INTERNATIONAL STANDARD

ISO/IEC 9594-3

First edition
1990-12-15

Information technology — Open Systems Interconnection — The Directory —

Part 3:
Abstract service definition

Technologies de l'information — Interconnexion de systèmes ouverts — L'annuaire —

Partie 3: Définition du service abstrait

Reference number
ISO/IEC 9594-3 : 1990 (E)

Contents

Page

Foreword iii

Introduction iv

SECTION 1: GENERAL 1

1 Scope 1

2 Normative references 1

3 Definitions 1

4 Abbreviations 2

5 Conventions 2

SECTION 2: ABSTRACT SERVICE 3

6 Overview of the Directory Service 3

7 Information Types 3

8 Bind and Unbind Operations 8

9 Directory Read Operations 9

10 Directory Search Operations 10

11 Directory Modify Operations 12

12 Errors 15

Annex A — Abstract Service in ASN.1 19

Annex B — Directory Object Identifiers 26

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9594-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

- *Part 1: Overview of concepts, models and services*
- *Part 2: Models*
- *Part 3: Abstract service definition*
- *Part 4: Procedures for distributed operation*
- *Part 5: Protocol specifications*
- *Part 6: Selected attribute types*
- *Part 7: Selected object classes*
- *Part 8: Authentication framework*

Annexes A and B form an integral part of this part of ISO/IEC 9594.

Introduction

0.1 This part of ISO/IEC 9594, together with the other parts, has been produced to facilitate the interconnection of information processing systems to provide directory services. The set of all such systems, together with the directory information which they hold, can be viewed as an integrated whole, called the *Directory*. The information held by the Directory, collectively known as the Directory Information Base (DIB), is typically used to facilitate communication between, with or about objects such as application entities, people, terminals, and distribution lists.

0.2 The Directory plays a significant role in Open Systems Interconnection, whose aim is to allow, with a minimum of technical agreement outside of the interconnection standards themselves, the interconnection of information processing systems:

- from different manufacturers;
- under different managements;
- of different levels of complexity; and
- of different ages.

0.3 This part of ISO/IEC 9594 defines the capabilities provided by the Directory to its users.

0.4 Annex A provides the ASN.1 module which contains all the definitions associated with the Abstract Service

Information technology — Open Systems Interconnection — The Directory —

Part 3:
Abstract service definition

SECTION 1: GENERAL

1 Scope

1.1 This part of ISO/IEC 9594 defines in an abstract way the externally visible service provided by the Directory.

1.2 This part of ISO/IEC 9594 does not specify individual implementations or products.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9594. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9594 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 7498:1984, *Information Processing Systems — Open Systems Interconnection — Basic Reference Model.*

ISO/IEC 8824:1990, *Information Technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 9072-1:1989, *Information Processing Systems — Text Communication — Remote Operations — Part 1: Model, Notation and Service Definition.*

ISO/IEC 9072-2:1989, *Information Processing Systems — Text Communication — Remote Operations — Part 2: Protocol Specification.*

ISO/IEC 9594-1:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 1: Overview of Concepts, Models and Services.*

ISO/IEC 9594-2:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 2: Models.*

ISO/IEC 9594-4:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 4: Procedures for Distributed Operation.*

ISO/IEC 9594-5:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 5: Protocol Specifications.*

ISO/IEC 9594-6:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 6: Selected Attribute Types.*

ISO/IEC 9594-7:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 7: Selected Object Classes.*

ISO/IEC 9594-8:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 8: Authentication Framework.*

ISO/IEC 10021-3:1990, *Information Technology — Text Communication — Message Oriented Text Interchange System (MOTIS) — Part 3: Abstract Service Definition.*

3 Definitions

3.1 Basic Directory Definitions

This part of ISO/IEC 9594 makes use of the following terms defined in ISO/IEC 9594-1:

a) Directory;

b) Directory Information Base;

c) (Directory) User.

3.2 Directory Model Definitions

This part of ISO/IEC 9594 makes use of the following terms defined in ISO/IEC 9594-2:

a) Directory System Agent;

b) Directory User Agent.

3.3 Directory Information Base Definitions

This part of ISO/IEC 9594 makes use of the following terms defined in ISO/IEC 9594-2:

a) alias entry;

b) Directory Information Tree;

c) (Directory) entry;

d) immediate superior;

e) immediately superior entry/object;

f) object;

g) object class;

h) object entry;

i) subordinate;

j) superior.

