INTERNATIONAL STANDARD

ISO/IEC 9594-5

Sixth edition
2008-12-15

Information technology — Open Systems Interconnection — The Directory: Protocol specifications

Technologies de l'information — Interconnexion de systèmes ouverts (OSI) — L'annuaire: Spécifications du protocole

Reference number
ISO/IEC 9594-5:2008(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published by ISO in 2009

Published in Switzerland

CONTENTS

Page

Page

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 9594-5:2008 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems,* in collaboration with ITU-T. The identical text is published as ITU-T Rec. X.519 (01/2008).

This sixth edition cancels and replaces the fifth edition (ISO/IEC 9594-5:2005), which has been technically revised.

ISO/IEC 9594 consists of the following parts, under the general title *Information technology — Open Systems Interconnection — The Directory*:

— *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: Public-key and attribute certificate frameworks*

— *Part 9: Replication*

— *Part 10: Use of systems management for administration of the Directory*

Introduction

This Recommendation | International Standard, together with the other Recommendations | International Standards, has been produced to facilitate the interconnection of information processing systems to provide directory services. A set of such systems, together with the directory information that 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.

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.

This Recommendation | International Standard specifies the application service elements and application contexts for two protocols – the Directory Access Protocol (DAP) and the Directory System Protocol (DSP). The DAP provides for access to the Directory to retrieve or modify Directory information. The DSP provides for the chaining of requests to retrieve or modify Directory information to other parts of the distributed Directory System where the information may be held.

In addition, this Recommendation | International Standard specifies the application service elements and application contexts for the Directory Information Shadowing Protocol (DISP) and the Directory Operational Binding Management Protocol (DOP). The DISP provides for the shadowing of information held in one DSA to another DSA. The DOP provides for the establishment, modification and termination of bindings between pairs of DSAs for the administration of relationships between the DSAs (such as for shadowing or hierarchical relationships).

This Recommendation | International Standard provides the foundation frameworks upon which industry profiles can be defined by other standards groups and industry forums. Many of the features defined as optional in these frameworks may be mandated for use in certain environments through profiles. This sixth edition technically revises and enhances, but does not replace, the fifth edition of this Recommendation | International Standard. Implementations may still claim conformance to the fifth edition. However, at some point, the fifth edition will not be supported (i.e., reported defects will no longer be resolved). It is recommended that implementations conform to this sixth edition as soon as possible.

This sixth edition specifies versions 1 and 2 of the Directory protocols.

The first and second editions specified only version 1. Most of the services and protocols specified in this edition are designed to function under version 1. However some enhanced services and protocols, e.g., signed errors, will not function unless all Directory entities involved in the operation have negotiated version 2. Whichever version has been negotiated, differences between the services and between the protocols defined in the six editions, except for those specifically assigned to version 2, are accommodated using the rules of extensibility defined in this edition of ITU-T Rec. X.519 | ISO/IEC 9594-5.

Annex A, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module for the common specifications for the Directory protocols.

Annex B, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module for the OSI protocol specification.

Annex C, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module for the Directory OSI protocols.

Annex D, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module for the IDM protocol specification.

Annex E, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module for the Directory IDM protocols.

Annex F, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module which contains all the ASN.1 object identifiers assigned to identify operational binding types in this series of Recommendations | International Standards.

Annex G, which is not an integral part of this Recommendation | International Standard, lists the amendments and defect reports that have been incorporated to form this edition of this Recommendation | International Standard.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – Open Systems Interconnection – The Directory: Protocol specifications

1 Scope

This Recommendation | International Standard specifies the Directory Access Protocol, the Directory System Protocol, the Directory Information Shadowing Protocol, and the Directory Operational Binding Management Protocol fulfilling the abstract services specified in ITU-T Rec. X.511 | ISO/IEC 9594-3, ITU-T Rec. X.518 | ISO/IEC 9594-4, ITU-T Rec. X.525 | ISO/IEC 9594-9, and ITU-T Rec. X.501 | ISO/IEC 9594-2.

2 References

2.1 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The basic model*.
- ITU-T Recommendation X.213 (2001) | ISO/IEC 8348:2002, *Information technology – Open Systems Interconnection – Network service definition*.
- ITU-T Recommendation X.214 (1995) | ISO/IEC 8072:1996, *Information technology – Open Systems Interconnection – Transport service definition*.
- ITU-T Recommendation X.500 (2008) | ISO/IEC 9594-1:2008, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services*.
- ITU-T Recommendation X.501 (2008) | ISO/IEC 9594-2:2008, *Information technology – Open Systems Interconnection – The Directory: Models*.
- ITU-T Recommendation X.509 (2008) | ISO/IEC 9594-8:2008, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.
- ITU-T Recommendation X.511 (2008) | ISO/IEC 9594-3:2008, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition*.
- ITU-T Recommendation X.518 (2008) | ISO/IEC 9594-4:2008, *Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation*.
- ITU-T Recommendation X.520 (2008) | ISO/IEC 9594-6:2008, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types*.
- ITU-T Recommendation X.521 (2008) | ISO/IEC 9594-7:2008, *Information technology – Open Systems Interconnection – The Directory: Selected object classes*.
- ITU-T Recommendation X.525 (2008) | ISO/IEC 9594-9:2008, *Information technology – Open Systems Interconnection – The Directory: Replication*.
- ITU-T Recommendation X.530 (2008) | ISO/IEC 9594-10:2008, *Information technology – Open Systems Interconnection – The Directory: Use of systems management for administration of the Directory*.

- ITU-T Recommendation X.680 (2008) | ISO/IEC 8824-1:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (2008) | ISO/IEC 8824-2:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (2008) | ISO/IEC 8824-3:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (2008) | ISO/IEC 8824-4:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*
- ITU-T Recommendation X.690 (2008) | ISO/IEC 8825-1:2008, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

2.1.2 ISO/IEC Standards

- ISO/IEC 10646:2003, *Information technology – Universal Multiple-Octet Coded Character Set (UCS).*

2.1.3 Other references

- ITU-T Recommendation E.164 (2005), *The international public telecommunication numbering plan.*
- ITU-T Recommendation X.121 (2000), *International numbering plan for public data networks.*
- IETF RFC 2025 (1996), *The Simple Public-Key GSS-API Mechanism (SPKM).*
- IETF RFC 793 (1981), *Transmission Control Protocol – DARPA Internet Program – Protocol Specification.*
- IETF RFC 1738 (1994), *Uniform Resource Locators (URL).*
- IETF RFC 2246 (1999), *The TLS Protocol Version 1.0.*
- IETF RFC 4511 (2006), *Lightweight Directory Access Protocol (LDAP): The Protocol.*
- IETF RFC 3546 (2003), *Transport Layer Security (TLS) Extensions.*
- IETF RFC 3986 (2005), *Uniform Resource Identifier (URI): Generic Syntax.*

2.2 Non-normative references

- ITU-T Recommendation X.217 (1995) | ISO/IEC 8649:1996, *Information technology – Open Systems Interconnection – Service definition for the Association Control Service Element.*
- ITU-T Recommendation X.224 (1995) | ISO/IEC 8073:1997, *Information technology – Open Systems Interconnection – Protocol for providing the connection-mode transport service.*
- ITU-T Recommendation X.225 (1995) | ISO/IEC 8327-1:1996, *Information technology – Open Systems Interconnection – Connection-oriented Session protocol: Protocol specification.*
- ITU-T Recommendation X.226 (1994) | ISO/IEC 8823-1:1994, *Information technology – Open Systems Interconnection – Connection-oriented Presentation protocol: Protocol specification.*
- ITU-T Recommendation X.227 (1995) | ISO/IEC 8650-1:1996, *Information technology – Open Systems Interconnection – Connection-oriented protocol for the Association Control Service Element: Protocol specification.*
- ITU-T Recommendation X.881 (1994) | ISO/IEC 13712-2:1995, *Information technology – Remote Operations: OSI realizations – Remote Operations Service Element (ROSE) service definition.*
- IETF RFC 1006 (1987), *ISO Transport Service on top of the TCP Version: 3.*
- IETF RFC 1277 (1991), *Encoding Network Addresses to Support Operation over Non-OSI Lower Layers.*
- IETF RFC 2126 (1997), *ISO Transport Service on top of TCP (ITOT).*

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply:

3.1 Basic Directory definitions

The following terms are defined in ITU-T Rec. X.501 | ISO/IEC 9594-2:

a) *the Directory*;

b) *(Directory) user*;

c) *Directory System Agent (DSA)*;

d) *Directory User Agent (DUA)*.

3.2 Distributed Operation Definitions

The following terms are defined in ITU-T Rec. X.518 | ISO/IEC 9594-4:

a) *chaining*;

b) *referral*.

3.3 Protocol specification definitions

The following terms are defined in this Recommendation | International Standard.

NOTE – The terms defined in this subclause are generalized definitions to cover both the OSI and the TCP/IP case, except exceptions as indicated.

3.3.1 abstract syntax: The specification of a data types and/or data values by using notation rules which are independent of the encoding technique used to represent them.

3.3.2 application-association: A cooperative relationship between two application-entities established by the Bind operation.

3.3.3 application-context: (OSI only definition) A set of rules shared in common by two application-entities in order to support an application-association.

3.3.4 application-context-name: An ASN.1 object identifier that identifies (names) an application-context.

3.3.5 Application Layer: The top layer of the OSI seven layer model representing the semantics of the communication.

3.3.6 application-entity: A representation of the external behaviour of an application process in the form of its communication capabilities.

3.3.7 application-entity title: The Directory distinguished name of an application-entity, and in particular, an application-entity representing a Directory application process.

3.3.8 application process: A process within a system which performs information processing for a particular purpose, in particular processing Directory operations.

3.3.9 Bind operation: An operation type used for establishing an application-association.

3.3.10 Directory operation: An operation type for exchange of Directory information.

3.3.11 directory protocol-data-unit: A unit of data for a Directory protocol consisting of control information and in the general case also application data as specified by Directory operations.

NOTE 1 – A Directory PDU in the OSI environment includes all the protocol elements of the OSI Presentation Layer and if relevant, protocol elements of ACSE in addition to the Directory-specific protocol elements.

NOTE 2 – The term "application-protocol-data-unit (APDU)" is a unit of data defined by an OSI application protocol. This term is not used for edition 5 and subsequent editions of these Directory Specifications. However, the abbreviation may appear in certain ASN.1 elements.

3.3.12 initiator: The application process that initiates an application-association by issuing a Bind request.

3.3.13 local matter: A decision made by a system concerning its behaviour that is not subject to the requirements of these Directory Specifications.

3.3.14 operation: An exchange between two application processes to perform a particular task. It consists of a request from one application-process to the other one and the return of zero or more responses (result and/or errors). An operation implies a certain process to be performed by the application process receiving the request.

3.3.15 protocol-data-unit: Comprised of the presentation protocol elements or the ACSE protocol elements of a Directory protocol-data-unit.

3.3.16 **Presentation Layer**: The sixth layer of the OSI Reference Model.

3.3.17 **protocol error**: An unrecognized or unexpected protocol-data-unit or a protocol-data-unit with an unexpected or invalid parameter is received.

3.3.18 **responder**: The application-process that receives a Bind request and either accepts or refuses the application-association.

NOTE – Initiator and responder are defined with respect to a single transport-connection. The initiator is also the application process that initiated the transport-connection (see 8.4). A DSA can be both an initiator and responder simultaneously.

3.3.19 **session layer**: The fifth layer of the OSI Reference Model.

3.3.20 **session-protocol-data-unit**: (OSI only definition) A unit of data at the OSI Session Layer consisting of control information and in the general case also carries a Directory protocol-data-unit.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

AC	Application Context
ACSE	Association Control Service Element
AE	application-entity
AFI	Authority and Format Identifier
APDU	application-protocol-data-unit
DAP	Directory Access Protocol
DISP	Directory Information Shadowing Protocol
DOP	Directory Operational Binding Management Protocol
DSA	Directory System Agent
DSP	Directory System Protocol
DSP	Domain Specific Part
DUA	Directory User Agent
IDI	Initial Domain Identifier
IDM	Internet Directly Mapped
IPv4	Internet Protocol, Version 4
IPv6	Internet Protocol, Version 6
ITOT	ISO Transport Service on top of TCP
LDAP	Lightweight Directory Access Protocol
NSAP	Network-Service-Access-Point
PDU	protocol-data-unit
PPDU	presentation-protocol-data-unit
SPDU	session-protocol-data-unit
TCP/IP	Transmission Control Protocol/Internet Protocol
TPDU	transport-protocol-data-unit
TPKT	Transport Packet
TSDU	transport-service-data-unit
URI	Uniform Resource Identifier

5 Conventions

The term "Directory Specification" (as in "this Directory Specification") shall be taken to mean ITU-T Rec. X.519 | ISO/IEC 9594-5. The term "Directory Specifications" shall be taken to mean the X.500-series Recommendations and all parts of ISO/IEC 9594.

This Directory Specification uses the term *first edition systems* to refer to systems conforming to the first edition of the Directory Specifications, i.e., the 1988 edition of the series of CCITT X.500 Recommendations and the ISO/IEC 9594:1990 edition.

This Directory Specification uses the term *second edition systems* to refer to systems conforming to the second edition of the Directory Specifications, i.e., the 1993 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:1995 edition.

This Directory Specification uses the term *third edition systems* to refer to systems conforming to the third edition of the Directory Specifications, i.e., the 1997 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:1998 edition.

This Directory Specification uses the term *fourth edition systems* to refer to systems conforming to the fourth edition of the Directory Specifications, i.e., the 2001 editions of ITU-T Recs X.500, X.501, X.511, X.518, X.519, X.520, X.521, X.525, and X.530, the 2000 edition of ITU-T Rec. X.509, and parts 1-10 of the ISO/IEC 9594:2001 edition.

This Directory Specification uses the term *fifth edition systems* to refer to systems conforming to the fifth edition of the Directory Specifications, i.e., the 2005 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:2005 edition.

This Directory Specification uses the term *sixth edition systems* to refer to systems conforming to the sixth edition of the Directory Specifications, i.e., the 2008 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:2008 edition.

This Directory Specification presents ASN.1 notation in the bold Helvetica typeface. When ASN.1 types and values are referenced in normal text, they are differentiated from normal text by presenting them in the bold Helvetica typeface. The names of procedures, typically referenced when specifying the semantics of processing, are differentiated from normal text by displaying them in bold Times. Access control permissions are presented in italicized Times.

If the items in a list are numbered (as opposed to using "–" or letters), then the items shall be considered steps in a procedure.

6 Common protocol specification

6.1 Directory associations and operations

The protocols for these Directory Specifications are described as a set of *operations*. An operation is defined in terms of a request sent from one system to another system expecting this other system to process the request, and if applicable, returns one or more replies constituting the result. An operation can either be a *Bind operation* or an operation invoked to access Directory information (a *Directory operation*).

If exception conditions are encountered, one or more errors may be returned instead of or in addition to possible results.

NOTE 1 – The currently defined operations will return either one or more results or a single error.

Directory protocols defined by these Directory Specifications may use an OSI protocol stack, a TCP/IP protocol stack or both. The specification provided by this clause is independent of the particular protocol stack. The OSI specific specification is given in clauses 7 and 8, while the TCP/IP specific specification is given in clauses 9 and 10.

A process within a system that processes Directory operations is called an *application process*. An *application-entity* is the reflection of the external behaviour of an application process.

Before Directory operations can be invoked between two Directory application processes, an *application-association* has to be established between the corresponding application-entities. An application-association is a cooperative relationship between two application-entities formed by exchange of control information within the request and result of a Bind operation and by the use of a common underlying service.

NOTE 2 – This is a modified definition of application-association as given by ITU-T Rec. X.217 | ISO/IEC 8649, and is intended to cover both the use of an underlying OSI protocol stack and an underlying TCP/IP stack.

An application-association is terminated using an unbind exchange. The unbinding of an application-association is not defined as an operation.

6.2 Specification for Directory operations

These Directory Specifications specify several operation types. An operation type is specified by the **OPERATION** ASN.1 information object class. Possible errors associated with an operation type are defined by the **ERRORS** ASN.1 information object class.

```
OPERATION ::= CLASS {
        &ArgumentType     OPTIONAL,
        &ResultType       OPTIONAL,
        &Errors           ERROR OPTIONAL,
        &operationCode    Code UNIQUE OPTIONAL }
WITH SYNTAX {
        [ARGUMENT &ArgumentType]
        [RESULT           &ResultType]
        [ERRORS           &Errors]
        [CODE             &operationCode] }

ERROR ::= CLASS {
        &ParameterType,
        &errorCode        Code UNIQUE OPTIONAL }
WITH SYNTAX {
        PARAMETER         &ParameterType
        [CODE             &errorCode] }

Code ::= CHOICE {
        local       INTEGER,
        global      OBJECT IDENTIFIER }
```

The **OPERATION** information object class is a convenient way to express the syntax of Directory requests, results and errors for a particular operation type.

This ASN.1 information object class has the following fields:

a) The **&ArgumentType** field specifies an open data type for the request part of an operation.

b) The **&ResultType** field specifies an open data type for one or more replies constituting the result of the request. If this field is absent, there is no result associated with the operation.

c) The **&Errors** field specifies one or more errors that can occur as the result of processing the request. If this field is absent, there is no error associated with the operation.

d) The **&operationCode** field specifies the type of Directory operation to be performed. This field is absent for the Bind operation. See 6.4 for currently defined operation codes.

Directory operations may in principle be performed in two different modes:

a) if a Directory operation shall be completed before a new Directory operation may be invoked, the mode of operation is *synchronous*; or

b) if several operations may be in progress at the same time, the mode of operation is *asynchronous*.

If all Directory operations defined for a particular type of application-association:

a) consist of both a request and one or more results and/or errors; and

b) only are allowed to be invoked by a designated system,

such operation may be executed in either synchronous or asynchronous mode. Otherwise, the mode of operation is always asynchronous.

The **OPERATION** information object class does not in itself imply any sequencing. A Directory request may have no result and/or error, or a request may have several results and/or errors. However, it does tie together a request with possible responses (results and errors) by carrying the same operation code and the same invoke id (see below). However, specification of a particular operation type may dictate sequencing restrictions.

An error is a report of the unsuccessful performance of an operation. An error is represented by the **ERROR** ASN.1 Information Object Class. The different fields are described below:

a) the **&ParameterType** field specifies the data type of the parameter of the error specifying the nature of the error; and

b) the **&errorCode** field specifies the code that identifies the error (see 6.5 for the defined error codes).

Although not reflected by the **OPERATION** or the **ERRORS** information object classes, each invocation of a Directory operation is assigned an **InvokeId**, which is carried in the protocol. This makes it possible to indicate to what Directory operation a particular request, result or error belongs. The definition of the **InvokeId** is as follows:

```
InvokeId ::= CHOICE {
        present     INTEGER,
        absent      NULL }
```

If an operation type does not specify an **&operationCode**, operations of this type cannot have **InvokeId** assigned.

6.3 Directory protocol overview

6.3.1 Use of underlying services

When two application processes from different open systems interact, the application-association is realized as an Application Layer protocol using either an OSI or a TCP/IP underlying service.

Details on the use of the OSI service are given in clause 8, while the details on the use of the TCP/IP service are given in clause 10.

The OSI Transport Layer may either be supported using the service as defined in ITU-T Rec. X.214 | ISO/IEC 8072 or by using the specification in 8.5. In this latter case, the OSI upper layer protocols stack are placed on the top of a TCP/IP protocol stack.

6.3.2 The Directory Access Protocol (DAP)

Before a DUA and a DSA from different open systems can interact, a Bind operation has to be invoked between them to establish an application-association supporting a Directory protocol called the Directory Access Protocol (DAP).

The Bind operation (**directoryBind**) for establishing a DAP application-association is defined in clause 8 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

This edition and all previous editions of the Directory Specifications only allow a DUA to invoke a Bind operation and to initiate subsequent Directory operations. If the OSI underlying stack is used, Directory operations may be invoked either in synchronous mode or in asynchronous mode. If the TCP/IP underlying stack is used, Directory operations are always invoked in asynchronous mode.

All Directory operations require either a single reply or a single error to be returned.

6.3.3 The Directory System Protocol (DSP)

Before a pair of DSAs from different open systems can interact, a Bind operation has to be invoked between them to establish an application-association supporting a Directory protocol called the Directory System Protocol (DSP).

The Bind operation (**dSABind**) for establishing a DSP application association is defined in clause 11 of ITU-T Rec. X.518 | ISO/IEC 9594-4.

Either DSA may invoke a Bind operation. Both the initiating and responding DSA may invoke subsequent Directory operations. Directory operations are always invoked in asynchronous mode on the DSP.

All Directory operations require either a single reply or a single error to be returned.

6.3.4 The Directory Information Shadowing Protocol (DISP)

Before a pair of DSAs from different open systems can interact for the purpose of exchanging shadowing information, a Bind operation has to be invoked between them to establish an application-association supporting a Directory protocol called the Directory Information Shadowing Protocol (DISP).

The Bind operation (**dSAShadowBind**) for establishing a DISP application-association is defined in clause 7.4.1 of ITU-T Rec. X.525 | ISO/IEC 9594-9.

If the OSI underlying stack is used, the mode of operation is synchronous or asynchronous depending on the application-context selected for the Bind operation. If the TCP/IP underlying stack is used, Directory operations are always invoked in asynchronous mode.

All Directory operations require either a single reply or a single error to be returned.

6.3.5 The Directory Operational Binding Management Protocol (DOP)

Before a pair of DSAs from different open systems can interact for the purpose of maintaining operational bindings, a Bind operation has to be invoked to establish an application-association supporting a Directory protocol called the Directory Operational Binding Management Protocol (DOP).

The DSA that may assume the role of initiator of the Bind operation depends on the DSA roles assigned for the operational binding(s) to be managed using the Directory operations on the application-association. Only the initiator may invoke Directory operations. More than one operational binding type may only be managed within this application-association if the DSA roles for the distinct types are compatible (e.g., a DSA assumes Role A for each binding type).

All Directory operations require either a single reply or a single error to be returned.

6.4 Operation codes

6.4.1 Operation codes for DAP and DSP

The following operation codes are used in the DAP and the DSP:

id-opcode-read **Code ::= local : 1**
id-opcode-compare **Code ::= local : 2**
id-opcode-abandon **Code ::= local : 3**
id-opcode-list **Code ::= local : 4**
id-opcode-search **Code ::= local : 5**
id-opcode-addEntry **Code ::= local : 6**
id-opcode-removeEntry **Code ::= local : 7**
id-opcode-modifyEntry **Code ::= local : 8**
id-opcode-modifyDN **Code ::= local : 9**

The use of these operation codes is specified in ITU-T Rec. X.511 | ISO/IEC 9594-3.

6.4.2 Operation codes for DISP

The following operation codes are used in the DISP.

id-opcode-requestShadowUpdate **Code ::= local : 1**
id-opcode-updateShadow **Code ::= local : 2**
id-opcode-coordinateShadowUpdate **Code ::= local : 3**

The use of these operation codes is specified in ITU-T Rec. X.525 | ISO/IEC 9594-9.

6.4.3 Operation codes for DOP

The following operation codes are used in the DOP.

id-op-establishOperationalBinding **Code ::= local : 100**
id-op-modifyOperationalBinding **Code ::= local : 102**
id-op-terminateOperationalBinding **Code ::= local : 101**

The use of these operation codes is specified in ITU-T Rec. X.501 | ISO/IEC 9594-2.

6.5 Error codes

6.5.1 Error codes for DAP and DSP

The following error codes are used in the DAP and the DSP. The code **id-errcode-referral** is only used in the DAP. The code **id-opcode-dsaReferral** is only used in the DSP:

id-errcode-attributeError **Code ::= local : 1**
id-errcode-nameError **Code ::= local : 2**
id-errcode-serviceError **Code ::= local : 3**
id-errcode-referral **Code ::= local : 4**
id-errcode-abandoned **Code ::= local : 5**
id-errcode-securityError **Code ::= local : 6**
id-errcode-abandonFailed **Code ::= local : 7**
id-errcode-updateError **Code ::= local : 8**
id-errcode-dsaReferral **Code ::= local : 9**

6.5.2 Error codes for DISP

The following error code is used in the DISP:

id-errcode-shadowError **Code ::= local : 1**

6.5.3 Error codes for DOP

The following error code is used in the DOP:

id-err-operationalBindingError **Code ::= local : 100**

6.6 Abstract syntaxes

A protocol specification includes a specification of the data types that may be transferred as part of the protocol exchanges. The data types are defined using an abstract notation like the ASN.1 notation and constitute the abstract syntax for the protocol. The abstract syntaxes are quite similar for OSI communication and for TCP/IP communication, although there are differences. Four abstract syntaxes are defined for each of these types of communication corresponding to the four different Directory protocols. Only for the OSI communication are the abstract syntaxes assigned object identifiers. When establishing an OSI application-association the relevant object identifier for the abstract syntax is signalled in the Bind (see 7.6.1).

