ETSI TECHNICAL REPORT

ETR 125

March 1994

Source: ETSI TC-TE, EWOS

Reference: DTR/TE-06013
EWOS ETG 005

ICS: 33.080

Key words: OSI, directory

Introduction to OSI Directory functional standards

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
X.400: c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

Contents

Foreword 5

1 Introduction 5

2 OSI Directory 5

3 The Directory Information Base (DIB) 6

4 Use of the Directory for specific purposes 8

5 Static capabilities of co-operating DSAs 8

6 Directory functional standards 8
6.1 Directory A-profiles 9
6.2 Directory F-profiles 9

7 ETSI Technical Reports for the Directory 10
7.1 Error Handling for the Directory 10
7.2 Security architecture for the directory 10
7.3 Interconnection of Directory Management Domains 10

8 Conformance 11

Annex A: Object classes and DIT structure 12

Annex B: Requirements on matching rules for attributes 16

History 18

Blank page

Foreword

ETSI Technical Reports (ETRs) are informative documents resulting from ETSI studies which are not yet appropriate for European Telecommunication Standard (ETS) or Interim European Telecommunication Standard (I-ETS) status. An ETR may be used to publish material which is either of an informative nature, relating to the use or application of ETSs or I-ETSs, or which is immature and not yet suitable for formal adoption as an ETS or I-ETS.

This ETR has been produced by the Terminal Equipment (TE) Technical Committee of the European Telecommunications Standards Institute (ETSI). More specifically, it is the result of a joint effort of experts from the European Workshop for Open Systems (EWOS) EGDIR and ETSI STC-TE 6 (Directory systems). Due to the similarity of objectives, EWOS and ETSI have agreed to issue common texts. The EWOS equivalent to this ETR is known as EWOS Technical Guide (ETG) 005.

1 Introduction

This ETR is intended to give an introduction to the OSI Directory profile work. This version of the ETR is mainly concerned with the European work as specified in M-TI-02. Later versions may be directed toward the International Standardized Profiles (ISPs) work, especially when the current European profiles (ENVs) are replaced by the ISPs.

The Directory functional standards described in this ETR are based on the 1988 edition of OSI Directory specifications, as specified in the International Standard ISO/IEC 9594:1990, Parts 1 to 8, and in the CCITT X.500 1988 Series of Recommendations. These specifications have been developed jointly by ISO/IEC and CCITT, and they are, in the context of functional standards, referred to as the base specifications.

In addition, this ETR introduces other Directory ETRs developed in conjunction with the profile work.

This ETR also includes annexes of a more tutorial nature intended to explain some areas that have been proven to be a little difficult for those not deeply involved in the Directory standardization work.

ISO/IEC and CCITT have completed a new edition of the Directory specifications referred to as the 1992 edition, and it was expected to be available during 1993. This edition will be the subject for future profile work.

2 OSI Directory

The Directory specifications comprise an OSI Application Layer standard specifying principles, protocols, and procedures for storage and retrieval of information about "objects".

Objects can be anything that can be given one or more unambiguous names and for which it is of interest to store information for later retrieval.

The Directory is not intended to be a general purpose database, but has mainly been developed for storing information about objects relevant in the telecommunications arena, such as OSI application-entities, persons, files, distribution lists, etc. The information stored about objects is typically information relevant for communications involving those objects.

The Directory specifications provide a model for the information structure, specify protocols for communication between open systems about directory information and specify procedures that allow the directory information to be distributed among any number of open systems, including procedures for navigation to the open system containing the information to be accessed.

An open system can locally maintain its part of the directory information using any suitable database technique.

An object represented by the Directory always has a so-called distinguished name, which is the "primary" name for the object. In addition, it can also have one or more alias names.

Several independent directories may be created. However, if the names of the objects represented by these directories are all drawn from the same name space, i.e., all names are assured to be different,

such directories can be merged into one directory - **The Directory** (spelled with a capital D). It is also called the global Directory, and it is global in two respects. It is world-wide, and it is common for all possible usages. The base specification assumes such a global Directory.

The set of Directory functional standards described ensures that complying Directory implementations can be integrated into a single Directory.

The element within an open system maintaining and communicating about Directory information is called a Directory System Agent (DSA). A DSA is, in OSI terms, an application-process. The collection of DSAs comprises the Directory. As a special case, the Directory may consist of just a single DSA. Such a non-distributed Directory is called a centralised Directory. The DSA in a non-distributed Directory is called a Centralised DSA; a DSA being part of a distributed Directory is called a Co-operating DSA.

DSAs in different open systems communicate with each other using an OSI Application Layer protocol called the Directory System Protocol (DSP).

An open system can access the Directory by establishing an application-association with one of the DSAs in the Directory. The element of an open system performing that access is called a Directory User Agent (DUA), and represents a single user. An open system may have any number of DUAs. A DUA communicates with a DSA using an Application Layer protocol called the Directory Access Protocol (DAP). The Directory gives a DUA access to Directory information independent of the location of that information and the location of the DUA.

The communication between a DUA and a DSA is always initiated by the DUA and the DSA will never send anything unsolicited to a DUA (except possibly for an A-ABORT). Therefore an inoperative DUA, in contrast to an inoperative DSA, only affects the local user, and only the local user puts requirements on the availability of the DUA.

