INTERNATIONAL TELECOMMUNICATION UNION

ITU-T X.882

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(07/94)

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

OSI APPLICATIONS – REMOTE OPERATIONS

INFORMATION TECHNOLOGY – REMOTE OPERATIONS: OSI REALIZATIONS – REMOTE OPERATIONS SERVICE ELEMENT (ROSE) PROTOCOL SPECIFICATION

ITU-T Recommendation X.882

(Previously "CCITT Recommendation")

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. Some 179 member countries, 84 telecom operating entities, 145 scientific and industrial organizations and 38 international organizations participate in ITU-T which is the body which sets world telecommunications standards (Recommendations).

The approval of Recommendations by the Members of ITU-T is covered by the procedure laid down in WTSC Resolution No. 1 (Helsinki, 1993). In addition, the World Telecommunication Standardization Conference (WTSC), which meets every four years, approves Recommendations submitted to it and establishes the study programme for the following period.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC. The text of ITU-T Recommendation X.882 was approved on 1st of July 1994. The identical text is also published as ISO/IEC International Standard 13712-3.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

ITU-T X-SERIES RECOMMENDATIONS

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS
(February 1994)

ORGANIZATION OF X-SERIES RECOMMENDATIONS

Subject area	Recommendation series
PUBLIC DATA NETWORKS	
Services and facilities	X.1-X.19
Interfaces	X.20-X.49
Transmission, signalling and switching	X.50-X.89
Network aspects	X.90-X.149
Maintenance	X.150-X.179
Administrative arrangements	X.180-X.199
OPEN SYSTEMS INTERCONNECTION	
Model and notation	X.200-X.209
Service definitions	X.210-X.219
Connection-mode protocol specifications	X.220-X.229
Connectionless-mode protocol specifications	X.230-X.239
PICS proformas	X.240-X.259
Protocol identification	X.260-X.269
Security protocols	X.270-X.279
Layer managed objects	X.280-X.289
Conformance testing	X.290-X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300-X.349
Mobile data transmission systems	X.350-X.369
Management	X.370-X.399
MESSAGE HANDLING SYSTEMS	X.400-X.499
DIRECTORY	X.500-X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600-X.649
Naming, addressing and registration	X.650-X.679
Abstract Syntax Notation One (ASN.1)	X.680-X.699
OSI MANAGEMENT	X.700-X.799
SECURITY	X.800-X.849
OSI APPLICATIONS	
Commitment, concurrency and recovery	X.850-X.859
Transaction processing	X.860-X.879
Remote operations	X.880-X.899
OPEN DISTRIBUTED PROCESSING	X.900-X.999

CONTENTS

Summary

This Recommendation | International Standard describes the behaviour of ROSE itself, and the way the Association Control Service Element (ACSE), the Reliable Transfer Service Element (RTSE) and the presentation layer are employed to transfer the ROSE Protocol Control Information (PCI) in an OSI realization. This Recommendation | International Standard makes no changes to the ROSE PCI as defined in CCITT Recommendation X.229 (1988).

Introduction

Remote operations (ROS) is a paradigm for interactive communication between objects. As such it can be used in the design and specification of distributed applications. The basic interaction involved is the invocation of an operation by one object (the invoker), its performance by another (the performer), possibly followed by a report of the outcome of the operation being returned to the invoker.

The concepts of ROS, as specified in ITU-T Rec. X.880 | ISO/IEC 13712-1, are abstract, and may be realized in many ways. For example, objects whose interactions employ ROS concepts may be separated by a software interface or by an OSI network.

ITU-T Rec. X.881 | ISO/IEC 13712-2 provides the framework for the realization of an association contract as an OSI application context. Such an application context is specified primarily in terms of a collection of application service elements. From a ROS perspective, these ASEs fall into three broad categories:

a) operation-specific ASEs, which embody knowledge of the definitions of the operations in the contract;

b) the Remote Operations ASE (ROSE) which drives the general-purpose protocol required to invoke and report returns of arbitrary operations;

c) information transfer ASEs concerned with the establishment and release of associations where necessary, and the communication of the ROSE protocol information.

This Recommendation | International Standard describes the behaviour of ROSE itself, and the way in which different collections of information transfer ASEs (specifically, the Reliable Transfer Service Element (RTSE) and the Association Control Service Element (ACSE)) are employed to transfer its protocol control information (PCI) in an OSI realization.

This Recommendation | International Standard is a revision of CCITT Rec. X.229 | ISO/IEC 9072-2. The existing usage of ROSE in conjunction with ACSE, RTSE and the Presentation layer as defined in CCITT Rec. X.229 | ISO/IEC 9072-2 remains valid after this revision. In addition, this revision makes no change to the ROSE PCI.

Annex A forms an integral part of this Recommendation | International Standard.

Annex B forms an integral part of this Recommendation | International Standard.

Annex C does not form an integral part of this Recommendation | International Standard.

Annex D does not form an integral part of this Recommendation | International Standard.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – REMOTE OPERATIONS: OSI REALIZATIONS – REMOTE OPERATIONS SERVICE ELEMENT (ROSE) PROTOCOL SPECIFICATION

1 Scope

This Recommendation | International Standard specifies the protocol (abstract syntax) and procedures for the Remote Operation Service Element. The terms, definitions and mechanisms defined in ITU-T Rec. X.880 | ISO/IEC 13712-1 apply here and are specialized for an OSI realization as specified in this Recommendation | International Standard. The ROSE services, defined in ITU-T Rec. X.881 | ISO/IEC 13712-2, are provided in conjunction with the Association Control Service Element (ACSE) services (ITU-T Rec. X.217 | ISO/IEC 8649) and the ACSE protocol (ITU-T Rec. X.227 | ISO/IEC 8650-1), optionally the Reliable Transfer Service Element (RTSE) services (ITU-T Rec. X.218 | ISO/IEC 9066-1) and the RTSE protocol (ITU-T Rec. X.228 | ISO/IEC 9066-2), and the Presentation service (ITU-T Rec. X.216 | ISO/IEC 8822).

The ROSE procedures are defined in terms of:

a) the interactions between peer ROSE protocol machines through the use of RTSE services or the Presentation service;

b) the interactions between the ROSE protocol machine and its service-user.

This Recommendation | International Standard specifies conformance requirements for systems implementing these procedures.

2 Normative references

The following ITU-T Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Specification. 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 Specification are encouraged to investigate the possibility of applying the most recent editions of the Recommendations and Standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunications Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

– ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

– ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*

– ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*

– ITU-T Recommendation X.683 (1994) | ISO/IEC 8824-4:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Parametrization of ASN.1 specifications.*

– ITU-T Recommendation X.690 (1994) | ISO/IEC 8825-1:1995, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

– 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.210 (1993) | ISO/IEC 10731:1993, *Information technology – Open Systems Interconnection – Conventions for the definitions of OSI services.*

- ITU-T Recommendation X.215 (1994) | ISO/IEC 8326:1994, *Information technology – Open Systems Interconnection – Session service definition*.
- ITU-T Recommendation X.216 (1994) | ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition*.
- ITU-T Recommendation X.217 (1995) | ISO/IEC 8649:1995, *Information technology – Open Systems Interconnection – Service definition for the Association Control Service Element*.
- ITU-T Recommendation X.227 (1995) | ISO/IEC 8650-1:1995, *Information technology – Open Systems Interconnection – Connection-oriented protocol for the Association Control Service Element: Protocol specification*.
- ITU-T Recommendation X.880 (1994) | ISO/IEC 13712-1:1995, *Information technology – Remote Operations: Concepts, model and notation*.
- ITU-T Recommendation X.881 (1994) | ISO/IEC 13712-2:1995, *Information technology – Remote Operations: OSI realizations – Remote Operations Service Element (ROSE) service definition*.

2.2 Paired Recommendations | International Standards equivalent in technical content

- ITU-T Recommendation X.218 (1993), *Reliable transfer: Model and service definition*.

  ISO/IEC 9066-1:1989, *Information processing systems – Text communication – Reliable transfer – Part 1: Model and service definition.*
- ITU-T Recommendation X.228 (1988), *Reliable transfer: Protocol specification*.

  ISO/IEC 9066-2:1989, *Information processing systems – Text communication – Reliable transfer – Part 2: Protocol specification.*
- CCITT Recommendation X.219 (1988), *Remote operations: Model, notation and service definition*.

  ISO/IEC 9072-1:1989, *Information processing systems – Text communication – Remote operations – Part 1: Model, notation and service definition.*
- CCITT Recommendation X.229 (1988), *Remote operations: Protocol specification*.

  ISO/IEC 9072-2:1989, *Information processing systems – Text communication – Remote operations – Part 2: Protocol specification*.

2.3 Additional references

- CCITT Recommendation X.410 (1984), *Message handling systems: Remote operations and reliable transfer service*

3 Definitions

3.1 Reference model definitions

This Recommendation | International Standard is based on the concepts developed in ITU-T Rec. X.200 | ISO/IEC 7498-1 and makes use of the following terms defined in it:

a) application layer;

b) application-process;

c) application-entity;

d) application-service-element;

e) application-protocol-data-unit;

f) application-protocol-control-information;

g) presentation-service;

h) presentation-connection;

i) session-service;

j) session-connection; and

k) transfer syntax.

3.2 Service conventions definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.210 | ISO/IEC 10731:

a) service-provider;

b) service-user;

c) confirmed service;

d) non-confirmed service;

e) provider-initiated service;

f) primitive;

g) request (primitive);

h) indication (primitive);

i) response (primitive); and

j) confirm (primitive).

3.3 Presentation service definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.216 | ISO/IEC 8822:

a) abstract syntax;

b) abstract syntax name;

c) presentation context;

d) defined context set.