7 Directory protocols using the OSI stack

This clause defines the Directory protocols and their mapping onto the OSI Session Protocol. It incorporates the relevant elements of the OSI Presentation Protocol as defined by ITU-T Rec. X.226 | ISO/IEC 8823-1 and the Association Control Service Element (ACSE) as defined by ITU-T Rec. X.227 | ISO/IEC 8650-1.

The relevant part of the OSI session protocol is defined in 8.3.

7.1 OSI-PDUs

The messages of the OSI based protocols are conveyed over an OSI application-association as Directory protocol-data-units represented by the **OSI-PDU** data type as follows:

```
OSI-PDU {APPLICATION-CONTEXT:protocol}::= TYPE-IDENTIFIER.&Type (
        OsiBind { {protocol} } |
        OsiBindResult { {protocol} } |
        OsiBindError { {protocol} } |
        OsiOperation { {protocol.&Operations} } |
        PresentationAbort )
```

7.2 Directory PDU structure

A Directory PDU in the OSI environment consists of protocol elements from the OSI Presentation Layer as defined by ITU-T Rec. X.226 | ISO/IEC 8823-1, of ACSE protocol elements as defined by ITU-T Rec. X.227 | ISO/IEC 8650-1, if relevant, and Directory specific protocol elements for the protocol in question.

The **OsiBind**, the **OsiBindResult** and the **OsiBindError** have presentation protocol elements and ACSE protocol elements in addition to the Directory specific protocol elements, while the **OsiOperation** only has presentation protocol elements in addition to the Directory specific protocol elements. The **PresentationAbort** has only presentation protocol elements.

The Presentation Layer protocol elements included within a specific Directory PDU comprise a PPDU.

NOTE 1 – The term PPDU (presentation-protocol-data-unit) is introduced here, as it is referenced when discussing presentation protocol errors and by the **Abort-reason** data type. The term is otherwise not relevant for these Directory Specifications.

The ACSE protocol elements included within a specific Directory PDU comprise an ACSE PDU.

NOTE 2 – ITU-T Rec. X.227 | ISO/IEC 8650-1 uses the term APDU (application-protocol-data-unit) for an ACSE PDU. As the Directory specific protocol elements of a specific Directory PDU in principle also comprise an APDU, the term ACSE PDU is used here to avoid confusion.

The following PPDUs are used by this Directory Specification:

a) CP PPDU, which is reflected by the **CP-type** data type defined by ITU-T Rec. X.226 | ISO/IEC 8823-1. It is a part of the **OsiBind** data type;

b) CPA PPDU, which is reflected by the **CPA-PPDU** data type defined by ITU-T Rec. X.226 | ISO/IEC 8823-1. It is a part of the **OsiBindResult** data type;

c) CPR PPDU, which is reflected by the **CPR-PPDU** data type defined by ITU-T Rec. X.226 | ISO/IEC 8823-1. It is part of the **OsiBindError** data type;

d) TD PPDU, which is reflected by the **User-data** data type defined by ITU-T Rec. X.226 | ISO/IEC 8823-1. It is part of the **OsiOperation** data type;

e) ARU PPDU, which is reflected by the **ARU-PPDU** defined by ITU-T Rec. X.226 | ISO/IEC 8823-1. It is part of the **ARU-PPDU** data type as defined by this Directory specification; and

f) ARP PPDU, which is reflected by the **ARP-PPDU** defined by ITU-T Rec. X.226 | ISO/IEC 8823-1. It constitutes the **ARP-PPDU** data type as defined by this Directory specification.

There is no PPDU defined for the release of an application-association (**OsiUnbind** and **OsiUnbindResult**). However, the **User-data** data type defined by ITU-T Rec. X.226 | ISO/IEC 8823-1 is used to carry the **OsiUnbind** and **OsiUnbindResult**.

The following ACSE PDUs are used by this Directory Specification:

a) **AARQ-apdu** is a part of the **OsiBind** data type;

b) **AARE-apdu** is part of the **OsiBindResult** data type and the **OsiBindError** data type;

c) **RLRQ-apdu** is part of the **OsiUnbind** data type;

d) **RLRE-apdu** is part of the **OsiUnbindResult** data type; and

e) **ABRT-apdu** is part of the **ARU-PPDU** data type.

7.3 Session PDUs

In addition to the Directory PDUs, this Directory Specification also defines session-protocol-data-units (SPDUs). All the Directory PDUs are carried within a SPDU.

The following SPDUs are used by this Directory Specification:

a) CONNECT SPDU used to carry the **OsiBind**;

b) ACCEPT SPDU used to carry the **OsiBindResult**;

NOTE – The AARE ACSE PDU (as represented by **AARE-apdu** and **AAREerr-apdu**) is according to 8.1.3 of ITU-T Rec. X.227 | ISO/IEC 8650-1 mapped onto the P-CONNECT response/confirm, where result is set as 'user rejection'. According to 6.2.5.6 of ITU-T Rec. X.226 | ISO/IEC 8823-1, a CPR PPDU shall be issued at the Presentation Layer. Also, according to 7.1.3 of ITU-T Rec. X.226 | ISO/IEC 8823-1, the CPR PPDU is conveyed in the S-CONNECT response and confirm session primitives.

c) REFUSE SPDU is used to carry **OsiBindError** and it is used for rejecting an application-association due to Session Layer conditions;

d) FINISH SPDU is used to carry the **OsiUnbind** to initiate termination of an application-association;

e) DISCONNECT SPDU is used to carry the **OsiUnbindResult** to complete termination of an application-association;

f) ABORT SPDU is used to carry the **ARU-PPDU** and **ARP-PPDU** in addition to be used on its own when aborting due to a Session Layer problem;

g) ABORT ACCEPT SPDU carries no upper layer information, but it does indicate that an abort has been received by the peer system; and

h) DATA TRANSFER SPDU is used to carry **OsiOperation**.

Details on SPDUs are given in 8.3.

7.4 OSI addressing

OSI defines addresses for the Network Layer and up to and including the Presentation Layer. The Address on the top of the Network Layer is called the network-service-access-point (NSAP) address. The structure of an NSAP address is defined in ITU-T Rec. X.213 | ISO/IEC 8348. A transport-address on top of the Transport Layer is defined as the NSAP address plus an optional transport-selector. A session-address on top of the Session Layer is defined as the transport-address plus an optional session-selector. A presentation-address is defined as a session-address plus an optional presentation-selector. Only session-selector and presentation-selector are referenced by this Directory Specification.

7.5 Procedure and sequencing

An application-association between two application processes is initiated by one of the application processes issuing an **OsiBind** as defined in 7.6.1. The initiating application process shall then wait for an **OsiBindResult** to confirm the application-association establishment before sending any Directory PDU on that application-association.

Independent of any sequencing rule, the initiating application process may at any time issue an **ARU-PPDU** or **ARP-PPDU** (see 7.6.7) after having issued an **OsiBind**. Likewise, the responding application may at any time issue an **ARU-PPDU** or **ARP-PPDU** after having received an **OsiBind**.

If an **OsiBindResult** is received, the initiating application process may send **OsiOperation** containing **OsiReq**, **OsiRes**, **OsiErr** and **OsiRej** as governed by the protocol in question.

An application-association is not established if an **OsiBindError** (see 7.6.3) is received in response to the **OsiBind**, or if the application-association is refused at the session level (see 8.3.5).

Two application processes may almost simultaneously issue **OsiBind** to each other. This shall be considered as two independent application-association establishment attempts. If they both succeed, the result will be two application-associations.

Protocol errors can occur within the session protocol, within the presentation protocol elements, within the ACSE protocol elements and within the Directory-specific protocol elements.

A protocol error can be caused by:

a) an unrecognized or unexpected PDU received; or

b) one or more parameters on a received PDU are invalid or unexpected.

NOTE 1 – According to the rules of extensibility specified in clause 12, unknown parameters shall be ignored. Clauses 8.5 of ITU-T Rec. X.226 | ISO/IEC 8823-1 and 7.4 of ITU-T Rec. X.227 | ISO/IEC 8650-1 specify similar rules.

NOTE 2 – Clauses 6.4.4.2 and 6.4.4.3 of ITU-T Rec. X.226 | ISO/IEC 8823-1 make a distinction between a protocol error and an invalid PPDU. As the two cases cause the same type of abort, this Directory Specification does not make that distinction. Clause 7.3.3.4 of ITU-T Rec. X.227 | ISO/IEC 8650-1 does not make that distinction either.

In both cases, the application-association or an application-association under establishment/termination shall be aborted.

If the problem is detected within the session protocol, an ABORT SPDU shall be issued (see 8.3.8) with no User Data.

If the problem is detected within the presentation protocol, an **ARP-PPDU** (see 7.6.7.2) shall be issued.

If the problem is detected within the ACSE protocol, an **ARU-PPDU** with **abort-source** set to **acse-service-provider** (see 7.6.7.1) shall be issued.

If the problem is detected within a Directory protocol, an **ARU-PPDU** with **abort-source** set to **acse-service-user** shall be issued.

7.6 Directory PDU specifications

7.6.1 OSI Bind request

```
OsiBind {APPLICATION-CONTEXT:Protocols}  ::=  SET {
        mode-selector                           [0]     IMPLICIT SET {mode-value [0] IMPLICIT INTEGER (1) },
        normal-mode-parameters                  [2]     IMPLICIT SEQUENCE {
                protocol-version                        [0]     IMPLICIT BIT STRING {version-1(0)}
                                                                        DEFAULT {version-1},
                calling-presentation-selector           [1]     IMPLICIT Presentation-selector OPTIONAL,
                called-presentation-selector            [2]     IMPLICIT Presentation-selector OPTIONAL,
                presentation-context-definition-list    [4]     IMPLICIT Context-list,
                user-data                                       CHOICE {
                        fully-encoded-data                      [APPLICATION 1] IMPLICIT
                                                                        SEQUENCE SIZE (1) OF SEQUENCE {
                                transfer-syntax-name                    Transfer-syntax-name OPTIONAL,
                                presentation-context-identifier         Presentation-context-identifier,
                                presentation-data-values                CHOICE {
                                        single-ASN1-type                [0]     AARQ-apdu {{Protocols}} } } } } }

Presentation-selector  ::=  OCTET STRING(SIZE (1..4, ..., 5..MAX))

Context-list ::= SEQUENCE SIZE (2) OF
        SEQUENCE {
                presentation-context-identifier   Presentation-context-identifier,
                abstract-syntax-name              Abstract-syntax-name,
                transfer-syntax-name-list         SEQUENCE OF Transfer-syntax-name }

Presentation-context-identifier  ::=  INTEGER(1..127, ..., 128..MAX)

Abstract-syntax-name  ::=  OBJECT IDENTIFIER

Transfer-syntax-name  ::=  OBJECT IDENTIFIER

AARQ-apdu {APPLICATION-CONTEXT:Protocols}  ::=  [APPLICATION 0]  IMPLICIT SEQUENCE {
        protocol-version                        [0]     IMPLICIT BIT STRING {version1(0)} DEFAULT {version1},
        application-context-name                [1]     Application-context-name,
        called-AP-title                         [2]     Name                                    OPTIONAL,
        called-AE-qualifier                     [3]     RelativeDistinguishedName               OPTIONAL,
        called-AP-invocation-identifier         [4]     AP-invocation-identifier                OPTIONAL,
        called-AE-invocation-identifier         [5]     AE-invocation-identifier                OPTIONAL,
```

```
calling-AP-title                    [6]   Name                                  OPTIONAL,
calling-AE-qualifier                [7]   RelativeDistinguishedName            OPTIONAL,
calling-AP-invocation-identifier    [8]   AP-invocation-identifier             OPTIONAL,
calling-AE-invocation-identifier    [9]   AE-invocation-identifier             OPTIONAL,
implementation-information          [29]  IMPLICIT Implementation-data        OPTIONAL,
user-information                    [30]
        IMPLICIT SEQUENCE SIZE(1) OF [UNIVERSAL 8] IMPLICIT SEQUENCE {
    direct-reference                    OBJECT IDENTIFIER OPTIONAL,
    indirect-reference                  Presentation-context-identifier,
    encoding                            CHOICE {
        single-ASN1-type                    [0]   TheOsiBind {{Protocols}} } } }
```

NOTE – The **user-information** component is in ITU-T Rec. X.226 | ISO/IEC 8823-1 defined as an **EXTERNAL**. As the content of the external is known it could be an aid for implementers if the exact encoding of the **EXTERNAL** is provided. The external is here presented according to the encoding as defined by ITU-T Rec. X.690 | ISO/IEC 8825-1, ITU-T Rec. X.691 | ISO/IEC 8825-2 and ITU-T Rec. X.693 | ISO/IEC 8825-4. This is not completely legal ASN.1. The formal and legal ASN.1 specification using the **EXTERNAL** notation is provided in Annex B.

```
Application-context-name ::= OBJECT IDENTIFIER

AP-invocation-identifier ::= INTEGER

AE-invocation-identifier ::= INTEGER

Implementation-data ::= GraphicString

TheOsiBind {APPLICATION-CONTEXT:Protocols} ::=
        [16]  APPLICATION-CONTEXT.&bind-operation.&ArgumentType ({Protocols})
```

The **OsiBind** is used for initiating an application-association. The **OsiBind** includes presentation protocol elements (see 7.6.1.1), ACSE protocol elements (see 7.6.1.2) and the Directory Bind protocol elements (see 7.6.1.3). The Bind request shall be formatted according to the specification given in mentioned subclauses.

The **OsiBind** is carried in the User Data parameter or the Extended User Data parameter of the Session CONNECT SPDU (see 8.3.3).

The responder of the application-association shall check the protocol elements in the following sequence:

1) The session protocol elements shall be checked. If one or more of these protocol elements are unacceptable, a REFUSE SPDU (see 8.3.5) shall be returned. Otherwise, continue.
2) The presentation protocol elements shall be checked. If one or more of these protocol elements are unacceptable, an **OsiBindError** including a **provider-reason** component and excluding a **user-data** component (see 7.6.3.1) shall be returned. Otherwise, continue.
3) The ACSE protocol elements shall be checked. If one or more of these protocol elements are unacceptable, an **OsiBindError** shall be returned with **result** and **result-source-diagnostic** components of the **AAREerr-apdu** present and with the **user-information** component absent as specified in 7.6.3.2. Otherwise, continue.
4) The Directory Bind shall be checked according to rules for the Directory protocol in question. An **OsiBindResult** (see 7.6.2) shall be returned if the responder is able to accept the Directory Bind. Otherwise, an **OsiBindError** shall be returned with the **user-information** component of the **AAREerr-apdu** present.

If a protocol error is detected at any time during that sequence, the appropriate abort shall be issued as specified in 7.5.

7.6.1.1 Presentation protocol elements

The presentation protocol elements constituting a CP PPDU are those defined by the **OsiBind** data type above except for the embedded **AARQ-apdu**.

The **mode-selector** component shall always be set to 1.

NOTE 1 – ITU-T Rec. X.226 | ISO/IEC 8823-1 defines two modes of presentation-connection. These Directory Specifications always use the **normal-mode**.

The **normal-mode-parameters** component has the following subcomponents:

a) The **protocol-version** subcomponent shall be omitted or set to **version-1**. If specified differently, the responder shall return an **OsiBindError** with **provider-reason** set to **protocol-version-not-supported**.
b) The value of the **calling-presentation-selector** subcomponent, if supplied, shall be obtained from locally held information.

For a definition of presentation-selector, see 7.4.

c) The value of the **called-presentation-selector** subcomponent, if supplied, shall be obtained from:

- information obtained from the **AccessPoint** value of a **ContinuationReference** as the result of a previous Directory operation (see ITU-T Rec. X.518 | ISO/IEC 9594-4); or
- locally held information.

If the responder does not use presentation-selector addressing or if the supplied presentation-selector is not one for a Directory application-process, then the responder shall return an **OsiBindError** with **provider-reason** set to **called-presentation-address-unknown**.

d) The **presentation-context-definition-list** subcomponent shall have two elements, each being a sequence type with:

- a **presentation-context-identifier** that is selected by the initiator. It shall be an uneven integer and shall be different for the two elements;
- an **abstract-syntax-name**, which
  - i) for one of the elements shall be an object identifier identifying the ACSE abstract syntax (**id-acseAS**); and
  - ii) for the other element shall be an object identifier for a Directory abstract syntax corresponding to the type of application-association to be established (**id-as-directoryAccessAS**, **id-as-directorySystemAS**, **id-as-directoryShadowAS** or **id-as-directoryOperationalBindingManagementAS**, as appropriate);
- a **transfer-syntax-name-list**, which shall consist of a single element being the object identifier for the Basic Encoding Rules (BER);

NOTE 2 – ITU-T Rec. X.226 | ISO/IEC 8823-1 allows several transfer syntaxes to be suggested, where one of those is then elected by the responder. The extensibility rules defined in clause 12 require the use of BER.

See 8.1 for details on abstract syntaxes and transfer syntaxes.

e) The **user-data** subcomponent has the following elements:

NOTE 3 – The **user-data** subcomponent reflects the **fully-encoded-data** choice of the **user-data** of the CP PPDU defined by ITU-T Rec. X.226 | ISO/IEC 8823-1. The **fully-encoded-data** consists of a sequence of **PVD-list**. This Directory Specification requires exactly one **PVD-list**. Therefore the sequence-of type specifies exactly one value.

- the **transfer-syntax-name** subcomponent, if present, shall be the object identifier for the Basic Encoding Rules (BER);

NOTE 4 – According to 8.4.2.7 of ITU-T Rec. X.226 | ISO/IEC 8823-1: "The transfer syntax name shall be present when more than one transfer syntax name was proposed for the presentation context of the presentation data values".

- the **presentation-context-identifier** subcomponent shall be given the same value as the **presentation-context-identifier** of the element of the **presentation-context-definition-list** that specifies the ACSE abstract syntax;
- the **presentation-data-values** subcomponent shall hold the ACSE protocol elements as specified in 7.6.1.2.

7.6.1.2 ACSE protocol elements

The ACSE protocol elements are those defined by the **AARQ-apdu** data type above except for the embedded **TheOsiBind**.

NOTE 1 – The ACSE protocol elements are the relevant components of the **AARQ-apdu** as defined by ITU-T Rec. X.227 | ISO/IEC 8650-1. Only the kernel functional unit of ACSE is used by these Directory Specifications. According to 9.1 of ITU-T Rec. X.227 | ISO/IEC 8650-1, the **sender-acse-requirements**, the **mechanism-name**, the **calling-authentication-value** and the **application-context-name-list** components are not relevant.

The **protocol-version** component shall be omitted or set to **version1,** i.e., bit 0 set. If the component is present, the initiator shall not include any bit after bit 0. If the responder receives a Bind request with this component present and bit 0 is set and one or more other bits are set, those bits shall be ignored. If bit 0 is not set, but some other bit is set, the responding application process shall reply with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **no-common-acse-version**.

The **application-context-name** component shall:

a) for the DAP, be set to **id-ac-directoryAccessAC**;

b) for the DSP, be set to **id-as-directorySystemAC**;

c) for the DISP, be set to either:

- **id-ac-shadowConsumerInitiatedAC**;

- **id-ac-shadowSupplierInitiatedAC**;
- **id-ac-shadowSupplierInitiatedAsynchronousAC**; or
- **id-ac-shadowConsumerInitiatedAsynchronousAC**;

d) for the DOP, be set to **id-ac-directoryOperationalBindingManagementAC**.

If the responder does not support the specified **application-context-name**, it shall reply with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **application-context-name-not-supported**.

The **called-AP-title** component, if present, shall be obtained from:

- information returned in a **ContinuationReference** as the result of a previous Directory operation; or
- locally held information.

If the responder does not recognize the **called-AP-title**, it shall reply with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **called-AP-title-not-recognized**.

The **called-AE-qualifier** component, if present, shall be obtained from:

- information returned in a **ContinuationReference** as the result of a previous Directory operation; or
- locally held information.

If the responder does not recognize the **called-AE-qualifier**, it shall reply with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **called-AE-qualifier-not-recognized**.

The **called-AP-invocation-identifier** component may optionally be supplied if information about its value is retained from a previous application-association. If the responder does not recognize the **called-AP-invocation-identifier**, it shall reply with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **called-AP-invocation-identifier-not-recognized**.

The **called-AE-invocation-identifier** component may optionally be supplied if information about its value is retained from a previous application-association. If the responder does not recognize the **called-AE-invocation-identifier**, it shall reply with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **called-AE-invocation-identifier-not-recognized**.

The **calling-AP-title** component, if supplied, shall be obtained from locally held information. If the responder wants to ensure the identity of the initiator, but does not recognize the **calling-AP-title**, it may reject the application-association with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **calling-AP-title-not-recognized**.

The **calling-AE-qualifier** component, if supplied, shall be obtained from locally held information. If the responder wants to ensure the identity of the initiator, but does not recognize the **calling-AE-qualifier**, it may reject the application-association with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **calling-AE-qualifier-not-recognized**.

The **calling-AP-invocation-identifier** component may optionally be supplied. A receiving system may ignore this value, if present. If the responder wants to ensure the identity of the initiator, but does not recognize the **calling-AP-invocation-identifier**, it may reject the application-association with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **calling-AP-invocation-identifier-not-recognized**.

The **calling-AE-invocation-identifier** component may optionally be supplied. A responding system may ignore this value, if present. If the responder wants to ensure the identity of the initiator, but does not recognize the **calling-AE-invocation-identifier**, it may reject the application-association with an **OsiBindError** (see 7.6.3) with **Associate-source-diagnostic** set to **calling-AE-invocation-identifier-not-recognized**.

The **implementation-information** component may hold implementation-specific information. This information does not affect the application-association establishment procedure.

The **user-information** component has the following subcomponents:

a) **direct-reference**, if present, shall hold the object identifier for the Basic Encoding Rules (BER);

b) **indirect-reference** shall identify the Directory abstract syntax within the **presentation-context-definition-list** as defined in 7.6.1.1 d); and

c) the **single-ASN1-type** shall hold the Bind protocol elements as specified in 7.6.1.3.

NOTE 2 – The **user-information** component corresponds to the **user-information** component of **AARQ-apdu** as defined by ITU-T Rec. X.227 | ISO/IEC 8650-1. This component is a **SEQUENCE OF EXTERNAL**. These Directory Specifications require exactly one occurrence of **EXTERNAL** (see NOTE in 7.6.1).

7.6.1.3 The Bind protocol elements

The **TheOsiBind** shall be the Bind request argument as defined for the Directory protocol in question.

NOTE – The Bind argument starts with the [16] tag as defined in ITU-T Rec. X.880 | ISO/IEC 13712-1.

7.6.2 OSI Bind result

An **OsiBindResult** is returned by the responder if the **OsiBind** is acceptable and the responder decides to engage in the application-association.

```
OsiBindResult {APPLICATION-CONTEXT:Protocols}  ::=  SET {
        mode-selector                       [0]     IMPLICIT SET {mode-value [0] IMPLICIT INTEGER (1) },
        normal-mode-parameters              [2]     IMPLICIT SEQUENCE {
                protocol-version                    [0]     IMPLICIT BIT STRING {version-1(0)} DEFAULT {version-1},
                responding-presentation-selector
                                                    [3]     IMPLICIT Presentation-selector OPTIONAL,
                presentation-context-definition-result-list
                                                    [5]     IMPLICIT SEQUENCE SIZE (2) OF SEQUENCE {
                        result                              [0]     IMPLICIT Result (acceptance),
                        transfer-syntax-name                [1]     IMPLICIT Transfer-syntax-name },
                user-data                                   CHOICE {
                        fully-encoded-data                  [APPLICATION 1] IMPLICIT
                                                                    SEQUENCE SIZE (1) OF SEQUENCE {
                                transfer-syntax-name                Transfer-syntax-name OPTIONAL,
                                presentation-context-identifier     Presentation-context-identifier,
                                presentation-data-values            CHOICE {
                                        single-ASN1-type            [0]     AARE-apdu {{Protocols}} } } } } }

Result ::= INTEGER {
        acceptance              (0),
        user-rejection          (1),
        provider-rejection      (2) }

AARE-apdu {APPLICATION-CONTEXT:Protocols}  ::=  [APPLICATION 1] IMPLICIT SEQUENCE {
        protocol-version                    [0]
                                IMPLICIT BIT STRING {version1(0)} DEFAULT {version1},
        application-context-name            [1]     Application-context-name,
        result                              [2]     Associate-result (accepted),
        result-source-diagnostic            [3]     Associate-source-diagnostic,
        responding-AP-title                 [4]     Name                                    OPTIONAL,
        responding-AE-qualifier             [5]     RelativeDistinguishedName               OPTIONAL,
        responding-AP-invocation-identifier [6]     AP-invocation-identifier                OPTIONAL,
        responding-AE-invocation-identifier [7]     AE-invocation-identifier                OPTIONAL,
        implementation-information          [29]    IMPLICIT Implementation-data            OPTIONAL,
        user-information                    [30]
                IMPLICIT SEQUENCE SIZE(1) OF [UNIVERSAL 8] IMPLICIT SEQUENCE {
                direct-reference                    OBJECT IDENTIFIER OPTIONAL,
                indirect-reference                  Presentation-context-identifier,
                encoding                            CHOICE {
                        single-ASN1-type            [0]     TheOsiBindRes {{Protocols}} } } }
```

NOTE – See Note in 7.6.1.

```
Associate-result ::= INTEGER {
        accepted                (0),
        rejected-permanent      (1),
        rejected-transient      (2) }(0..2, ...)

Associate-source-diagnostic ::= CHOICE {
        acse-service-user       [1]     INTEGER {
                        null                                                    (0),
                        no-reason-given                                         (1),
                        application-context-name-not-supported                  (2),
                        calling-AP-title-not-recognized                         (3),
                        calling-AP-invocation-identifier-not-recognized         (4),
                        calling-AE-qualifier-not-recognized                     (5),
                        calling-AE-invocation-identifier-not-recognized         (6),
                        called-AP-title-not-recognized                          (7),
                        called-AP-invocation-identifier-not-recognized          (8),
                        called-AE-qualifier-not-recognized                      (9),
                        called-AE-invocation-identifier-not-recognized          (10) } (0..10, ...),
```

```
    acse-service-provider      [2]     INTEGER {
                   null                                              (0),
                   no-reason-given                                   (1),
                   no-common-acse-version                            (2) } (0..2, ...) }

TheOsiBindRes {APPLICATION-CONTEXT:Protocols}  ::=
                   [17]    APPLICATION-CONTEXT.&bind-operation.&ResultType ({Protocols})
```

The **OsiBindResult** is carried in the User Data parameter of the Session ACCEPT SPDU (see 8.3.4).