When a DUA or a DSA sends a request to a DSA to access Directory information in a distributed Directory, and that DSA contains none or only part of the information in question, it can continue to operate in either of two modes:

a) it can forward the request to another DSA supposedly more capable of handling the request and eventually combine the received reply with local results before sending a single result back to the originator. This mode of operation is called Chaining mode. As a special case, identical requests may be chained by a DSA to two or more other DSAs either in parallel or sequentially in order to fulfil a single incoming request;

b) the DSA can return information to the requester about DSAs better able to handle the request. This mode is called referral mode.

The decision to use referral mode, chaining mode, or a combination of the two may be taken independently for each Directory operation.

3 The Directory Information Base (DIB)

An object is represented in the Directory by an entry. An entry is an abstraction of the information stored about the object it represents.

The complete information maintained by the Directory is called the Directory Information Base (DIB).

Every entry in the Directory has a unique name, which is also a name for the corresponding object. The Directory entries are organized into a hierarchical structure, and the entry names reflect that hierarchy. This hierarchical structure can be depicted as a reversed tree, where each node in the tree is an entry. This is illustrated in figure 1. This model of the information structure is called the Directory Information Tree (DIT). The root of the tree is not a real entry, and it has no name. In principle, the naming hierarchy could be arranged in any arbitrary way, but it has been established that the naming hierarchy has to reflect the way names are allocated by naming authorities. As allocation of names is mostly a national undertaken, the first level below the root is made up of entries representing countries. The Directory specification dictates that the country codes defined in ISO 3166 shall be used. Recognised international organizations can also be represented by entries just below the root, but this requires an international registration to ensure unique names for such organizations.

The next level under country entries may reflect localities within the country, like counties, but may also represent organizations having unique names within the country. The latter case is shown in figure 1. Entries below country entries have a name composed of the country name plus one additional naming component, which has to be unique within the country.

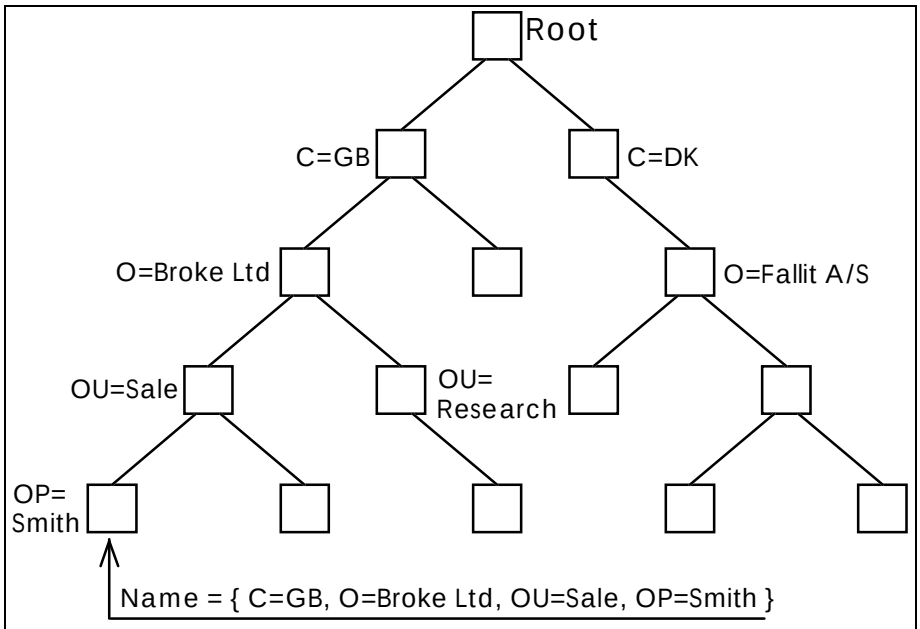

Figure 1 - Directory Information Tree (DIT)

This principle continues down through the tree. In figure 1 the next level represents organizational units and the lowest level is the organizational person. There could be many more objects represented, like OSI application-process entries just below an organization entry. The tree could of course be much deeper and does not need to have the same depth everywhere. For each level going down, a new naming component is added, and this component has to be unique with respect to the immediately superior entry. The naming component added for each level going down is called the Relative Distinguished Name (RDN) for the entry in question. An entry name is thus the concatenation of all the RDNs from the root down to and including the entry itself.

Entries at the bottom of the tree are called leaf entries.

An entry directly representing an object is called an object entry, and the name of the object entry is the distinguished name for the object it represents.

The DIT may contain alias entries. An alias entry points to an object entry, and the alias entry name is an alternative, or alias name for the object represented by that object entry. An alias entry cannot have any subordinate entries, i.e., it is always a leaf entry.

Due to the way Directory names are structured, an object's distinguished name is part of the distinguished name of any subordinate object. However, if an object has an alias name, this alias name can replace the distinguished name as prefix for names for subordinate objects and thus allows alternate, or alias names, also for the subordinate objects.

The DIT can be distributed among DSAs in a general way. A contiguous part of the DIT wholly contained within one DSA is called a naming context. To allow navigation to specific entries, a DSA must have some minimum knowledge about naming contexts in other DSAs in order to perform proper chaining or to return valid referrals.