3.4 Association control definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.217 | ISO/IEC 8649 :

a) application-association; association;

b) application context;

c) association control service element.

3.5 Reliable transfer definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.218 | ISO/IEC 9066-1:

– Reliable Transfer Service Element.

3.6 ROSE service definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.881 | ISO/IEC 13712-2:

a) association-initiating-application-entity; association-initiator;

b) association-responding-application-entity; association-responder;

c) invoking-application-entity; invoker;

d) performing-application-entity; performer;

e) requestor;

f) acceptor;

g) linked-operations;

h) parent-operation;

i) child-operation;

j) remote operation service element;

k) ROSE-provider;

l) ROSE-user;

m) RTSE-user.

3.7 Remote operation protocol specification definitions

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

3.7.1 remote-operation-protocol-machine: The protocol machine for the remote operation service element specified in this Recommendation | International Standard.

3.7.2 requesting-remote-operation-protocol-machine: The remote-operation-protocol-machine whose service-user is the requestor of a particular remote operation service element service.

3.7.3 accepting-remote-operation-protocol-machine: The remote-operation-protocol-machine whose service-user is the acceptor for a particular remote operation service element service.

4 Abbreviations

4.1 Data units

APDU	Application-protocol-data-unit
PCI	Protocol control information
PDV	Presentation data value

4.2 Types of application-protocol-data-units

The following abbreviations have been given to the application-protocol-data-units defined in this Recommendation | International Standard:

Invoke	RO-INVOKE service application-protocol-data-unit
ReturnResult	RO-RESULT service application-protocol-data-unit
ReturnError	RO-ERROR service application-protocol-data-unit
Reject	RO-REJECT-U/P service application-protocol-data-unit

4.3 Other abbreviations

The following abbreviations are used in this Recommendation | International Standard:

AE	Application entity
ACSE	Association control service element
ASE	Application service element
ASN.1	Abstract Syntax Notation One
RO (or ROS)	Remote Operations
ROPM	Remote Operations Protocol Machine
ROSE	Remote Operations Service Element
RT (or RTS)	Reliable Transfer
RTSE	Reliable Transfer Service Element

5 Conventions

This Recommendation | International Standard employs a tabular presentation of the parameters of its pseudo-primitives and for the fields of its APDUs. In clause 7, tables are presented for each pseudo-primitive and each ROSE APDU. Each parameter or field is summarized using the following notation:

blank	Not applicable
M	Presence is mandatory
U	Presence is a ROSE-user option
C	Conditional
req	Source is related request primitive
ind	Sink is related indication primitive
resp	Source is related response primitive
conf	Sink is related confirm primitive
sp	Source or sink is the ROPM

In addition, the notation (=) indicates that a parameter value is semantically equal to the value to its left in the table.

This Recommendation | International Standard employs ASN.1, as specified in ITU-T Rec. X.681 | ISO/IEC 8824-2, to define the **REALIZATION** information object class. It also provides notation by which designers of ROS realizations can specify particular instances of the class.

The structure of each ROSE APDU is specified in ITU-T Rec. X.880 | ISO/IEC 13712-1 using ASN.1.

6 Overview

6.1 Service provision

The protocol specified in this Recommendation | International Standard provides the ROSE services defined in ITU-T Rec. X.881 | ISO/IEC 13712-2. These services are listed in Table 1 reproduced from Table 1 of ITU-T Rec. X.881 | ISO/IEC 13712-2.

Table 1 – ROSE services

Service	Type
RO-INVOKE	Non-confirmed
RO-RESULT	Non-confirmed
RO-ERROR	Non-confirmed
RO-REJECT-U	Non-confirmed
RO-REJECT-P	Provider-initiated
RO-BIND	Confirmed
RO-UNBIND	Confirmed

6.2 Association and transfer services

The ROSE protocol specified herein needs a transfer service to pass information in the form of ROSE APDUs between peer application-entities, and, if a connection package is involved in the association contract, an association service to establish and release associations between the application-entities. These services are provided by use of various ASEs together with the OSI Presentation service.

This specification is structured into the description of a generic protocol (see clause 7), together with a number of specific realizations of the association service (see clause 8) and of the transfer service (see clause 9). The generic protocol is independent of the particular realizations chosen.

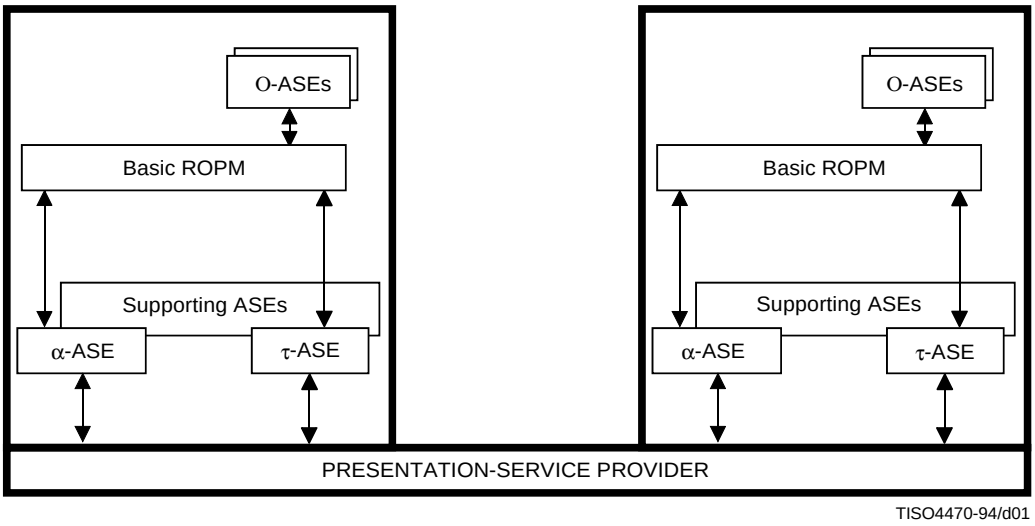
NOTE – It is envisaged that other association and transfer realizations may be defined, both as future extensions to this standard, and on a proprietary basis.

Two specific association realizations are included, one based upon ACSE and one on RTSE. Two specific transfer realizations are included, based respectively on the use of P-DATA and RT-DATA to transfer the APDUs.

6.3 Protocol model

The services of ROSE, as defined in ITU-T Rec. X.881 | ISO/IEC 13712-2, are provided by the Remote Operations Protocol Machine (ROPM). The ROPM uses the services provided by the OSI Presentation service provider, together with a collection of ASEs which shall include an α-ASE, may include a τ-ASE, and may also include ASEs supporting these. The collection always includes ACSE. Different OSI realizations of ROS result from the use of different collections.

This model is depicted in Figure 1.



α-ASE	ASE providing association establishment and release
τ-ASE	ASE providing information transfer
ROPM	Remote Operations protocol machine
O-ASEs	Operation-specific ASEs

Figure 1 – Protocol model

In general, ROSE does not assume that it has exclusive use of the services of the α-ASE, τ-ASE, supporting ASEs, or the Presentation layer. Furthermore, the use of service parameters whose values are unconstrained by the ROSE protocol specification can be specified as appropriate by the application context designer. Any exceptions to this are indicated in the specification of the appropriate realization.

7 Basic ROSE elements of procedure

The basic ROSE protocol consists of the following elements of procedure:

a) association-establishment;

b) association-release;

c) association-abort;

d) invocation;

e) return-result;

f) return-error;

g) user-reject;

h) provider-reject.

In the following subclauses, a specification of each of these elements of procedure is presented. In so doing, a number of pseudo-primitives are used to describe the use of the association and transfer services. Each realization of these services in clauses 8 and 9 describes the actual primitives which are used if that realization is employed.

For the association services, the pseudo-primitives are shown in Table 2.

Table 2 – Pseudo-primitives for association realizations

Assumed service	Pseudo-primitive	req.	ind.	resp.	conf.
Association establishment	*ESTABLISH*				
	application context	M	M(=)		
	release can fail		M	C	C(=)
	user data	U	C(=)	U	C(=)
	result			M	M(=)
Association release	*RELEASE*				
	user data	U	C(=)	U	C(=)
	result			M	M(=)
Association abort-user	*ABORT*				
	source		M		
	user data	U	C(=)		
Association abort-provider	*ABORT-P*				
	provider reason		C		

The result parameter of *ESTABLISH* takes the symbolic values “accepted” and “rejected”.

The release can fail parameter takes the symbolic values “true” and “false”. In successive primitives of this service, the value of the parameter can change from “true” to “false” but not vice versa. This parameter is present on the response or confirm if and only if the result parameter takes the value “accepted”.

The result parameter of *RELEASE* takes the three symbolic values “accepted”, “rejected-released” and “rejected-not-released”.

The source parameter of the *ABORT* takes the symbolic values "association-control service-user" or "association-control-service-provider.

The value of user information parameter depends on the application context in place.

The provider-reason parameter of the *ABORT-P* takes the symbolic values defined in ITU-T Rec. X.216 | ISO/IEC 8822.

For the transfer services, the pseudo-primitives are shown in Table 3.

Table 3 – Assumed primitives for transfer realizations

Assumed service	Pseudo-primitive	req.	ind.
Information transfer	*TRANSFER*		
	user-data	M	M(=)

The use of the components of the various APDUs are described in this clause. In ITU-T Rec. X.880 | ISO/IEC 13712-1, the data types corresponding to these APDUs are specified using ASN.1.

7.1 Association-establishment

7.1.1 Purpose

The attempted establishment of an association through the invocation of a bind operation.