7.6.2.1 Presentation protocol elements

The presentation protocol elements constituting a CPA PPDU are those defined by the **OsiBindResult** data type above except for the embedded **AARE-apdu**.

The **mode-selector** component shall always be set to 1.

The **normal-mode-parameters** component has the following subcomponents:

a) The **protocol-version** subcomponent shall be omitted or set to **version-1**.

b) The **responding-presentation-selector** subcomponent, if supplied, shall be obtained from locally held information.

c) The **presentation-context-definition-result-list** subcomponent shall have two elements in a sequence corresponding to the sequence of elements provided in the **presentation-context-definition-list** of the Bind request, each providing the result of the context negotiation for the corresponding element as follows:

   – The **result** shall be present and set to **acceptance**.

   – The **transfer-syntax-name** shall be present and specify the object identifier for the Basic Encoding Rules (BER).

d) The **user-data** subcomponent has the following elements:

   – The **transfer-syntax-name** subcomponent, if present, shall be the object identifier for the Basic Encoding Rules (BER).

   – The **presentation-context-identifier** subcomponent shall be given the same value as the **presentation-context-identifier** of the element of the **presentation-context-definition-list** of the Bind request that specified the ACSE abstract syntax name.

   – The **presentation-data-values** subcomponent shall hold the ACSE protocol elements as specified in 7.6.2.2.

7.6.2.2 ACSE protocol elements

The **protocol-version** component shall be omitted or set to **version1,** i.e., bit 0 set. If the component is present, the responder shall not include any bit after bit 0.

The **result** component shall be set to **accepted** by the responder.

The **result-source-diagnostic** component shall take the **acse-service-user** choice and take the value **null** or **no-reason-given**.

The **application-context-name** component shall be present and set to the value of the corresponding component of the Bind request.

The **responding-AP-title** component, if supplied, shall be obtained from locally held information.

The **responding-AE-qualifier** component, if supplied, shall be obtained from locally held information.

The **responding-AP-invocation-identifier** component may optionally be supplied. The initiator may ignore this component, if present.

The **responding-AE-invocation-identifier** component may optionally be supplied. The initiator may ignore this component, if present.

The **implementation-information** component may hold implementation-specific information. This information does not affect the application-association establishment procedure.

The **user-information** component has the following subcomponents:

a) **direct-reference**, if present, shall hold the object identifier for the ASN.1 Basic Encoding Rules (BER).

b) **indirect-reference** shall identify the Directory abstract syntax within the **presentation-context-definition-list** as defined in 7.6.1.1 d).

c) The **single-ASN1-type** shall hold the Bind result protocol elements as specified in 7.6.2.3.

7.6.2.3 The Bind result protocol elements

The **TheOsiBindRes** shall be the Bind result type as defined for the Directory protocol in question.

NOTE – The Bind result starts with the [17] tag as defined in ITU-T Rec. X.880 | ISO/IEC 13712-1.

7.6.3 OSI Bind error

```
OsiBindError {APPLICATION-CONTEXT:Protocols}  ::=  CHOICE {
        normal-mode-parameters  SEQUENCE {
                protocol-version                [0]     IMPLICIT BIT STRING {version-1(0)} DEFAULT {version-1},
                responding-presentation-selector
                                                [3]     IMPLICIT Presentation-selector OPTIONAL,
                presentation-context-definition-result-list
                                                [5]     IMPLICIT Result-list OPTIONAL,
                provider-reason                 [10]    IMPLICIT Provider-reason OPTIONAL,
                user-data                               CHOICE {
                        fully-encoded-data              [APPLICATION 1] IMPLICIT SEQUENCE SIZE (1) OF SEQUENCE {
                                transfer-syntax-name            Transfer-syntax-name OPTIONAL,
                                presentation-context-identifier Presentation-context-identifier,
                                presentation-data-values        CHOICE {
                                        single-ASN1-type        [0]     AAREerr-apdu {{Protocols}} } } } OPTIONAL }
}

Result-list ::= SEQUENCE SIZE (2) OF SEQUENCE {
        result                          [0]     IMPLICIT Result,
        transfer-syntax-name            [1]     IMPLICIT Transfer-syntax-name OPTIONAL,
        provider-reason                 [2]     IMPLICIT INTEGER {
                reason-not-specified                            (0),
                abstract-syntax-not-supported                   (1),
                proposed-transfer-syntaxes-not-supported        (2) } OPTIONAL }

Provider-reason ::= INTEGER {
        reason-not-specified                    (0),
        temporary-congestion                    (1),
        local-limit-exceeded                    (2),
        called-presentation-address-unknown     (3),
        protocol-version-not-supported          (4),
        default-context-not-supported           (5),
        user-data-not-readable                  (6),
        no-PSAP-available                       (7) }

AAREerr-apdu {APPLICATION-CONTEXT:Protocols}  ::=  [APPLICATION 1] IMPLICIT SEQUENCE {
        protocol-version                [0]     IMPLICIT BIT STRING {version1(0)}
                                                DEFAULT {version1},
        application-context-name        [1]     Application-context-name,
        result                          [2]     Associate-result (rejected-permanent..rejected-
transient),
        result-source-diagnostic        [3]     Associate-source-diagnostic,
        responding-AP-title             [4]     Name
        OPTIONAL,
        responding-AE-qualifier         [5]     RelativeDistinguishedName       OPTIONAL,
        responding-AP-invocation-identifier [6] AP-invocation-identifier        OPTIONAL,
        responding-AE-invocation-identifier [7] AE-invocation-identifier        OPTIONAL,
        implementation-information      [29]    IMPLICIT Implementation-data    OPTIONAL,
        user-information                [30]
                IMPLICIT SEQUENCE SIZE(1) OF [UNIVERSAL 8] IMPLICIT SEQUENCE {
                direct-reference                OBJECT IDENTIFIER OPTIONAL,
                indirect-reference              Presentation-context-identifier,
                encoding                        CHOICE {
                        single-ASN1-type        [0]     TheOsiBindErr {{Protocols}} } OPTIONAL } }
```

NOTE – See Note in 7.6.1.

```
TheOsiBindErr {APPLICATION-CONTEXT:Protocols}  ::=
                [18]    APPLICATION-CONTEXT.&bind-operation.&Errors.&ParameterType ({Protocols})
```

The **OsiBindError** is carried in the Reason Code field of the Session REFUSE SPDU (see 8.3.5).

7.6.3.1 Presentation protocol elements

The presentation protocol elements constituting a CPR PPDU are those defined by the **OsiBindError** data type above except for the embedded **AAREerr-apdu**.

The **normal-mode-parameters** component has the following subcomponents:

NOTE 1 – The CPR-PPDU is a choice between X.410 mode and normal mode. These Directory Specifications only use the normal mode. The CHOICE statement is retained to ensure bitwise backward compatibility when using other than BER or similar encoding.

a) The **protocol-version** subcomponent shall be as specified in 7.6.2.1.

b) The **responding-presentation-selector** subcomponent, if supplied, shall be as specified in 7.6.2.1.

c) The **presentation-context-definition-result-list** subcomponent shall be specified as follows:

- if the rejection is not related to presentation context negotiation, the **result** element shall be set to **acceptance**, **transfer-syntax-name** shall be present specifying the object identifier for the Basic Encoding Rules (BER), and **provider-reason** element shall be absent;
- if the abstract syntax in question is not supported by any of the proposed transfer syntaxes, the **result** element shall set to **provider-rejection** and the **provider-reason** element shall be present with the appropriate value; or
- if the abstract syntax in question is not supported at all and the previous bullet does not apply, the **result** element shall set to **user-rejection** and the **provider-reason** element shall be present with the appropriate value.

d) The **provider-reason** subcomponent shall be present if the application-association is rejected due to problems detected within the presentation protocol elements of the Bind request. Otherwise it shall be absent.

NOTE 2 – Clause 6.2.4.9 of ITU-T Rec. X.226 | ISO/IEC 8823-1 states for **provider-reason**: "If present, this shall indicate that the rejection is by the responding presentation-service-provider; if absent this shall indicate that the rejection is by the responding PS-user."

e) The **user-data** subcomponent shall be absent if **provider-reason** subcomponent is present. Otherwise, it shall be present with the following elements:

- The **transfer-syntax-name** subcomponent, if present, shall be the object identifier for the ASN.1 Basic Encoding Rules (BER).
- The **presentation-context-identifier** subcomponent shall be given the same value as the **presentation-context-identifier** of the element of the **presentation-context-definition-list** of the Bind request that specified the ACSE abstract syntax name.
- The **presentation-data-values** subcomponent shall hold the ACSE protocol elements as specified in 7.6.3.2.

7.6.3.2 ACSE protocol elements

The **protocol-version** component shall be as specified in 7.6.2.2.

The **application-context-name** component shall be present and set to the value of the corresponding component of the Bind request.

The **result** component shall be set to **rejected-permanent** or **rejected-transient** based on local considerations.

NOTE – According to 11.1.1 of ITU-T Rec. X.881 | ISO/IEC 13712-2, a Bind Error is carried in the A-ASSOCIATE response/confirm with the Result parameter value of the A-ASSOCIATE service primitives set to "rejected (permanent)" or "rejected (transient)", and the error value of the Bind operation is mapped on the User Information parameter of these service primitives. At the protocol level, that translates to the **result** component being set to either **rejected-permanent** or **rejected-transient**. Most of the Bind errors reflect a permanent condition. However, the **serviceError** with problem **unavailable** might be considered as being transient.

The **result-source-diagnostic** component shall take values as follows depending on the condition:

a) if the rejection is within a directory protocol, the **acse-service-user** choice shall be taken with the value **null** or **no-reason-given**; or

b) if the rejection is ACSE related or due to errors in specified application process title, application-entity title or application-context the **acse-service-user** choice shall be taken with the appropriate value.

The value of the **responding-AP-title** component, if present, shall be obtained from locally held information.

The **responding-AE-qualifier** component, if present, shall be obtained from locally held information.

The **responding-AP-invocation-identifier** component, if present, may be ignored or retained for a future association with that DSA.

The **responding-AE-invocation-identifier** component, if present, may be ignored or retained for a future association with that DSA.

The **implementation-information** component may hold implementation-specific information.

The **user-information** component has the following subcomponents:

a) **direct-reference**, if present, shall hold the object identifier for the ASN.1 Basic Encoding Rules (BER);

b) **indirect-reference** shall identify the Directory abstract syntax within the **presentation-context-definition-list** as defined in 7.6.1.1 d);

c) the **single-ASN1-type** shall hold the Bind error protocol elements as specified in 7.6.3.3.

7.6.3.3 The Bind error protocol elements

The **TheOsiBindErr** shall be the Bind error type as relevant for the type of error.

NOTE – The Bind error starts with the [18] tag as defined in ITU-T Rec. X.880 | ISO/IEC 13712-1.

7.6.4 OSI unbind request

```
OsiUnbind  ::=  CHOICE {
        fully-encoded-data  [APPLICATION 1] IMPLICIT SEQUENCE SIZE (1) OF SEQUENCE {
                presentation-context-identifier  Presentation-context-identifier,
                presentation-data-values   CHOICE {
                        single-ASN1-type    [0]      TheOsiUnbind } } }

TheOsiUnbind ::= [APPLICATION 2] IMPLICIT SEQUENCE {
        reason          [0]      IMPLICIT Release-request-reason OPTIONAL }

Release-request-reason  ::=  INTEGER {
        normal                   (0) }
```

The **OsiUnbind** is carried in the User Data of the Session FINISH SPDU (see 8.3.6).

Only the initiator of an application-association may invoke an unbind request.

NOTE 1 – Clause 8.5 of ITU-T Rec. X.880 | ISO/IEC 13712-1 defines a **CONNECTION-PACKAGE** information object class, where the field **&responderCanUnbind** specifies whether the responder may issue an unbind or not. It defaults to **FALSE**. The fourth edition of this Directory Specification did not add the **&responderCanUnbind** for any of the protocols. The IDM protocol allows the responder to issue an unbind, except for the DAP protocol (see 9.2.2).

NOTE 2 – Clause 8.5 of ITU-T Rec. X.880 | ISO/IEC 13712-1 also defines an **&unbindCanFail** field of the **CONNECTION-PACKAGE** information object class with default equal **FALSE**. The fourth edition of this Directory Specification did not add the **&unbindCanFail** for any of the protocols.

7.6.4.1 Presentation protocol elements

The presentation protocol elements are only those defined by the **User-data** data type defined by ITU-T Rec. X.226 | ISO/IEC 8823-1.

The **presentation-context-identifier** component shall be given the same value as the **presentation-context-identifier** of the element of the **presentation-context-definition-list** of the Bind request that specified the ACSE abstract syntax.

The **presentation-data-values** component shall hold the ACSE protocol elements as specified in 7.6.4.2.

7.6.4.2 ACSE protocol elements

The **reason** component shall be set to **normal** or be absent. The absence of **reason** component indicates normal release.

NOTE 1 – According to 11.1.2 of ITU-T Rec. X.881 | ISO/IEC 13712-2, the **reason** shall always be set to **normal**.

NOTE 2 – According to ITU-T Rec. X.226 | ISO/IEC 8823-1, there are no presentation protocol elements for normal release of a connection. Normal release is accomplished by the normal release of the underlying session connection.

7.6.5 OSI unbind result

```
OsiUnbindResult  ::=  CHOICE {
        fully-encoded-data  [APPLICATION 1] IMPLICIT SEQUENCE SIZE (1) OF SEQUENCE {
                presentation-context-identifier    Presentation-context-identifier,
                presentation-data-values           CHOICE {
                        single-ASN1-type           [0]      TheOsiUnbindRes } } }
```

```
TheOsiUnbindRes  ::= [APPLICATION 3] IMPLICIT SEQUENCE {
        reason          [0]     IMPLICIT Release-response-reason OPTIONAL }

Release-response-reason  ::=  INTEGER {
        normal                  (0) }
```

NOTE – Pre-edition 5 specifications specify that the Result parameter of the A-RELEASE service as defined by ITU-T Rec. X.217 | ISO/IEC 8649 shall be set to 'affirmative'.

The **OsiUnbindResult** is carried in the User Data of the Session DISCONNECT SPDU (see 8.3.7).

7.6.5.1 Presentation protocol elements

The presentation protocol elements are only those defined by the **User-data** data type defined by ITU-T Rec. X.226 | ISO/IEC 8823-1.

The **presentation-context-identifier** component shall be given the same value as the **presentation-context-identifier** of the element of the **presentation-context-definition-list** of the Bind request that specified the ACSE abstract syntax.

The **presentation-data-values** component shall hold the ACSE release request.

7.6.5.2 ACSE protocol elements

The absence of reason component indicates normal release.

7.6.6 OSI operations

```
OsiOperation {OPERATION:Operations}  ::=  CHOICE {
        fully-encoded-data  [APPLICATION 1] IMPLICIT SEQUENCE SIZE (1) OF SEQUENCE {
                presentation-context-identifier    Presentation-context-identifier,
                presentation-data-values           CHOICE {
                        single-ASN1-type           [0]     CHOICE {
                                request                            OsiReq {{Operations}},
                                result                             OsiRes {{Operations}},
                                error                              OsiErr {{Operations}},
                                reject                             OsiRej } } } }
```

The **OsiOperation** is carried in the User Information Field of the Session DATA TRANSFER SPDU (see 8.3.10).

7.6.6.1 Presentation protocol elements

The **presentation-context-identifier** component shall be given the same value as the **presentation-context-identifier** of the element of the **presentation-context-definition-list** of the Bind request that specified the Directory abstract syntax name for the Directory protocol in question.

The **presentation-data-values** component shall hold the Directory request, result, error or reject.

7.6.6.2 OSI Request

```
OsiReq {OPERATION:Operations}  ::=  [1]  IMPLICIT SEQUENCE {
        invokeId                invokeId,
        opcode                  OPERATION.&operationCode ({Operations}),
        argument                OPERATION.&ArgumentType ({Operations} {@opcode}) }
```

NOTE 1 – The Request starts with the [1] tag as defined in ITU-T Rec. X.880 | ISO/IEC 13712-1.

The **invokeId** component identifies the particular invocation. It shall not be a value that has been used for a previous request that requires a response (result and/or error) and which is still in progress. If this is violated, the receiver shall issue an **OsiReject** with an **InvokeProblem** set to **duplicateInvocation**. If the request does not necessarily result in a response, it is a local option as to the time passed before reusing an **invokeId**.

NOTE 2 – All currently defined Directory operations require a response.

The **opcode** component shall hold the operation code for the particular type of operation. If an unknown operation code is specified, the receiver shall issue an **OSIReject** with an **InvokeProblem** set to **unrecognizedOperation**.

The **argument** component shall hold the argument formed in accordance with the **&ArgumentType** field of the operation type identified by the **opcode** component for the protocol in question.

7.6.6.3 OSI result

```
OsiRes { OPERATION:Operations}  ::=  [2] IMPLICIT SEQUENCE {
        invokeId        InvokeId,
        result          SEQUENCE {
```

```
        opcode      OPERATION.&operationCode ({Operations}),
        result      OPERATION.&ResultType ({Operations} {@.opcode}) } }
```

NOTE – The Result starts with the [2] tag as defined in ITU-T Rec. X.880 | ISO/IEC 13712-1.

The **invokeID** component shall be equal to the one specified in the corresponding request.

The **opcode** component shall be equal to the one specified in the corresponding request.

The **result** component shall hold the result formed in accordance with the **&ResultType** field of the operation type identified by the **opcode** component for the protocol in question.

7.6.6.4 OSI error

```
OsiErr {OPERATION:Operations}  ::=  [3]  IMPLICIT  SEQUENCE {
        invokeID            InvokeId,
        errcode             OPERATION.&Errors.&errorCode ({Operations}),
        error               OPERATION.&Errors.&ParameterType ({Operations} {@.errcode}) }
```

NOTE – The Error starts with the [3] tag as defined in ITU-T Rec. X.880 | ISO/IEC 13712-1.

The **invokeID** component shall be equal to the one specified in the corresponding **OsiRequest**.

The **errcode** component shall be set to the code for one of the errors identified by the **ERRORS** field of the **OPERATION** information object identified by the **opcode** of the corresponding **OsiRequest**.

The **error** component shall hold the parameters as identified by the **errcode** component.

7.6.6.5 OSI reject

The type **OsiRej** is used for reporting erroneous use of the other Directory PDUs. It is specified as follows:

```
OsiRej  ::=  [4] IMPLICIT SEQUENCE {
        invokeId        InvokeId,
        problem         CHOICE {
                general             [0]     GeneralProblem,
                invoke              [1]     InvokeProblem,
                returnResult        [2]     ReturnResultProblem,
                returnError         [3]     ReturnErrorProblem } }
```

NOTE – The Reject starts with the [4] tag as defined in ITU-T Rec. X.880 | ISO/IEC 13712.

The **invokeId** component shall be equal to the one specified in the PDU to be rejected, except if the **invokeId** cannot be determined, it shall take the **absent** choice instead (see 6.2).

The **problem** component shall hold the Reject problem as defined by 7.6.6.6.

7.6.6.6 Reject problems

```
GeneralProblem ::= INTEGER      {
        unrecognizedPDU         (0),
        mistypedPDU             (1),
        badlyStructuredPDU      (2) }
```

A **GeneralProblem** is a fundamental problem with the form or structure of a Directory PDU. The possibilities are specified as follows:

a) **unrecognizedPDU** – The leading tag of the PDU indicates that it is not an **OsiRequest**, an **OsiResult**, an **OsiError** or an **OsiReject**;

b) **mistypedPDU** – The structure of the PDU does not conform to the appropriate definition; or

c) **badlyStructuredPDU** – The structure of the PDU cannot be determined based on the expected abstract syntax.

```
InvokeProblem ::= INTEGER {
        duplicateInvocation     (0),
        unrecognizedOperation   (1),
        mistypedArgument        (2),
        resourceLimitation      (3),
        releaseInProgress       (4) }
```

An **InvokeProblem** indicates that some component of an **OsiRequest** was erroneous. The possibilities are specified as follows:

a) **duplicateInvocation** – See 7.6.6.2;

b) **unrecognizedOperation** – The operation code is not one of those defined for the Directory protocol in question;

c) **mistypedArgument** – The argument is not formed according to the **&ArgumentType** field of the operation identified by the **opcode** component;

d) **resourceLimitation** – The intended performer is not willing to perform the operation due to a resource limitation; or

e) **releaseInProgress** – The intended performer is not willing to perform the operation because it is about to release the application-association.

```
ReturnResultProblem ::= INTEGER {
    unrecognizedInvocation      (0),
    resultResponseUnexpected    (1),
    mistypedResult              (2) }
```

A **ReturnResultProblem** indicates that some component of an **OsiResult** was erroneous. The possibilities are specified as follows:

a) **unrecognizedInvocation** – The **InvokeId** was not one that identifies an outstanding request;

b) **resultResponseUnexpected** – A result was received for an operation for which no result is defined;

NOTE 1 – All the currently defined Directory operation types specify a result.

c) **mistypedResult** – The result is not formed according to the **&ResultType** field of the operation identified by the **opcode** component.

```
ReturnErrorProblem ::= INTEGER {
    unrecognizedInvocation      (0),
    errorResponseUnexpected     (1),
    unrecognizedError           (2),
    unexpectedError             (3),
    mistypedParameter           (4) }
```

A **ReturnErrorProblem** indicates that some component of an **OsiError** was erroneous. The possibilities are specified as follows:

a) **unrecognizedInvocation** – The **InvokeId** was not one that identifies an outstanding request;

b) **errorResponseUnexpected** – An error was received for an operation for which no error is defined;

NOTE 2 – All the currently defined Directory operation types specify one or more errors.

c) **unrecognizedError** – An error was received that was not one of those specified by these Directory Specifications;

d) **unexpectedError** – An error was received that was not one of those identified by the **&Errors** field of the operation identified by the **opcode** component; or

e) **mistypedParameter** – The parameter of the error result is not formed according to the **&ParameterType** field of the error identified by the **errcode** component.

7.6.7 Presentation abort

Abort can be caused by an application problem (**ARU-PPDU**) or a Presentation Layer problem (**ARP-PPDU**).

```
PresentationAbort ::= CHOICE {
    aru-ppdu    ARU-PPDU,
    arp-ppdu    ARP-PPDU }
```

7.6.7.1 OSI application abort

```
ARU-PPDU  ::=  CHOICE {
    normal-mode-parameters          [0]     IMPLICIT SEQUENCE {
        presentation-context-identifier-list        [0]     IMPLICIT Presentation-context-identifier-list,
        user-data                                           CHOICE {
            fully-encoded-data [APPLICATION 1]      IMPLICIT SEQUENCE SIZE (1) OF SEQUENCE {
                presentation-context-identifier             Presentation-context-identifier,
                presentation-data-values                    CHOICE {
                    single-ASN1-type                        [0]     ABRT-apdu } } } } }

Presentation-context-identifier-list ::=
    SEQUENCE SIZE (1) OF SEQUENCE {
```

```
        presentation-context-identifier    Presentation-context-identifier,
        transfer-syntax-name               Transfer-syntax-name}

ABRT-apdu  ::=  [APPLICATION 4]  IMPLICIT SEQUENCE {
        abort-source ABRT-source }

ABRT-source  ::=  INTEGER {
        acse-service-user          (0),
        acse-service-provider      (1) }
```

The **ABRT-PPDU** is used if the abort is caused by problems at the Directory Protocol level or within ACSE rather than within the presentation protocol elements.