The Directory information contained by a Directory entry is modelled as a number of user attributes, each attribute holding an attribute type identification and one or more attribute values. Each object entry belongs to a specific object class characterising the type of object it represents. A list of the types of user attributes an entry must and may hold is part an object class specification (see Annex A for further details).

The main function of the Directory is to provide information related to attributes. Attributes are carried as part of the protocol exchanges. It is therefore necessary to have complete agreement on the syntax of the attributes between the DSAs holding the attributes and the DUAs accessing them. It is only necessary in special cases to attach semantics to an attribute as part of the Directory specifications, and this is when the value of an attribute affects the operation of the Directory (e.g. Alias Object Name).

Entries must hold a special Object Class attribute that characterises it and other entries of the same type. From this attribute it can be deduced which other mandatory attributes must be part of the entry and which optional attributes can be part of the entry.

4 Use of the Directory for specific purposes

The Directory can be used for many different purposes. A Message Handling System (MHS) user may wish to map the Directory name of an Originator/Recipient to its O/R address; an FTAM user may want to map file names into locality information; etc. Accordingly, the Directory may contain entries related to a wide variety of different usages. However, it is believed that the naming strategy for many applications will have common elements allowing some standard name form at the top of the DIT, and only the leaf entries or entries close to leaf entries will have application specific forms of name components.

The structure of naming and the content of attributes is regulated by the Directory Schema; each DSA is regulated by a subset of the Directory Schema, which is termed a Subschema. The Subschema for a particular DSA is in turn regulated by the DSA's Administrative Authority.

A basic set of object classes is considered of general applicability independent of the specific use of the Directory. Such a set of object-classes is standardized by the Directory specification [ISO/IEC 9594-7 | CCITT X.521] - "Selected Object Classes".

Other standards may define object classes for specific use. As an example, MHS-specific object classes are defined by [ISO/IEC 10021-2 | CCITT X.402-1988] - "MOTIS (Message Handling Systems) - Overall Architecture".

5 Static capabilities of co-operating DSAs

DSAs co-operating to provide a Directory Service can have different static capabilities. The static capabilities of a DSA reflect what capabilities have been implemented and possibly can be invoked, and not what capabilities are used for a specific operation. A static capability that has been implemented is also referred to as a supported capability.

A DSA shall, according to the base specifications, as minimum support either the DAP or DSP and may support both.

The static DAP capabilities of a DSA are independent of its static DSP capabilities. Even in the case where a DSA tries to protect a DUA from referrals it cannot in general do that completely and has to be able to submit referrals on the DAP, if the DAP is supported. It is therefore possible to define a DAP profile independent of the DSP capabilities. Such a profile is defined by the Functional Standard A/DI1.

A DSA may have static capabilities as 1) the acceptor of an operation on the DSP or 2) as both the requester and the acceptor of an operation. The Functional Standard A/DI2 specifies the profiles for these two cases. The case where a DSA is acting only as a requester on the DSP is a theoretical possibility that is out of scope of A/DI2.

A DSA that supports the DAP only can only work in referral mode. However, if a DSA restricts access by DUAs not being within a specific domain, and if knowledge of this DSA is made available outside this domain, such a DSA is required to support at least incoming chained requests.

A DSA that supports the DSP only as the acceptor can receive and act on chained requests, but cannot issue chained requests, and will therefore respond with referrals, if appropriate, on the DSP.

A DSA, that supports the DSP both as the requester and as the acceptor, has the capability to chain requests.

6 Directory functional standards

Several functional standards have been defined for the OSI Directory. Each of these functional standards gives a profile for a particular area, selecting options within the scope of the base specifications. These functional standards also provide material that summarises or clarifies particular aspects of the base specifications.

The primary objective of the selections made by the functional standards is to ensure or promote interworking at several levels, by:

a) avoiding situations where elements of the Directory do not co-operate properly in the provision of the Directory service;

b) avoiding situations where two distinct but otherwise similar elements of the Directory behave inconsistently under a particular set of circumstances;

c) ensuring that DSAs have a minimum level of capability, so that user expectations of the Directory can be uniformly fulfilled.

There are two sets of Directory profiles defined by these functional standards. One set, the A-profiles, relates to protocol and distributed operations capabilities. The other set, the F-profiles, relates to information and naming structure.

6.1 Directory A-profiles

A number of A-profiles has been defined which are concerned with the following implementation options related to protocol and distributed operation capabilities:

A/DI1 - ENV 41210 (A/DI11 & A/DI12	Directory Access: DUA Support of Directory Access & DSA Support of Directory Access) In the M-IT-02 taxonomy, these two functional standards, A/DI11 and A/DI12, are identified separately to align with the corresponding ISO/IEC TR 10000-2 taxonomy for ISPs. However, only one functional standard, A/DI1 - "Directory Access" with the combined scope of A/DI11 and A/DI12, has been developed and it specifies a profile of the DAP Protocol as it applies both to Centralised and Co-operating DSAs; it also specifies a profile of the Abstract Service Definition of DSAs within the scope of [ISO/IEC 9594-3 | CCITT X.511].
A/DI2 - ENV 41212 (A/DI21 & A/DI22	Directory System Protocol: DSA - Responder Role & DSA - Initiator Role) In line with above only one functional standard, A/DI2 - Directory System Protocol, has been developed. This functional standard specifies a profile for the DSP protocol.
A/DI31 - ENV 41217	Behaviour of DUAs for Distributed Operations: This functional specifies a profile of the dynamic behaviour of the DUA in respect to referrals, i.e., when the DUA has to execute a series of operation to perform a specific user request.
A/DI32 - ENV 41215	Behaviour of DSAs for Distributed Operations: This functional standard provides requirements, guide-lines and clarification on how DSAs are to participate in distributed operations in the Directory; it augments the material provided by [ISO/IEC 9594-4 | CCITT X.518].