3.4 Directory Entry Definitions

This part of ISO/IEC 9594 makes use of the following terms defined in ISO/IEC 9594-2:

a) attribute;

b) attribute type;

c) attribute value;

d) attribute value assertion.

3.5 Name Definitions

This part of ISO/IEC 9594 makes use of the following terms defined in ISO/IEC 9594-2:

a) alias, alias name;

b) distinguished name;

c) (directory) name

d) purported name;

e) relative distinguished name.

3.6 Distributed Operations Definitions

This part of ISO/IEC 9594 makes use of the following terms defined in ISO/IEC 9594-4:

a) chaining;

b) referral.

3.7 Abstract Service Definitions

For the purpose of this part of ISO/IEC 9594 the following definitions apply:

3.7.1 *filter*: An assertion about the presence or value of certain attributes of an entry in order to limit the scope of a search;

3.7.2 *service controls*: Parameters conveyed as part of an abstract-operation which constrain various aspects of its performance.

3.7.3 *originator*: The user that originated an operation.

4 Abbreviations

This part of ISO/IEC 9594 makes use of the following abbreviations:

AVA	Attribute Value Assertion
DIB	Directory Information Base
DIT	Directory Information Tree
DSA	Directory System Agent
DUA	Directory User Agent
DMD	Directory Management Domain
RDN	Relative Distinguished Name

5 Conventions

This part of ISO/IEC 9594 makes use of the abstract service definition conventions defined in ISO/IEC 10021-3.

SECTION 2: ABSTRACT SERVICE

6 Overview of the Directory Service

6.1 As described in ISO/IEC 9594-2, the services of the Directory are provided through access points to DUAs, each acting on behalf of a user. These concepts are depicted in figure 1.

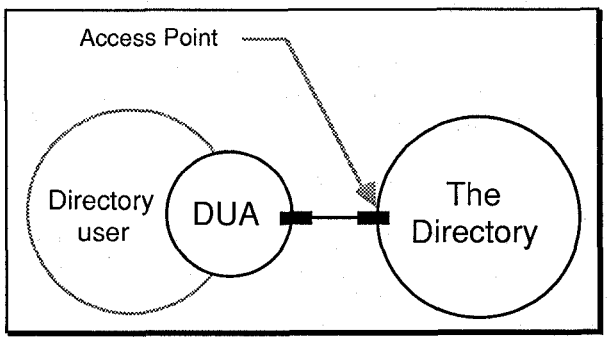

Figure 1 —Access to the Directory

6.2 In principle, access points to the Directory may be of different types, providing different combinations of services. It is valuable to consider the Directory as an *object*, supporting a number of types of *port*. Each port defines a particular kind of interaction which the Directory can participate in with a DUA. Each access point corresponds to a particular combination of port types.

6.3 Using the notation defined in ISO/IEC 8505-3, the Directory can be defined as follows:

```
directory
        OBJECT
                PORTS {         readPort [S],
                                searchPort [S],
                        modifyPort  [S]}
        ::=     id-ot-directory
```

The Directory supplies operations via: Read Ports, which support reading information from a particular named entry in the DIB; Search Ports, which allow more 'exploration' of the DIB; and Modify Ports, which enable the modification of entries in the DIB.

Note - It is intended that in the future there may be other types of Directory port.

6.4 Similarly, a DUA (from the viewpoint of the Directory) can be defined as follows:

```
dua
        OBJECT
                PORTS {         readPort [C],
                                searchPort [C],
                        modifyPort [C]}
        ::=     id-ot-dua
```

The DUA consumes the services provided by the Directory.

6.5 The ports cited from 6.2 through 6.4 can be defined as follows:

```
readPort
        PORT
                CONSUMER INVOKES {
                        Read, Compare, Abandon}
        ::=     id-pt-read

searchPort
        PORT
                CONSUMER INVOKES {
                        List, Search}
        ::=     id-pt-search

modifyPort
                PORT
        CONSUMER INVOKES {
                        AddEntry, RemoveEntry,
                        ModifyEntry, ModifyRDN}
        ::=     id-pt-modify
```

6.6 The operations from the **readPort**, **searchPort** and the **modifyPort** are defined in clauses 9, 10, and 11 respectively.