The **ABRT-PPDU** is carried in the User Data of the Session ABORT SPDU and the Transport Disconnect field bit 2 shall be set and bit 3 shall be reset (see 8.3.8).

The **ABRT-PPDU** may cause loss of information in transfer.

The receipt of an **ABRT-PPDU** on an association supporting the DAP terminates all request processing. Except for certain conditions described in ITU-T Rec. X.518 | ISO/IEC 9594-4, this is also true for the DSP. It is a Directory user responsibility to confirm if requested modifications to the DIB occurred.

The receipt of an **ABRT-PPDU** on an association supporting the DISP is described in ITU-T Rec. X.525 | ISO/IEC 9594-9.

The receipt of an **ABRT-PPDU** on an association supporting the DOP is described in ITU-T Rec. X.518 | ISO/IEC 9594-4.

7.6.7.1.1 Presentation protocol elements

The **normal-mode-parameters** component has the following subcomponents:

a) The **presentation-context-identifier-list** subcomponent indicates what transfer syntax is used for the user data. Only ACSE information is included in the user data. It shall have one element being a sequence type with:
   - the **presentation-context-identifier** subcomponent shall be given the same value as the **presentation-context-identifier** of the element of the **presentation-context-definition-list** of the Bind request that specified the ACSE abstract syntax name;
   - the **transfer-syntax-name** shall be the object identifier for the Basic Encoding Rules (BER).

b) The **user-data** subcomponent has the following elements:
   - the **presentation-context-identifier** subcomponent shall be given the same value as the **presentation-context-identifier** of the element of the **presentation-context-definition-list** that specifies the ACSE abstract syntax;
   - the **presentation-data-values** subcomponent shall hold the ACSE protocol elements as defined in 7.6.7.1.2.

7.6.7.1.2 ACSE protocol elements

If the abort is caused at the Directory Protocol level, the **ABRT-source** shall be set to **acse-service-user**. If the abort is caused at the ACSE level, the **ABRT-source** shall be set to **acse-service-provider**.

NOTE – **ABRT-apdu** as defined by ITU-T Rec. X.227 | ISO/IEC 8650-1 have two additional parameters. The **abort-diagnostics** shall not be present if only the Kernel is used, which implies that the abort is only used for signalling a protocol error. The **user-information** is not used, as these Directory Specifications provide no abort information.

7.6.7.2 OSI Presentation abort

```
ARP-PPDU  ::=  SEQUENCE {
        provider-reason     [0]     IMPLICIT Abort-reason OPTIONAL,
        event-identifier    [1]     IMPLICIT Event-identifier OPTIONAL }

Abort-reason  ::=  INTEGER {
        reason-not-specified                        (0),
        unrecognized-ppdu                           (1),
        unexpected-ppdu                             (2),
        unexpected-session-service-primitive        (3),
        unrecognized-ppdu-parameter                 (4),
        unexpected-ppdu-parameter                   (5),
        invalid-ppdu-parameter-value                (6) }
```

```
Event-identifier ::= INTEGER {
        cp-PPDU                 (0),
        cpa-PPDU                (1),
        cpr-PPDU                (2),
        aru-PPDU                (3),
        arp-PPDU                (4),
        td-PPDU                 (7),
        s-release-indication    (14),
        s-release-confirm       (15) }
```

The **ARP-PDU** is used if the abort is caused by problems within the presentation protocol level.

The **ARP-PDU** is carried in the User Data of the Session ABORT SPDU and the Transport Disconnect field bit 2 shall be set and bit 3 shall be reset (see 8.3.8).

The **ARP-PDU** may cause loss of information in transfer.

The receipt of an **ARP-PDU** shall be treated as specified for **ARU-PDU** in 7.6.7.1.

The **provider-reason** component may take one of the following values:

a) **reason-not-specified**;

b) **unrecognized-ppdu** indicating that an unknown PPDU was received;

NOTE – This may be PPDU as defined by ITU-T Rec. X.226 | ISO/IEC 8823-1, but not used by this Directory Specification. Some implementation may signal that as an **unexpected-ppdu**. However, it is no requirement that an implementation shall recognize PPDUs not defined by this Directory Specification.

c) **unexpected-ppdu** indicating that a PPDU identified by the **Event-identifier** was received out of sequence;

d) **unexpected-session-service-primitive** as indicated by the **Event-identifier**;

e) **unrecognized-ppdu-parameter** – Should not be used according to the rules of extensibility (see Note 1 in 7.5);

f) **unexpected-ppdu-parameter** indicating that even a parameter was recognized, but it was not expected at this particular time or place in a PPDU as identified by the **Event-identifier**;

g) **invalid-ppdu-parameter-value** indicating that a parameter had an invalid value in a PPDU as identified by the **Event-identifier**.

The **Event-identifier** shall be present when referenced above. Otherwise, it shall be absent.

a) **s-release-indication** indicates the application-association unexpectedly has been terminated by the Session Layer function of the peer system;

b) **s-release-confirm** indicates the application-association unexpectedly has been terminated by the local Session Layer function.

8 Directory protocol mapping onto OSI services

8.1 Abstract syntaxes and transfer syntaxes

As part of an application-association the protocol elements of the supporting protocol have to be agreed between the two parties. This is done by signalling the relevant abstract syntaxes as part of the Bind operation. An abstract syntax is assigned an object identifier, which is then carried in the Bind.

The directory protocols each requires two abstract syntaxes, one reflecting the protocol element of the ACSE protocol and one reflecting the actual Directory protocol (Directory abstract syntax).

NOTE – The protocol elements of ACSE are part of the Directory Specifications for edition 5 and subsequent editions. However, for backward compatibility, it is still necessary to signal two abstract syntaxes in the Bind operation.

The object identifiers for Directory abstract syntaxes are:

```
id-as-directoryAccessAS                     OBJECT IDENTIFIER    ::=    {id-as 1}
id-as-directorySystemAS                     OBJECT IDENTIFIER    ::=    {id-as 2}
id-as-directoryShadowAS                     OBJECT IDENTIFIER    ::=    {id-as 3}
id-as-directoryOperationalBindingManagementAS  OBJECT IDENTIFIER ::=    {id-as 4}
```

The ACSE abstract syntax is identified by:

```
id-acseAS                                   OBJECT IDENTIFIER   ::=
        { joint-iso-itu-t association-control(2) abstract-syntax(1) apdus(0) version(1) }
```

The ASN.1 encoding rules for an abstract syntax are signalled by an object identifier.

The object identifiers for the ASN.1 encoding rules are defined in ITU-T Rec. X.690 | ISO/IEC 8825-1. For convenience, the object identifier for BER is supplied here:

```
{ joint-iso-itu-t asn1(1) basic-encoding(1) }
```

8.2 Application-contexts

An *application-context* is a set of common rules shared by two application-entities in order to support an application-association. An application-context is identified by an *application-context-name* in the form of an object identifier. The application-context-name is signalled by the Bind operation.

An application-context is defined using the following ASN.1 information object class:

```
APPLICATION-CONTEXT  ::=  CLASS {
        &bind-operation                 OPERATION,
        &Operations                     OPERATION,
        &applicationContextName         OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        BIND-OPERATION                  &bind-operation
        OPERATIONS                      &Operations
        APPLICATION CONTEXT NAME        &applicationContextName }
```

The **&bind-operation** field is used for specifying the type of Bind operation corresponding to the specified application-context name.

The **&Operations** field is used for listing all the Directory operations relevant for the application-context.

The **&applicationContextName** field is used for supplying the object identifier for the application-context.

NOTE – This ASN.1 information object class is an abbreviated version of the one defined by ITU-T Rec. X.881 | ISO/IEC 13712-2 and is provided here as certain specifications use the ASN.1 information object reference rather than the assigned object identifier.

8.2.1 Applications-context for DAP

```
directoryAccessAC  APPLICATION-CONTEXT  ::=  {
        BIND-OPERATION                  directoryBind
        OPERATIONS                      { read | compare | abandon | list | search
                                        | addEntry | removeEntry | modifyEntry | modifyDN }
        APPLICATION CONTEXT NAME        id-ac-directoryAccessAC }
```

The **directoryAccessAC** application-context is the one defining the DAP. Support of this application-context requires support of the **id-acseAS** and the **id-as-directoryAccessAS** abstract syntaxes.

For a DUA it implies support for at least one DAP operation type, beyond possibly the Abandon operation type. For a DSA it implies support of all the DAP operations.

8.2.2 Applications-context for DSP

```
directorySystemAC  APPLICATION-CONTEXT  ::=  {
        BIND-OPERATION                  dSABind
        OPERATIONS                      { chainedRead | chainedCompare | chainedAbandon
                                        | chainedList | chainedSearch
                                        | chainedAddEntry | chainedRemoveEntry
                                        | chainedModifyEntry | chainedModifyDN }
        APPLICATION CONTEXT NAME  id-ac-directorySystemAC }
```

The **directorySystemAC** application-context is the one defining the DSP. Support of this application-context requires support of the **id-acseAS** and the **id-as-directorySystemAS** abstract syntaxes.

It implies support of all the DSP operations as listed above.

8.2.3 Applications-contexts for DISP

```
shadowSupplierInitiatedAC  APPLICATION-CONTEXT  ::= {
        BIND-OPERATION                  dSAShadowBind
        OPERATIONS                      { updateShadow
                                        | coordinateShadowUpdate }
        APPLICATION CONTEXT NAME        id-ac-shadowSupplierInitiatedAC }
```

The **shadowSupplierInitiatedAC** application-context is a DISP application-context for an application-association where shadow updating is initiated by the supplier and the operation mode is synchronous.

NOTE – The terms "consumer" and "supplier" are used to designate two roles. These roles correspond to the two terms "shadow consumer" and "shadow supplier", respectively, used in ITU-T Rec. X.525 | ISO/IEC 9594-9.

```
shadowConsumerInitiatedAC  APPLICATION-CONTEXT  ::= {
        BIND-OPERATION                  dSAShadowBind
        OPERATIONS                      { requestShadowUpdate
                                        | updateShadow }
        APPLICATION CONTEXT NAME        id-ac-shadowConsumerInitiatedAC }
```

The **shadowConsumerInitiatedAC** application-context is a DISP application-context for an application-association where shadow updating is initiated by the consumer and the operation mode is synchronous.

```
shadowSupplierInitiatedAsynchronousAC  APPLICATION-CONTEXT  ::= {
        BIND-OPERATION                  dSAShadowBind
        OPERATIONS                      { updateShadow
                                        | coordinateShadowUpdate }
        APPLICATION CONTEXT NAME        id-ac-shadowSupplierInitiatedAsynchronousAC }
```

The **shadowSupplierInitiatedAsynchronousAC** application-context is a DISP application-context for an application-association where shadow updating is initiated by the supplier and the operation mode is asynchronous.

```
shadowConsumerInitiatedAsynchronousAC  APPLICATION-CONTEXT  ::= {
        BIND-OPERATION                  dSAShadowBind
        OPERATIONS                      { requestShadowUpdate
                                        | updateShadow }
        APPLICATION CONTEXT NAME        id-ac-shadowConsumerInitiatedAsynchronousAC }
```

The **shadowConsumerInitiatedAsynchronousAC** application-context is a DISP application-context for an application-association where shadow updating is initiated by the consumer and the operation mode is asynchronous.

8.2.4 Applications-context for DOP

```
directoryOperationalBindingManagementAC  APPLICATION-CONTEXT  ::= {
        BIND-OPERATION                  dSAOperationalBindingManagementBind
        OPERATIONS                      { establishOperationalBinding
                                        | modifyOperationalBinding
                                        | terminateOperationalBinding}
        APPLICATION CONTEXT NAME        id-ac-directoryOperationalBindingManagementAC }
```

The **directoryOperationalBindingManagementAC** application-context is the one defining the DOP.

8.3 Session Layer specification

8.3.1 Structure of session-protocol-data-unit (SPDU)

A *session-protocol-data-unit* (SPDU) consists of an *SPDU identifier* (SI), zero or more parameters each identified by a *parameter identifier* (PI) and possibly, a *parameter value* (PV) field. Related parameters can be grouped and then be identified by a *parameter group identifier* (PGI).

The first part of an SPDU is the SPDU identifier (SI) field. It consists of a single octet. The value is a binary number.

A length indicator (LI) is used to indicate the length of an SPDU, the length of a parameter or the length of a group of parameters. LI fields indicating lengths within the range 0-254 shall comprise one octet. LI fields indicating lengths within the range 255-65.535 shall comprise three octets. The first octet shall then be coded 1111 1111 and the second and third octets shall contain the length of the associated parameter field with the high order bits in the first of these two octets.

The value of the LI field does not include either itself or any subsequent User Information field.

NOTE – Of the SPDU used by this Directory Specification, only the DATA TRANSFER SPDU has a User Information field.

The bits of an octet are numbered from 1 to 8, where bit 1 is the least significant bit.

Figure 1 illustrates the case where a SPDU has no parameters. The ABORT ACCEPT SPDU is an example. The LI field then has the value 0.

Figure 1 – SPDU without parameters

Figure 2 illustrates the case where a SPDU has two separate parameters, each identified by a PI. The first LI field indicates the length of the SPDU, excluding the SI field and the LI field itself. The two other LI fields indicate the length of the parameters.

As an example: If the first PV is 3 octets and the second PV is 4 octets, then the first LI field has the value 11, the second LI field has the value 3 and the third LI field has the value 4.

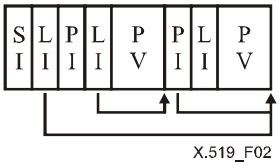

Figure 2 – SPDU with parameters – Not grouped

Figure 3 illustrates the case where a SPDU has two parameters grouped together, each identified by a PI. The group is identified by a PGI field. The first LI field indicates the length of the SPDU, excluding the SI field and the LI field itself. The next LI field indicates the length of the group excluding the PGI field and the LI field itself. The two other LI fields indicate the length of the parameters.

As an example: If the first PV is 5 octets and the second PV is 3 octets, then the first LI field has the value 14, the second LI field has the value 12, the third LI field has the value 5 and the fourth LI field has the value 3.

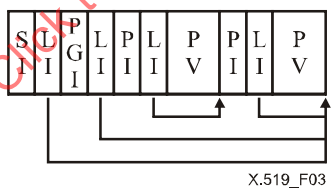

Figure 3 – SPDU with parameters – Grouped

8.3.2 TSDU size and segmenting

The maximum transport-service-data-unit (TSDU) size expresses the maximum number of octets to be presented to the Transport Layer for transmission. The maximum TSDU size is negotiated during application-association establishment for each direction of transmission (see 8.3.3 and 8.3.4). If a Directory PDU including session protocol overhead exceeds that maximum, it is necessary to segment the Directory PDU into multiple SPDUs.

Each application process proposes a maximum TSDU size that the initiator is permitted to send. The lesser of the two numbers is used. A zero value is interpreted to mean unlimited TSDU size. If either application process proposes zero, the initiator shall not send segmented data on the application-association.

Each application process also proposes a maximum TSDU size that the responder is permitted to send. The lesser of the two numbers is used. A zero value is interpreted to mean unlimited TSDU size. If either application process proposes zero, the responder shall not send segmented data on the application-association.

8.3.3 Session CONNECT SPDU

Table 1 – Parameters of the CONNECT SPDU

PGI	M/O	Code	PI	M/O	Code	Length
Connection Identifier	O	1	Calling SS-user Reference	O	10	64 octets maximum
			Common reference	O	11	64 octets maximum
			Additional Reference Information	O	12	4 octets maximum
Connect/Accept Item	M	5	Protocol options	M	19	1 octet
			TSDU Maximum Size	O	21	4 octets
			Version Number	M	22	1 octet
			Session User Requirements	M	20	2 octets
			Calling Session Selector	O	51	16 octets maximum
			Called Session Selector	O	52	16 octets maximum
User data	M	193				512 octets maximum
Extended User Data	M	194				10 240 octets maximum

The SI field shall be given the value 13 ('0D'H).

The Connection Identifier is an optional parameter group that is filled with locally generated data that allows identification of this session connection. It may have the following optional parameters:

a) The Calling SS-user Reference, i.e., a reference selected by the initiator;

NOTE 1 – An SS-user or session-service-user is according to ITU-T Rec. X.200 | ISO/IEC 7498-1 a Presentation Layer function using the Session Service.

b) Common Reference; and

c) Additional Reference Information

Connect/Accept Item is a mandatory parameter group with the following parameters:

a) Protocol Options – Extended concatenation, as defined by ITU-T Rec. X.225 | ISO/IEC 8327-1, is not relevant for this Directory Specification. This field shall be absent or the value shall be set to '00'H (default value). However, an implementation should accept the value '01'H.

b) The TSDU Maximum Size PV field shall be present if a TSDU Maximum Size is proposed. If the TSDU Maximum Size PV field is present:

   i) the first two octets of the PV field shall contain the proposed maximum TSDU size, expressed in octets, in the direction from the initiator to the responder, encoded as a binary number, where the first of the two octets is the high order part of the number;

   ii) the second two octets of the PV field shall contain the proposed maximum TSDU size, expressed in octets, in the direction from the responder to the initiator, encoded as a binary number, where the first of the two octets is the high order part of the number.

   If this parameter is absent, the TSDU Maximum Size is not limited. If either pair of octets has the value zero, the TSDU size is not limited in the direction of transfer associated with that pair of octets.

c) Version Number – This field shall be given the value '02'H.

Session User Requirements field shall be set to '0002'H.

NOTE 2 – Only the session duplex functional unit is used by the Directory.

Calling Session Selector field shall hold the value of the initiator's session-selector, if one is assigned, and shall have a value obtained from locally held information. If the initiator does not have a session-selector within its presentation-address, this field shall be absent.

Called Session Selector field shall be present if it is known to be part of the addressing for the receiving system. Otherwise, it shall be absent. If present, the value shall be obtained from:

– information returned in a **ContinuationReference** as the result of a previous Directory operation; or

– locally held information.

Both the User Data parameter and the Extended User Data parameter shall be supported, but only one of these two parameters may be used for an instance of communication. If the user data to be included is 512 octets or less, the User

Data Parameter shall be used. If user data is larger than 512 octets, the Extended User Data parameter shall be used and the User Data parameter shall not be used.

The OSI Bind request is carried as user data of the session CONNECT SPDU (see 7.6.1). The OSI Bind Request shall not exceed 10240 octets.

8.3.4 Session ACCEPT SPDU

Table 2 – Parameters of the ACCEPT SPDU

PGI	M/O	Code	PI	M/O	Code	Length
Connection Identifier	O	1	Called SS-user Reference	O	9	64 octets maximum
			Common Reference	O	11	64 octets maximum
			Additional Reference Information	O	12	4 octets maximum
Connect/Accept Item	O	5	Protocol options	M	19	1 octet
			TSDU maximum size	O	21	4 octets
			Version number	M	22	1 octet
			Session User Requirements	M	20	2 octets
			Calling Session Selector	O	51	16 octets maximum
			Responding Session Selector	O	52	16 octets maximum
User data	M	193				

The SI field shall be given the value 14 ('0E'H).

The Connection Identifier is an optional parameter group that is filled with locally generated data that allows identification of this session connection. It may have the following optional parameters:

a) the Called SS-user Reference;

b) Common Reference; and

c) Additional Reference Information.

Connect/Accept Item is a mandatory parameter group with the following parameters:

a) Protocol Options – This field shall be absent or the value shall be set to '00'H (default value). However, an implementation should accept the value '01'H.

b) TSDU Maximum Size – This field shall be present if a TSDU Maximum Size is proposed by the responder. The encoding and default for this field is as for the CONNECT SPDU (see 8.3.3).

c) Version Number – This field shall be given the value '02'H.

Session User Requirements field shall be set to '0002'H.

Calling Session Selector field shall be present if the corresponding field was present in the CONNECT SPDU and shall then hold the same value as that field. Otherwise, this field shall be absent.

Responding Session Selector field, if supplied, shall have a value obtained from locally held information.

The User Data parameter shall be supported. It shall be used to carry the **OsiBindResult** (see 7.6.2).

The length of the ACCEPT SPDU shall not exceed 65 539 octets.

8.3.5 Session REFUSE SPDU

The session REFUSE SPDU is used by the responder to refuse an application-association.

Table 3 – Parameters of the REFUSE SPDU

PGI	M/O	Code	PI	M/O	Code	Length
Connection Identifier	O	1	Called SS-user Reference	O	9	64 octets maximum
			Common Reference	O	11	64 octets maximum
			Additional Reference Information	O	12	4 octets maximum
			Transport Disconnect	O	17	1 octet
			Session User Requirements	O	20	2 octets
			Version Number	O	22	1 octet
			Reason Code	M	50	See below.

The SI field shall be given the value 12 ('0C'H).

The Connection Identifier is an optional parameter group that is filled with locally generated data that allows identification of this session connection. It may have the following optional parameters:

a) the Called SS-user Reference;

b) Common Reference; and

c) Additional Reference Information

The Transport Disconnect field indicates whether the underlying transport-connection shall be released or kept. The encoding for this field shall be

a) bit 1 = 0: Transport connection is kept;

b) bit 1 = 1: Transport connection is released.

Bits 2-8 are reserved.

If this field is absent, the transport connection is released.

Session User Requirements field shall not be present if the Reason Code field is not set to 2. If the Reason Code field is set to 2, this field shall be present and be set to '0002'H.

The Reason Code field shall contain a reason code in the first octet. Depending on the value of this first octet, additional octets may be used. The following values are defined for the first octet:

a) 0: Rejection by called SS-user; reason not specified.

b) 1: Rejection by called SS-user due to temporary congestion.

c) 2: Rejection by called SS-user. Subsequent octets may be used for user data up to a length of 512 octets if Protocol Version 1 has been selected, and up to a length such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets if Protocol Version 2 has been selected.

d) * 128 + 1: Session Selector unknown.

e) * 128 + 2: SS-user not attached to SSAP.

f) 128 + 3: Session Protocol Machine congestion at connect time.

g) * 128 + 4: Proposed protocol versions not supported.

h) * 128 + 5: Rejection by the Session Protocol Machine, reason not specified.

i) * 128 + 6: Rejection by the Session Protocol Machine; implementation restriction stated in the Protocol Implementation Conformance Statement.

NOTE – Reasons marked with an asterisk (*) may be considered persistent, others may be considered as transient.

All other values are reserved.

8.3.6 Session FINISH SPDU

Table 4 – Parameters of the FINISH SPDU

PGI	M/O	Code	PI	M/O	Code	Length
			Transport Disconnect	O	17	
User Data	M	193				

The SI field shall be given the value 9.

The Transport Disconnect field indicates whether the underlying transport-connection shall be released or kept. The encoding for this field shall be:

a) bit 1 = 0: Transport connection is kept; or

b) bit 1 = 1: Transport connection is released.

If this field is absent, the transport connection is released.

The User Data field shall hold the **OsiUnbind** (see 7.6.4). The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

NOTE – The Enclosure Item parameter as defined for the FINISH SPDU by ITU-T Rec. X.225 | ISO/IEC 8327-1 is not relevant, as only a limited amount of user data will be passed.

8.3.7 Session DISCONNECT SPDU

Table 5 – Parameters of the DISCONNECT SPDU

PGI	M/O	Code	PI	M/O	Code	Length
User Data	M	193				

The SI field shall be given the value 10.

The User Data field shall hold the **OsiUnbindResult** (see 7.6.5). The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

NOTE – The Enclosure Item parameter as defined for the DISCONNECT SPDU by ITU-T Rec. X.225 | ISO/IEC 8327-1 is not relevant, as only a limited amount of user data will be passed.

8.3.8 Session ABORT SPDU

Table 6 – Parameters of the ABORT SPDU

PGI	M/O	Code	PI	M/O	Code	Length
			Transport Disconnect	M	17	
			Reflect Parameter Value	O	49	9 octets maximum
User Data	O	193				

The SI field shall be given the value 25.

The Transport Disconnect field shall indicate whether or not the transport connection is to be kept, together with an optional reason code. The encoding for this field shall be:

a) bit 1 = 0: Transport connection is kept;

b) bit 1 = 1: Transport connection is released;

c) bit 2 = 1: User abort;

d) bit 3 = 1: Protocol error;

e) bit 4 = 1: No reason;

f) bit 5 = 1: Implementation restriction stated in the Protocol Implementation Conformance Statement.

Bits 6-8 are reserved.

The Reflect Parameter Values field shall only be present if the Transport Disconnect field indicates protocol error and shall contain an implementation-defined value and semantics.

The User Data field shall only be present if the Transport Disconnect field indicates user abort and shall contain the **ARU-PPDU** (see 7.6.7.1) or **ARP-PPDU** (see 7.6.7.2). The length of the User Data parameter is limited such that the total length (including SI and LI) of the SPDU does not exceed 65 539 octets.

NOTE – The Enclosure Item parameter as defined for the ABORT SPDU by ITU-T Rec. X.225 | ISO/IEC 8327-1 is not relevant, as only a limited amount of user data will be passed.

8.3.9 Session ABORT ACCEPT SPDU

The SI field shall be given the value 26.

