESC 'information about an informal comment created in the
        schema registry'
  STRUCTURAL
  MUST ( srListingInformalCommentNumber $ srListingInformalComment
        srListingInformalCommentCreatedTime $
        srListingInformalCommentAuthorMail )
  MAY ( srListingInformalCommentType $
        srListingInformalCommentAuthor ) )
```

7. DIT Structure for the Directory Schema Registry

The root of the schema registry is modelled with the objectClass srLdapSchemaNodeObject. The RDN of this entry is "cn= Schema Registry".

The current implementation of the DSR contains two main subtrees, one for the actual schema data and one for additional information about the OIDs. For an overall picture see Diagram 1. In future versions of the registry additional subtrees may be added, e.g. for storing XML schema data.

7.1. Structure of the OID subtree

The OID subtree is modelled with the attribute types and objectclass as described in section 6.2, where each part of the numeric OID is an entry and thus the OID hierarchy is exactly followed by the DIT hierarchy. The objectclass `srOidRegistry` is used to model the entry that specifies the root of this OID subtree. The RDN of this entry is "cn=OidTree", thus its DN is "cn=Schema Registry, cn=OidTree".

7.2. Structure of the schema data subtree

7.2.1. The root entry of the schema data subtree

The subtree for the LDAP schema data has an entry modelled by the object class `srLdapSchemaNodeObject`. The RDN of this entry is "cn=LdapRegistry", thus its DN is "cn=Schema Registry, cn=LDAPRegistry". In case that the DSR will in future be used as registry for other entities than schema, like for example LDAP extensions, another level will be inserted into the hierarchy to create two or more subtrees, one for LDAP schema and others for LDAP extensions, etc.

7.2.2. Schema descriptor

As defined in [RFC2927], each `LdapSchema` has its schema descriptor, which contains a unique name for the schema.

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.30
  NAME 'srLdapSchemaDescriptor'
  DESC 'description of the schema'
  SUP name )
```

For a definition of the attribute name see section 4.2.1.

Every single schema has one entry as root of its subtree. This entry is modelled with the objectclass `srLdapSchemaNodeObject` as described in 7.2.5.