For a centralised Directory the Functional Standard A/DI1 by itself profiles the externally visible behaviour of the accessed DSA. For a distributed Directory a co-operating DSA behaviour is profiled by the Functional Standard A/DI32 together with Functional Standards A/DI1 and/or A/DI2, depending on whether the DSA supports the DAP, the DSP or both.

The Directory A-profiles are organized in such a way that an implementation may claim conformance to any of the functional standards without necessarily claiming conformance to the other ones. However, it is recommended that an implementation claims conformance to all relevant functional standards.

6.2 Directory F-profiles

The purpose of the Directory F-profiles is to define groups of object classes for recognised usages of the Directory, and to define the naming structure and the use of attributes for the selected object classes.

A DSA may claim conformance to a subset of F-profiles, depending on what applications should be supported. It is likely that the number of F-profiles will increase as new applications are found.

A major aspect of Directory F-profiles is to avoid unnecessary variety in the way information is structured as new usages are introduced. Therefore the F/DI11 Profile is intended to provide schema definitions usable by a wide variety of applications.

F/DI11 - ENV 41512 Common Directory Use:
This functional standard specifies a profile of the Schema, within the scope of [ISO/IEC 9595-6,-7 | CCITT X.520, X.521]. It specifies Naming- and Structure rules, Object Classes, Attribute Types, Attribute Syntaxes and their properties and implications, to be supported by conforming DSAs. This Functional Standard is intended to provide naming structure rules to be general applicable by a wide variety of Directory usages. Other Directory F-profiles are expected, as far as possible, to rely on this naming structure and only to specify the provision of more specific information.

Other F-profiles relate to specific application areas not covered by the Directory specifications. The currently proposed profiles are:

F/DI2 This Functional Standard specifies Object Classes, the Attribute Types and Syntaxes, as relating to X.400 systems. This profile is specified as an increment to F/DI1.

F/DI3 As F/DI2, but for FTAM systems.

7 ETSI Technical Reports for the Directory

In addition to the actual profiles a number of Directory ETRs has been developed or is under development. Beyond this ETR, the following ETRs are available.

a) ETR 124 - Error Handling for Directory;

b) ETR 097 - Security Architecture for the Directory;

c) ETR xxx - Interconnection of Directory Management Domains (still under development).

7.1 Error Handling for the Directory

A number of errors can be detected by a DSA during its execution of Directory requests. The base specifications have defined a few error categories, each category with a number of different error codes that can be returned in response to Directory requests.

The base specifications are not very particular on how and when to use the different error codes, and the purpose of ETR 124 is to give further guidance on that subject by identifying error conditions and recommending an appropriate error to be returned for each such error condition.

7.2 Security architecture for the directory

The purpose of this ETR on Directory Security is to identify the security threats that are specific to Directory, to list procedures on how to protect against such security threats and to specify capabilities to be implemented in DSAs to provide tools for such procedures.

7.3 Interconnection of Directory Management Domains

For a European wide, and in the end a world wide, Directory Service to be effective, it is not sufficient for all involved parties to conform to relevant standards and profiles. It is necessary to have a number of agreements among operators of Directory Management Domains. The ETR on Interconnection of Directory Management Domains (under study) describes the types of agreements necessary and proposals for some of these agreements, such as service agreements.

8 Conformance

Conformance requirements for DUAs are specified in ISO/IEC 9494-5 CCITT X.519, section 9.1. Additionally, a DUA implementation may claim conformance to:

- functional standard A/DI31.

Conformance requirements for DSAs are specified in ISO/IEC 9494-5 CCITT X.519, section 9.2.

A co-operating DSA implementation may additionally claim conformance to:

- any combination of the functional standards A/DI1, A/DI2, and A/DI32; and

- zero or more F-profiles.

Conformance to an A-profile addresses the required functionality of implementations. Note that it is possible to conform to the base specifications, without conforming to an A-profile. Conformance to a base specification mainly embraces the question of protocol conformance, thus allowing a response like "unwillingToPerform" on nearly all types of requests. In contrast, conformance to an A-profile also demands provision of the functionality associated with the protocol elements. A conforming implementation should act upon a certain protocol element (classified as "supported" by the A-profile under question) as indicated (implicitly or explicitly) by the base specifications.

F-profiles address the scope of information supported by conforming implementations, and the structure of this information. Implementations claiming conformance to particular F-profiles are required to support all the object classes, attribute types and related properties that are defined within those profiles.

Note, that these rules do not restrict conforming implementations to this scope. They may support information outside this scope, as entries of object classes not defined within the F-profile(s) under question or entries of an object class defined within the profile, but not obeying the defined structure or naming rules (e.g. a person as an immediate subordinate to country or an entry with more than four organizationalUnits within its name). Such entries are outside the scope of the F-profiles, and thus there are no conformance requirements for access to such entries.