There is no parameter field associated with this SPDU.

8.3.10 Session DATA TRANSFER SPDU

The Session DATA Transfer SPDU consists in principle of two concatenated SPDU, where the first one is a so-called GIVE TOKEN SPDU. It consists in the form used by this Directory Specification only of the SI field, which has the value 1, and a length field having the value zero.

NOTE – ITU-T Rec. X.225 | ISO/IEC 8327-1 defines basic and extended concatenation. Extended concatenation is not used by this Directory Specification. Basic concatenation is only relevant for the DATA TRANSFER SPDU and Table 7 of ITU-T Rec. X.225 | ISO/IEC 8327-1 specifies that the DATA TRANSFER SPDU shall be concatenated with the GIVE TOKEN SPDU. As we are only using the full duplex functional unit, the Token item is not needed and neither is the User data.

Table 7 – Parameters of the DATA TRANSFER SPDU

PGI	M/O	Code	PI	M/O	Code	Length
			Enclosure Item	O	25	1 octet
User Information field						

The SI field for the DATA TRANSFER SPDU shall also be given the value 1.

User Information field holds the complete or part of a Directory PDU. The LI field following the SI field does not include the User Information Field.

The Enclosure Item PV field, if present, shall indicate whether or not this SPDU is the beginning or end of the Directory PDU. This field shall be present if segmenting may be used. This field shall not be present if segmenting shall not be used. The encoding for this field shall be:

a) bit 1 = 1: Beginning of Directory PDU;
   bit 1 = 0: Not beginning of Directory PDU;
b) bit 2 = 1: End of Directory PDU;
   bit 2 = 0: Not end of Directory PDU.

Bits 3-8 are reserved.

If this field is not present, segmenting is not allowed and this SPDU contains a complete Directory PDU.

Example of encoding:

If the Enclose Item is not included, the encoding of the concatenated SPDUs would be: '01 00 01 00'H.

If the Enclose is included and the SPDU holds the complete Directory PDU, the encoding of the concatenated SPDUs would be: '01 00 01 03 19 01 03'H.

8.4 Use of transport service

Before an application-association can be established, a transport-connection, as defined by ITU-T Rec. X.214 | ISO/IEC 8072, has to be established.

Only the initiator of a transport-connection is allowed to initiate an application-association.

NOTE – This restriction is specified in 6.1.4 of ITU-T Rec. X.225 | ISO/IEC 8327-1.

A transport-connection may be established by mapping onto the service as defined by ITU-T Rec. X.214 | ISO/IEC 8072 or by establishing a transport-connection according to the specification in 8.5. In the former case, all the session SPDUs are mapped onto T-DATA request and T-DATA indication. In the latter case, all session SPDUs are carried by the User parameter of the DT TPDU.

When an application-association is refused, or has been successfully connected and subsequently disconnected, by abort or normal release, the supporting transport connection may be either disconnected or reused.

The transport connection may be kept for reuse provided that:

a) the application process that established the transport connection requests retention of the transport connection by parameter in an ABORT SPDU or a FINISH SPDU; or

b) the application process that established the transport connection receives a REFUSE SPDU or an ABORT SPDU which indicates by parameter that the transport connection is to be retained.

To avoid contention for a retained transport connection, only the transport connection initiator may reuse the transport connection by issuing a Bind request to establish a new application-association.

Transport expedited flow is not used.

8.5 OSI Transport Layer on top of TCP

8.5.1 Scope and limitation

This subclause defines ISO Transport Service on top of TCP (ITOT).

NOTE 1 – This includes specifying those functions of ITU-T Rec. X.224 | ISO/IEC 8073 together with the enhancements defined by IETF RFC 2126 that are relevant for these Directory Specifications.

NOTE 2 – ITU-T Rec. X.224 | ISO/IEC 8073 defines several protocol classes to cope with different qualities of network connections. Protocol classes 0 and 2 are the only protocol classes considered here. These protocol classes have been designed to be used over a type A network connection. A type A network connection is according to ITU-T Rec. X.224 | ISO/IEC 8073 defined as: "Network connection with acceptable residual error rate (for example, not signalled by disconnect or reset) and acceptable rate of signalled errors".

NOTE 3 – A TCP connection enhanced by IETF RFC 2126 is assumed to correspond to a type A network connection.

NOTE 4 – ITU-T Rec. X.224 | ISO/IEC 8073 specifies mandatory features not needed by these Directory Specifications. That means that an implementation only supporting the OSI Transport Layer on top of TCP as defined here does not conform to ITU-T Rec. X.224 | ISO/IEC 8073.

Network address structure is specified in 11.3 and 11.4.

Direct use of the OSI Network Layer, e.g. direct use of an OSI connectionless-mode network protocol, is not considered here. If that is required, the relevant OSI documentation has to be consulted.

8.5.2 Overview of the transport-protocol

8.5.2.1 Functions of the transport-protocol

The functions of the transport-protocol are those necessary to bridge the gap between the service available from the TCP and what is required for supporting the session-protocol as specified in 8.3.

The transport-protocol specified here is a connection mode protocol, i.e., a formal transport-connection has to be established before transfer of data may take place.

A message transmitted between two systems at the transport level is called a transport-protocol-data-unit (TPDU).

Data transfer is in full duplex mode by means of a two-way simultaneously communication.

The transport-connection is released when it is not needed anymore or as the result of an error condition.

8.5.3 Protocol classes and options

8.5.3.1 General

The functions of the Transport Layer are organized into classes and options.

A class defines a set of functions. Optional functions defined within a class may or may not be implemented.

This Directory Specification specifies two classes, class 0 and class 2, for OSI transport over TCP.

NOTE – These two classes correspond to class 0 and class 2 as specified by ITU-T Rec. X.224 | ISO/IEC 8073.

During the connection establishment the characteristics of the connection are determined as follows:

– selection of protocol class; and

– the maximum TPDU size.

8.5.3.2 Characteristics of class 0

Class 0 gives basic capabilities and shall be supported by an implementation claiming conformance to OSI transport over TCP. It includes the following capabilities:

– connection establishment over an already existing TCP connection;

– data transfer with segmentation;

– flow control as provided by underlying TCP;

– error reporting; and
– implicit disconnect by TCP disconnection.

Class 0 provides the simplest type of transport-connection.

8.5.3.3 Characteristics of class 2

Class 2 may optionally be supported by implementations claiming conformance to OSI transport over TCP. It includes the following additional capabilities:

– explicit disconnect instead of implicit disconnect.

NOTE – According to 4.2.1 of IETF RFC 2126, multiplexing of multiple transport-connections over a TCP connection is not supported by ITOT.

A transport-connection may be terminated either with a Disruptive Disconnect or a Non-Disruptive Disconnect. A DR TPDU and a DC TPDU are exchanged in both cases.

Non-Disruptive Disconnect is performed when there is no application-association using this transport-connection, which has not to be kept for reuse.

A Disruptive Disconnect is performed when there is an application-association under establishment, established or under termination on this transport-connection at the time the disconnection of the transport-connection is initiated.

In both cases, the DR TPDU Reason code is set to 128+0 ('80'H). In case of Non-Disruptive Disconnect, the Additional Information parameter is set to '80'H.

8.5.4 TPDU types

The following TPDU types are relevant for this Directory Specification.

TPDU type	Validity within classes		See subclause	Code
	Class 0	Class 2		
CR TPDU (connection request)	x	x	8.5.6.1	1110 0000
CC TPDU (connection confirm)	x	x	8.5.6.2	1101 0000
DR TPDU (disconnect request)	x	x	8.5.6.3	1000 0000
DC TPDU (disconnect confirm)		x	8.5.6.4	1100 0000
DT TPDU (data)	x	x	8.5.6.5	1111 0000
ER TPDU (TPDU error)	x	x	8.5.6.6	0111 0000

NOTE – Class 0 only uses the DR TPDU to reject a suggested transport-connection establishment, not to initiate a release of an existing transport-connection.

8.5.5 General TPKT structure

The TCP does not have the concept of a delimited protocol-data-unit, but manages a continuous stream of octets. Delimitation has to be performed by the overlying protocol. When a TPDU is exchanged using the TCP support, it is prefixed with additional header fields, which together with the TPDU constitute a transport packet (TPKT). This TPKT provides the needed delimitation. The structure of a TPKT with the imbedded TPDU is shown in Figure 4. The TPKT header fields are identical for all TPDU types.

Subclauses 8.5.5.1-8.5.5.3 describe the TPKT header fields, while 8.5.5.4-8.5.5.7 describe the TPDU parameters.

The TPKT special header fields and the TPDU shall each consists of an integral number of octets. The octets in a TPDU are numbered starting from 1 and increasing in the order they are presented to TCP or received from TCP. The bits in an octet are numbered from 1 to 8, where bit 1 is the lowest order bit. When consecutive octets are used to represent a binary number, the lower octet number has the most significant value.

The encoding of TPDUs is shown as follows:

a) octets are shown with the lowest numbered octet to the left; higher numbered octets being further to the right;

b) within an octet, bits are shown with bit 8 to the left and bit 1 to the right, where bit 8 is the most significant bit.

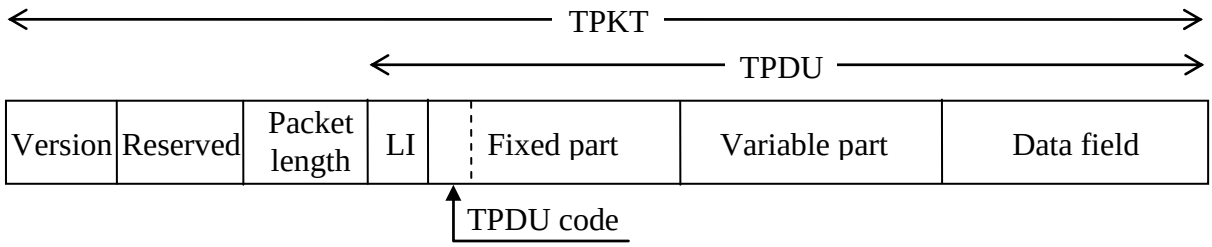

Figure 4 – TPKT structure

8.5.5.1 Version field

This is an 8-bit field that signals the version of the TPKT. It is an unsigned binary encoded integer and it shall have the value 3.

NOTE – This is the value specified by IETF RFC 1006 (version 3). This value is kept to allow compatibility with current IETF RFC 1006 implementations. It is for the same reason the value specified by IETF RFC 2126.

8.5.5.2 Reserved field

This is an 8-bit field. It shall be set to binary zeroes. A receiving system should not try to interpret this field, but should ignore its content.

8.5.5.3 Packet length

This is a 16-bit field. It is an unsigned binary encoded integer that gives the total length in octets of the TPKT, including all the TPKT special headers.

8.5.5.4 Length indicator field

This field is contained in the first octet of the TPDU. The value shall be an unsigned binary number indicating the length in octets of the TPDU, excluding the length indicator field itself and excluding the user data, if any. The maximum value shall be 254 (1111 1110).

8.5.5.5 Fixed part

The fixed part contains frequently occurring parameters including the TPDU code. The length and the structure of the fixed part depend on the TPDU type and in certain cases on the protocol class. If any of the parameters of the fixed part have an invalid value, or if the fixed part cannot be contained within the header (as defined by LI), this is a protocol error.

The TPDU code field is contained in octet 2 of the TPDU. It is used to signal the type of TPDU.

8.5.5.6 Variable part

The variable part is used to define less frequently used parameters. If the variable part is present, it shall contain one or more parameters.

Each parameter contained within the variable part is structured as follows:

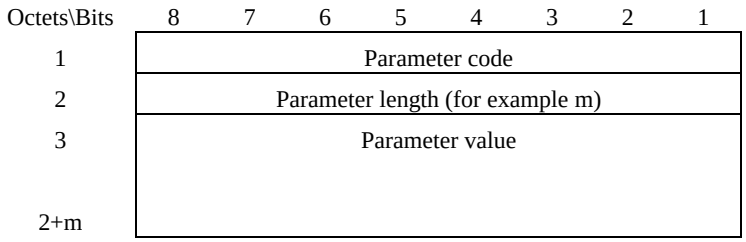

Figure 5 – Structure of parameter within variable part

The parameter code is an unsigned binary encoded integer.

The parameter length is an unsigned binary integer that shall hold the length in octets of the parameter value.

The parameters defined in the variable part may be in any order. If any parameter is duplicated, then the last value shall be used. A parameter not defined in this Directory Specification shall be treated as a protocol error in any received TPDU except in a CR TPDU; in a CR TPDU such a parameter shall be ignored.

If the responder selects a class for which a parameter of the CR TPDU is not defined, it may ignore this parameter, except for the alternative protocol class parameter, which shall always be interpreted.

A parameter defined in this Directory Specification but having an invalid value shall be treated as a protocol error in any received TPDU.

8.5.5.7 User data field

This field contains transparent user data. Only the DT TPDU has a data field.

NOTE – ITU-T Rec. X.224 | ISO/IEC 8073 also defines user data for the CR TPDU, the CC TPDU and the DR TPDU. However, all SPDUs defined in 8.3 are mapped onto the user data field of the DT TPDU. These Directory Specifications have no further use of the user data field.

8.5.6 Structure and encoding of TPDUs

8.5.6.1 Connection request (CR TPDU)

8.5.6.1.1 Structure

The procedure for connection establishment is used in both classes to create a new transport-connection.

The initiator initiates a transport-connection by transmitting a CR TPDU to the responder, which replies with a CC TPDU if the connection is accepted. Otherwise, a DR TPDU is returned.

Before sending the CR TPDU, the initiator assigns the transport-connection being created to a TCP connection.

During the connection exchange, all information and parameters needed for the operation shall be exchanged or negotiated.

NOTE – It is recommended that the initiator start a timer at the time the CR TPDU is sent. This timer should be stopped when the connection is considered as accepted or refused. If the timer expires, the initiator should disconnect the TCP connection.

The format of the CR TPDU is shown in Figure 6. The fixed part parameters are specified in 8.5.6.1.2 to 8.5.6.1.6, while the parameters within the variable part field are specified in 8.5.6.1.7 to 8.5.6.1.13.

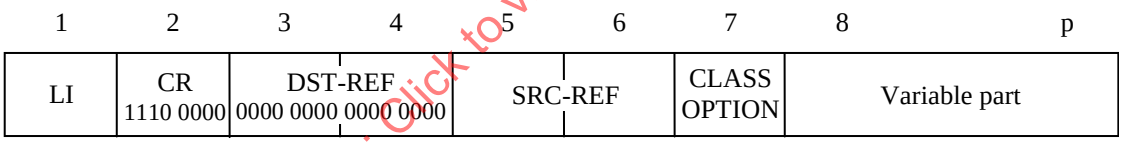

Figure 6 – Connection request (CR TPDU)

8.5.6.1.2 Length indicator (LI) parameter

See 8.5.5.4.

8.5.6.1.3 TPDU code field

This is the CR TPDU code parameter and shall take the value '1110 0000'B.

8.5.6.1.4 DST-REF parameter

The DST-REF parameter shall be set to binary zeroes by the initiator.

8.5.6.1.5 SRC-REF parameter

SRF-REF parameter shall hold the reference identification of the requested transport-connection as seen by the initiator. This value shall not be zero and it shall not be a value already in use for an existing transport-connection.

This mechanism is symmetrical and provides identification of the transport-connection independent of the network connection. The range of references used for transport-connections, in a given system, is a local matter.

8.5.6.1.6 CLASS OPTION parameter

This parameter is used by the initiator to specify the preferred transport protocol class.

Bits 8 to 5 of octet 7 are used to negotiate the transport protocol class to be operated over the requested transport-connection. This parameter shall take the value:

– '0000'B for class 0; or

– '0010'B for class 2.

The initiator may specify class 0 as an alternative protocol class in the variable part of the CR TPDU if it has specified class 2 as the preferred protocol class.

Bits 4 to 1 of octet 7 shall be set to 0001. If the bits are not set as indicated, the responder shall refuse the connection with Reason 128+2 ('82'H).

NOTE – ITU-T Rec. X.224 | ISO/IEC 8073 specifies that bit 4 and 3 shall always be set to 0. Bit 1 shall be set to 1, if non use of explicit flow control is selected, as it is specified by IETF RFC 2126. ITU-T Rec. X.224 | ISO/IEC 8073 specifies that extended format shall not be selected if non use of explicit flow control is selected, which requires bit 2 to be set to 0.

8.5.6.1.7 Transport-selector parameters

The CR TPDU may hold two transport-selectors, one for the initiator (calling transport-selector) and one for the responder (called transport-selector).

The parameter codes are:

1100 0001 for the calling transport-selector

1100 0010 for the called transport-selector

A parameter length is the length in octets of the corresponding transport-selector.

NOTE – The maximum length for a transport-selector is not defined. However, the length is restricted by the 254 octets maximum CR TPDU size.

8.5.6.1.8 TPDU size parameter

This parameter defines the proposed maximum TPDU size (in octets including the header) to be used over the requested transport-connection. The coding of this parameter is:

Parameter code:	1100 0000	
Parameter length:	1 octet	
Parameter value:	0000 1101	8192 octets
	0000 1100	4096 octets
	0000 1011	2048 octets
	0000 1010	1024 octets
	0000 1001	512 octets
	0000 1000	256 octets
	0000 0111	128 octets
Default value:	65531 octets.	

8.5.6.1.9 Preferred TPDU size parameter

This parameter defines the proposed maximum TPDU size (in octets including the header) to be used over the requested transport-connection.

The coding of this parameter is:

Parameter code:	1111 0000
Parameter length:	Up to 3
Parameter value:	A binary value. The binary value indicates the maximum TPDU size, expressed as a multiple of 128 octets. This binary value shall be greater than or equal to 1.
Maximum value	65531 octets

NOTE 1 – ITU-T Rec. X.224 | ISO/IEC 8073 allows the parameter length to be up to 4 octets. However, as the maximum TPDU size is 65531, the maximum value this parameter may take is 511.

NOTE 2 – The maximum value is determined by having a TPKT length of 'FFFF'H minus 4 octets for the TPKT header fields.

8.5.6.1.10 Version number parameter

This field shall not be included if class 0 is the preferred class.

This field may be included if the preferred class is 2. If included, it shall have the following format.

Parameter code: 1100 0100

Parameter length: 1 octet

Parameter value: 0000 0001

Default value: 0000 0001

8.5.6.1.11 Protection parameter

The use of this parameter is not defined by this Directory Specification. A receiving system may ignore this parameter if it is present. This parameter shall not be present if class 0 is the preferred class.

Parameter code: 1100 0101

Parameter length: Not defined by this Directory Specification

Parameter value: Not defined by this Directory Specification

8.5.6.1.12 Additional option selection parameter

This parameter shall not be present if class 0 is the preferred class.

Parameter code: 1100 0110

Parameter length: 1

Parameter value:

Bit	Option
8-2	Not applicable
1	Always set to 0

Not applicable bits may take any value and shall not be checked by the responder. Default for bit 1 is 0, which is identical to the required setting. If this parameter is present, the responder is only required to check bit 1 for correct setting. If bit 1 is set to 1, the responder shall refuse the connection with Reason 128+2.

8.5.6.1.13 Alternative protocol class parameter

This parameter shall not be present if class 0 is the preferred class. It is optional if class 2 is the preferred class.

The coding of this parameter is:

Parameter code: 1100 0111

Parameter length: 1

Parameter value: 0000 0000 indicating that class 0 is an alternative protocol class.

8.5.6.2 Connection confirm (CC TPDU)

8.5.6.2.1 Structure

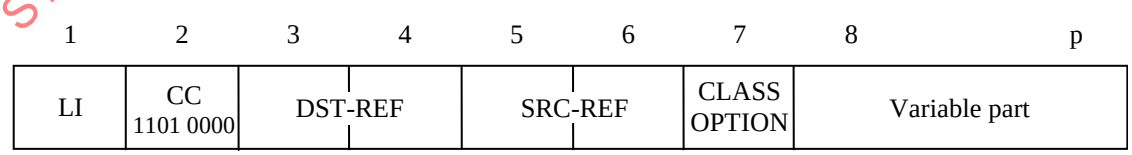

Figure 7 – Connection confirm (CC TPDU)

8.5.6.2.2 Length indicator (LI) field

See 8.5.5.4.

8.5.6.2.3 TPDU code parameter

This is the CC TPDU code parameter and shall take the binary value 1101 0000.

8.5.6.2.4 DST-REF parameter

DST-REF parameter shall hold the reference of the initiator, i.e., it shall echo the value of the SCR-REF parameter of the corresponding CR TPDU.

8.5.6.2.5 SRC-REF parameter

SRF-REF parameter shall hold the identification reference of the transport-connection as seen by the responder. This value shall not be zero and it shall not be a value already in use by the responder.

8.5.6.2.6 CLASS OPTION parameter

Bits 8 to 5 of octet 7 are used by the responder to specify the selected transport protocol class. These bits shall take the value:

– 0000 for class 0; or

– 0010 for class 2.

Bits 4 to 1 of octet 7 shall be set to 0001. If the bits are not set as indicated, the initiator shall consider it a protocol error.

If class 2 is the only class proposed by the initiator and class 2 is not supported by the responder, a DR TPDU shall be returned with REASON 128+2 and the SRC-REF parameter set to zero to indicate an unassigned reference.

NOTE – An implementation based on IETF RFC 1006 may not recognize the above situation and might accept the transport-connection specifying class 0 as the selected class.

8.5.6.2.7 Transport-selector parameters in the variable part parameter

The CR TPDU may hold two transport-selectors, one for the initiator and one for the responder.

The parameter codes are:

1100 0001 for the calling transport-selector

1100 0010 for the responding transport-selector

A parameter length is the length in octets of the corresponding transport-selector.

8.5.6.2.8 TPDU size parameter

This parameter defines the selected maximum TPDU size (in octets including the header) to be used over the accepted transport-connection. The coding of this parameter is as in 8.5.6.1.8.

8.5.6.2.9 Preferred TPDU size parameter

This parameter defines the selected maximum TPDU size (in octets including the header) to be used over the accepted transport-connection.

The coding of this parameter is as in 8.5.6.1.9.

8.5.6.2.10 Protection parameter

The use of this parameter is not defined by this Directory Specification. A receiving system may ignore this parameter if it is present. This parameter shall not be present if class 0 is the preferred class. For the encoding, see 8.5.6.1.11.

8.5.6.2.11 Additional option selection parameter

This parameter shall not be present if class 0 is the preferred class.

If present, this parameter shall be encoded as specified in 8.5.6.1.12. If bit 1 is not set correctly, the initiator shall consider it a protocol error.

8.5.6.3 Disconnect request (DR TPDU)

8.5.6.3.1 Structure

The disconnect request (DR TPDU) is used for refusal of a connection request for both protocol classes.

For class 2, the DR TPDU is also used for initiating an explicit release of a transport-connection without necessarily release of the underlying TCP connection. Either one of the two sides may issue a DR TPDU to initiate release (see 14.1 of ITU-T Rec. X.214 | ISO/IEC 8072).

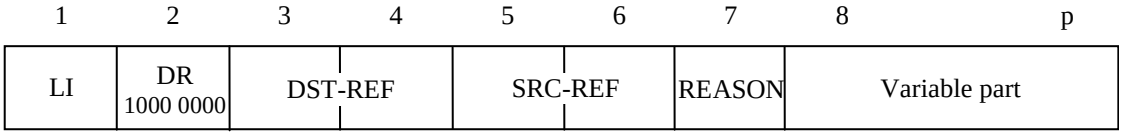

Figure 8 – Disconnect request

8.5.6.3.2 Length indicator (LI) parameter

See 8.5.5.4.

8.5.6.3.3 TPDU code parameter

This is the DR TPDU code parameter and shall take the binary value 1000 0000.

8.5.6.3.4 DST-REF parameter

This parameter shall hold the transport-protocol reference of the receiver.

8.5.6.3.5 SRC-REF parameters

If the DR TPDU is sent to reject a requested transport-connection, this parameter shall be filled with binary zeroes to indicate that no reference has been allocated by the responder.

If the DR TPDU is sent to initiate release of an existing transport-connection, this parameter shall hold the transport-protocol reference of the sender.

8.5.6.3.6 REASON parameter

The Reason parameter defines the reason for disconnecting the transport-connection. This parameter shall take one of the following values.

The following values may be used for class 2:

1) 128+0 Normal disconnect initiated by session entity.
2) 128+1 Responder congestion at connect request time.
3) *128+2 Connection negotiation failed (i.e. proposed class 2 not supported).
4) 128+3 Duplicate source reference detected for the same pair of NSAPs.
5) 128+4 Mismatched references.
6) 128+5 Protocol error.
7) 128+6 Not used.
8) 128+7 Reference overflow.
9) 128+8 Connection request refused on this TCP connection.
10) 128+9 Not used.
11) 128+10 Header or parameter length invalid.

The following values can be used for both classes:

12) 0 Reason not specified.
13) 1 Congestion.
14) *2 No session functionality associated with the transport-address.
15) *3 Address unknown.

NOTE – Reasons marked with an asterisk may be considered as persistent, other reasons as transient.