6.7 These ports are used only as a method of structuring the description of the Directory service. Conformance to Directory operations is specified in ISO/IEC 9594-5.

7 Information Types

7.1 Introduction

7.1.1 This clause identifies, and in some cases defines, a number of information types which are subsequently used in the definition of Directory operations. The information types concerned are those which are common to more than one operation, are likely to be in the future, or which are sufficiently complex or self-contained as to merit being defined separately from the operation which uses them.

7.1.2 Several of the information types used in the definition of the Directory service are actually defined elsewhere. Subclause 7.2 identifies these types and indicates the source of their definition. Each of the remaining subclauses (7.3 through 7.11) identifies and defines an information type.

7.2 Information Types Defined Elsewhere

7.2.1 The following information types are defined in ISO/IEC 9594-2:

a) **Attribute**;

b) **AttributeType**;

c) **AttributeValue**;

d) **AttributeValueAssertion**;

e) **DistinguishedName**;

f) **Name**;

g) **RelativeDistinguishedName**.

7.2.2 The following information type is defined in ISO/IEC 9594-6:

a) **PresentationAddress**.

7.2.3 The following information types are defined in ISO/IEC 9594-8:

a) ***Certificate***;

b) **SIGNED**;

c) **CertificationPath**.

7.2.4 The following information types are defined in ISO 9072-1:

a) **InvokeID**.

7.2.5 The following information types are defined in ISO 9594-4:

a) **OperationProgress**;

b) **ContinuationReference**.

7.3 Common Arguments

7.3.1 The CommonArguments information may be present to qualify the invocation of each operation that the Directory can perform.

```
CommonArguments   ::=   SET {
        [30]   ServiceControls DEFAULT {},
        [29]   SecurityParameters DEFAULT {},
        requestor     [28]   DistinguishedName
                                 OPTIONAL,
        [27]   OperationProgress DEFAULT {notStarted},
        aliasedRDNs [26] INTEGER OPTIONAL,
        extensions [25] SET OF Extension
                             OPTIONAL }

Extension      ::=    SET {
        identifier     [0]    INTEGER,
        critical       [1]    BOOLEAN DEFAULT FALSE,
        item           [2]    ANY DEFINED BY identifier}
```

7.3.2 The various components have the meanings as defined in 7.3.2.1 through 7.3.2.4.

7.3.2.1 The **ServiceControls** component is specified in 7.5. Its absence is deemed equivalent to there being an empty set of controls.

7.3.2.2 The **SecurityParameters** component is specified in 7.9. Its absence is deemed equivalent to there being an empty set of security parameters.

7.3.2.3 The **requestor** Distinguished Name identifies the originator of a particular abstract-operation. It holds the name of the user as identified at the time of binding to the Directory. It may be required when the request is to be signed (see 7.10), and shall hold the name of the user who initiated the request.

7.3.2.4 The **OperationProgress** defines the role that the DSA is to play in the distributed evaluation of the request. It is more fully defined in ISO/IEC 9594-4.

7.3.2.5 The **aliasedRDNs** component indicates to the DSA that the **object** component of the operation was created by the dereferencing of an alias on an earlier operation attempt. The integer value indicates the number of RDNs in the object that came from dereferencing the alias. (The value would have been set in the referral response of the previous operation.)

7.3.2.6 The **extensions** component provides a mechanism to express standardized extensions to the form of the argument of a Directory abstract-operation.

Note — The form of the result of such an extended abstract-operation is identical to that of the non-extended version. (Nonetheless, the result of a particular extended abstract-operation may differ from its non-extended counterpart).

The subcomponents are as defined in 7.3.2.6.1 through 7.3.2.6.3.

7.3.2.6.1 The **identifier** serves to identify a particular extension. Values of this component shall be assigned only by future versions of ISO/IEC 9594.

7.3.2.6.2 The **critical** subcomponent allows the originator of the extended abstract-operation to indicate that the performance of only the extended form of the abstract-operation is acceptable (i.e. that the non-extended form is not acceptable). In this case the extensions is a critical extension. If the Directory, or some part of it, is unable to perform a critical extension it returns an indication of **unavailableCriticalExtension** (as a **serviceError** or **PartialOutcomeQualifier**). If the Directory is unable to perform an extension which is not critical, it ignores the presence of the extension.