This allows conforming implementations:

a) to support additional F-profiles or other functional standards defining Directory usage with different scope; and

b) to support private applications, which have requirements for information structured quite differently than generally applicable structure rules. Such private information must be located in a separate part of the DIT, obeying other structure and naming rules than defined within the F-profile(s) to which the implementation claims conformance.

Annex A: Object classes and DIT structure

The 1988 edition of the Directory specifications includes some rudimentary Directory schema concepts concerning object classes and DIT Structure. It has been found necessary to expand those concepts during the development of ENV 41512. The 1992 edition of the Directory specifications has adopted these concepts and further progressed them.

The 1992 edition of the Directory specifications has introduced restrictions not violated by ENV 41512, but not expressed explicitly by ENV 41512 either. It would be beneficial to recognise these restrictions when developing FDI profiles, even for the 1988 edition of the Directory specifications.

A number of groups are developing Directory FDI profiles based on the principles described in ENV 41512, but these principles require some background knowledge, and this section of the document therefore has two purposes:

1) explore which 1992 concepts can be used directly for FDI profiles based on the 1988 edition of the Directory specifications without requiring non-1988 features to be implemented; and

2) to provide a general tutorial on the subject.

A (Directory) object represented in the Directory is a reflection of certain aspects of some "real world" object. Different aspects of the same real world object can be viewed as different Directory objects. Examples of such real world object are countries, organizational persons, OSI application-entities, etc.

An object class is the specification of a type of Directory object that can be represented by entries in the Directory. An object class specification includes a list of what attributes shall and what attributes may be held by an entry created according to that object class, i.e., the specification includes a list of mandatory and optional attributes. For an object class to be useful it has to be officially registered within the user community in which it is used. As a part of this registration it is given a unique identifier in the form of an ASN.1 object identifier.

Object classes can be defined and registered by any committee or organization that has been authorised to allocate ASN.1 object identifiers. This includes all committees defining International Standards and International Standardized Profiles; Regional Workshops (such as EWOS); CCITT; private organizations; etc.

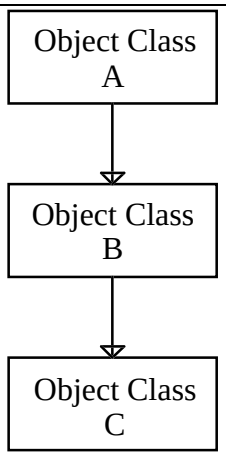

Figure A.1: Object Class Inheritance

These possibilities can, of course, lead to a proliferation of object classes, some of which may be closely related in the sense that they represent similar real world objects and specify almost the same set of attributes. To minimise such proliferation and to minimise redundant specifications, the concept of inheritance, and in particular multiple inheritance, has been invented. Object class inheritance allows one object class to inherit attributes of one or more other object classes. This is illustrated in figure A.1, where object class B is said to be a subclass of object class A. The concept of deriving subclasses can be extended to derive subclasses of subclasses to any depth. Object class C in figure A.1 is a subclass of object class B, and is also said to be a subclass of object class A.

Object class A is said to be a superclass of object classes B and C, whilst object class B is a superclass of object class C. Object class A is said to be a direct superclass of object class B, and so is object class B of object class C.

The object class (type) hierarchy illustrated in figure A.1 should not be confused with the entry (instance) hierarchy represented by the DIT. In the latter, an entry superior/subordinate pair has not an inheritance relationship.

Figure A.2 illustrates further the concept of inheritance. Object class B is here a subclass of object class A. Object class B does not only include the mandatory and optional attributes listed as part of its specification, but also inherits those of the object class A specification not already part of the object class B specification. In case the object class B specification directly includes a particular attribute and inherits the same attribute from a superclass, this does not result in multiple occurrences of the same attribute type. A mandatory specification takes presence as illustrated by Attr-3 in figure A.2. The object class B to the right in figure A.2 shows the result of object class B being a subclass of object class A.