8.5.6.3.7 Additional clearing information parameter (variable part)

This parameter allows additional information related to the clearing of the connection.

The coding of this parameter is:

Parameter code: 1110 0000

Parameter length: Any value provided that the length of the DR TPDU does not exceed the maximum agreed TPDU size or 128 when the DR TPDU is used during the connection refusal procedure.

Parameter value: Additional information. The content of this parameter is not defined by this Directory Specification.

8.5.6.4 Disconnect confirm (DC TPDU)

8.5.6.4.1 Structure

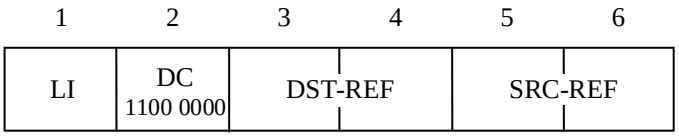

Figure 9 – Disconnect confirm

8.5.6.4.2 Length indicator (LI) parameter

See 8.5.5.4.

8.5.6.4.3 TPDU code parameter

This is the DC TPDU code parameter and shall take the binary value 1100 0000.

8.5.6.4.4 DST-REF parameter

DST-REF parameter holds the identification reference of the transport connection at the remote transport entity and shall be set to the SRC-REF parameter of the DR TPDU received (it may be equal to zero).

8.5.6.4.5 SRC-REF parameter

SRC-REF parameter holds the identification reference of the transport connection at the local transport entity and shall be set to the DST-REF parameter of the DR TPDU received (it may be equal to zero).

8.5.6.5 Data (DT TPDU)

8.5.6.5.1 Structures

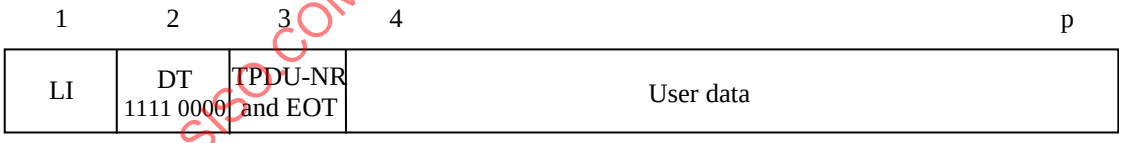

Figure 10 – Data TPDU for class 0

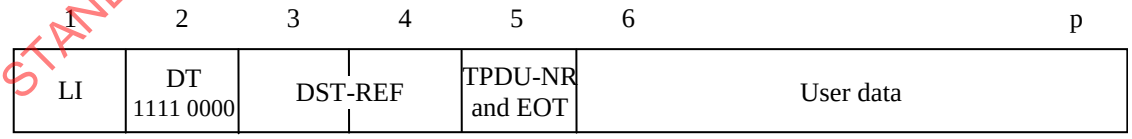

Figure 11 – Data TPDU for class 2

NOTE – Extended format as defined in ITU-T Rec. X.224 | ISO/IEC 8073 is not used, as it has no meaning if explicit flow control at the transport level is not used. Clause 6.5.4 of ITU-T Rec. X.224 | ISO/IEC 8073 specifies: In class 2, whenever a transport entity requests or agrees to the transport expedited data transfer service or to the use of extended formats, it shall also request or agree (respectively) to the use of explicit flow control.

8.5.6.5.2 Length indicator (LI) parameter

See 8.5.5.4.

8.5.6.5.3 TPDU code parameter

This is the DT TPDU code parameter and shall take the binary value 1111 0000.

8.5.6.5.4 DST-REF parameter

This parameter shall hold the transport-protocol reference of the receiver (only if class 2 has been negotiated).

8.5.6.5.5 TPDU-NR and EOT parameter

EOT – When set to ONE, it indicates that the current DT TPDU is the last data unit of a complete DT TPDU sequence (end of TSDU). EOT is bit 8 of octet 3 in class 0 and bit 8 of octet 5 for class 2.

TPDU-NR – TPDU send sequence number (zero in class 0). It may take any value in class 2 without explicit flow control. TPDU-NR is bits 7 to 1 of octet 3 for class 0, bits 7 to 1 of octet 5 for normal formats in class 2.

8.5.6.5.6 User data field

This field contains (part of) the SPDU or SPDU segment being transmitted.

NOTE – The length of this field is limited to the negotiated TPDU size minus 3 octets in class 0 and minus 5 octets in class 2.

8.5.6.6 TPDU error (ER TPDU)

8.5.6.6.1 Structure

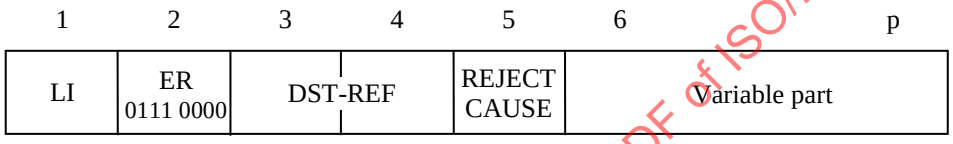

Figure 12 – Error TPDU

8.5.6.6.2 Length indicator (LI) parameter

See 8.5.5.4.

8.5.6.6.3 TPDU code parameter

This is the ER TPDU code parameter and shall take the binary value 0111 0000.

8.5.6.6.4 DST-REF parameter

DST-REF parameter holds the identification reference of the transport connection at the remote transport entity and shall be set to the SRC-REF parameter of the rejected TPDU.

8.5.6.6.5 REJECT CAUSE parameter

0000 0000	Reason not specified
0000 0001	Invalid parameter code
0000 0010	Invalid TPDU type
0000 0011	Invalid parameter value

8.5.6.6.6 Invalid TPDU parameter

Parameter code:	1100 0001
Parameter length:	Number of octets of the value parameter
Parameter value:	Contains the bit pattern of the rejected TPDU header up to and including the octet which caused the rejection. This parameter is mandatory in class 0.

8.5.7 Use of the service provided by TCP

The use of the service provided by TCP is expressed by reference to the conceptual calls as defined in 3.8 of IETF RFC 793.

8.5.7.1 TCP connection establishment (socket connection)

A connection is established by the DUA or DSA issuing an active OPEN call and by the replying system having an outstanding passive OPEN call (see 3.4 of IETF RFC 793).

When issuing an active OPEN, the initiator shall:

a) Specify the socket (IP address and TCP port number) of the responder as determined from:
   – information returned in a **ContinuationReference** as the result of a previous communication; or
   – locally held information.

b) Set the active flag.

NOTE 1 – Because many operating systems do not support fixed outgoing TCP ports, only dynamic allocation of local port numbers is assumed here, but it is not an error to specify a local port number if support is provided for that.

An active OPEN will fail if the responder has not issued a passive OPEN.

When issuing a passive OPEN, the responder shall:

a) Specify the local port number to be used. The "well known" TCP port number 102 is reserved for ITOT. It is allowed to use another port number.

NOTE 2 – It is suggested that the TCP keep alive mechanism be selected, as this ensures reporting of network connection loss.

8.5.7.2 Data transfer

When a DUA or DSA issues a SEND call to send a TPDU:

– the PUSH flag shall be set to ensure immediate delivery; and
– the URGENT flag shall not be set.

NOTE – For performance reason it is suggested that the Nagle algorithm (RFC 896) be disabled (using the TCP_NODELAY socket option). This feature allows TPKT data to be sent without delay.

8.5.7.3 TCP connection release

The connection can be released by either:

– closing the connection, which leads to terminating gracefully the message flow; or
– aborting the connection, which leads to deletion of ongoing messages.

8.5.7.3.1 Orderly release

An orderly release ensures that data in transit is not lost (see 3.8 of IETF RFC 793).

Either side may at any time initiate an orderly release by issuing a CLOSE.

An orderly release is disruptive under the following conditions:

– when supporting a transport-protocol class 0 class and the overlying application-association has not been terminated; and
– when supporting a transport-protocol class 2 class and the overlying transport-connection has not been terminated.

8.5.7.3.2 TCP abort

A system should only issue an ABORT whenever it detects a serious exception, such as an abnormally functioning communication partner.

8.5.8 Elements of procedures for the transport-protocol

8.5.8.1 Segmenting and reassembling

The segmenting and reassembling procedure is used in both classes to map a SPDU onto TPDUs.

A system shall map a SPDU onto an ordered sequence of one or more DT TPDUs. This sequence shall not be interrupted by other DT TPDUs on the same transport-connection.

All DT TPDUs except the last DT TPDU in a sequence greater than one shall have a length of data greater than zero.

The EOT parameter of a DT TPDU indicates whether or not there are subsequent DT TPDUs in the sequence.

8.5.8.2 Connection establishment

A transport-connection is established by means of the initiator transmitting a CR TPDU to the responder, which replies with a CC TPDU.

Before sending the CR TPDU, the initiator assigns the transport-connection being created to one TCP connection. It is this TCP connection over which the TPDUs are sent.

Only the initiator of a TCP connection may initiate a transport-connection on that TCP connection.

During this exchange, all information and parameters needed for the two parties to operate shall be exchanged or negotiated.

8.5.8.2.1 References

Each system chooses a reference for the transport connection to be used by the peer system in the DST-REF parameter when sending TPDUs to that system.

This mechanism is symmetrical and provides identification of the transport-connection independent of the TCP connection. The range of references used for transport-connections for a given system is a local matter.

8.5.8.2.2 Transport selectors

Calling, Called and Responding Transport-Selectors (optional) – When the TCP/IP addressing (IP address and Port Number) unambiguously defines the transport address, this information may be omitted.

A transport-selector parameter with a length indicator set to zero shall be treated as having the "nil selector value".

NOTE – This requirement specified in 9.5.2 of ITU-T Rec. X.650 | ISO/IEC 7498-3 for the calling and called transport-selector. For the responding transport-selector, this requirement is specified in 13.2.3 of ITU-T Rec. X.224 | ISO/IEC 8073.

8.5.8.2.3 Protection parameter

This parameter and its semantics are not defined by this Directory Specification. A system shall be able to receive this parameter, but may ignore its content.

8.5.8.2.4 Protocol class negotiation

Protocol class negotiation is optional. If the initiator does not support protocol class negotiation, it shall specify class 0 as the preferred protocol class.

If the initiator supports protocol class negotiation, it may specify either class 0 or 2 as the preferred protocol class. If the preferred class is class 2, it may propose class 0 as an alternative class. If the preferred class is class 0, it shall not propose an alternative class. The initiator should assume when it sends the CR TPDU that its preferred class will be agreed to, and commence the procedures associated with that class.

If the initiator does not specify the class options according to the above, it shall by the responder be considered as a protocol error.

If the responder does not support protocol class negotiation, it may return class 0 as the selected protocol class independent on what the initiator specifies.

When the responder supports protocol class negotiation, it shall select one class according to the following:

- if the preferred transport protocol class in the CR TPDU is class 0, then class 0 shall be returned in the CC TPDU:
- if the preferred transport protocol class in the CR TPDU is class 2, and the alternative class is class 0, then class 0 or class 2 shall be returned; and
- if the preferred transport protocol class in the CR TPDU is class 2, and no alternative class is specified, then class 2 shall be returned. If class 2 is not supported by the responder, a DR TPDU shall be returned with REASON 128+2 and the SRC-REF parameter set to zero to indicate an unassigned reference.

The responder shall indicate the selected class in the CC TPDU and shall follow the procedures for the selected class.

If the preferred class is not selected, then on receipt of the CC TPDU, the initiator shall adjust its operation according to the procedures of the selected class.

The initiator shall terminate the transport connection if:

- the initiator specifies the preferred class as class 2 and does not specify class 0 as an alternative class; and
- the responder accepts the connection specifying class 0 as the selected class.

8.5.8.2.5 TPDU size negotiation

There are two mechanisms for negotiating maximum TPDU size as specified by 8.5.6.1.8 and 8.5.6.1.9 and further developed under a) and b) below.

a) *TPDU size* can be considered as the primary mechanism. It is optional and does not have to be supported by the initiator or responder. However, in its absence it has the default value of 65531 octets.

NOTE 1 – ITU-T Rec. X.224 | ISO/IEC 8073 specifies a default of 128 octets, which is also the minimum TPDU size. IETF RFC 1006 and IETF RFC 2126 (and this Directory Specification) specify a default of 65531 octets, which is the maximum TPDU size. This means that the maximum TPDU size is used when no TPDU size negotiation is performed.

The initiator may propose a maximum size for TPDUs, and the responder may accept this value or respond with any value between 128 and the proposed value in the set of values available (see 8.5.6.1.8).

If this parameter is absent, the TPDU size defaults to 65531 octets (unless the preferred maximum TPDU size parameter is included in the CR TPDU and supported by the responder).

An initiator shall support all the listed values for the maximum TPDU size as given in 6.5.6.1.8 up to and including the maximum TPDU size proposed in the CR TPDU.

b) *Preferred maximum TPDU size* – The value of this parameter, multiplied by 128, yields the proposed or accepted maximum TPDU size in octets. The initiator may propose a preferred maximum size for TPDUs and the responder may accept this value or respond with a smaller value.

NOTE 2 – If this parameter is used in a CR TPDU without also including the TPDU size parameter, this will result in a maximum TPDU size of 65531 octets being selected if the responder does not recognize the preferred TPDU size parameter. Therefore, it is recommended that both parameters be included in the CR TPDU.

If the preferred maximum TPDU size parameter is present in a CR TPDU, the responder shall either:

– ignore the preferred maximum TPDU size parameter and follow TPDU size negotiation as defined in a) above; or

– use the preferred maximum TPDU size parameter to determine the maximum TPDU size requested by the initiator and ignore the TPDU size parameter or its default value. In this case, the responder shall use the preferred maximum TPDU size parameter in the CC TPDU and shall not include the TPDU size parameter in the CC TPDU.

If the preferred maximum TPDU size parameter is not present in the CR TPDU, it shall not be included in the corresponding CC TPDU. In this case, TPDU size negotiation is as defined in a) above.

NOTE 3 – If the resulting suggested maximum TPDU size based on the considerations above is not the default maximum TPDU size of 65531 octets, then the responder should include a selected TPDU size in the CC TPDU, as omission of this will result in an implied selected value of 65531 octets, which would violate a) or b) above.

8.5.8.2.6 Option negotiation

There is no option negotiation.

NOTE – ITU-T Rec. X.224 | ISO/IEC 8073 and IETF RFC 2126 define protocol options to be set according to the communications requirements. However, these Directory Specifications require all the options to be fixed (see 8.5.6.1.6 and 8.5.6.1.12).

8.5.8.2.7 Version number

This parameter is only used in the CR TPDU and only for class 2 (see 8.5.6.1.10).

8.5.8.3 Connection refusal

The connection refusal procedure is used in both classes when the responder refuses a transport-connection in response to a CR TPDU.

If a transport-connection cannot be accepted, the responder shall respond to the CR TPDU with a DR TPDU. The Reason shall indicate why the connection was not accepted. The source reference parameter in the DR TPDU shall be set to zero to indicate an unassigned reference.

If a DR TPDU is received, the initiator shall regard the connection as released.

The responder shall respond to an invalid CR TPDU by sending an ER or DR TPDU. If an ER TPDU is received in response to a CR TPDU, the initiator shall regard the connection as released.

NOTE 1 – When the invalid CR TPDU can be identified as having class 0 as the preferred class, it is recommended to respond with an ER TPDU. For class 2 either an ER TPDU or DR TPDU may be sent.

NOTE 2 – If the optional supervisory timer TS1 has been set for this connection, then the initiator should stop the timer on receipt of the DR or ER TPDU.

NOTE 3 – It is a local matter whether the initiator releases the network connection if no transport-connections are currently assigned to it.

8.5.8.4 Normal release

The release procedure is used to terminate a transport-connection. The implicit variant is used only in class 0. The explicit variant is used only in class 2.

NOTE – When the implicit variant is used for class 0, the lifetime of the transport-connection is directly correlated with the lifetime of the TCP connection. The use of the explicit variant of the release procedure for class 2 enables the transport-connection to be released independently of the underlying TCP connection.

8.5.8.4.1 Procedure for implicit variant

In the implicit variant, both the initiator and the responder disconnect a transport-connection by disconnecting the TCP connection to which it is assigned. See also 8.5.7.3.1.

8.5.8.4.2 Procedure for explicit variant

When the release of a transport-connection is to be initiated, a transport entity:

a) If it has previously sent or received a CC TPDU shall:

   1) send a DR TPDU;

   2) discard all subsequently received TPDUs other than a DR, DC or ER TPDU;

   3) consider the transport-connection released on receipt of a DR, DC or ER TPDU.

b) If a) is not applicable and if there is an outstanding CR TPDU, it shall wait for the acknowledgement of the outstanding CR TPDU; if it receives a CC TPDU, it shall follow the procedures in item a) above.

NOTE 1 – This requirement ensures that the transport entity is aware of the remote reference for the transport-connection.

A transport entity that receives a DR TPDU shall:

c) If it has previously sent a DR TPDU for the same transport-connection, consider the transport-connection released.

d) If it has previously sent a CR TPDU that has not been acknowledged by a CC TPDU, consider the connection refused. If the SRC-REF is not zero, a DC TPDU shall be sent using the SRC-REF as the DST-REF.

NOTE 2 – In this case, the DR is associated to that connection regardless of the SRC-REF parameter.

e) If c) and d) are not applicable, send a DC TPDU and consider the transport-connection released. If the received DR has the DST-REF parameter set to zero, then a DC with SRC-REF set to zero shall be sent, regardless of the local reference.

NOTE 3 – If the entity receiving such a DR TPDU has previously decided to negotiate down the class, this entity is always entitled to consider such a DR TPDU as spurious. Since no association has been made the transport-connection is not released at the responder side but the CC TPDU, when sent, will be answered by a DR TPDU (spurious CC TPDU).

NOTE 4 – When the transport-connection is considered as released, the local reference is available for re-use.

NOTE 5 – After the release of a transport-connection, the network connection can be released or retained to enable its re-use for the assignment of other transport-connections.

NOTE 6 – When a transport entity is waiting for a CC TPDU before sending a DR TPDU and the TCP connection is released, it should consider the transport-connection released.

8.5.8.5 Error release

This procedure is used only in classes 0 and 2 to release a transport-connection when either the local system is issuing a TCP abort or a TCP abort is received from the peer system.

When either a TCP abort is issued or a TCP abort is received for a TCP connection, a transport-connection using this TCP connection, the system shall consider that the transport-connection is released and so inform the TS-users.

On receipt of an N-RESET indication:

– In class 0, an N-DISCONNECT request shall be issued.

– In class 2, it is a local choice to issue an N-RESET response or an N-DISCONNECT request; one of these primitives shall be issued.

9 IDM protocol

This clause defines the Internet Directly Mapped Protocol (IDM), a mapping of request-response service elements directly onto the Internet TCP/IP protocol, bypassing the ACSE, Presentation, Session and Transport layers of the OSI model. The protocol is deliberately minimal and is designed for simplicity of implementation. It is connection-oriented and is fully asynchronous.

The protocol makes use of a number of protocol-data-units to transfer bind, request, response and error messages.

9.1 IDM-PDUs

The messages of the Internet Directly Mapped protocol are conveyed over a TCP/IP connection as protocol-data-units called IDM-PDUs and are mapped onto TCP/IP as specified in 9.6. The TCP/IP connection may optionally be protected using TLS, as specified in 9.8. TLS is specified in IETF RFC 2246 and IETF RFC 3546. The ASN.1 definition for an IDM-PDU follows.

```
IDM-PDU {IDM-PROTOCOL:protocol} ::= CHOICE {
        bind            [0]     IdmBind{ {protocol} },
        bindResult      [1]     IdmBindResult{ {protocol} },
        bindError       [2]     IdmBindError{ {protocol} },
        request         [3]     Request{ {protocol.&Operations} },
        result          [4]     IdmResult{ {protocol.&Operations} },
        error           [5]     Error{ {protocol.&Operations} },
        reject          [6]     IdmReject,
        unbind          [7]     Unbind,
        abort           [8]     Abort,
        startTLS        [9]     StartTLS,
        tLSResponse     [10]    TLSResponse }

IdmBind {IDM-PROTOCOL:Protocols} ::= SEQUENCE {
        protocolID              IDM-PROTOCOL.&id ({Protocols}),
        callingAETitle  [0]     GeneralName     OPTIONAL,
        calledAETitle[1]        GeneralName     OPTIONAL,
        argument        [2]     IDM-PROTOCOL.&bind-operation.&ArgumentType
                                        ({Protocols} {@protocolID}) }

IdmBindResult {IDM-PROTOCOL:Protocols} ::= SEQUENCE {
        protocolID              IDM-PROTOCOL.&id ({Protocols}),
        respondingAETitle [0]   GeneralName     OPTIONAL,
        result          [1]     IDM-PROTOCOL.&bind-operation.&ResultType
                                        ({Protocols} {@protocolID}) }

IdmBindError {IDM-PROTOCOL:Protocols} ::= SEQUENCE {
        protocolID              IDM-PROTOCOL.&id ({Protocols}),
        -- errcode              IDM-PROTOCOL.&bind-operation.&Errors.&errorCode
                                        OPTIONAL,
        respondingAETitle [0]   GeneralName     OPTIONAL,
        aETitleError            ENUMERATED {
                                        callingAETitleNotAccepted (0),
                                        calledAETitleNotRecognized (1) } OPTIONAL,
        error           [1]     IDM-PROTOCOL.&bind-operation.&Errors.&ParameterType
                                        ({Protocols} {@protocolID}) }

Request {OPERATION:Operations} ::= SEQUENCE {
        invokeID        INTEGER,
        opcode          OPERATION.&operationCode ({Operations}),
        argument        OPERATION.&ArgumentType ({Operations} {@opcode}) }

IdmResult {OPERATION:Operations} ::= SEQUENCE {
        invokeID        INTEGER,
        opcode          OPERATION.&operationCode ({Operations}),
        result          OPERATION.&ResultType ({Operations} {@opcode}) }

Error {OPERATION:Operations} ::= SEQUENCE {
        invokeID        INTEGER,
        errcode         OPERATION.&Errors.&errorCode ({Operations}),
        error           OPERATION.&Errors.&ParameterType
                                ({Operations} {@errcode}) }
```

```
IdmReject ::= SEQUENCE {
        invokeID            INTEGER,
        reason              ENUMERATED {
                            mistypedPDU                     (0),
                            duplicateInvokeIDRequest        (1),
                            unsupportedOperationRequest     (2),
                            unknownOperationRequest         (3),
                            mistypedArgumentRequest         (4),
                            resourceLimitationRequest       (5),
                            unknownInvokeIDResult           (6),
                            mistypedResultRequest           (7),
                            unknownInvokeIDError            (8),
                            unknownError                    (9),
                            mistypedParameterError          (10) } }

Unbind ::= NULL

Abort ::= ENUMERATED {
                    mistypedPDU             (0),
                    unboundRequest          (1),
                    invalidPDU              (2),
                    resourceLimitation      (3),
                    connectionFailed        (4),
                    invalidProtocol         (5),
                    reasonNotSpecified      (6) }

StartTLS ::= NULL

TLSResponse ::= ENUMERATED {
                    success                 (0),
                    operationsError         (1),
                    protocolError           (2),
                    unavailable             (3) }
```

A **bind** PDU is sent to request a binding between the sender and the responder. **protocolID** identifies the **IDM-PROTOCOL** protocol to be used (see 9.4). **argument** is a value for the **ARGUMENT** field of the **BIND-OPERATION** of the identified protocol. **callingAETitle** is the name of the local application entity sending the **bind** PDU. **calledAETitle** is the name of the remote application entity to which the **bind** PDU is being sent.

A **bindResult** PDU is returned in response to a successful bind request. **protocolID** is the same value sent in the corresponding **bind** PDU. **result** is a value for the **RESULT** field of the **BIND-OPERATION** of the identified protocol. **respondingAETitle** is the name of the remote application entity which sent the **bindResult**.

An **IdmBindError** PDU is returned in response to an unsuccessful bind request. **protocolID** is the same value sent in the corresponding **bind** PDU. **error** is a value for the **PARAMETER** field of the **ERROR**. **respondingAETitle** is the name of the remote application entity which sent the **IdmbindError**. **aETitleError** is set to **callingAETitleNotAccepted** if an **Idmbind** PDU is received and the supplied **callingAETitle** is not acceptable to the called system. **aETitleError** is set to **calledAETitleNotRecognized** if an **IdmBind** PDU is received and the remote application entity knows the application entity which is binding, but does not accept the **calledAETitle** sent in the **IdmBind** PDU as its own name.

A **request** PDU is sent to request an operation. **invokeID** identifies a particular request and its associated responses, and is a positive integer chosen to be different to the value sent in any previous request over that TCP/IP connection. **opcode** is the code for one of the operations listed against the **OPERATIONS** field of the chosen protocol. **argument** is a value for the **ARGUMENT** field of the **OPERATION** identified by **opcode**.

NOTE – **InvokeId** in X.500 systems is semantically equivalent to `messageID` in LDAP systems, as defined in 4.1.1.1 of IETF RFC 4511.