7.1.2 APDUs used

The association-establishment procedure uses the BindInvoke, BindResult, and BindError APDUs. These APDUs are defined if and only if, respectively, the **&ArgumentType**, **&ResultType** and **&ParameterType** fields are defined for the **bind** operation and its associated error used in the **connection package** that is used for dynamic association control (see ITU-T Rec. X.880 | ISO/IEC 13712-1 for a definition of the corresponding information object classes).

7.1.2.1 BindInvoke

The BindInvoke APDU is used in the request to establish an association. The fields of this APDU are listed in Table 4.

Table 4 – BindInvoke APDU fields

Field name	Presence	Source	Sink
Argument	U	req.	ind.

The Argument field is derived from the **&ArgumentType** field of the **bind** operation.

7.1.2.2 BindResult

The BindResult APDU is used to indicate the successful establishment of an association. The fields of this APDU are listed in Table 5.

The Result field is derived from the **&ResultType** field of the **bind** operation.

Table 5 – BindResult APDU fields

Field name	Presence	Source	Sink
Result	U	resp.	conf.

7.1.2.3 BindError

The BindErrorAPDU is used to indicate that the attempt to establish an association was unsuccessful. The fields of this APDU are listed in Table 6.

The Error-Parameter field is derived from the **&ParameterType** field of the error associated with the **bind** operation.

Table 6 – BindError APDU fields

Field name	Presence	Source	Sink
Error-Parameter	U	resp.	conf.

7.1.3 Association-establishment procedure

This procedure is driven by the following events:

a) an RO-BIND request;

b) a BindInvoke APDU as user data on an *ESTABLISH* indication primitive;

c) an RO-BIND response with outcome of “result”;

d) a BindResult APDU as user data on an *ESTABLISH* confirm primitive with result of “accepted”.

e) an RO -BIND response with outcome of “error”;

f) a BindError APDU as user data on an *ESTABLISH* confirm primitive with result of “rejected”.

Sending the BindInvoke APDU or BindResult APDU or BindError APDU is optional when, respectively, the **&argumentTypeOptional** or **&resultTypeOptional** or **&parameterTypeOptional** fields of the **bind** operation and error are set to **TRUE**.

7.1.3.1 RO-BIND request

The requesting ROPM forms a BindInvoke APDU from the Argument parameter of the RO-BIND request, and conveys it in the user-data parameter of an *ESTABLISH* request. The release can fail parameter takes its settings from the **&unbindCanFail** field of the connection package identified by the application context parameter.

7.1.3.2 BindInvoke APDU

The accepting ROPM issues an RO-BIND indication, whose Argument parameter is derived from the BindInvoke APDU.

7.1.3.3 RO-BIND response with outcome of “result”

The accepting ROPM forms a BindResult APDU from the Bind-Result parameter of the RO-BIND response, and conveys it in the user-data parameter of an *ESTABLISH* response whose result parameter takes the value “accepted”. The unbind can fail parameter of the RO-BIND response governs the setting of the release can fail parameter of the *ESTABLISH* response.

7.1.3.4 BindResult APDU

The requesting ROPM issues an RO-BIND confirm, whose Bind-Result parameter is derived from the BindResult APDU.

7.1.3.5 RO-BIND response with outcome of “error”

The accepting ROPM forms a BindError APDU from the Bind-Error parameter of the RO-BIND response, and conveys it in the user-data parameter of an *ESTABLISH* response.

7.1.3.6 BindError APDU

The requesting ROPM issues an RO-BIND confirm, whose Bind-Error parameter is derived from the BindError APDU.

7.2 Association-release

7.2.1 Purpose

The attempted release of an association through the invocation of an **unbind** operation.

7.2.2 APDUs used

The association-release procedure uses the UnbindInvoke, UnbindResult, and UnbindError APDUs. These APDUs are defined if and only if, respectively, the **&ArgumentType**, **&ResultType** and **&ParameterType** fields are defined for the **unbind** operation and its associated error used in the **connection package** that is used for dynamic association control (see ITU-T Rec. X.880 | ISO/IEC 13712-1 for a definition of the corresponding information object classes).

7.2.2.1 UnbindInvoke

The UnbindInvoke APDU is used in the request to release an association. The fields of this APDU are listed in Table 7.

The Argument field is derived from the **&ArgumentType** field of the **unbind** operation.

Table 7 – UnbindInvoke APDU fields

Field name	Presence	Source	Sink
Argument	U	req.	ind.

7.2.2.2 UnbindResult

The UnbindResult APDU is used to indicate the successful release of an association. The fields of this APDU are listed in Table 8.

The Result field is derived from the **&ResultType** field of the **unbind** operation.

Table 8 – UnbindResult APDU fields

Field name	Presence	Source	Sink
Result	U	resp.	conf.

7.2.2.3 UnbindError

The UnbindErrorAPDU is used to indicate that the request to release an association is refused. The fields of this APDU are listed in Table 9.

The Parameter field is derived from the **&ParameterType** field of the error associated with this **unbind** operation.

Table 9 – UnbindError APDU fields

Field name	Presence	Source	Sink
Parameter	U	resp.	conf.

7.2.3 Association-release procedure

This procedure is driven by the following events:

a) an RO-UNBIND request;

b) an UnbindInvoke APDU as user data on an *RELEASE* indication primitive;

c) RO-UNBIND response with outcome of “result”;

d) an UnbindResult APDU as user data on an *RELEASE* confirm primitive with result of "success".

e) RO-UNBIND response with outcome of “error-bound” or “error-unbound”

f) an UnbindError APDU as user data on an *RELEASE* confirm primitive with result of "failure".

Sending the UnbindInvoke APDU or UnbindResult APDU or UnbindError APDU is optional when, respectively, the **&argumentTypeOptional** or **&resultTypeOptional** or **&parameterTypeOptional** fields of the **unbind** operation or its associated error is set to **TRUE**.

7.2.3.1 RO-UNBIND request

The requesting ROPM forms an UnbindInvoke APDU from the Argument parameter of the RO-UNBIND request, and conveys it in the user data parameter of a *RELEASE* request.

7.2.3.2 UnbindInvoke APDU

The accepting ROPM issues an RO-UNBIND indication, whose Argument parameter is derived from the UnbindInvoke APDU.

7.2.3.3 RO-UNBIND response with outcome of “result”

The accepting ROPM forms an UnbindResult APDU from the Unbind-Result parameter of the RO-UNBIND response and conveys it in the user-data parameter of a *RELEASE* response.

7.2.3.4 UnbindResult APDU

The requesting ROPM issues an RO-UNBIND confirm, whose Unbind-Result parameter is derived from the UnbindResult APDU.

7.2.3.5 RO-UNBIND response with outcome of “error-bound” or “error-unbound”

The accepting ROPM forms an UnbindError APDU from the Unbind-Error parameter of the RO-UNBIND response, and conveys it in the user-data parameter of an *RELEASE* response. If the outcome of “error-bound” occurs, the association continues to exist.

7.2.3.6 UnbindError APDU

The requesting ROPM issues an RO-UNBIND confirm with outcome of “error-bound” or “error-unbound”, and whose Unbind-Error parameter is derived from the UnbindError APDU. If the outcome of “error-bound” occurs, the association continues to exist.

7.3 Association-abort

7.3.1 Association-abort purpose

The abnormal release of an association by the association control service user or the association control service provider.

NOTE – This may also occur as a result of an event signalled by the underlying communications infrastructure.

7.3.2 Association-abort-procedures

The association abort procedures is driven by the following events:

a) an ABORT request or indication primitive; or

b) an ABORT-P indication primitive.

Case a) signals the abnormal release of an association by the association service-user or the association-control-service provider. Case b) signals the release of the association because of an abnormal event signalled by the underlying communications infrastructure. The association is released immediately and any PDUs in transit are lost.

7.4 Invocation

7.4.1 Purpose

The invocation procedure is used by one AE (the invoker) to request an operation to be performed by the other AE (the performer).

7.4.2 APDUs used

The invocation procedure uses the Invoke APDU. The fields of the Invoke APDU are listed in Table 10.

Table 10 – Invoke APDU fields

Field name	Presence	Source	Sink
Invoke-id	M	req.	ind.
Linked-id	U	req.	ind.
Operation-id	M	req.	ind.
Argument	U	req.	ind.

7.4.3 Invocation procedure

This procedure is driven by the following events:

a) an RO-INVOKE request primitive from the requestor;

b) an Invoke APDU as user-data of a *TRANSFER* indication primitive.

7.4.3.1 RO-INVOKE request primitive

The requesting ROPM forms an Invoke APDU from the parameter values of the RO-INVOKE request primitive. It issues a *TRANSFER* request primitive. The user-data parameter of the *TRANSFER* request primitive contains the Invoke APDU.

The requesting ROPM waits either for a *TRANSFER* indication primitive from the *TRANSFER* service-provider or any other primitive from the requestor.

7.4.3.2 Invoke APDU

The accepting ROPM receives an Invoke APDU from its peer as user-data on a *TRANSFER* indication primitive. If any of the fields of the Invoke APDU are unacceptable to this ROPM, the provider-reject procedure is performed, and no RO-INVOKE indication primitive is issued by the ROPM.

If the Invoke APDU is acceptable to the accepting ROPM, it issues an RO-INVOKE indication primitive to the acceptor. The RO-INVOKE indication primitive parameters are derived from the Invoke APDU.

The accepting ROPM waits either for a *TRANSFER* indication primitive from the *TRANSFER* service-provider or any other primitive from the acceptor.

7.4.4 Use of the Invoke APDU fields

The fields of the Invoke APDU are used as follows.

7.4.4.1 Invoke-id

This is the Invoke-id parameter value of the RO-INVOKE request primitive. It appears as the Invoke-id parameter value of the RO-INVOKE indication primitive.

The value of this field is transparent to the ROPM and shall not be the value **absent:NULL**; however the value may be used in the provider reject procedure.