7.3.2.6.3 The **item** subcomponent provides the information needed for the Directory to perform the extended form of the abstract-operation.

7.4 Common Results

7.4.1 The **CommonResults** information should be present to qualify the result of each retrieval operation that the Directory can perform.

```
CommonResults ::=   SET {
                [30]    SecurityParameters   OPTIONAL,
performer       [29]    DistinguishedName OPTIONAL,
aliasDereferenced       [28]    BOOLEAN  DEFAULT FALSE}
```

7.4.2 The various components have the meanings as defined in 7.4.2.1 through 7.4.2.3.

7.4.2.1 The **SecurityParameters** component is specified in 7.9. Its absence is deemed equivalent to there being an empty set of security parameters.

7.4.2.2 The **performer** Distinguished Name identifies the performer of a particular operation. It may be required when the result is to be signed (see 7.10) and shall hold the name of the DSA which signed the result.

7.4.2.3 The **aliasDereferenced** component is set to **TRUE** when the purported name of an object or base object which is the target of the operation included an alias which was dereferenced.

7.5 Service Controls

7.5.1 A **ServiceControls** parameter contains the controls, if any, that are to direct or constrain the provision of the service.

```
ServiceControls         ::=     SET {
        options [0] BIT STRING {
                preferChaining(0),
                chainingProhibited (1),
                localScope (2),
                dontUseCopy (3),
                dontDereferenceAliases(4)}
        DEFAULT {},

        priority [1] INTEGER {
                low (0),
                medium (1),
                high (2) } DEFAULT medium,

        timeLimit       [2]     INTEGER OPTIONAL,

        sizeLimit       [3]     INTEGER OPTIONAL,

        scopeOfReferral         [4]     INTEGER{
                                        dmd(0),
                                        country(1)}
                                        OPTIONAL }
```

7.5.2 The various components have the meanings as defined in 7.5.2.1 through 7.5.2.5.

7.5.2.1 The **options** component contains a number of indications, each of which, if set, asserts the condition suggested. Thus:

a) **preferChaining** indicates that the preference is that chaining, rather than referrals, be used to provide the service. The Directory is not obliged to follow this preference;

b) **chainingProhibited** indicates that chaining, and other methods of distributing the request around the Directory, are prohibited;

c) **localScope** indicates that the operation is to be limited to a local scope . The definition of this option is itself a local matter, for example, within a single DSA or a single DMD;

d) **dontUseCopy** indicates that copied information (as defined in ISO/IEC 9594-4 shall not be used to provide the service;

e) **dontDereferenceAliases** indicates that any alias used to identify the entry affected by an operation is not to be dereferenced.

Note - This is necessary to allow reference to an alias entry itself rather than the aliased entry, e.g., in order to read the alias entry.

If this component is omitted, the following are assumed: no preference for chaining but chaining not prohibited, no limit on the scope of the operation, use of copy permitted, and aliases shall be dereferenced (except for modify operations where aliases shall never be dereferenced).

7.5.2.2 The priority (low, medium or high) at which the service is to be provided. Note that this is not a guaranteed service in that the Directory, as a whole, does not implement queuing. There is no relationship implied with the use of 'priorities' in underlying layers.

7.5.2.3 The timeLimit indicates the maximum elapsed time, in seconds, within which the service shall be provided. If the constraint cannot be met, an error is reported. If this component is omitted, no time limit is implied. In the case of time limit exceeded on a List or Search, the result is an arbitrary selection of the accumulated results.

Note — This component does not imply the length of time spent processing the request during the elapsed time: any number of DSAs may be involved in processing the request during the elapsed time.

7.5.2.4 The **sizeLimit** is only applicable to List and Search operations. It indicates the maximum number of objects to be returned. In the case of size limit exceeded, the results of List and Search may be an arbitrary selection of the accumulated results, equal in number to the size limit. Any further results shall be discarded.

7.5.2.5 The **scopeOfReferral** indicates the scope to which a referral returned by a DSA should be relevant. Depending on whether the values **dmd** or **country** are

selected, only referrals to other DSAs within the selected scope shall be returned. This applies to the referrals in both a **Referral** error and the **unexplored** parameter of List and Search results.