A **result** PDU is returned in response to a successful operation request. **invokeID** and **opcode** are the same values as sent in the request PDU to which this PDU is a reply. **result** is a value for the **RESULT** field of the **OPERATION** identified by **opcode**.

An **error** PDU is returned in response to an unsuccessful operation request. **invokeID** has the same value as sent in the request PDU to which this PDU is a reply. **errcode** is the code for one of the errors listed against the **ERRORS** field of the operation in the request PDU. **error** is a value for the **PARAMETER** field of the **ERROR** identified by **errcode**.

A **reject** PDU is returned in response to a protocol error detected in a received **request**, **result** or **error** PDU from which an invoke ID can be recovered. **invokeID** is the invoke ID of the received PDU that was in error. **reason** is an integer code for the error, as described in 9.4.

An **unbind** PDU is sent to close a binding in an orderly manner, as described in 9.2. It has no parameters.

A **startTLS** PDU is sent by the TCP/IP initiator to request TLS establishment.

A **tLSResponse** PDU is sent by the TCP/IP responder following receipt of a **startTLS** PDU. A **tLSResponse** of **success** indicates that the responder is willing and able to negotiate TLS. A **tLSResponse** other than **success** indicates that the responder is either unwilling or unable to negotiate TLS. The responder shall return an **operationsError** if it detects any incorrect operations sequencing, such as receipt of a **startTLS** PDU after TLS has already been established. The responder shall return a **protocolError** if it does not support TLS, either by design or current configuration. The responder shall return **unavailable** if it supports TLS but is unable to establish TLS at the time of the **startTLS** request.

9.2 Sequencing requirements

9.2.1 Binding

The initiator of the TCP/IP connection shall send the **bind** PDU to the responder. The responder shall reply by sending either a **bindResponse** or a **bindError** PDU. Once the **bindResponse** PDU has been received, an *association* is said to be in place between the initiator and the responder.

The initiator shall send a **bind** PDU before sending **request** PDUs. It may send **request** PDUs after sending the **bind** PDU but before receiving a **bindResponse** or **bindError**. The responder shall process and reply to a received **bind** PDU before processing and replying to received **request** PDUs.

If the protocol permits the responder to initiate requests, the responder may initiate such requests as soon as it has sent a **bindResponse** PDU. The initiator shall process the **bindResponse** before replying to received **request** PDUs.

If a **bindError** is received, the initiator may choose whether to attempt another bind by sending a new bind PDU or whether to close the TCP/IP connection.

If both application entities use the **AETitle** information of the **bind** PDU, a **bindError** PDU with **aETitleError** set to **callingAETitleNotAccepted** or **calledAETitleNotRecognized** can be received as a response to a **bind** PDU.

9.2.2 Unbinding

When the DAP protocol is being used, only the initiator of the bind shall send an **unbind** PDU. For any other protocol, either the initiator or responder may send an **unbind** PDU. An **unbind** is destructive in that the results of any outstanding operations are lost (undefined). To avoid loss of data, the initiator should only unbind when all requests have been responded to.

Either the initiator or responder may close the underlying TCP/IP connection at any time. Any outstanding requests are lost.

9.2.3 Requests and responses

A **request** PDU may be sent at any time after sending a **bind** PDU or **bindResult** PDU, and requests the recipient of the PDU to perform the indicated operation. The recipient of the **request** PDU shall reply with a **result**, **error**, or **reject** PDU.

Requests are asynchronous and the order of the responses is not guaranteed to be the same as that of the requests.

The receiver of a response shall use the invoke ID as the primary indicator of the request to which the response belongs, and shall reject the response if the invoke ID is in error.

9.2.4 Rejects

The **reject** PDU shall be used to indicate that a problem was encountered in processing a **request**, **result**, or **error** PDU.

If any other protocol error occurs or if the invoke ID cannot be determined, the connection shall be closed.

9.3 Protocols

Protocols for use within the IDM protocol are defined through use of the **IDM-PROTOCOL** information object class, defined as follows:

```
IDM-PROTOCOL ::= CLASS {
        &bind-operation          OPERATION,
        &Operations              OPERATION,
        &id                      OBJECT IDENTIFIER UNIQUE }
```

WITH SYNTAX {
BIND-OPERATION **&bind-operation**
OPERATIONS **&Operations**
ID **&id }**

Each instance of an **IDM-PROTOCOL** class defines the Bind operation and request/response operations for use within the IDM protocol. The **bindOperation** field defines the operation to be used for binding; the **ARGUMENT** field of this operation is used with the **bind** PDU that signals the protocol, the **RESULT** field is used with the **bindResult** PDU, and one of the errors given in the **ERRORS** field of this operation is used with the **bindError** PDU. The **Operations** field defines the operations that may be used within the **request**, **result** and **error** PDUs of the IDM protocol. The **id** field is the protocol identifier. It also implicitly determines the application context for a Bind operation. As a consequence, a separate **IDM-PROTOCOL** is defined for each required application context.

9.4 Reject reasons

A **reject** PDU is returned in response to various error conditions. The error conditions and the reason code with which they are signalled are described below:

A **mistypedPDU** reason is returned if the PDU is invalidly constructed.

A **duplicateInvokeIDRequest** reason is returned if a **request** PDU is received and the **invokeID** has previously been used since the connection was established.

An **unsupportedOperationRequest** reason is returned if a **request** PDU is received and the requested operation is not supported.

An **unknownOperationRequest** reason is returned if a **request** PDU is received and the requested operation is unknown.

A **mistypedArgumentRequest** reason is returned if a **request** PDU is received and the **argument** is invalidly constructed.

A **resourceLimitationRequest** reason is returned if a **request** PDU is received and no operations can be performed because of resource limitations.

An **unknownInvokeIDResult** reason is returned if a **result** PDU is received and the **invokeID** does not match that of an operation to which a response is expected.

A **mistypedResultRequest** reason is returned if a **result** PDU is received and the **result** is invalidly constructed, or the **opcode** does not match that of the corresponding **request** PDU.

An **unknownInvokeIDError** reason is returned if an **error** PDU is received and the **invokeID** does not match that of an operation to which a response is expected.

An **unknownError** reason is returned if an **error** PDU is received and the indicated **error** does not belong to the indicated protocol or is not permitted as a response to the operation.

A **mistypedParameterError** reason is returned if an **error** PDU is received and the **parameter** is invalidly constructed, or the **opcode** does not match that of the corresponding **request** PDU.

9.5 Abort reasons

An **Abort** PDU is returned in response to various error conditions which are not covered by the **Reject** nor the **BindError** PDUs. The error conditions and the reason code with which they are signalled are described below:

A **mistypedPDU** reason is returned if the PDU received has an invalid construction.

An **unboundRequest** reason is returned if a **request** PDU request is received before an association has been established.

An **invalidPDU** reason is returned if a DSA gets a PDU which is not a IDM-PDU.

A **resourceLimitation** reason is returned if a **Bind** PDU is received and no operations can be performed because of resource limitations, e.g., maximum number of connections exceeded.

A **connectionFailed** reason is returned if the DSA was not able to create the TCP/IP connection in order to send a **Bind** PDU.

An **invalidProtocol** reason is returned if a **resultBind**, a **BindResult** or a **BindError** PDU is received and the **protocolID** is unknown or not supported.

A **reasonNotSpecified** reason is returned if the initiator or the responder wants to close the association for any other reason.

NOTE – An abort may be generated by the underlying service of the initiator, resulting in protocol that will not flow across the connection, e.g., returning an abort with **unboundRequest** would be initiated by the underlying service as opposed to the target system which cannot be reached.

9.6 Mapping onto TCP/IP

Each IDM-PDU is encoded using the ASN.1 Basic Encoding Rules without restriction. The binary data resulting from the encoding is then partitioned and placed in one or more segments to be sent over the TCP/IP connection. Each segment has a *header* and carries the next *fragment* or portion of the encoded data. The division of an IDM-PDU into fragments and the size of any fragment are at the choice of the sender and carry no significance. All fragments of an IDM-PDU shall be sent before another IDM-PDU is sent.

The format for a segment (header plus fragment of an IDM-PDU) is as follows:

version (1 octet)	final (1 octet)	length (4 octets)	data (*length* octets)

`version` indicates the version of the IDM-PDU and its mapping onto TCP/IP. The version described in this Directory Specification shall be indicated with the value 1. All packets on a connection shall have the same value of `version`.

NOTE – How the communicating parties negotiate the version number is for further study.

`final` indicates whether `data` holds a non-final IDM-PDU fragment (value 0), or the whole value or final fragment (value 1).

`length` is the length of `data` field in octets. It is sent in 'network octet order' with more significant octets preceding less significant octets. The minimum value of length is 1. For performance reasons, it is recommended that the whole IDM-PDU be contained in one segment if the length can be expressed in the 4 octets of the length field; IDM fragmentation should only be used if the length of the IDM-PDU cannot be expressed in 4 octets.

`data` holds the next fragment of the IDM-PDU being conveyed, or the whole IDM-PDU if the whole value is conveyed in one fragment.

9.7 Addressing

An IDM-style communications endpoint is defined by its IP address and its port number, and can be written in the notation of IETF RFC 1738 as:

```
idm://host:port
```

An OSI Network address format for an IDM access point is specified in 11.3.2. Alternatively, the URI format as specified in 11.4 may be used.

9.8 Use of TLS

9.8.1 TLS establishment

The initiator of the TCP/IP connection may at any time request the establishment of TLS by sending a **StartTLS** PDU. The initiator shall not send any PDUs following this request until it has received a **TLSResponse** PDU.

9.8.2 TLS closure

Two forms of TLS closure are supported: graceful and abrupt.

9.8.2.1 Graceful closure

Either the TCP/IP initiator or responder may terminate the TLS connection by sending a TLS closure alert. Upon sending this alert, it shall cease sending any further TLS Record Protocol PDUs and shall ignore any received TLS Record Layer PDUs until it receives a TLS closure alert from the other party. Once it has received the TLS closure alert, it may continue to send and receive IDM PDUs.

Upon receipt of a TLS closure alert that it did not solicit, a party may choose whether to leave the underlying TCP/IP connection intact. If it chooses to leave the connection intact, it shall immediately respond with a TLS closure alert, after which it may send and receive IDM PDUs. After a TLS connection has been closed, a DSA shall not respond to any requests that were received prior to closure of the TLS connection.

Either party may choose to drop the underlying TCP/IP connection after sending or receiving a TLS closure alert.

9.8.2.2 Abrupt closure

Either the TCP/IP initiator or responder may abruptly close a TLS connection by closing the underlying TCP/IP connection.

10 Directory protocol mapping onto the IDM protocol

This clause gives definitions for mapping the Directory protocols onto the IDM protocol. The complete **DirectoryIDMProtocols** module is given in Annex E. The components are repeated in this clause for clarity.

10.1 DAP-IP protocol

The DAP-IP protocol **dap-ip** (Directory Access Protocol over TCP/IP) is used to invoke operations of the **DirectoryAbstractService** abstract service. It is defined as:

```
DAP-IDM-PDUs  ::=  IDM-PDU {dap-ip}

dap-ip IDM-PROTOCOL ::= {
        BIND-OPERATION          directoryBind
        OPERATIONS              { read | compare | abandon | list | search
                                | addEntry | removeEntry | modifyEntry | modifyDN }
        ID                      id-idm-dap }
```

The operation and error codes for this protocol are the same as those given in 6.4.1 and 6.5.1.

Only DUAs shall initiate connections using this protocol. Only the initiator of a connection shall request operations of the protocol.

10.2 DSP-IP protocol

The DSP-IP protocol **dsp-ip** (Directory System Protocol over TCP/IP) is used to invoke operations of the **DistributedOperations** abstract service. It is defined as:

```
DSP-IDM-PDUs  ::=  IDM-PDU {dsp-ip}

dsp-ip IDM-PROTOCOL  ::=  {
        BIND-OPERATION          directoryBind
        OPERATIONS              { chainedRead | chainedCompare | chainedAbandon
                                | chainedList | chainedSearch
                                | chainedAddEntry | chainedRemoveEntry
                                | chainedModifyEntry | chainedModifyDN }
        ID                      id-idm-dsp }
```

The operation and error codes for this protocol are the same as those given in 6.4.1 and 6.5.1.

DSAs may use this protocol, and both the initiator and the acceptor of a connection may request operations of the protocol.

10.3 DISP-IP protocol

The DISP-IP protocol **disp-ip** (Directory Information Shadowing Protocol over TCP/IP) is used to invoke operations of the **DirectoryShadowAbstractService** abstract service. It is defined as:

```
DISP-IDM-PDUs  ::=  IDM-PDU {disp-ip}

disp-ip IDM-PROTOCOL  ::=  {
        BIND-OPERATION          directoryBind
        OPERATIONS              { requestShadowUpdate
                                | updateShadow
                                | coordinateShadowUpdate }
        ID                      id-idm-disp }
```

The operation and error codes for this protocol are the same as those given in 6.4.2 and 6.5.2.

DSAs may use this protocol, and both the initiator and the acceptor of a connection may request operations of the protocol.

10.4 DOP-IP protocol

The DOP-IP protocol **dop-ip** (Directory Operational Binding Protocol over TCP/IP) is used to invoke operations of the **OperationalBindingManagement** abstract service. It is defined as:

```
DOP-IDM-PDUs  ::=  IDM-PDU {dop-ip}

dop-ip IDM-PROTOCOL  ::=  {
        BIND-OPERATION          directoryBind
        OPERATIONS              { establishOperationalBinding
                                | modifyOperationalBinding
                                | terminateOperationalBinding}
        ID                      id-idm-dop }
```

The operation and error codes for this protocol are the same as those given in 6.4.3 and 6.5.3.

DSAs may use this protocol, and both the initiator and the acceptor of a connection may request operations of the protocol.

11 Protocol stack coexistence

Subclause 9.7 defined an OSI network address format for an IDM communications endpoint. This clause recommends an approach for coexistence between DSAs supporting different protocol stacks, such as OSI, IDM and LDAP. In order to allow referrals to contain LDAP access points, this clause also specifies an OSI network address format for an LDAP communications endpoint. In order to allow referrals to contain Uniform Resource Identifiers (URIs) this clause also defines a NSAP address like format that does not have the length restriction imposed on NSAP addresses (see 11.4).

11.1 Coexistence between OSI and IDM stacks

A conformant implementation shall implement either the OSI stack as defined by clauses 7 and 8, the IDM stack as defined by clauses 9 and 10, or both.

If a chaining DSA needs to forward a request to a target DSA and if the two DSAs do not support a protocol stack in common, then the chaining DSA shall return instead a referral. That referral will be returned through each DSA that chained the request. If any one of these DSAs supports the target DSA's protocol stack, it may choose to send the request directly to the target DSA identified in the referral.

If none of the chaining DSAs support the target DSA's protocol stack, the referral shall be returned to the DUA. That DUA may be able to send the request directly to the target DSA.

If deploying within a domain a mixture of DSA products some of which support only one protocol stack, it is recommended that either:

a) DSAs holding knowledge of DSAs that support only one protocol stack should support that protocol stack; or

b) the DSA to which the DUA binds should support both protocol stacks.

11.2 Coexistence in the presence of LDAP

DSAs supporting either the OSI upper layer protocol stack or the IDM protocol stack may also choose to support LDAP. Interoperability between such DSAs may be accomplished through the use of chaining or referrals. Interoperability between such DSAs and DUAs may be accomplished through the use of LDAP or DAP.

In order for a DSA to be able to provide useful referrals for DUAs supporting only LDAP, it is necessary to represent the LDAP access point of a potential target DSA in an OSI presentation address. Subclause 11.3 defines an NSAP format for LDAP. A DSA getting a referral containing an NSAP of this type can convert it to an LDAP referral and send it back to the connected LDAP client.

11.3 Defining network addresses for Internet Protocol, Version 4 support

Directory addressing format as it is transferred in referrals and cross reference information is in the OSI presentation addressing format as defined in 6.9.1 of ITU-T Rec. X.520 | ISO/IEC 9594-6. For consistency, the same format is used for non-OSI addressing.

Systems that simultaneously support a combination of OSI, OSI over TCP/IP, IDM and LDAP stacks can have a single OSI presentation address containing multiple network addresses for those protocol stacks. If an NSAP address is for an

OSI stack, possibly on top of TCP/IP, **pSelector**, **sSelector** and the **tSelector** components, if present, shall be recognized. Otherwise, they shall be ignored.

The OSI network address (NSAP address) format is specified in ITU-T Rec. X.213 | ISO/IEC 8348. It consists of three parts:

a) the Authority and Format Identifier (AFI), which is a value allocated within ITU-T Rec. X.213 | ISO/IEC 8348 and specifies the IDI format, the authority for allocating IDI values and the DSP format (see below);

b) the Initial Domain Identifier (IDI) identifies the authority for allocating DSP values (see below); and

c) the Domain Specific Part (DSP) holds the actual network address information.

An NSAP address holding IPv4 and possibly a TCP port number is encoded as a sequence of hexadecimal and decimal digits each occupying a semi-octet (4 bits). A hexadecimal digit is encoded in the range from '0000'B to '1111'B. A decimal digit is encoded in the range from '0000'B to '1001'B.

Subclauses 11.3.1-11.3.3 specify NSAP address structures for communication over the Internet Protocol, Version 4 (IPv4) for different types of communication. They all have a common structure:

a) the AFI has the value 54, which according to ITU-T Rec. X.213 | ISO/IEC 8348 is the F.69 format;

b) the IDI is a telex number encoded as 8 decimal digits (4 octets), where the value shall be 00 72 87 22;

c) The DSP has a substructure as follows:

- the first octet is a prefix indicating the type of communication over the IPv4;
- the next 6 octets hold the IPv4 address which is encoded according to the 4-component dotted address. It is encoded in 12 decimal digits with three digits per component (without the dots);
- a 5 decimal digit TCP port number, that may be absent if a default value is available; and
- a trailing semi-octet with the value 'F' to pack out the DSP to a full octet if the TCP port number is present.

NOTE – The structures of the IDI and DSP are in accordance with IETF RFC 1277.

11.3.1 Definition of NSAP address for LDAP

An NSAP address for an LDAP access point is encoded as follows:

- the DSP prefix shall have the value '11';
- the TCP port number shall be present; and
- a trailing hex 'F' shall be added.

11.3.2 Definition of NSAP address for IDM over IPv4

An NSAP address for an IDM endpoint is encoded as follows:

- the DSP prefix shall have the value '10';
- the TCP port number shall be present; and
- a trailing hex 'F' shall be added.

11.3.3 Definition of NSAP address for ITOP over IPv4

An NSAP address for an ISO Transport on top of TCP (ITOT) access point is encoded as follows:

- the DSP prefix shall have the value '03';
- the TCP port number may be absent and then defaults to 102; and
- a trailing hex 'F' shall be added if the port number is present.

11.4 Definition of NSAP like address for long addressing information

NOTE 1 – An OSI Network address (NSAP address) is restricted to 20 octets in length, while the **nAddress** component in **PresentationAddress** data type does not have a length restriction (see ITU-T Rec. X.520 | ISO/IEC 9594-6). It is therefore possible to define NSAP address like addresses with no length restriction. Such an address can also be part of an instance of a **PresentationAddress** data type.

NOTE 2 – The format defined in this subclause may be used in all situations where a URI may be expressed and it allows for IPv6 support.

Octets 0-2	AFI = FF	IDI (octets 1-2)
Octets 3-n	DSP=Address information	

An NSAP like address with no inherent length restriction is encoded as follows:

a) the AFI has the value FF, which is a value that will never be allocated by ITU-T Rec. X.213 | ISO/IEC 8348;

b) the following values are defined for IDI:

– 0000: The DSP is a Uniform Resource Identifier (URI) as defined by IETF RFC 3986 for an ITOT access point; and

– 0001: The DSP is a Uniform Resource Identifier (URI) for non-OSI access points (LDAP, IDM, etc.)

NOTE 3 – Other values may be defined in the future.

12 Versions and the rules for extensibility

This clause describes version negotiation rules and rules for extensibility for the OSI-mapped protocols defined in clause 7, and the IDM-mapped protocols defined in clause 10.

The Directory may be distributed and more than two Directory application-entities (AEs) may interoperate to service a request. The Directory AEs may be implemented conforming to different editions of the Directory specification of the Directory service which may or may not be represented by different protocol version numbers. The version number is negotiated to the highest common version number between two directly binding Directory AEs.

NOTE 1 – There are currently two versions of each Directory protocol. The first and second editions are of version 1. Most features added in subsequent editions are also available in version 1. However, some enhanced services and protocols, e.g., signed errors, require that version 2 has been negotiated among all involved parties.

A DUA may issue a request as specified in the latest edition of the Directory specification to which the DUA was implemented. Using the rules of extensibility defined below, that request shall be forwarded to the appropriate DSA that will respond to that request, regardless of the edition of the intervening DSAs. The responding DSA shall function as defined below.

NOTE 2 – An intermediate DSA only chaining the request may choose to examine selected elements of the Directory PDU that is needed to perform its function, e.g., name resolution.

12.1 DUA to DSA

12.1.1 Version negotiation

When accepting an association, i.e., binding, utilizing the DAP, the version negotiated shall only affect the point-to-point aspects of the protocol exchanged between the DUA and the DSA to which it is connected. Subsequent requests or responses on the association shall not be constrained by the version negotiated.

NOTE – There are no point-to-point aspects of the DAP that are currently indicated by different protocol versions.

12.1.2 Request and response processing

The DUA may initiate requests using the highest edition of the specification of that request it supports. If one or more elements of the request are critical, it shall indicate the extension number(s) in the **criticalExtensions** parameter.

NOTE 1 – If a value defined by an extension is encoded in a **CHOICE**, **ENUMERATED**, or **INTEGER** (used as **ENUMERATED**) type and if that type is essential for proper operation in a DSA implemented according to an earlier edition of these Directory Specifications, it is recommended that the extension be marked critical.

When processing a request from a DUA, a DSA shall follow the rules defined in 12.2.2.

When processing a response, a DUA shall:

a) ignore all unknown bit name assignments within a bit string;

b) ignore all unknown named numbers in an **ENUMERATED** type or **INTEGER** type that is being used in the enumerated style, provided the number occurs as an optional element of a **SET** or **SEQUENCE**;

c) ignore all unknown elements in **SET**s, at the end of **SEQUENCE**s, or in **CHOICE**s where the **CHOICE** is itself an optional element of a **SET** or **SEQUENCE**;

NOTE 2 – Implementations may as a local option ignore certain additional elements in a Directory PDU. In particular, some unknown named numbers and unknown **CHOICE**s in mandatory elements of **SET**s and **SEQUENCE**s can be ignored without invalidating the operation. The identification of such elements is for further study.

d) not consider the receipt of unknown attribute types and attribute values as a protocol violation; and

e) optionally report the unknown attribute types and attribute values to the user.

12.1.3 Extensibility rules for error handling

When processing a known error type with unknown indicated problems and parameters, a DUA shall:

a) not consider the receipt of unknown indicated problems and parameters as a protocol violation (i.e., it shall not issue a **OsiReject** or a **Reject**, as appropriate, or abort the application association); and

b) optionally report the additional error information to the user.

When processing an unknown error type, a DUA shall:

a) not consider the receipt of unknown error type as a protocol violation (i.e., it shall not issue a **OsiReject** or an **Reject**, as appropriate, or abort the application association); and

b) optionally report the error to the user.

12.2 DSA to DSA

12.2.1 Version negotiation

When establishing or accepting an association, i.e., binding, utilizing the DSP, the version negotiated shall only affect the point-to-point aspects of the protocol exchanged between the DSAs. Subsequent requests or responses on the association shall not be constrained by the version negotiated.

NOTE 1 – There are no point-to-point aspects of the DSP that are currently indicated by different protocol versions.

When establishing or accepting an association, i.e., binding, utilizing the DISP, the version negotiated shall define all aspects of the protocol exchanged between the DSAs. Subsequent requests or responses on the association shall be constrained by the version negotiated.

NOTE 2 – There is currently only one version of the DISP protocol.

When establishing or accepting an association, i.e., binding, utilizing the DOP, the version negotiated shall define all aspects of the protocol exchanged between the DSAs. Subsequent requests or responses on the association shall be constrained by the version negotiated.

NOTE 3 – There is currently only one version of the DOP protocol.

12.2.2 Rules of extensibility for operation processing

If any DSA performing an operation (after name resolution is completed) detects an element of **criticalExtensions** whose semantic is unknown, it shall return an **unavailableCriticalExtension** indication as a **serviceError** or in a **PartialOutcomeQualifier**.

NOTE – If a **criticalExtensions** string with one or more zero values is received, this indicates either that the extensions corresponding to the values are not present in the operation or are not critical. The presence of a zero value in a **criticalExtensions** string shall not be inferred as either the presence or absence of the corresponding extension in the Directory PDU.