7.4.4.2 Linked-id

This is the Linked-id parameter value of the RO-INVOKE request primitive. It appears as the Linked-id parameter value of the RO-INVOKE indication primitive.

The value of this field is transparent to the ROPM and shall not be the value **absent:NULL**.

7.4.4.3 Operation-id

This is the Operation-id parameter value of the RO-INVOKE request primitive. It appears as the Operation-id parameter value of the RO-INVOKE indication primitive.

The value of this field is transparent to the ROPM.

7.4.4.4 Argument

This is the Argument parameter value of the RO-INVOKE request primitive. It appears as the Argument parameter value of the RO-INVOKE indication primitive.

The value of this field is transparent to the ROPM.

7.5 Return-result

7.5.1 Purpose

The return-result procedure is used by one AE (the performer) to request the transfer of the result of a successfully performed operation to the other AE (the invoker).

7.5.2 APDUs used

The return-result procedure uses the ReturnResult APDU.

The fields of the ReturnResult APDU are listed in Table 11.

Table 11 – ReturnResult APDU fields

Field name	Presence	Source	Sink
Invoke-id	M	req.	ind.
Operation-id	C	req.	ind.
Result	U	req.	ind.

7.5.3 Return-result procedure

This procedure is driven by the following events:

a) an RO-RESULT request primitive from the requestor;

b) an ReturnResult APDU are user-data of a *TRANSFER* indication primitive.

7.5.3.1 RO-RESULT request primitive

The requesting ROPM forms an ReturnResult APDU from the parameter values of the RO-RESULT request primitive. It issues a *TRANSFER* request primitive. The user-data parameter of the *TRANSFER* request primitive contains the ReturnResult APDU.

The requesting ROPM waits either for a *TRANSFER* indication primitive from the *TRANSFER* service-provider or any other primitive from the requestor.

7.5.3.2 ReturnResult APDU

The accepting ROPM receives an ReturnResult APDU from its peer as user-data on a *TRANSFER* indication primitive. If any of the fields of the ReturnResult APDU are unacceptable to this ROPM, the provider-reject procedure is performed, and no RO-RESULT indication primitive is issued by the ROPM.

If the ReturnResult APDU is acceptable to the accepting ROPM, it issues an RO-RESULT indication primitive to the acceptor. The RO-RESULT indication primitive parameters are derived from the ReturnResult APDU.

The accepting ROPM waits either for a *TRANSFER* primitive from the *TRANSFER* service-provider or any other primitive from the acceptor.

7.5.4 Use of the ReturnResult APDU fields

The ReturnResult fields are used as follows.

7.5.4.1 Invoke-id

This is the Invoke-id parameter value of the RO-RESULT request primitive. It appears as the Invoke-id parameter value of the RO-RESULT indication primitive.

The value of this field is transparent to the ROPM and shall not be the value **absent:NULL;**, however the value may be used in the provider-reject procedure.

7.5.4.2 Operation-id

This is the Operation-id parameter value of the RO-RESULT request primitive. It appears as the Operation-id parameter value of the RO-RESULT indication primitive.

The value of this field is transparent to the ROPM.

This field shall be present only if the Result field is present.

7.5.4.3 Result

This is the Result parameter value of the RO-RESULT request primitive. It appears as the Result parameter value of the RO-RESULT indication primitive.

The value of this field is transparent to the ROPM.

7.6 Return-error

7.6.1 Purpose

The return-error procedure is used by one AE (the performer) to request the transfer of the error information in the case of an unsuccessfully performed operation to the other AE (the invoker).

7.6.2 APDUs used

The return-error procedure uses the ReturnError APDU.

The fields of the ReturnError APDU are listed in Table 12.

Table 12 – ReturnError APDU fields

Field name	Presence	Source	Sink
Invoke-id	M	req.	ind.
Error-id	M	req.	ind.
Parameter	U	req.	ind.

7.6.3 Return-error procedure

This procedure is driven by the following events:

a) an RO-ERROR request primitive from the requestor;

b) an ReturnError APDU as user-data of a *TRANSFER* indication primitive.

7.6.3.1 RO-ERROR request primitive

The requesting ROPM forms an ReturnError APDU from the parameter values of the RO-ERROR request primitive. It issues a *TRANSFER* request primitive. The user-data parameter of the *TRANSFER* request primitive contains the ReturnError APDU.

The requesting ROPM waits either for a *TRANSFER* primitive from the *TRANSFER* service-provider or any other primitive from the requestor.

7.6.3.2 ReturnError APDU

The accepting ROPM receives an ReturnError APDU from its peer as user-data on a *TRANSFER* indication primitive. If any of the fields of the ReturnError APDU are unacceptable to this ROPM, the provider-reject procedure is performed, and no RO-ERROR indication primitive is issued by the ROPM.

If the ReturnError APDU is acceptable to the accepting ROPM, it issues an RO-ERROR indication primitive to the acceptor. The RO-ERROR indication primitive parameters are derived from the ReturnError APDU.

The accepting ROPM waits either for a *TRANSFER* indication primitive from the *TRANSFER* service-provider or any other primitive from the acceptor.

7.6.4 Use of the ReturnError APDU fields

The ReturnError fields are used as follows.

7.6.4.1 Invoke-id

This is the Invoke-id parameter value of the RO-ERROR request primitive. It appears as the Invoke-id parameter value of the RO-ERROR indication primitive.

The value of this field is transparent to the ROPM and shall not be the value **absent:NULL**; however the value may be used in the provider-reject procedure.

7.6.4.2 Error-id

This is the Error-id parameter value of the RO-ERROR request primitive. It appears as the Error-id parameter value of the RO-ERROR indication primitive.

The value of this field is transparent to the ROPM.

7.6.4.3 Parameter

This is the Parameter parameter value of the RO-ERROR request primitive. It appears as the Parameter parameter value of the RO-ERROR indication primitive.

The value of this field is transparent to the ROPM.

7.7 User-reject

7.7.1 Purpose

The user-reject procedure is used by one AE to reject the request (invocation) or reply (result or error) of the other AE.

7.7.2 APDUs used

The user-reject procedure uses the Reject APDU. This Reject APDU is used in addition by the provider-reject procedure.

The fields of the Reject APDU used for the user-reject procedure are listed in Table 13.

Table 13 – Reject APDU fields

Field name	Presence	Source	Sink
Invoke-id	M	req.	ind.
Problem (choice of):	M	req.	ind.
Invoke-problem Return-result-problem Return-error-problem			

7.7.3 User-reject procedure

This procedure is driven by the following events:

a) an RO-REJECT-U request primitive from the requestor;

b) an Reject APDU as user-data of a *TRANSFER* indication primitive.

7.7.3.1 RO-REJECT-U request primitive

The requesting ROPM forms an Reject APDU from the parameter values of the RO-REJECT-U request primitive. It issues a *TRANSFER* request primitive. The user-data parameter of the *TRANSFER* request primitive contains the Reject APDU.

The requesting ROPM waits either for a *TRANSFER* indication primitive from the *TRANSFER* service-provider or any other primitive from the requestor.

7.7.3.2 Reject APDU

The accepting ROPM receives an Reject APDU from its peer as user-data on a *TRANSFER* indication primitive. If any of the fields of the Reject APDU are unacceptable to this ROPM, no RO-REJECT-U indication primitive is issued by the ROPM.

If the Reject APDU is acceptable to the accepting ROPM and the fields of the Reject APDU indicates a user reject (i.e. Invoke-problem, Return-result-problem, or Return-error-problem), it issues an RO-REJECT-U indication primitive to the acceptor. The RO-REJECT-U indication primitive parameters (Invoke-id and Reject-reason) are derived from the Reject APDU.

The accepting ROPM waits either for a *TRANSFER* indication primitive from the *TRANSFER* service-provider or any other primitive from the acceptor.

7.7.4 Use of the Reject APDU fields

The Reject fields are used as follows.

7.7.4.1 Invoke-id

This is the Invoke-id parameter value of the RO-REJECT-U request primitive. It appears as the Invoke-id parameter value of the RO-REJECT-U indication primitive.

The value of this field is transparent to the ROPM.

7.7.4.2 Problem

This is the Problem parameter value of the RO-REJECT-U request primitive. It appears as the Problem parameter value of the RO-REJECT-U indication primitive.

The values used by the user-reject procedure are:

a) **Invoke problem**: user-reject of an RO-INVOKE indication primitive with values defined in 8.4.1 of ITU-T Rec. X.881 | ISO/IEC 13712-2.

b) **Return-result-problem**: user-reject of an RO-RESULT indication primitive with values defined in 8.4.1 of ITU-T Rec. X.881 | ISO/IEC 13712-2.

c) **Return-error-problem**: user-reject of an RO-ERROR indication primitive with values defined in 8.4.1 of ITU-T Rec. X.881 | ISO/IEC 13712-2.

7.8 Provider-reject

7.8.1 Purpose

The provider-reject procedure is used to inform the ROSE user and the peer ROPM, if an ROPM detects a problem.

7.8.2 APDUs used

The provider-reject procedure uses the Reject APDU. This Reject APDU is used in addition by the user-reject procedure.

The fields of the Reject APDU used for the provider-reject procedure are listed in Table 14.

Table 14 – Reject APDU fields

Field name	Presence	Source	Sink
Invoke-id	M	sp	ind.
Problem (choice of): General-problem	M	sp	ind.

7.8.3 Provider-reject procedure

This procedure is driven by the following events:

a) an unsuccessful APDU as user-data of a *TRANSFER* indication indication primitive.

b) a Reject APDU with the Problem parameter choice **General-problem** as user-data of a *TRANSFER* indication primitive;

c) unsuccessful APDU transfer (e.g. association abort).