7.5.3 Certain combinations of **priority**, **timeLimit**, and **sizeLimit** may result in conflicts. For example, a short time limit could conflict with low priority; a high size limit could conflict with a low time limit, etc.

7.6 Entry Information Selection

7.6.1 An **EntryInformationSelection** parameter indicates what information is being requested from an entry in a retrieval service.

```
EntryInformationSelection::=        SET {
    attributeTypes
            CHOICE {
     allAttributes [0]        NULL,
     select        [1]   SET OF AttributeType
     --        empty set implies no attributes
     --        are requested - - }
               DEFAULT allAttributes NULL,

          infoTypes   [2]   INTEGER {
            attributeTypesOnly (0),
            attributeTypesAndValues (1) }
     DEFAULT attributeTypesAndValues }
```

7.6.2 The various components have the meanings as defined in 7.6.2.1 and 7.6.2.2.

7.6.2.1 The **attributeTypes** component specifies the set of attributes about which information is requested:

a) If the **select** option is chosen, then the attributes involved are listed. If the list is empty, then no attributes shall be returned. Information about a selected attribute shall be returned if the attribute is present. An **attributeError** with the **noSuchAttribute** problem shall only be returned if none of the attributes selected is present.

b) If the **allAttributes** option is selected, then information is requested about all attributes in the entry.

Attribute information is only returned if access rights are sufficient. A **securityError** (with an **insufficientAccessRights** problem) shall only be returned in the case where access rights preclude the reading of all attribute values requested.

7.6.2.2 The **infoTypes** component specifies whether both attribute type and attribute value information (the default) or attribute type information only is requested. If the **attributeTypes** component (7.6.2.1) is such as to request no attributes, then this component is not meaningful.

7.7 Entry Information

7.7.1 An **EntryInformation** parameter conveys selected information from an entry.

```
EntryInformation      ::=     SEQUENCE {
        DistinguishedName,
        fromEntry      BOOLEAN DEFAULT TRUE,
        SET OF CHOICE {
              AttributeType,
              Attribute } OPTIONAL }
```

7.7.2 The **DistinguishedName** of the entry is always included.

7.7.3 The **fromEntry** parameter indicates whether the information was obtained from the entry (**TRUE**) or a copy of the entry (**FALSE**).

7.7.4 A set of **AttributeTypes or Attributes** are included, if relevant, each of which may be alone or accompanied by one or more attribute values.

7.8 Filter

7.8.1 A **Filter** parameter applies a test that is either satisfied or not by a particular entry. The filter is expressed in terms of assertions about the presence or value of certain attributes of the entry, and is satisfied if and only if it evaluates to **TRUE**.

Note — A Filter may be **TRUE**, **FALSE** or undefined.

```
Filter     ::=     CHOICE {
        item    [0]    FilterItem,
        and     [1]    SET OF Filter,
        or      [2]    SET OF Filter,
        not     [3]    Filter }

FilterItem      ::=    CHOICE {
                  equality    [0]  AttributeValueAssertion,
                substrings    [1]  SEQUENCE {
                             type   AttributeType,
                          strings   SEQUENCE OF CHOICE {
                                       initial  [0]   AttributeValue,
                                         any    [1]   AttributeValue,
                                         final  [2]
                          AttributeValue}},
        greaterOrEqual [2]   AttributeValueAssertion,
        lessOrEqual    [3]   AttributeValueAssertion,
        present        [4]   AttributeType,
   approximateMatch    [5]   AttributeValueAssertion}
```

7.8.2 A **Filter** is either a **FilterItem** (see 7.8.3), or an expression involving simpler Filters composed together using the logical operators **and**, **or**, and **not**. The **Filter** is undefined if it is a **FilterItem** which is undefined, or if it involves one or more simpler **Filter**s, all of which are undefined. Otherwise, where the **Filter** is:

a) an **item**, it is **TRUE** if and only if the corresponding **FilterItem** is **TRUE**;

b) an **and**, it is **TRUE** unless any of the nested **Filter**s is **FALSE**.

Note - thus, if there are no nested **Filter**s the **and** evaluates to **TRUE**;

c) an **or**, it is **FALSE** unless any of the nested **Filter**s is **TRUE**.