Otherwise, when processing a Directory PDU a DSA shall:

a) ignore all unknown bit name assignments within a bit string; and

b) ignore all unknown named numbers in an **ENUMERATED** type or **INTEGER** type that is being used in the enumerated style, provided the number occurs as an optional element of a **SET** or **SEQUENCE**; and

c) ignore all unknown elements in **SET**s, at the end of **SEQUENCE**s, or in **CHOICE**s where the **CHOICE** is itself an optional element of a **SET** or **SEQUENCE**.

12.2.3 Rules of extensibility for chaining

If the PDU is a request, the DSA shall forward the request containing the unknown types and values to any additional DSAs determined by the name resolution process.

If the PDU is a response, the DSA shall process the unknown types and values as it would process known types and values (see clause on results merging in the Directory Specification on Distributed Operations) and forward to the initiating DSA or DUA.

A DSA implementing fifth edition or subsequent editions acting as an intermediate DSA that is only chaining a request shall forward a request with an unknown operation. A pre-fifth DSA may optionally forward a request containing an unknown operation.

12.2.4 Rules of extensibility for error handling

When processing a known error type with unknown indicated problems and parameters, a DSA:

a) shall not consider the receipt of unknown indicated problems and parameters as a protocol violation (i.e., it shall not issue a **OsiReject**, or an **Reject**, as appropriate, or abort the application association); and

b) may attempt to recover as appropriate to its understanding of just the error type, or may just return the error (and its unknown indicated problems and parameters) to the next appropriate DSA or DUA.

When processing an unknown error type, a DSA which is only involved in chaining the request shall:

a) not consider the unknown error type as a protocol violation (i.e., it shall not issue a **OsiReject** or an **Reject**, as appropriate, or abort the application association); and

b) not attempt to correct or recover from the error and its indicated problems and parameters; and

c) return the unknown error type to the next appropriate DSA or DUA.

When processing an unknown error, a DSA which is correlating multiple responses shall:

a) not consider the unknown error type as a protocol violation (i.e., it shall not issue a **OsiReject** or an **Reject**, as appropriate, or abort the application association); and

b) not attempt to correct or recover from the error and its indicated problems and parameters; and

c) put the unknown error in **PartialOutcomeQualifier**; and

d) continue correlating results as usual.

12.3 Rules of extensibility for NSAP addresses

A DUA or a DSA may receive a reference that has one or more NSAP addresses with an unknown format. If this is the case, then the DUA or DSA shall:

– not consider this an error;

– not attempt to use an NSAP address with unknown structure; and

– ignore the reference if all contained NSAP addresses have an unknown structure.

12.4 Rules of extensibility for object classes

Optional user attributes may be added to an existing object class without assigning a new object identifier.

A DSA not supporting an object class extension may reject any operation that attempts to create or modify an entry resulting in an extension attribute to be present in the entry.

12.5 Rules of extensibility for user attribute types

A user attribute type definition may be extended in such a way that its matching characteristics are not changed. This may include:

– adding values to **ENUMERATED** and **INTEGER** types that is being used in the enumerated style;

– adding bits to a bitstring.

A DSA is not required to handle an attribute value that includes such extensions.

A DUA shall not consider the receipt of an extended attribute value as an error.

13 Conformance

This clause defines the requirements for conformance to this Directory Specification.

13.1 Conformance by DUAs

A DUA implementation claiming conformance to this Directory Specification shall satisfy the requirements specified in 13.1.1 through 13.1.3.

13.1.1 Statement requirements

The following shall be stated:

a) the operations of the **directoryAccessAC** application-context and/or *dap-ip* protocol that the DUA is capable of invoking for which conformance is claimed;

b) the bind security level(s) for which conformance is claimed (none, simple, strong – and if simple, then whether without password, with password or with protected-password); and whether the DUA can generate signed arguments or validate signed results;

c) the extensions listed Table 1 of ITU-T Rec. X.511 | ISO/IEC 9594-3, that the DUA is capable of initiating for which conformance is claimed;

d) whether conformance is claimed to Rule-based Access Control; and

e) if conformance is claimed for strong authentication, or signed operations, identification of the Certificate and CRL extensions for which conformance is claimed.

13.1.2 Static requirements

A DUA shall:

a) have the capability of supporting the **directoryAccessAC** application-context as defined by its abstract syntax in clause 7; and/or the **dap-ip** protocol defined in clause 10;

b) conform to the extensions for which conformance was claimed in 13.1.1 c);

c) if conformance is claimed to Rule-based Access Control, have the capability of supporting security labels as identified in 19.4 of ITU-T Rec. X.501 | ISO/IEC 9594-2; and

d) conform to clauses 8 and 15 of ITU-T Rec. X.509 | ISO/IEC 9594-8 for the Certificate and CRL extensions for which conformance was claimed in 13.1.1 e).

13.1.3 Dynamic Requirements

A DUA shall:

a) conform to the mapping onto the used service defined in clause 8 or clause 10 or both; and

b) conform to the rules of extensibility procedures defined in 12.1.

13.2 Conformance by DSAs

A DSA implementation claiming conformance to this Directory Specification shall satisfy the requirements specified in 13.2.1 through 13.2.3.

13.2.1 Statement requirements

The following shall be stated:

a) The application-contexts and IDM protocols for which conformance is claimed: **directoryAccessAC**, **directorySystemAC**, **directoryOperationalBindingManagementAC**, **dap-ip**, **dsp-ip**, **dop-ip**, or a combination of these. A DSA that claims conformance to the **directoryOperationalBindingManagementAC** or to the **dop-ip** in support of hierarchical operational bindings shall also support the **directorySystemAC** or **dsp-ip**. If a DSA is such that knowledge of it has been disseminated, causing knowledge references to the DSA to be held in other DSAs outside of its own DMD, then it shall claim conformance to the **directorySystemAC** or **dsp-ip**.

NOTE 1 – An application context shall not be divided except as stated herein; in particular, conformance shall not be claimed to particular operations.

b) The operational binding types for which conformance is claimed: **shadowOperationalBindingID**, **specificHierarchicalBindingID**, **non-specificHierarchicalBindingID**, or a combination of these. A DSA that claims conformance to the **shadowOperationalBindingID** shall support one or more of the application contexts for shadow suppliers and/or shadow consumers indicated in 13.3 and 13.4.

c) Whether or not the DSA is capable of acting as a first level DSA, as defined in ITU-T Rec. X.518 | ISO/IEC 9594-4.

d) If conformance is claimed to the application-context specified by **directorySystemAC** and/or associated with the **dap-ip** protocol, whether or not the chained mode of operation is supported, as defined in ITU-T Rec. X.518 | ISO/IEC 9594-4.

e) If conformance is claimed to the application-context specified by **directoryAccessAC** and/or associated with the **dap-ip** protocol, the bind security level(s) for which conformance is claimed (none, simple,

strong – and if simple, then whether without password, with password, or with protected password); whether the DSA can perform originator authentication as defined in 22.1 of ITU-T Rec. X.518 | ISO/IEC 9594-4 and if so, whether identity-based or signature-based; and whether the DSA can perform result authentication as defined in 22.2 of ITU-T Rec. X.518 | ISO/IEC 9594-4.

f) If conformance is claimed to the application-context specified by **directorySystemAC** and/or associated with the **dsp-ip** protocol, the bind security level(s) for which conformance is claimed (none, simple, strong – and if simple, then whether without password, with password, or with protected password); whether the DSA can perform originator authentication as defined in 22.1 of ITU-T Rec. X.518 | ISO/IEC 9594-4 and if so, whether identity-based or signature-based; and whether the DSA can perform result authentication as defined in 22.2 of ITU-T Rec. X.518 | ISO/IEC 9594-4.

g) The selected attribute types defined in ITU-T Rec. X.520 | ISO/IEC 9594-6, and any other attribute types, for which conformance is claimed and whether for attributes based on the syntax **DirectoryString**, conformance is claimed for the **UniversalString**, **BMPString**, or **UTF8String** choices.

h) The selected object classes defined in ITU-T Rec. X.521 | ISO/IEC 9594-7, and any other object classes, for which conformance is claimed.

i) The extensions listed in Table 1 of ITU-T Rec. X.511 | ISO/IEC 9594-3, that the DSA is capable of responding to for which conformance is claimed.

j) Whether conformance is claimed for collective attributes as defined in 8.9 of ITU-T Rec. X.501 | ISO/IEC 9594-2 and 7.6, 7.8.2 and 9.2.2 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

k) Whether conformance is claimed for hierarchical attributes as defined in 7.6, 7.8.2 and 9.2.2 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

l) The operational attribute types defined in ITU-T Rec. X.501 | ISO/IEC 9594-2 and any other operational attribute types for which conformance is claimed.

m) Whether conformance is claimed for return of alias names as described in 7.7.1 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

n) Whether conformance is claimed for indicating that returned entry information is complete, as described in 7.7.1 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

o) Whether conformance is claimed for modifying the object class attribute to add and/or remove values identifying auxiliary object classes, as described in 11.3.2 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

p) Whether conformance is claimed to Basic Access Control.

q) Whether conformance is claimed to Simplified Access Control.

r) Whether the DSA is capable of administering the subschema for its portion of the DIT, as defined in ITU-T Rec. X.501 | ISO/IEC 9594-2.

NOTE 2 – The capability to administer a subschema shall not be divided; specifically, the capability to administer particular subschema definitions shall not be claimed.

s) The selected name bindings defined in ITU-T Rec. X.521 | ISO/IEC 9594-7 and any other name bindings, for which conformance is claimed.

t) Whether the DSA is capable of administering collective attributes, as defined in ITU-T Rec. X.501 | ISO/IEC 9594-2.

u) The selected context types defined in ITU-T Rec. X.520 | ISO/IEC 9594-6, and any other context types, for which conformance is claimed.

v) Whether conformance is claimed for contexts as defined in 8.8, 8.9 and 12.8 of ITU-T Rec. X.501 | ISO/IEC 9594-2, and in 7.3 and 7.6 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

w) Whether conformance is claimed for the use of contexts in RDNs, as defined in 8.5 and 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2, 7.7 of ITU-T Rec. X.511 | ISO/IEC 9594-3, and ITU-T Rec. X.518 | ISO/IEC 9594-4.

x) Whether conformance is claimed for the management of the DSA Information Tree, as defined in 7.12 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

y) Whether conformance is claimed for the use of systems management for administration of the Directory, as defined in ITU-T Rec. X.530 | ISO/IEC 9594-10.

z) The selected managed objects and management attribute types defined in ITU-T Rec. X.530 | ISO/IEC 9594-10, and any other managed objects and attributes, for which conformance is claimed.

aa) Whether conformance is claimed to Rule-based Access Control.

NOTE 3 – The support of security labels requires the following minimal support of contexts: Context lists as per 8.8 of ITU-T Rec. X.501 | ISO/IEC 9594-2 and **returnContexts** per 7.6 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

bb) Whether conformance is claimed to integrity of Directory operations.

cc) Whether conformance is claimed that the DSA can hold and provide access to encrypted and digitally signed information.

dd) If conformance is claimed for strong authentication, signed operations, or protected operations, identification of the Certificate and CRL extensions for which conformance is claimed.

13.2.2 Static requirements

A DSA shall:

a) have the capability of supporting the application-contexts whose abstract syntaxes are defined in clause 7, and the IDM protocols defined in clause 10, for which conformance is claimed;

b) have the capability of supporting the information framework defined by its abstract syntax in ITU-T Rec. X.501 | ISO/IEC 9594-2;

c) conform to the minimal knowledge requirements defined in ITU-T Rec. X.518 | ISO/IEC 9594-4;

d) if conformance is claimed as a first-level DSA, conform to the requirements support of the root naming context, as defined in ITU-T Rec. X.501 | ISO/IEC 9594-2;

e) have the capability of supporting the attribute types for which conformance is claimed, as defined by their abstract syntaxes;

f) have the capability of supporting the object classes for which conformance is claimed, as defined by their abstract syntaxes;

g) conform to the extensions for which conformance was claimed in 13.2.1 i);

h) if the capability to administer subschema as defined in ITU-T Rec. X.501 | ISO/IEC 9594-2 is claimed, the DSA shall be able to do this administration;

i) if conformance is claimed for collective attributes, have the capability of performing the related procedures defined in 7.6, 7.8.2 and 9.2.2 of ITU-T Rec. X.511 | ISO/IEC 9594-3;

j) if conformance is claimed for hierarchical attributes, have the capability of performing the related procedures defined in 7.6, 7.8.2 and 9.2.2 of ITU-T Rec. X.511 | ISO/IEC 9594-3;

k) have the capability of supporting the operational attribute types for which conformance is claimed;

l) if conformance is claimed to Basic Access Control, have the capability of holding ACI items that conform to the definitions of Basic Access Control;

m) if conformance is claimed to Simplified Access Control, have the capability of holding ACI items that conform to the definitions of Simplified Access Control;

n) have the capability of supporting the context types for which conformance is claimed, as defined by their abstract syntaxes;

o) if conformance is claimed for contexts, have the capability of performing the related procedures defined in ITU-T Rec. X.511 | ISO/IEC 9594-3;

p) if conformance is claimed for the use of contexts in RDNs, have the capability of performing the related procedures as defined in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2, 7.7 of ITU-T Rec. X.511 | ISO/IEC 9594-3, and ITU-T Rec. X.518 | ISO/IEC 9594-4;

q) if conformance is claimed for the management of the DSA Information Tree, have the capability of performing the related procedures as defined in 7.5 and 7.12 of ITU-T Rec. X.511 | ISO/IEC 9594-3;

r) if conformance is claimed for the support of the families of entries feature, have the capabilities as defined in 7.3.2, 7.6.4 and 7.8.3 of ITU-T Rec. X.511 | ISO/IEC 9594-3;

s) if conformance is claimed to the search relaxation feature, have the capabilities as defined in 13.6.2 of ITU-T Rec. X.501 | ISO/IEC 9594-2 and in 10.2.2 of ITU-T Rec. X.511 | ISO/IEC 9594-3. In particular an implementation shall specify:

- whether it supports the inclusion of the **RelaxationPolicy** construct in a search request;
- whether it supports mapping-based matching, matching rule substitution, or both; and
- if it supports mapping-based matching, what mappings are supported;

t) if conformance is claimed to the hierarchical group feature, have the capabilities as defined in 7.5 of ITU-T Rec. X.511 | ISO/IEC 9594-3;

in addition, the implementation shall declare:

- what hierarchy options are supported;

u) if conformance is claimed to basic administration of services, have the capabilities as defined in clause 16 of ITU-T Rec. X.501 | ISO/IEC 9594-2, and the basic checking procedures as defined in clause 13 of ITU-T Rec. X.511 | ISO/IEC 9594-3. This support includes:

- support for entry count;
- support of the service controls options **entryCount** and **performExactly**;
- support of the **notification** extension defined in 7.4 of ITU-T Rec. X.511 | ISO/IEC 9594-3;

in addition, the implementation shall declare whether it supports:

- service-specific administrative points different from autonomous administrative points;
- the context feature within search-rules;
- the families of entries facility within search-rules, which also requires general conformance to that feature;
- the search relaxation feature within search-rules detailed as above in s), which also requires that the implementation claims general conformance to the search relaxation feature;
- hierarchical groups within search-rules;

v) if conformance is claimed for the use of systems management for administration of the Directory, have the capability of performing the related procedures as defined in ITU-T Rec. X.530 | ISO/IEC 9594-10 for the managed objects for which conformance is claimed;

w) if conformance is claimed to Rule-Based Access Control, have the capability of holding ACI items that conform to the definition of Rule-Based Access Control;

x) if conformance is claimed to integrity of Directory operations, be capable of signing all Directory operations supported;

y) if conformance is claimed to integrity of directory information in storage be capable of supporting the **attributeValueIntegrityInfoContext** to protect directory information;

z) conform to clause 8 of ITU-T Rec. X.509 | ISO/IEC 9594-8 for the Certificate and CRL extensions for which conformance was claimed in 13.2.1 dd).

13.2.3 Dynamic requirements

A DSA shall:

a) if claiming conformance to any application-contexts defined in 8.2.2, 8.2.3 and 8.2.4, conform to the mapping onto used OSI services defined in clause 8;

b) conform to the procedures for distributed operation of the Directory related to referrals, as defined in ITU-T Rec. X.518 | ISO/IEC 9594-4;

c) if conformance is claimed to the application-context specified by **directoryAccessAC** and/or associated with the **dap-ip** protocol, conform to the procedures of ITU-T Rec. X.518 | ISO/IEC 9594-4 as they relate to the referral mode of the DAP;

d) if conformance is claimed to the application-context specified by **directorySystemAC** and/or associated with the **dsp-ip** protocol, conform to the referral mode of interaction, as defined in ITU-T Rec. X.518 | ISO/IEC 9594-4;

e) if conformance is claimed to the chained mode of interaction, conform to the chained mode of interaction, as defined in ITU-T Rec. X.518 | ISO/IEC 9594-4;

NOTE – Only in this case is it necessary for a DSA to be capable of invoking operations of the **directorySystemAC** and/or **dsp-ip**.

f) conform to the rules of extensibility procedures defined in 12.2;

g) if conformance is claimed to Basic Access Control, have the capability of protecting information within the DSA in accordance with the procedures of Basic Access Control;

h) if conformance is claimed to Simplified Access Control, have the capability of protecting information within the DSA in accordance with the procedures of Simplified Access Control;

i) if conformance is claimed for the **shadowOperationalBindingID**, conform to the procedures of ITU-T Rec. X.525 | ISO/IEC 9594-9 and ITU-T Rec. X.501 | ISO/IEC 9594-2 as they relate to the DOP;

j) if conformance is claimed for the **specificHierarchicalBindingID**, conform to the procedures of ITU-T Rec. X.518 | ISO/IEC 9594-4 and ITU-T Rec. X.501 | ISO/IEC 9594-2 as they relate to specific hierarchical operational bindings;

k) if conformance is claimed for the **non-specificHierarchicalBindingID**, conform to the procedures of ITU-T Rec. X.518 | ISO/IEC 9594-4 and ITU-T Rec. X.501 | ISO/IEC 9594-2 as they relate to non-specific hierarchical operational bindings;

l) if conformance is claimed for the use of contexts in RDNs, conform to name resolution involving contexts as defined in 9.4 of ITU-T Rec. X.501 | ISO/IEC 9594-2, and 10.3, 10.4, 10.6, 10.10, 10.11 and 15.5.4 of ITU-T Rec. X.518 | ISO/IEC 9594-4;

m) if conformance is claimed to Rule-based Access Control, have the capability of protecting information within the DSA in accordance with the procedures of Rule-based Access Control;

n) if conformance is claimed to basic administration of services, have the capability of handling the search-rules as specified in 19.3.2 of ITU-T Rec. X.518 | ISO/IEC 9594-4.

13.3 Conformance by a shadow supplier

A DSA implementation claiming conformance to this Directory Specification in the role of shadow supplier shall satisfy the requirements specified in 13.3.1 through 13.3.3.

13.3.1 Statement requirements

The following shall be stated:

a) The application context(s) for which conformance is claimed as a shadow supplier: **shadowSupplierInitiatedAC**, **shadowConsumerInitiatedAC**, **shadowSupplierInitiatedAsynchronousAC**, **shadowConsumerInitiatedAsynchronousAC**, and **disp-ip**.

A DSA implementation claiming conformance as a shadow supplier and not supporting **disp-ip** shall, at a minimum, support either the **shadowSupplierInitiatedAC** or the **shadowConsumerInitiatedAC**. If the DSA supports the **shadowSupplierInitiatedAC**, it may optionally support the **shadowSupplierInitiatedAsynchronousAC**. If the DSA supports the **shadowConsumerInitiatedAC**, it may optionally support the **shadowConsumerInitiatedAsynchronousAC**. If claiming conformance to **disp-ip**, it shall be stated whether the implementation is capable of invoking the **requestShadowUpdate** operation, responding to a **coordinateShadowUpdate**, or both.

b) The security-level(s) for which conformance is claimed (none, simple, strong).

c) To which degree the **UnitOfReplication** is supported. Specifically, which (if any) of the following optional features are supported:

- entry filtering on **objectClass**;
- selection/Exclusion of attributes via **AttributeSelection**;
- the inclusion of subordinate knowledge in the replicated area;
- the inclusion of extended knowledge in addition to subordinate knowledge;
- selection/Exclusion of attribute values based on contexts.

13.3.2 Static requirements

A DSA shall:

a) have the capability of supporting the application-contexts whose abstract syntaxes are defined in clause 7, and the IDM protocols defined in clause 10, for which conformance is claimed;

b) provide support for **modifyTimestamp** and **createTimestamp** operational attributes.

13.3.3 Dynamic requirements

A DSA shall:

a) if claiming conformance to any application-contexts defined in 8.2.3, conform to the mapping onto used OSI services defined in clause 8;

b) conform to the procedures of ITU-T Rec. X.525 | ISO/IEC 9594-9 as they relate to the DISP.

13.4 Conformance by a shadow consumer

A DSA implementation claiming conformance to this Directory Specification as a shadow consumer shall satisfy the requirements specified in 13.4.1 through 13.4.3.

13.4.1 Statement requirements

The following shall be stated:

a) The application context(s) for which conformance is claimed as a shadow consumer: **shadowSupplierInitiatedAC**, **shadowConsumerInitiatedAC**, **shadowSupplierInitiatedAsynchronousAC**, **shadowConsumerInitiatedAsynchronousAC**, and **disp-ip**.

   A DSA implementation claiming conformance as a shadow consumer and not supporting **disp-ip** shall, at a minimum, support either the **shadowSupplierInitiatedAC** or the **shadowConsumerInitiatedAC**. If the DSA supports the **shadowSupplierInitiatedAC**, it may optionally support the **shadowSupplierInitiatedAsynchronousAC**. If the DSA supports the **shadowConsumerInitiatedAC** it may optionally support the **shadowConsumerInitiatedAsynchronousAC**. If claiming conformance to **disp-ip**, it shall be stated whether the implementation is capable of responding to the **requestShadowUpdate** operation, requesting a **coordinateShadowUpdate**, or both;

b) The security-level(s) for which conformance is claimed (none, simple, strong);

c) Whether the DSA can act as a secondary shadow supplier (i.e., participate in secondary shadowing as an intermediate DSA);

d) Whether the DSA supports shadowing of overlapping units of replication.

13.4.2 Static requirements

A DSA shall:

a) have the capability of supporting the application-contexts whose abstract syntaxes are defined in clause 7, and the IDM protocols defined in clause 10, for which conformance is claimed;

b) provide support for **modifyTimestamp** and **createTimestamp** operational attributes if overlapping units of replication is supported;

c) provide support for the **copyShallDo** service control.

13.4.3 Dynamic requirements

A DSA shall:

a) if claiming conformance to any application-contexts, conform to the mapping onto used OSI services defined in clause 8;

b) conform to the procedures of ITU-T Rec. X.525 | ISO/IEC 9594-9 as they relate to the DISP.

Annex A

Common protocol specifications in ASN.1

(This annex forms an integral part of this Recommendation | International Standard)

```
CommonProtocolSpecification {joint-iso-itu-t ds(5) module (1) commonProtocolSpecification (35) 6}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --
-- The types and values defined in this module are exported for use in the
-- other ASN.1 modules contained within the Directory Specifications, and for
-- the use of other applications which will use them to access Directory
-- services. Other applications may use them for their own purposes, but this
-- will not constrain extensions and modifications needed to maintain or
-- improve the Directory service.

IMPORTS

-- from ITU-T Rec. X.501 | ISO/IEC 9594-2

        opBindingManagement
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 6}

        establishOperationalBinding, modifyOperationalBinding, terminateOperationalBinding
                FROM  OperationalBindingManagement opBindingManagement  ;

OPERATION  ::=  CLASS {
        &ArgumentType      OPTIONAL,
        &ResultType        OPTIONAL,
        &Errors            ERROR OPTIONAL,
        &operationCode     Code UNIQUE OPTIONAL }
WITH SYNTAX {
        [ARGUMENT &ArgumentType]
        [RESULT            &ResultType]
        [ERRORS            &Errors]
        [CODE              &operationCode] }

ERROR  ::=  CLASS {
        &ParameterType,
        &errorCode         Code UNIQUE OPTIONAL }
WITH SYNTAX {
        PARAMETER          &ParameterType
        [CODE              &errorCode] }

Code ::=  CHOICE  {
        Local          INTEGER,
        global         OBJECT IDENTIFIER }

InvokeId  ::=  CHOICE {
        present        INTEGER,
        absent         NULL }

--  operation codes for DAP and DSP

id-opcode-read                          Code  ::=     local : 1
id-opcode-compare                       Code  ::=     local : 2
id-opcode-abandon                       Code  ::=     local : 3
id-opcode-list                          Code  ::=     local : 4
id-opcode-search                        Code  ::=     local : 5
id-opcode-addEntry                      Code  ::=     local : 6
id-opcode-removeEntry                   Code  ::=     local : 7
id-opcode-modifyEntry                   Code  ::=     local : 8
id-opcode-modifyDN                      Code  ::=     local : 9

--  operation codes for DISP

id-opcode-requestShadowUpdate           Code  ::=     local : 1
id-opcode-updateShadow                  Code  ::=     local : 2
```