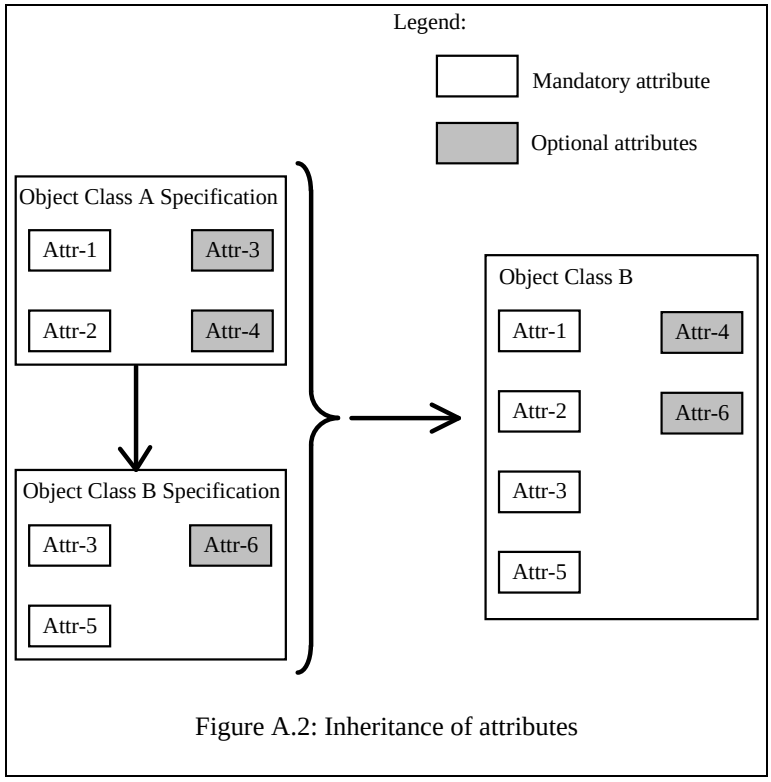

Figure A.2: Inheritance of attributes

An object class that is a subclass of other subclasses specifies as part of its specification what object class it has as direct superclass (or object classes if it has several direct superclasses - see later). When defining such a particular subclass, all attributes specified by its direct superclass, and possible other superclasses, are known. There is no need to repeat them when specifying the subclass (except when wanting to make otherwise optional attributes mandatory), although it is allowed. This is one of the purposes of the subclass concept, to avoid redundant specifications. Figure A.2 illustrates this principle.

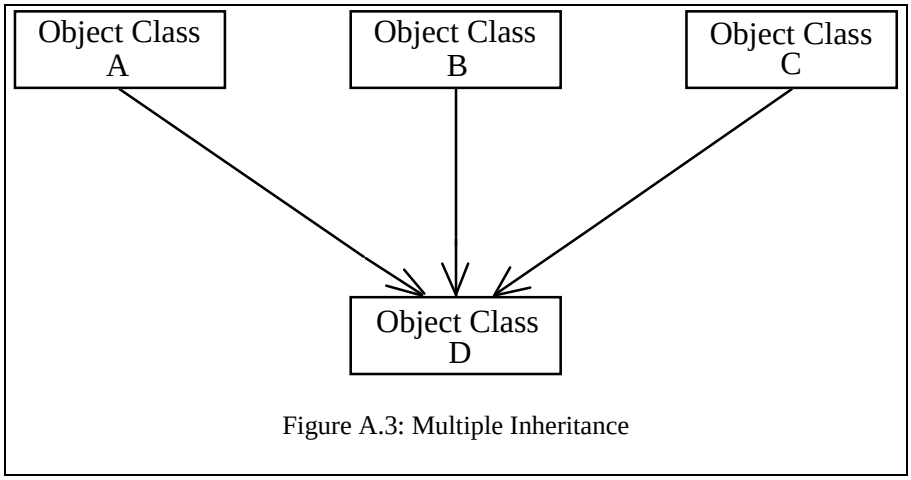

Figure A.3: Multiple Inheritance

A Directory entry is created according to some particular object class. Taking figure A.1 as an example, an entry can be created from any of the object classes A, B, and C. If an entry is created from object class A, it has only the A-properties. If created from object class C, it has the combined properties of A, B and C. Although the entry is created according to object class C, it is said to belong not only to C, but also to A and B.

Multiple inheritance implies that an object class can have more than one direct superclass. In figure A.3, object class D has three direct superclasses, A, B, and C. Object class D therefore inherits all the attribute types specified for object classes A, B, and C.

According to the 1988 edition of the Directory specifications an entry can only be created from a single object class. This has lead to the concept of unregistered object classes, which means object classes that have not been assigned ASN.1 object identifiers. An unregistered object class can be defined, as illustrated in figure A.4, where a set of specific entries shall have the properties of more than one object class that do not have a superclass/subclass relationship.

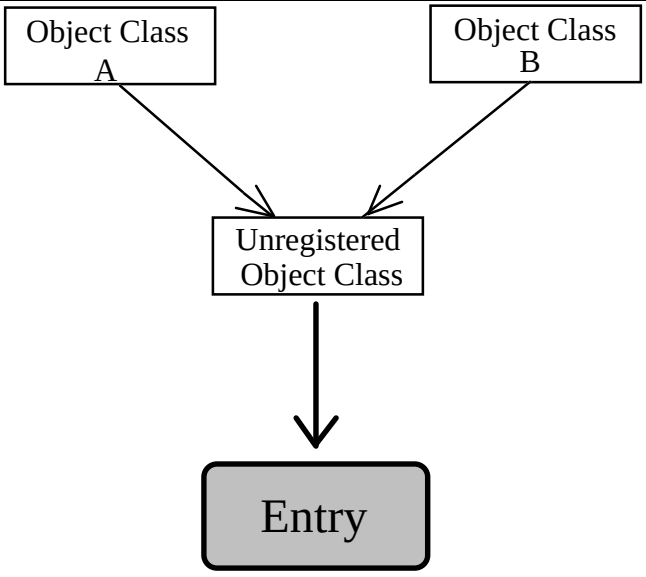

Figure A.4: Unregistered Object Class

In case the unregistered object does not add any new attributes as part of its own specification, the entry shown has exactly the combined attributes of object class A and B specifications.