7.8.3.1 Unacceptable APDU

The receiving ROPM receives an APDU from its peer as user data on a *TRANSFER* indication primitive. If any of the fields of the APDU (except Reject APDU) are unacceptable to this ROPM, it forms an Reject APDU with the Problem field choice General-problem and the Invoke-id of the rejected APDU. The receiving ROPM issues a *TRANSFER* request primitive. The user-data parameter of the *TRANSFER* request primitive contains the Reject APDU.

If the received unacceptable APDU is an Reject APDU no new Reject APDU is formed and transferred. In this case, or after the rejection of a locally specified number of APDUs, the application-association may be released abnormally.

If the application-association is not released abnormally, the receiving ROPM waits either for a *TRANSFER* indication primitive from the *TRANSFER* service-provider or any other primitive from the requestor.

7.8.3.2 Reject APDU

The receiving ROPM receives an Reject APDU from its peer as user-data on a *TRANSFER* indication primitive. If any of the fields of the Reject APDU are unacceptable to this ROPM, the provider-reject procedure for an unacceptable APDU is performed.

If the Reject APDU is acceptable to the accepting ROPM and the Problem field of the Reject APDU indicates a General-problem, it issues an RO-REJECT-P indication primitive to the acceptor. The RO-REJECT-P indication primitive parameters (Invoke-id and Reject-reason) are derived from the Reject APDU.

The receiving ROPM waits either for a *TRANSFER* indication primitive from the *TRANSFER* service-provider or any other primitive from the acceptor.

7.8.3.3 Unsuccessful APDU transfer

If a sending ROPM is not able to transfer an APDU by means of the *TRANSFER* request primitive (e.g. in the case of abnormal association release), the sending ROPM issues an RO-REJECT-P indication primitive to the requestor for each APDU not yet transferred.

The RO-REJECT-P indication primitive parameter Invoke-id contains the invoke id of the RO-INVOKE request, RO-RESULT request, RO-ERROR request or RO-REJECT-U request primitives.

After all the RO-REJECT-P ind primitives for the APDUs not transferred have been issued to the requestor, the application-association, if it still exists, is released abnormally.

7.8.4 Use of the Reject APDU fields

The Reject APDU fields are used as follows.

7.8.4.1 Invoke-id

This is the Invoke-id field of a rejected APDU and the Invoke-id parameter of the RO-REJECT-P indication primitive. The type and value of this field may be **NULL**, if the Invoke-id field of the rejected APDU is not detectable. In this case, the Invoke-id parameter of the RO-REJECT-P indication primitive is omitted.

7.8.4.2 Problem – General-problem

This is the Problem parameter value of the RO-REJECT-P indication primitive. The values used by the provider-reject procedure are:

a) **General-problem**: provider-reject of an APDU with values defined in 8.5.2 of ITU-T Rec. X.881 | ISO/IEC 13712-2.

8 Association realizations

8.1 Introduction

An association realization requires the inclusion in the application context of an ASE which provides services for the establishment and release of associations. The ASE may require a number of other supporting ASEs to be present. Two such realizations are specified in this clause. The ACSE realization is specified in 8.2, and uses the connection-oriented services of ACSE directly. The RTSE realization is specified in 8.3.

The specification of an association realization involves:

a) identifying the ASE which provides the association services, and any supporting ASEs;

b) specifying which of that ASE's services provide the *ESTABLISH* and *RELEASE* primitives;

c) specifying which transfer realization(s) are available for use on the established association;

d) defining any parameters which must be supplied to complete the realization;

e) providing an object of the class **REALIZATION**, specified as follows, to allow application context designers to use such a realization in their specification.

```
REALIZATION ::= TYPE-IDENTIFIER
```

8.2 Association realization by ACSE

8.2.1 This subclause specifies an association realization employing the connection-oriented services of ACSE.

8.2.2 The association services assumed by ROSE are provided as shown in Table 15.

Table 15 – Actual association primitives for the ACSE realization

Pseudo-primitive	Actual primitive(s)
ESTABLISH	A-ASSOCIATE
application context	Application Context Name
release can fail	Session requirements: negotiated release
user data	User information
result – accepted – rejected	Result – accepted – rejected(permanent) rejected(transient)
RELEASE	A-RELEASE
user data	User information
result – accepted – rejected	Result/Reason – affirmative/normal – affirmative/not finished negative/*

Table 15 – Actual association primitives for the ACSE realization (end)

ABORT	A-ABORT
user data	User information
source – association-control service user – association-control service provider	Abort source – service user – service provider
ABORT-P	A-P-ABORT
provider reason	Provider reason

8.2.3 This realization requires the establishment of a defined context set (DCS) which includes at least one presentation context for each of the abstract syntaxes necessary to convey the bind and unbind operations, as described in 11.2-3.

8.2.4 If user data is present, each *ESTABLISH* primitive and each *RELEASE* primitive conveys in the user data parameter a single ROSE APDU which is considered a presentation data value (PDV) from the appropriate presentation context.

8.2.5 In the *ESTABLISH* primitive, the single ROS APDU that may be conveyed in the user data parameter is formed from the **Bind{}** parameterised type (see 9.11 of ITU-T Rec. X.880 | ISO/IEC 13712-1). In the case of the *RELEASE* primitive, the single ROS APDU is formed from the **Unbind{}** parameterised type (see 9.12 of ITU-T Rec. X.880 | ISO/IEC 13712-1).

8.2.6 The P-DATA transfer realization (see 9.2) is available for use on the established association.

8.2.7 If this realization is used in conjunction with the P-DATA transfer realization specified in 9.2, then any PDVs representing *TRANSFER* requests which are conveyed in an A-ASSOCIATE primitive shall follow the PDV which represents the BindInvoke, BindResult, or BindError APDU. Similarly, any PDVs representing *TRANSFER* requests which are conveyed in an A-RELEASE primitive shall precede the PDV which represents the UnbindInvoke, UnbindResult, or UnbindError APDU.

8.2.8 The realization specified in this subclause can be included as the **&associationRealization** field of an **APPLICATION-CONTEXT** by referencing either the definition **acse** or the definition **acse-with-concatenation**:

```
acse REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(association-service)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) association-realizations(10)
acse-without-concatenation(0)}
 }

acse-with-concatenation REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type (association-service),
                                      concatenation (TRUE)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) association-realizations(10) acse-with-concatenation(1)}
 }
```

where

```
RealizationParameter ::= SEQUENCE
{
    realization-type    ENUMERATED {association-service(0), transfer-service(1)},
    concatenation       BOOLEAN DEFAULT FALSE
}
```

8.3 Association realization by RTSE

8.3.1 This subclause specifies an association realization employing the services of RTSE, which therefore must be included in the application context.

NOTE – RTSE requires the inclusion of ACSE in the application context.

8.3.2 The association services assumed by ROSE are provided as shown in Table 16.

Table 16 – Actual association primitives for the RTSE realization

Pseudo-primitive	Actual primitive(s)
ESTABLISH	RT-OPEN
application context	Application Context Name (normal mode) Application-protocol (X.410-1984 mode)
release can fail	*false*
user data	User-data
result – accepted – rejected	Result – accepted – rejected(permanent) rejected(transient)
RELEASE	RT-CLOSE
user data	User data
result – accepted – rejected	Reason – normal – not finished
ABORT	RT-U-ABORT
source	Abort source – service user
user data	User data
ABORT-P	RT-P-ABORT
provider reason	Provider reason
NOTE – The *ABORT* pseudo-primitive and the parameters "user data" and "result" of the *RELEASE* pseudo-primitive are not supported in the X.410-1984 mode.	

8.3.3 The RT-TRANSFER transfer realization (see 9.3) is available for use on the established association.

8.3.4 The realization specified in this subclause can be included as the **&associationRealization** field of an **APPLICATION-CONTEXT** by referencing the definition **association-by-rtse**:

```
association-by-RTSE REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(association-service)})
IDENTIFIED BY {joint-iso-itu-t association-realizations(10) association-by-rtse(2)}
 }
```

where, **RealizationParameter** is defined as

```
RealizationParameter ::= SEQUENCE
{
    realization-type    ENUMERATED {association-service(0), transfer-service(1)},
    concatenation       BOOLEAN DEFAULT FALSE
}
```

9 Transfer realizations

9.1 Introduction

A transfer realization may require the inclusion in the application context of an ASE which provides services for the transfer of information. Alternatively, the realization may involve direct use of the presentation service. The realization may require a number of supporting ASEs to be present. Two transfer realizations are specified in this clause. The P-DATA realization is specified in 9.2, and uses the connection-oriented services of the Presentation service directly. The RT-TRANSFER realization is specified in 9.3.

The specification of a transfer realization involves:

a) identifying the ASE, if any, which provides the *TRANSFER* services, and any supporting ASEs;

b) specifying which services provide the *TRANSFER* primitives;

c) specifying any additional rules or constraints.

d) defining any parameters which must be supplied to complete the realization;

e) providing an object of the class **REALIZATION**, specified as follows, to allow application context designers to use such a realization in their specification.

```
REALIZATION ::= TYPE-IDENTIFIER
```

9.2 P-DATA

9.2.1 This subclause specifies a transfer realization employing the P-DATA service of the Presentation layer, and, possibly, the information transfer capabilities of the A-ASSOCIATE and A-RELEASE services.

9.2.2 The *TRANSFER* service assumed by ROSE is provided as shown in Table 17.

Table 17 – Actual transfer primitives for the P-DATA realization

Pseudo-primitive	Actual primitive(s)
TRANSFER	P-DATA A-ASSOCIATE A-RELEASE
user-data	User data User information

9.2.3 This realization requires the prior establishment of a presentation connection between the two application-entities involved, and the establishment of a defined context set (DCS) which includes at least one presentation context for each of the abstract syntaxes necessary, as described in clause 10. The application context shall contain any rules which must be agreed between the application-entities for the establishment and release of the presentation connection.