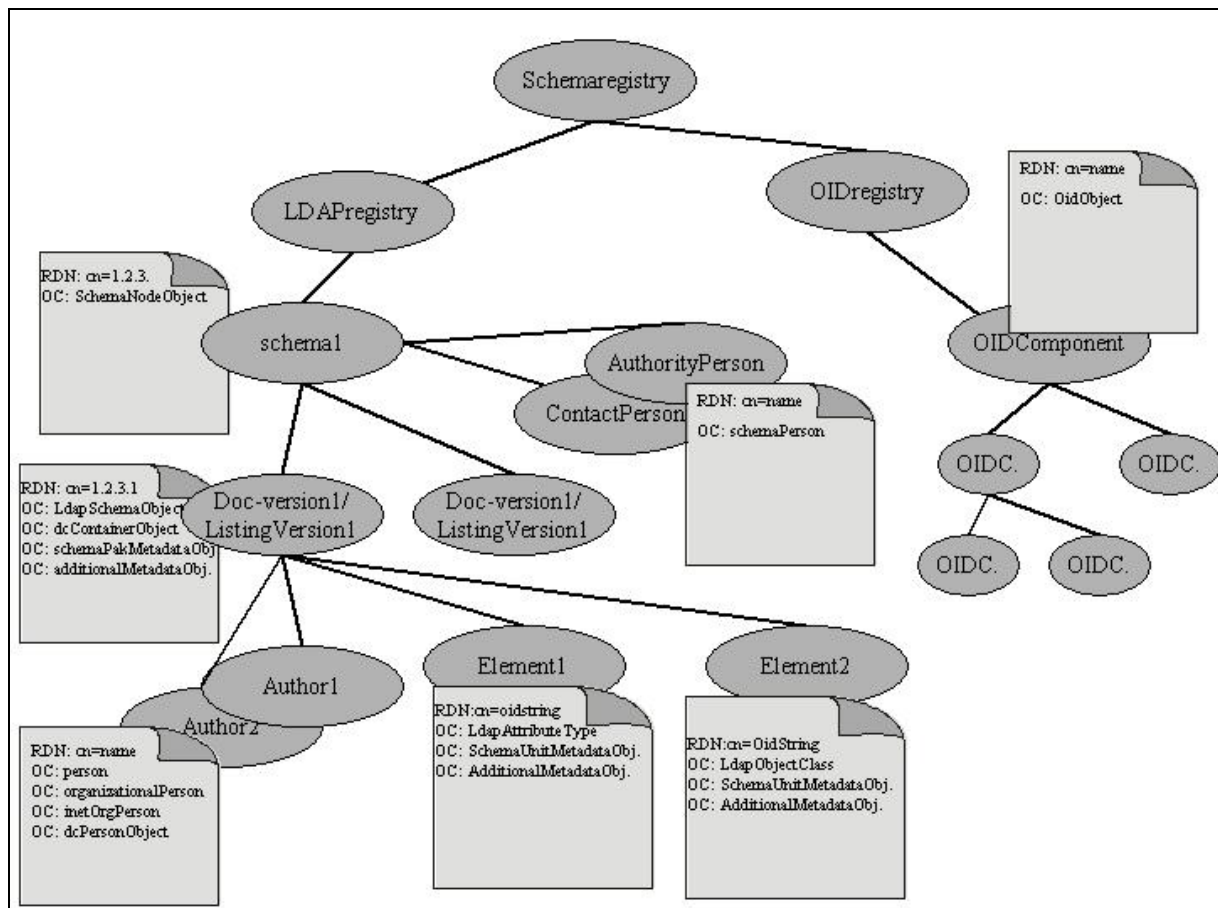


Diagram 1 DIT-Structure of the Directory Schema Registry

7.2.3. Schema versions

Pointers to all registered versions of a schema are stored in the root of the respective schema subtree. They are stored in the following attribute type:

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.31
  NAME 'srLdapSchemaVersions'
  DESC 'pointer to the entries of versions'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
```

7.2.4. Schema newest version

To know, which of the registered versions is the newest a pointer to the newest version is additionally stored in following single value attribute:

```
attributetype ( 1.3.6.1.4.1.10126.1.8.3.32
  NAME 'srLdapSchemaNewestVersion'
  DESC 'the newest version of the schema'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE )
```

7.2.5. Object class schemanode Object

All entries which don't represent a schemadocument/schemaversion, a schema element, an informal comment or a person are modelled with the object class SchemaNodeObject. The LDAP definition of this object class is as follows:

```
objectclass ( 1.3.6.1.4.1.10126.1.8.4.4
  NAME 'srLdapSchemaNodeObject'
  DESC 'structural objectclass for a schema toplevel node'
  STRUCTURAL
  MUST cn
  MAY ( srListingName $ srListingTitle $ srLdapSchemaDescriptor $
    srLdapSchemaVersions $ srLdapSchemaNewestVersion $ dcSubject $
    description) )
```

8. Pointer to References

[RegBib] Gietz, Peter, " Bibliography for the Directory Schema Registry Project", Version 1, Deliverable B of the TERENA project Directory Schema Registry, January 2003

[RegBib] contains all references used in the project documents.

A. MIME Directory Type Registrations

[TBD]

B. Detailed Table of Contents

- 1. Status of this document..... 1
- 2. Introduction..... 1
- 3. Requirements 2
- 4. Bibliographical references in LDAP 2
 - 4.1. The Dublin Core Metadata set and its LDAP representation 3
 - 4.1.1. The issues about DC Qualifiers 3
 - 4.1.2. DC.Title 4
 - 4.1.3. DC.Creator 4
 - 4.1.4. DC.Subject..... 4
 - 4.1.5. DC.Description 5
 - 4.1.6. DC.Publisher 5
 - 4.1.7. DC.Contributor 6
 - 4.1.8. DC.Date 6
 - 4.1.9. DC.Type..... 6
 - 4.1.10. DC.Format 7
 - 4.1.11. DC.Identifyer 7
 - 4.1.12. DC.Source 7
 - 4.1.13. DC.Language 8
 - 4.1.14. DC.Relation..... 8
 - 4.1.15. DC.Coverage 9
 - 4.1.16. DC.Rights..... 10
 - 4.1.17. DC.Audience 10
 - 4.1.18. Object class dcContainer 10
 - 4.2. Additional schema for person information..... 11
 - 4.2.1. Common name 11
 - 4.2.2. surName 11
 - 4.2.3. initials 11
 - 4.2.4. organizationName 12
 - 4.2.5. street 12
 - 4.2.6. locationName 12
 - 4.2.7. stateOrProvinceName 12
 - 4.2.8. postalCode 13
 - 4.2.9. countryName 13

4.2.10.	telephoneNumber	13
4.2.11.	mail	13
4.2.12.	labeledURI	14
4.2.13.	preferredLanguage	14
4.2.14.	description.....	14
4.2.15.	dcPersonObject object class.....	14
4.3.	The Front matter elements of [RFC 2629]	15
4.3.1.	The title Element.....	16
4.3.2.	The author Element	16
4.3.3.	The date Element.....	17
4.3.4.	The area Element	18
4.3.5.	The workgroup Element.....	18
4.3.6.	The keyword Element	18
4.3.7.	The abstract Element.....	18
4.3.8.	The note Element.....	18
4.3.9.	The copyright status.....	18
4.3.10.	The memo's status and table of contents.....	19
4.3.11.	The document name	19
4.4.	Other sections specified in [RFC 2629]	19
4.4.1.	the figure element	19
5.	Metadata specified by the IETF schema WG	20
5.1.	MIME types for schema metadata and their LDAP representation	20
5.1.1.	listingName	20
5.1.2.	listingTitle	21
5.1.3.	listingUse	22
5.1.4.	specFile	23
5.1.5.	relatedTo.....	24
5.1.6.	contactName.....	25
5.1.7.	contactLanguage	26
5.1.8.	contactEmail	27
5.1.9.	contactPhone	27
5.1.10.	contactAddress	28
5.1.11.	authName	28
5.1.12.	authLanguage	29
5.1.13.	authEmail	29
5.1.14.	authPhone.....	30

5.1.15.	authAddress.....	30
5.1.16.	specURL.....	31
5.1.17.	security.....	31
5.1.18.	created.....	32
5.1.19.	moreInfo.....	33
5.1.20.	caveat.....	34
5.1.21.	listingComments.....	35
5.1.22.	schemaPak.....	35
5.1.23.	pakMember.....	36
5.1.24.	Object class srSchemaUnitMetadataObject.....	37
5.1.25.	Object class srSchemaPakMetadataObject.....	38
5.1.26.	Object class srSchemaPerson.....	38
5.2.	MIME types for LDAP schema elements and their LDAP representation.....	38
5.2.1.	ldapSchemas.....	38
5.2.2.	attributeTypes.....	40
5.2.3.	matchingRules.....	41
5.2.4.	objectClasses.....	41
5.2.5.	matchingRuleUses.....	42
5.2.6.	ldapSyntaxes.....	42
6.	Additional schema for the DSR.....	43
6.1.	Schema for additional schema elements not specified in [RFC 2927].....	43
6.1.1.	DIT Structure Rules.....	43
6.1.2.	DIT Content Rules.....	44
6.1.3.	Name Forms.....	45
6.2.	Schema for storing an OID tree.....	46
6.2.1.	srOidComponent.....	46
6.2.2.	srNumericOid.....	46
6.2.3.	srOidAuthName.....	46
6.2.4.	srExternalLink.....	46
6.2.5.	srInternalLink.....	47
6.2.6.	srOidObject object class.....	47
6.2.7.	srOidRegistry object class.....	47
6.3.	Schema for storing the single parts of schema element definitions.....	48
6.3.1.	Schema element names.....	48
6.3.2.	schema element description.....	49
6.3.3.	Obsolete indicator.....	49

6.3.4.	Subtype of.....	50
6.3.5.	Matching rule pointer	50
6.3.6.	Syntax pointer	51
6.3.7.	Single value indicator	51
6.3.8.	Collective attribute indicator	52
6.3.9.	No user modification indicator	52
6.3.10.	Usage indicator	53
6.3.11.	Extensions	53
6.3.12.	Object class type.....	53
6.3.13.	MUST, MAY and NOT attributes.....	53
6.3.14.	Applies Attribute.....	54
6.3.15.	Structural Object class pointer	54
6.3.16.	Rule identifier	54
6.3.17.	Name Form Pointer.....	55
6.3.18.	Auxiliary object class pointer.....	55
6.3.19.	Imports pointer	55
6.3.20.	Object Class pointer.....	56
6.3.21.	Attribute Type pointer.....	56
6.3.22.	Matching Rule pointer	56
6.3.23.	Matching Rule Type	56
6.3.24.	Object Class LdapSchemaObject	56
6.3.25.	Object Class LdapElementObject	57
6.3.26.	Object Class LdapAttributeType	57
6.3.27.	Object Class LdapObjectClass	57
6.3.28.	Object Class LdapSyntax.....	57
6.3.29.	Object Class LdapMatchingRule	57
6.3.30.	Object Class LdapMatchingRuleUse	58
6.3.31.	Object Class LdapNameForm.....	58
6.3.32.	Object Class LdapDitStructureRule	58
6.3.33.	Object Class LdapDitContentRule	58
6.4.	Schema for additional metadata	58
6.4.1.	reference pointer.....	59
6.4.2.	syntax ok.....	59
6.4.3.	oid ok	59
6.4.4.	informal listing comment pointer	59
6.4.5.	Link to external information.....	59

6.4.6.	Objectclass additionalMetadataObject.....	60
6.5.	Schema for informal comments entries	60
6.5.1.	informal listing comment number	60
6.5.2.	informal listing comment	60
6.5.3.	informal listing comment type.....	60
6.5.4.	informal listing comment date	61
6.5.5.	informal listing comment author	61
6.5.6.	informal listing comment author email.....	61
6.5.7.	Objectclass informalCommentObject.....	61
7.	DIT Structure for the Directory Schema Registry	61
7.1.	Structure of the OID subtree	62
7.2.	Structure of the schema data subtree	62
7.2.1.	The root entry of the schema data subtree	62
7.2.2.	Schema descriptor	62
7.2.3.	Schema versions	63
7.2.4.	Schema newest version	63
7.2.5.	Object class schemanode Object.....	63
8.	Pointer to References.....	63
A.	MIME Directory Type Registrations	63
B.	Detailed Table of Contents	64