The 1988 edition of the Directory specifications allows unregistered object classes to add additional attributes not part of its superclasses. The 1992 edition has removed that capability by removing the concept of unregistered object class. Unregistered object classes should therefore not by any standardization body be used for adding additional attributes. The 1992 edition has added another mechanism to add such additional attributes, but this mechanism cannot be used by an 1988 implementation.

The 1992 edition of the Directory specifications allows an entry to be created from more than one object class, thus having the combined characteristics of the object classes from which it is created following the same rules as for inheritance. This is illustrated in figure A.5. This capability removes the necessity to define an unregistered object class.

If unregistered object classes are not used to add additional attribute types, the situation depicted in figure A.4 is completely equivalent to the one depicted in figure A.5. They are just different description techniques. It should not make any difference for an implementation. A profile based on the 1988 edition could therefore, if it chooses to, use this particular 1992 concept and does not necessarily have to define unregistered object classes.

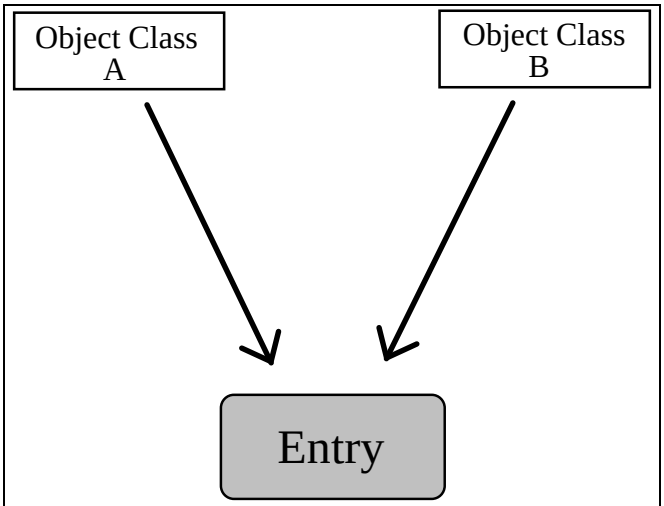

Figure A.5: Entry creation from Multiple Object Classes

The hierarchy of entries in the DIT should be regulated in some way by applying certain rules as to how entries of different object classes should be placed with respect to each other. An example of such a rule is to specify that an OSI application-process entry shall be below either a locality entry, an organization entry or an organizational unit entry. The Directory specifications or the ENVs do not mandate particular rules, but ENV 41512 suggests how such rules for structure and naming can be documented. It will be up to the user communities what particular rules they want. However, ENV 41512 does specify some minimum rules for naming and structure, which a conforming DSA has to support. For a particular entry it has to be decided what attribute type or attribute types have to be used for naming, i.e., to be used for Relative Distinguished Name. The base Directory specifications and ENV 41512 give some recommendations on what attribute(s) should be used for naming, and they should be followed unless there are very good reasons for the opposite.

In figure A.5 (or figure A.4) an entry is created from two object classes A and B. It is obvious that only one of these object classes can determine the entry's location in the DIT and only one object class can determine its name component. Otherwise conflicting rules for the same entry could be the result. This has lead to the concept of different categories of object classes. An object class is called a structural object class if it has rules for structure and naming as part of its specification. An object class that has no such associated structure rules is called an auxiliary object class. An entry is created from exactly one structural object and from zero or more auxiliary object classes as illustrated in figure A.6.

It should be noted that a structural object class that is a subclass of another structural object class does not automatically inherit the rules for naming and structure.

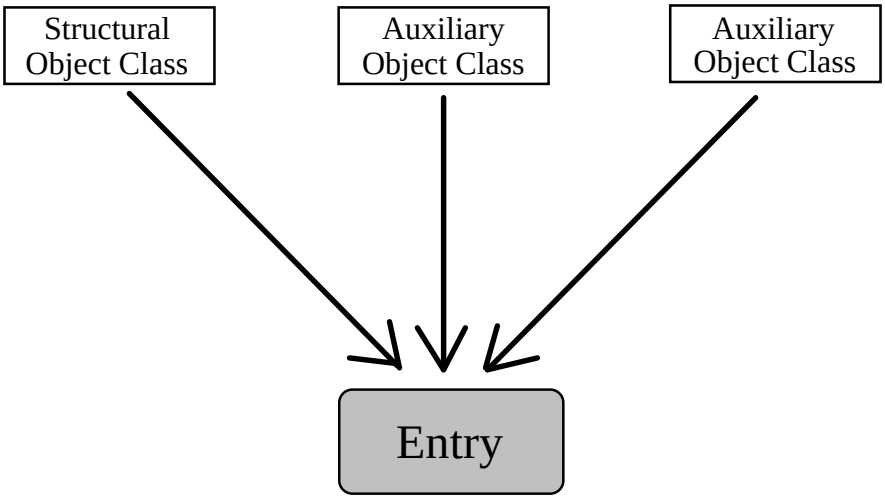

Figure A.6: Object Class Categories

The structural object class is the object class that actually represents the corresponding real world object, while auxiliary object classes, as the name indicates, do not in themselves represent a particular real world object, but only represents additional attributes that can be associated with some structural object class to supplement its specification.