NOTES

1 This requirement is automatically met if the application context includes the association realization specified in 8.2.

2 The usage of the term “DCS” is intended to include the initial DCS in the case of A-ASSOCIATE, although strictly-speaking the relevant presentation contexts will only be included in the DCS on the completion of the association establishment.

9.2.4 The object identifier value **{joint-iso-itu-t asn1(1) basic-encoding(1)}** specified in ITU-T Rec. X.690 | ISO/IEC 8825-1 may be used as a transfer syntax name. In this case, the ROSE-user protocol need not name and specify a transfer syntax.

9.2.5 Each *TRANSFER* request conveys a single ROSE APDU, which is considered a presentation data value (PDV) from the appropriate presentation context.

9.2.6 Two variants of this realization are provided:

a) **concatenation permitted** – The PDVs from several *TRANSFER* requests arising close to one another in time, can be concatenated together, preserving order, as a single User Data parameter in a single P-DATA request. In addition, the PDVs from one or more *TRANSFER* requests arising close in time to an A-ASSOCIATE or A-RELEASE request or A-ASSOCIATE or A-RELEASE response can be included in the User information of the appropriate primitive only by putting the *TRANSFER* requests within the User Data parameter, behind any user data already present in the A-ASSOCIATE or A-RELEASE request/response;

   TRANSFER requests in the bind and unbind pending states are possible if and only if this realization is chosen.

b) **concatenation prohibited** – Each *TRANSFER* request corresponds to a single P-DATA request.

9.2.7 The realization specified in this subclause can be included as the **&transferRealization** field of an **APPLICATION-CONTEXT** by referencing the definitions **pData or pData-with-concatenation:**

```
pData REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(transfer-service)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) transfer-realizations(11) pData-without-concatenation(0)}
 }

pData-with-concatenation REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(transfer-service),
                                      concatenation(TRUE)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) transfer-realizations(11) pData-with-concatenation(0)}
 }
```

where

```
RealizationParameter ::= SEQUENCE
{
    realization-type    ENUMERATED {association-service(0), transfer-service(1)},
    concatenation       BOOLEAN DEFAULT FALSE
}
```

9.3 RT-TRANSFER

9.3.1 This subclause specifies a transfer realization employing the RT-TRANSFER service of RTSE, which must therefore be included in the application context.

9.3.2 The *TRANSFER* service assumed by ROSE is provided as shown in Table 18.

Table 18 – Actual transfer primitives for the RT-TRANSFER realization

Pseudo-primitive	Actual primitive(s)
TRANSFER	RT-TRANSFER
user-data	APDU

9.3.3 This realization requires the prior establishment, by RTSE, of an application association between the two application-entities involved, and the establishment of a defined context set which includes at least one presentation context for each of the abstract syntaxes necessary, as described in 10. The application context shall contain any rules which must be agreed between the application-entities for the establishment and release of the association.

NOTE – This requirement is automatically met if the application context includes the association realization specified in 8.3.

9.3.4 Each APDU is transferred as user-data of the RT-TRANSFER service. The ROPM only issues an RT-TRANSFER request primitive if it possesses the Turn, and if there is no outstanding RT-TRANSFER confirm primitive. The use of the other RT-TRANSFER parameters is as follows:

RT-TRANSFER request

APDU	The APDU is to be transferred. Its maximum size is not restricted in this mapping.
Transfer-time	This is specified by a local rule of the sending ROPM. It may be related to the priority of the APDU.

RT-TRANSFER indication

APDU	The APDU is transferred. Its maximum size is not restricted in this mapping.

RT-TRANSFER confirm

APDU	The APDU is not transferred within the transfer time. This parameter is nly provided if the value of the result parameter is "APDU-not-transferred". In this case, the ROPM issues a RO-REJECT-P indication primitive with the invoke id of the returned ROSE APDU.
Result	The parameter value "APDU-transferred" indicates a positive confirm, while the parameter value "APDU-not-transferred" indicates a negative confirm.

9.3.5 Managing the turn

A ROPM shall possess the turn before it can use the RT-TRANSFER service. The ROPM without the turn may issue a RT-TURN-PLEASE request primitive the priority parameter of which reflects the highest priority APDU awaiting transfer.

The ROPM which has the turn, may issue an RT-TURN-GIVE request primitive when it has no further APDUs to transfer. It will issue an RT-TURN-GIVE request primitive in response to an RT-TURN-PLEASE indication when it has no further APDUs to transfer of priority equal to or higher than that indicated in the RT-TURN-PLEASE indication primitive. If it has APDUs of lower priority still to transfer, it may issue an RT-TURN-PLEASE request whose priority reflects the highest priority APDU remaining to be transferred.

9.3.5.1 Use of the RT-TURN-PLEASE service

The ROPM issues the RT-TURN-PLEASE request primitive to request the turn. It may do so only if it does not already possess the turn. The RT-TURN-PLEASE service is a non-confirmed service.

The use of the RT-TURN-PLEASE service parameters is as follows:

Priority – This reflects the highest priority APDU awaiting transfer.

9.3.5.2 Use of the RT-TURN-GIVE service

The ROPM issues the RT-TURN-GIVE request primitive to relinquish the turn to its peer. It may do so only if it possesses the turn. The RT-TURN-GIVE service is a non-confirmed service with no parameters.

NOTE – A ROPM will often have no APDU to be transferred when it has just transferred an Invoke APDU for a synchronous operation.

9.3.6 The realization specified in this subclause can be included as the **&transferRealization** field of an **APPLICATION-CONTEXT** by referencing the definition **transfer-by-RTSE** given by:

```
transfer-by-RTSE REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(transfer-service)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) transfer-realizations(11) rTSE-transfer(0)}
 }
```

where **RealizationParameter** is defined in 9.2.7.

10 Abstract syntaxes

10.1 Introduction

An application context realizing an association contract must include one or more abstract syntaxes to represent the required APDUs of ROSE, as referenced in the elements of procedure of 7. The APDUs themselves are as defined for the generic ROS protocol of ITU-T Rec. X.880 | ISO/IEC 13712-1. For ROSE, the following set of **InvokeId**s (or some subset thereof) is always employed:

```
ROSEInvokeIds InvokeId ::= {ALL EXCEPT noInvokeId}
```

ITU-T Rec. X.880 | ISO/IEC 13712-1 also includes a number of parameterised definitions which can be useful in defining suitable abstract syntaxes.

Throughout this clause, the object reference **ac** is used to represent the particular **APPLICATION-CONTEXT** involved.

10.2 Bind operation

If the application context **ac** realizes an association contract which includes a connection package, the values of the data type:

```
Bind{OPERATION:ac.&associationContract.&connection.&bind}
```

shall appear in at least one of the **ac.&AbstractSyntaxes**.

10.3 Unbind operation

If the application context **ac** realizes an association contract, which includes a connection package the values of the data type:

```
Unbind{OPERATION:ac.&associationContract.&connection.&unbind}
```

shall appear in at least one of the **ac.&AbstractSyntaxes**.

10.4 Other operations

An application context **ac** involves a set of operation packages.

```
OperationPackages{APPLICATION-CONTEXT:ac} OPERATION-PACKAGE ::=
{
     ac.&associationContract.&OperationsOf
|    ac.&associationContract.&InitiatorConsumerOf
|    ac.&associationContract.&ResponderConsumerOf
}
```

For each of the operations **op** which are involved in these packages, there shall be at least one of **ac.&AbstractSyntaxes** which includes:

```
Invoke{InvokeId:ROSEInvokeIds, OPERATION:op}
```

and at least one which includes:

```
ReturnResult {OPERATION:op}
```

and for each of the errors **err** in **Errors{OPERATION:op}** there shall be at least one of **ac.&AbstractSyntaxes** which includes:

```
ReturnError {ERROR:err}
```

At least one of **ac.&AbstractSyntaxes** shall include:

```
Reject
```

10.5 Defining the abstract syntaxes

10.5.1 ITU-T Rec. X.880 | ISO/IEC 13712-1 provides a number of parameterised definitions which facilitate the specification of the abstract syntaxes required.

NOTE – As explained in ITU-T Rec. X.880 | ISO/IEC 13712-1, some of these definitions cannot be used if the depth of linkage of linked operations exceeds a certain level.

10.5.2 Given a certain operation **package**, a single abstract syntax which allows for the invocation and reporting for all of its operations can be defined using the following data type:

```
ROS-SingleAS{InvokeId:ROSEInvokeIds, OPERATION-PACKAGE:package}
```

or alternatively a pair of abstract syntaxes can be defined based upon the pair of types:

```
ROS-ConsumerAS{InvokeId:ROSEInvokeIds, OPERATION-PACKAGE:package}
ROS-SupplierAS{InvokeId:ROSEInvokeIds, OPERATION-PACKAGE:package}
```

10.5.3 A single abstract syntax may accommodate a set of packages, provided that the operation and error codes are unique. For example, the following data type can be used as the basis of a single abstract syntax to accommodate an entire association contract:

```
AllValues {APPLICATION-CONTEXT:ac} ::= CHOICE
{
        bind{ac.&associationContract.&connection.&bind},
        unbind{ac.&associationContract.&connection.&unbind},
        ros-SingleAs
        {
            {ROSEInvokeIds},
            combine
            {
                {
                    ac.&associationContract.&OperationsOf
                |   ac.&associationContract.&InitiatorConsumerOf
                |   ac.&associationContract.&ResponderConsumerOf
                },
                {},
                {...}
            }
        }
}
```

11 Conformance

An implementation claiming conformance to this Recommendation | International Standard shall comply with the requirements in 11.1 through 11.3.