Note - thus, if there are no nested **Filter**s the **or** evaluates to **FALSE**;

d) a **not**, it is **TRUE** if and only if the nested **Filter** is **FALSE**.

7.8.3 A **FilterItem** is an assertion about the presence or value(s) of an attribute of a particular type in the entry under test. Each such assertion is **TRUE**, **FALSE**, or undefined.

7.8.3.1 Every **FilterItem** includes an **AttributeType** which identifies the particular attribute concerned.

7.8.3.2 Any assertion about the value of such an attribute is only defined if the **AttributeType** is known, and the purported **AttributeValue**(s) conforms to the attribute syntax defined for that attribute type.

Notes

1. Where these conditions are not met, the **FilterItem** is undefined.

2. Access control restrictions may require that the **FilterItem** be considered undefined.

7.8.3.3 Assertions about the value of an attribute are evaluated using the matching rules associated with the attribute syntax defined for that attribute type. A matching rule not defined for a particular attribute syntax cannot be used to make assertions about that attribute.

Note — Where this condition is not met, the **FilterItem** is undefined.

7.8.3.4 A **FilterItem** may be undefined (as described in 7.8.2–3 above). Otherwise, where the **FilterItem** asserts:

a) **equality**, it is TRUE if and only if there is a value of the attribute which is equal to that asserted;

b) **substrings**, it is TRUE if and only if there is a value of the attribute in which the specified substrings appear in the given order. The substrings shall be non-overlapping, and may (but need not) be separated from the ends of the attribute value and from one another by zero or more string elements.

If **initial** is present, the substring shall match the initial substring of the attribute value; if **final** is present, the substring shall match the final substring of the attribute value; if **any** is present, the substring may match any substring in the attribute value.

c) **greaterOrEqual**, it is TRUE if and only if the relative ordering (as defined by the appropriate ordering algorithm) places the supplied value before or equal to any value of the attribute;

d) **lessOrEqual**, it is TRUE if and only if the relative ordering (as defined by the appropriate ordering algorithm) places the supplied value after or equal to any value of the attribute;

e) **present**, it is TRUE if and only if such an attribute is present in the entry;

f) **approximateMatch**, it is TRUE if and only if there is a value of the attribute which matches that which is asserted by some locally-defined approximate matching algorithm (e.g., spelling variations, phonetic match, etc). There are no specific guidelines for approximate matching in this version of this part of ISO/IEC 9594. If approximate matching is not supported, this FilterItem should be treated as a match for equality.

7.9 Security Parameters

7.9.1 The **SecurityParameters** govern the operation of various security features associated with a Directory operation.

Note — These parameters are conveyed from sender to recipient. Where the parameters appear in the argument of an abstract-operation the requestor is the sender, and the performer is the recipient. In a result, the roles are reversed.


```
SecurityParameters::= SET {
  certification-path [0]  CertificationPath OPTIONAL,
  name               [1]  DistinguishedName OPTIONAL,
  time               [2]  UTCTime OPTIONAL,
  random             [3]  BIT STRING OPTIONAL,
  target             [4]  ProtectionRequest OPTIONAL }

ProtectionRequest ::=  INTEGER { none(0),
                                 signed (1)}
```


7.9.2 The various components have the meanings as defined in 7.9.2.1 through 7.9.2.5.

7.9.2.1 The **CertificationPath** component consists of the sender's certificate, and, optionally, a sequence of certificate pairs. The certificate is used to associate the sender's public key and distinguished name, and may be used to verify the signature on the argument or result. This parameter shall be present if the argument or result is signed. The sequence of certification pairs consists of certification authority cross certificates. It is used to enable the sender's certificate to be validated. It is not required if the recipient shares the same certification authority as the sender. If the recipient requires a valid set of certificate pairs, and this parameter is not present, whether the recipient rejects the signature on the argument or result, or attempts to generate the certification path, is a local matter.

7.9.2.2 The **name** is the distinguished name of the first intended recipient of the argument or result. For example, if a DUA generates a signed argument, the name is the distinguished name of the DSA to which the operation is submitted.

7.9.2.3 The **time** is the intended expiry time for the validity of the signature, when signed arguments are used.

It is used in conjunction with the random number to enable the detection of replay attacks.