Let us assume we have a type of real world object for which we want to define an object class. There may be available a more generic structural object class, that captures the major semantics of the object in question and most of the needed attributes, but we want to add some additional attributes. We now have two techniques available:

1) we can define a (structural) subclass to the more generic structural object class that includes the additional attributes; or

2) we can define an auxiliary object class specifying the additional attributes and then combining that with the generic structural object class.

Which of the two techniques that should be used depends on the circumstances and has to be judged for each particular case. If the set of attributes to be added is generally applicable to a number of cases, an auxiliary object class should be defined, as it can be combined with different structural object classes. If, however, the rules for structure and naming defined for the more generic object class do not apply for the situation at hand, a subclass being a structural object class has to be defined.

Annex B: Requirements on matching rules for attributes

This annex has been included because it contains valuable information for developer of F-profiles and for implementors, and it has not been possible to include it into any of the current F-profiles.

In the Directory, apart from matching Distinguished Name components, there are two primary motivations for the design of matching rules for equality:

- support of search; and

- support of administration of the Directory.

The second item is to do with the adding and removal of attribute values:

- two attribute values that match for equality cannot coexist in the same attribute; and

- to remove a specific attribute value, the value (or a value that matches it for equality) must be supplied in the modify-entry operation.

The search requirement is relatively unrestrictive - a purported value must match stored values in such a way that attributes that would be expected to match do match; for convenience, the matching rule may cause attributes to be returned that have matching values that do not precisely match the purported value. An example of this is with the Presentation Address attribute, which matches if the supplied network-addresses form a subset of the stored network-addresses, given that the P-, S- and T-selectors all match.

The administration requirement applies only to multi-valued attributes (Presentation Address is a single value attribute); here the requirement is that, whenever possible, the matching rule is "well-behaved" in having three basic properties:

- A matches A (reflexive property) - always required;

- if A matches B then B matches A (commutative property); and

- if A matches B and B matches C then A matches C (transitive property).

These rules are important for the following reasons:

- The reflexive property is required to remove a particular value.

- Suppose that a value A is placed in the Directory, and sometimes later B places in the same attribute, where A matches B but B does not match A; since B does not match A, the adding of the new attribute is successful.

- Using "remove value", quoting a value C that matches both A and B can result in the wrong value being removed.

- If, on the contrary, B is placed in the Directory first, adding A anomalously fails, since A matches B.

- The requirement of transitivity arises from the desirability, for administrative purposes, to have values fall into "equivalence classes"; suppose, for example that a (non-transitive) match on physical size meant that A matches B if and only if A and B differ by less than 1 cm; values could only be inserted if they were spaces at least 1 cm from any other value; this could be an unexpected outcome, and thus an unwanted one.

- Now if there were two values A and B that differed by just over 1 cm, attempting to remove A by specifying C, between A and B could result in removal of B instead.
- In general, transitivity ensures that if a value A was replaced by different but matching value B, either value can be removed can be removed by using a third (perhaps shorter and convenient) value C that matches A.

If the matching rule is non-commutative or non-transitive, indexation cannot be used with the attribute values, and scanning of a set of such values can produce unexpected results.

Similar, but mandatory, requirements of reflexivity, commutativity and transitivity exist on attributes whose values are to be found in Distinguished Names.

There is a further implicit requirement that applies to matching rules supporting administration:

> Two values A and B shall not match for equality if it is reasonable for them to coexist in the same attribute;

and conversely:

> Two values A and B should ideally match when they both represent a description of the same logical entity.

Thus, ACI attribute values (holding access control information) do not match even if they have the same ACI but differ in their identifier; conversely if two facsimile telephone numbers are {F,G}, {F,G'} where F represents one telephone number and G,G' represent different G3 fax non-basic-parameters, then the two values could usefully match because they both purportedly describe the same logical entity (a facsimile machine located by telephone number). This is more obvious if G (say) is empty, and implies that you do not have to know or read the G3 non-basic-parameters of a value to remove it.

The ACI and facsimile telephone number values are examples of a general situation where a value is composed of two components, P and Q; P is relevant to discriminating between two values in the sense that it fully identifies the described logical entity; Q is additional information that is not relevant to discrimination in this way. Then an appropriate matching rule should state:

> {P,Q} and {P',Q'} should match for equality if and only if P and P' match for equality.

A methodology for finding the correct matching rule could take the following steps:

1. Determine what elements of a value represent a basic identifying property of the entity described; ignore the values of the remainder when matching.

2. Test the rule by attempting to construct two distinct values that differ only in one or more ignored elements; also attempt to construct two values that are not equivalent although matching on the ignored values; go back to step 1 if need be.

3. Describe a (potentially non-commutative) matching rule M from these steps.

4. Convert this to a commutative matching rule M' as follows:

   - M'(A,B) (i.e., the result of matching A and B) is true if and only if M(A,B) and M(B,A) are both true.

5. Convert this to a transitive matching rule M'' as follows:

   - M''(A,B) is true if and only if for each C, M'(A,C) equals M'(C,B).

Usually, steps 4 and 5 will be unnecessary. Step 4 follows because logical AND is commutative. Step 5 follows because if M''(A,B) is true and M''(B,C) is true, for each P, M'(A,P) equals M'(B,P) which equals M'(C,P), taking into account commutativity.

History

Document history	
March 1994	First Edition
March 1996	Converted into Adobe Acrobat Portable Document Format (PDF)