11.1 Statement requirements

An implementor shall state the following:

a) the application context for which conformance is claimed, including whether the system supports the mapping of ROSE onto RTSE, onto the Presentation service, or both.

11.2 Static requirements

The system shall:

a) conform to the abstract syntax definition of APDUs defined in clause 10.

11.3 Dynamic requirements

The system shall:

a) conform to the elements of procedure defined in clause 7;

b) conform to the mappings to the used services, for which conformance is claimed, as defined in clause 9.

Annex A

ROPM State Tables

(This annex forms an integral part of this Recommendation | International Standard)

A.1 General

This Annex defines a single Remote Operations Protocol Machine (ROPM) in terms of state tables. At any given time the state can be described by several state variables:

a) an operating state;

b) (if realizing an association contract with a connection package) a binding state;

c) (if realizing an application context with an association realization) an association state;

d) a transfer state.

The relationships between these state components that comprise the ROPM are shown in Figure A.1: a) covering the case where a connection package is involved and b) where no connection package is involved.

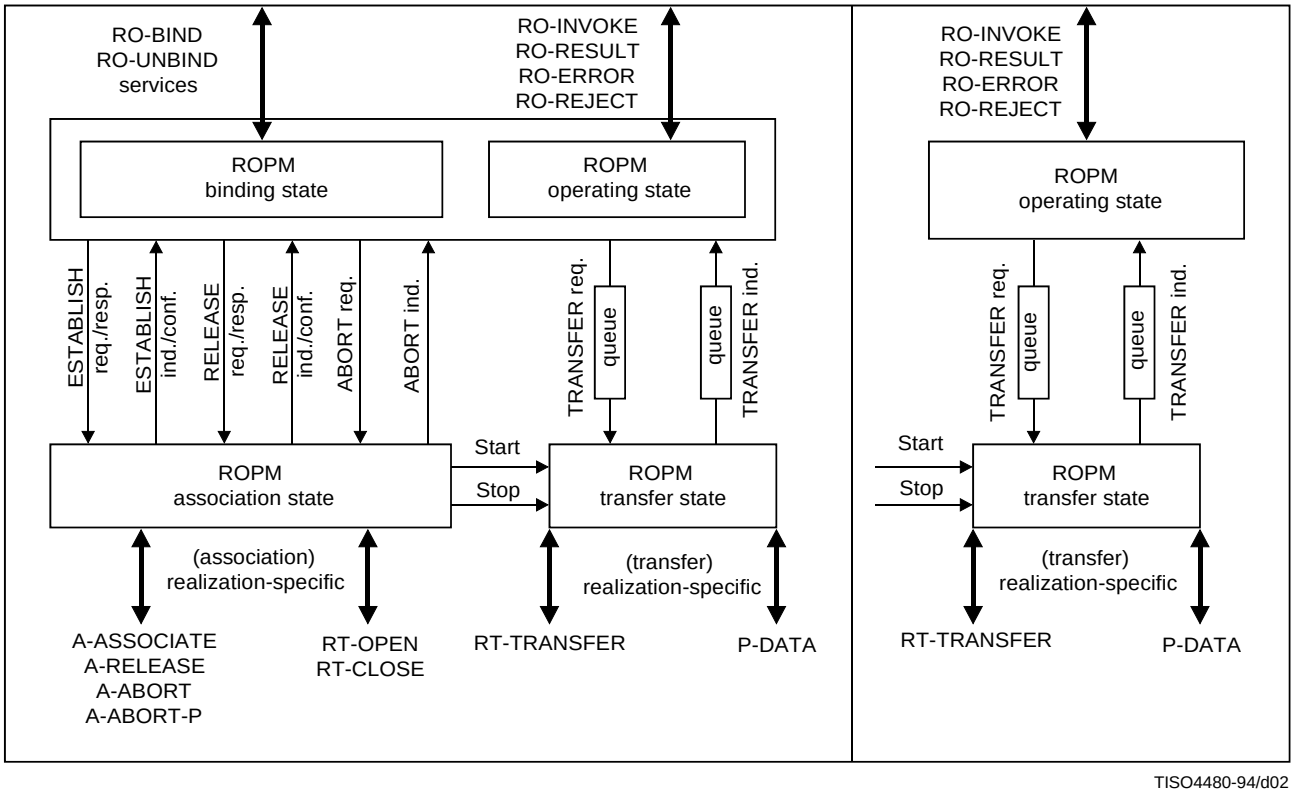


Figure A.1 – Components of the ROPM state

In Figure A.1, case a), the connection package defines the bind/unbind operations that are used to establish/release the association. If the (optional) bind/unbind operations are not used to establish/release the association, it is assumed that some other means (not described in this Recommendation | International Standard) are used to control the association.

In Figure A.1, case b), it is assumed that a transfer service has been made available by means not described in this Recommendation | International Standard.

The operating state is realization-independent while the other components depend on the particular realization in use.

Each state table shows the interrelationship between the relevant parts of the ROPM state, the events that occur, and the actions taken. The state tables have been separated into two, one where a connection package is involved and one without. The latter state table permits the realization-independent state transitions for the "operating state" to be combined with any permissible realizations of the association and transfer services.

The state tables do not constitute a formal specification. They are included to provide a more precise specification of the elements of procedure defined in clauses 7, 8, and 9. In general, the state tables assume that only valid local sequences of primitives occur.

A.2 Conventions

In the state tables, the intersection of an incoming event (row) and a state (column) forms a cell.

In the state table, a blank cell represents the combination of an incoming event and a state that is not defined for the ROPM. (See A.3.1.)

A non-blank cell represents an incoming event and a state that is defined for the ROPM. Such a cell contains one or more action lists. An action list may be either mandatory or conditional. If a cell contains a mandatory action list, it is the only action list in the cell.

A mandatory action list contains:

a) optionally one or more outgoing events; and

b) a resultant state.

A conditional action list contains:

a) a predicate expression comprising predicates and Boolean operators (¬ represents the Boolean NOT); and

b) a mandatory action list (this mandatory action list is used only if the predicate expression is TRUE).

A.3 Actions to be taken by the ROPM

The ROPM state table defines the action to be taken by the ROPM in terms of an optional outgoing event and the resultant state of the application-association.

A.3.1 Invalid intersections

Blank cells indicate an invalid intersection of an incoming event and state. If such an intersection occurs, one of the following actions is taken:

a) If the incoming event comes from the ROSE-user, any action taken by the ROPM is a local matter.

b) If the incoming event is related to a received APDU, Presentation-Service-provider, ACSE, or RTSE, either the ROPM issues an AA-ABreq event to the ROPM transfer state (ROPM-TR), or the ROPM-TR issues an ABORTreq to either the RTSE or ACSE and an AA-ABind to the ROPM binding_state.

A.3.2 Valid intersections

If the intersection of the state and incoming event is valid, the following actions is taken:

a) If the cell contains a mandatory action list, the ROPM takes the action specified.

b) If a cell contains one or more conditional action lists, for each predicate expression that is true, the ROPM takes the actions specified. If none of the predicate expressions are true, the ROPM takes one of the actions defined in A.3.1.

A.4 Tables

Each state table has either four or five components, as follows:

IN The incoming event list, which specifies the abbreviated name, source, and name or description of each incoming event;

ST The state list, which specifies the abbreviated name of each state;

OUT The outgoing event list, which specifies the abbreviated name, target, and name or description of each outgoing event;

PR The predicates, if any, used to specify conditional action lists;

TABLE The state table itself.

There are six state tables, as follows:

A.1 a) ROPM state table – Involving a connection package

A.1 b) ROPM state table – Not involving a connection package

A.2 Association state table – ACSE realization

A.3 Association state table – RTSE realization

A.4 Transfer state table – P-DATA realization

A.5 Transfer state table – RT-TRANSFER realization

Table A.1 a) (IN) – ROPM involving a connection package

Abbreviated name	Source	Name and description
RO-BINDreq	ROSE-user	RO-BIND request
RO-BINDrsp+	ROSE-user	RO-BIND response with outcome “result”
RO-BINDrsp–	ROSE-user	RO-BIND response with outcome “error”
RO-UNBDreq	ROSE-user	RO-UNBIND request
RO-UNBDrsp+	ROSE-user	RO-UNBIND response with outcome “result”
RO-UNBDrsp–	ROSE-user	RO-UNBIND response with outcome “error-bound” or “error-unbound”
BindInv	ROPM-peer	BindInvoke APDU as user data on ESTind event
BindRes	ROPM-peer	BindResult APDU as user data on ESTcnf+ event
BindErr	ROPM-peer	BindError APDU as user data on ESTcnf– event
UnbindInv	ROPM-peer	UnbindInvoke APDU as user data on RELind event
UnbindRes	ROPM-peer	UnbindResult APDU as user data on RELcnf+ event
UnbindErr	ROPM-peer	UnbindError APDU as user data on RELcnf– event
RO-INVreq	ROSE-user	RO-INVOKE request primitive
RO-RES req	ROSE-user	RO-RESULT request primitive
RO-ERR req	ROSE-user	RO-ERROR request primitive
RO-REJu req	ROSE-user	RO-REJECT-U request primitive
Invoke	ROPM-peer	Invoke APDU on a TRANSind event
RetRes	ROPM-peer	ReturnResult APDU on a TRANSind event
RetErr	ROPM-peer	ReturnError APDU on a TRANSind event
RejU	ROPM-peer	Reject APDU (user-reject) on a TRANSind event
RejP	ROPM-peer	Reject APDU (provider-reject) on a TRANSind event
APDUua	ROPM-peer	Unacceptable APDU on a TRANSind event
TRANSind	transfer	*TRANSFER* indication carrying an APDU
ESTind	association	*ESTABLISH* indication
ESTcnf+	association	*ESTABLISH* confirm (accepted)
ESTcnf–	association	*ESTABLISH* confirm (rejected)
RELind	association	*RELEASE* indication
RELcnf+	association	*RELEASE* confirm (accepted)
RELcnf–	association	*RELEASE* confirm (rejected)
ABTind	association	*ABORT* and *ABORT-P* indication