7.9.2.4 The **random number** is a number which should be different for each unexpired token. It is used in conjunction with the time parameter to enable the detection of replay attacks when the argument or result has been signed.

7.9.2.5 The **target ProtectionRequest** may appear only in the request for an operation to be carried out, and indicates the requestor's preference regarding the degree of protection to be provided to the result. Two levels are provided: **none** (no protection requested, the default), and **signed** (the Directory is requested to sign the result). The degree of protection actually provided to the result is indicated by the form of result and may be equal to or lower than that requested, based on the limitations of the Directory.

7.10 OPTIONALLY-SIGNED

7.10.1 An OPTIONALLY-SIGNED information type is one whose values may, at the option of the generator, be accompanied by their digital signature. This capability is specified by means of the following macro:

```
OPTIONALLY-SIGNED MACRO ::=
BEGIN

TYPE NOTATION      ::= type (Type)
VALUE NOTATION     ::= value (VALUE
                CHOICE { Type, SIGNED Type})

END
```

7.10.2 The SIGNED macro, which describes the form of the signed form of the information, is specified in ISO/IEC 9594-8.

8 Bind and Unbind Operations

The DirectoryBind and DirectoryUnbind operations, defined in 8.1 and 8.2 respectively, are used by the DUA at the beginning and end of a particular period of accessing the Directory.

8.1 Directory Bind

8.1.1 A DirectoryBind operation is used at the beginning of a period of accessing the Directory.

```
DirectoryBind ::= ABSTRACT-BIND
        TO { readPort, searchPort, modifyPort }
        BIND
        ARGUMENT    DirectoryBindArgument
        RESULT              DirectoryBindResult
        BIND-ERROR  DirectoryBindError

DirectoryBindArgument       ::=     SET {
        credentials     [0]     Credentials OPTIONAL,
        versions        [1]     Versions DEFAULT {v1988}}

Credentials     ::=     CHOICE {
        simple          [0]     SimpleCredentials,
        strong          [1]     StrongCredentials,
    externalProcedure   [2]     EXTERNAL}

SimpleCredentials ::= SEQUENCE{
        name            [0]     DistinguishedName,
        validity        [1]     SET {
        time1           [0]     UTCTime OPTIONAL,
        time2           [1]     UTCTime OPTIONAL,
        random1         [2]     BIT STRING  OPTIONAL,
        random2         [3]     BIT STRING  OPTIONAL}
                                        OPTIONAL,
                -- in most instances the arguments for
                -- time and random are relevant in
                -- dialogues employing protected
                -- password mechanisms and derive
                -- their meaning as per bilateral
                -- agreement
        password        [2]     OCTET STRING OPTIONAL}
                -- the value would be an unprotected
                -- password, or Protected1 or Protected2
                -- as specified in ISO/IEC 9594-8

StrongCredentials ::=           SET {
        certification-path      [0]     CertificationPath
                                                OPTIONAL,
        bind-token              [1]     Token }

Token ::=       SIGNED SEQUENCE {
        algorithm       [0]     AlgorithmIdentifier,
        name            [1]     DistinguishedName,
        time            [2]     UTCTime,
        random          [3]     BIT STRING }

Versions        ::=     BIT STRING {v1988(0)}

DirectoryBindResult ::= DirectoryBindArgument

DirectoryBindError      ::=     SET {
        versions        [0] Versions DEFAULT {v1988},
        CHOICE {
                serviceError    [1]
                ServiceProblem,
                securityError   [2]     SecurityProblem }}
```

8.1.2 The various arguments are defined in 8.1.2.1 and 8.1.2.2.

8.1.2.1 The **Credentials** of the DirectoryBindArgument allow the Directory to establish the identity of the user. They may be either simple, or strong or externally defined (external Procedure)(as described in ISO/IEC 9594-8].

8.1.2.1.1 SimpleCredentials consist of a name (always the distinguished name of an object) and (optionally) a password. This provides a limited degree of security. If the password is protected as described in clause 5 of ISO/IEC 9594-8, then **SimpleCredentials** includes **name**, **password**, and (optionally) time and/or random numbers which are used to detect replay. In some instances a protected password may be checked by an object which knows the password only after locally regenerating the protection to its own copy of the password and comparing the result with the value in the bind argument (**password**). In other instances a direct comparison may be possible.