Table A.1 b) (IN) – ROPM not involving a connection package

Abbreviated name	Source	Name and description
RO-INVreq	ROSE-user	RO-INVOKE request primitive
RO-RES req	ROSE-user	RO-RESULT request primitive
RO-ERR req	ROSE-user	RO-ERROR request primitive
RO-REJu req	ROSE-user	RO-REJECT-U request primitive
Invoke	ROPM-peer	Invoke APDU on a TRANSind event
RetRes	ROPM-peer	ReturnResult APDU on a TRANSind event
RetErr	ROPM-peer	ReturnError APDU on a TRANSind event
RejU	ROPM-peer	Reject APDU (user-reject) on a TRANSind event
RejP	ROPM-peer	Reject APDU (provider-reject) on a TRANSind event
APDUua	ROPM-peer	Unacceptable APDU on a TRANSind event
TRANSind	transfer	*TRANSFER* indication carrying an APDU
ABTind	association	*ABORT* and *ABORT-P* indication

Table A.1 a) (ST) – ROPM involving a connection package

Abbreviated name	Name and description
STA01	unbound
STA02	bound
STA03A	local bind pending
STA03B	remote bind pending
STA04A	local unbind pending
STA04B	remote unbind pending
STA04C	unbind collision – responder completion
STA04D	unbind collision – initiator completion

Table A.1 b) (ST) – ROPM not involving a connection package

Abbreviated name	Name and description
STA05	a transfer service is available
STA06	a transfer service is not available

Table A.1 a) (PR) – ROPM involving a connection package

Code	Name and description
p1	unacceptable APDU is not Reject APDU & the number of rejects does not exceed some locally specified value
p2	user is the initiator
p3	responder can unbind
p4	outcome is “error-bound”

Table A.1 b) (PR) – ROPM not involving a connection package

Code	Name and description
p1	unacceptable APDU is not Reject APDU & the number of rejects does not exceed some locally specified value

Table A.1 a) (OUT) – ROPM involving a connection package

Abbreviated name	Target	Name and description
RO-BINDind	ROSE-user	RO-BIND indication
RO-BINDcnf+	ROSE-user	RO-BIND confirm with outcome “result”
RO-BINDcnf–	ROSE-user	RO-BIND confirm with outcome “error”
RO-UNBDind	ROSE-user	RO-UNBIND indication
RO-UNBDcnf+	ROSE-user	RO-BIND confirm with outcome “result”
RO-UNBDcnf–	ROSE-user	RO-BIND confirm with outcome “error-bound” or “error-unbound”
BindInv	ROPM-peer	BindInvoke APDU as user data on ESTreq event
BindRes	ROPM-peer	BindResult APDU as user data on ESTrsp+ event
BindErr	ROPM-peer	BindError APDU as user data on ESTrsp– event
UnbindInv	ROPM-peer	UnbindInvoke APDU as user data on RELreq event
UnbindRes	ROPM-peer	UnbindResult APDU as user data on RELrsp+ event
UnbindErr	ROPM-peer	UnbindError APDU as user data on RELrsp– event
RO-INVind	ROSE-user	RO-INVOKE indication primitive
RO-RES ind	ROSE-user	RO-RESULT indication primitive
RO-ERR ind	ROSE-user	RO-ERROR indication primitive
RO-REJu ind	ROSE-user	RO-REJECT-U indication primitive
RO-REJp ind	ROSE-user	RO-REJECT-P indication primitive
Invoke	ROPM-peer	Invoke APDU on a TRANSreq event
RetRes	ROPM-peer	ReturnResult APDU on a TRANSreq event

Table A.1 a) (OUT) – ROPM involving a connection package (end)

RetErr	ROPM-peer	ReturnError APDU on a TRANSreq event
RejU	ROPM-peer	Reject APDU (user-reject) on a TRANSreq event
RejP	ROPM-peer	Reject APDU (provider-reject) on a TRANSreq event
TRANSreq	transfer	*TRANSFER* request carrying an APDU
ESTreq	association	*ESTABLISH* request
ESTrsp+	association	*ESTABLISH* response (accepted)
ESTrsp−	association	*ESTABLISH* response (rejected)
RELreq	association	*RELEASE* request
RELrsp+	association	*RELEASE* response (accepted)
RELrsp−	association	*RELEASE* response (rejected)
ABTreq	association	*ABORT* request

Table A.1 b) (OUT) – ROPM not involving a connection package

Abbreviated name	Target	Name and description
RO-INVind	ROSE-user	RO-INVOKE indication primitive
RO-RES ind	ROSE-user	RO-RESULT indication primitive
RO-ERR ind	ROSE-user	RO-ERROR indication primitive
RO-REJu ind	ROSE-user	RO-REJECT-U indication primitive
RO-REJp ind	ROSE-user	RO-REJECT-P indication primitive
Invoke	ROPM-peer	Invoke APDU on a TRANSreq event
RetRes	ROPM-peer	ReturnResult APDU on a TRANSreq event
RetErr	ROPM-peer	ReturnError APDU on a TRANSreq event
RejU	ROPM-peer	Reject APDU (user-reject) on a TRANSreq event
RejP	ROPM-peer	Reject APDU (provider-reject) on a TRANSreq event
TRANSreq	transfer	*TRANSFER* request carrying an APDU
ABTreq	association	*ABORT* request

Table A.1 a) (TABLE) – ROPM involving a connection package (Part 1 of 2)

	STA01	STA02	STA03A	STA03B
RO-BINDreq	p2: BindInv[a)] STA03A			
RO-BINDrsp+				¬p2:BindRes[b)] STA02
RO-BINDrsp–				¬p2:BindErr[c)] STA01
RO-UNBDreq		p2 ∨ p3: UnbindInv[d)] STA04A		
RO-UNBDrsp+				
RO-UNBDrsp–				
BindInv	¬p2: RO-BINDind STA03B			
BindRes			p2: RO-BINDcnf+ STA02	
BindErr			p2: RO-BINDcnf- STA01	
UnbindInv		¬p2 ∨ p3: RO-UNBDreq STA04B		
UnbindRes				
UnbindErr				
RO-INVreq		Invoke STA02 TRANSreq	p2: Invoke STA03A TRANSreq	
RO-RESreq		RetRes STA02 TRANSreq		
RO-ERRreq		RetErr STA02 TRANSreq		
RO-RJUreq		RejU STA02 TRANSreq		
Invoke		RO-INVind STA02		¬p2: RO-INVind STA03B
RetRes		RO-RESind STA02		
RetErr		RO-ERRind STA02		

Table A.1 a) (TABLE) – ROPM involving a connection package (Part 1 of 2) *(end)*

	STA01	STA02	STA03A	STA03B
RejU		RO-RJUind STA02		
RejP		RO-RJPind STA02		
APDUua		p1: RejP STA02 TRANSreq ¬p1: ABTreq STA01		
ABTind		STA01	STA01	STA01

a) Sending the BindInvokePDU is optional if the **&argumentTypeOptional** field in the definition of the **bind** operation is set to **TRUE**. The corresponding statement also hold for the **unbind** operation and error. (See 7.1.3 and 7.2.3.)

b) Sending the BindResult PDU is optional if the **&resultTypeOptional** field in the definition of the **bind** operation is set to **TRUE**. (See 7.1.3.)

c) Sending the BindError PDU is optional if the **&parameterTypeOptional** field in the definition of the **bind** error is set to **TRUE**. (See 7.1.3.)

d) Sending the UnbindInvokePDU is optional if the **&argumentTypeOptional** field in the definition of the **unbind** operation is set to **TRUE**. (See 7.2.3.)

Table A.1 a) (TABLE) – ROPM involving a connection package (Part 2 of 2)

	STA04A	STA04B	STA04C	STA04D
RO-BINDreq				
RO-BINDrsp+				
RO-BINDrsp–				
RO-UNBDreq				
RO-UNBDrsp+		UnbindRes[e] STA01	p2: UnbindRes STA04D	¬p2: UnbindRes STA01
RO-UNBDrsp–		UnbindErr[f] p4: STA02 ¬p4: STA01	p2: UnbindErr STA04D	¬p2: UnbindErr STA01
BindInv				
BindRes				
BindErr				
UnbindInv	RO-UNBDreq STA04C			
UnbindRes	RO-UNBDcnf+ STA01		¬p2: RO-UNBDcnf+ STA04D	p2: RO-UNBDcnf+ STA01
UnbindErr	RO-UNBDcnf– p4: STA02 ¬p4: STA01		¬p2: RO-UNBDcnf– STA04D	p2: RO-UNBDcnf– STA01
RO-INVreq				

Table A.1 a) (TABLE) – ROPM involving a connection package (Part 2 of 2) *(end)*

RO-RESreq				
RO-ERRreq				
RO-RJUreq				
Invoke	RO-INVind STA04A			
RetRes	RO-RESind STA04A			
RetErr	RO-ERRind STA04A			
RejU	RO-RJUind STA04A			
RejP	RO-RJPind STA04A			
APDUua	p1: STA04A ¬p1: ABTreq STA01			
ABTind	STA01	STA01	STA01	STA01

e) Sending the UnbindResultPDU is optional if the **&resultTypeOptional** field in the definition of the **unbind** operation is set to **TRUE**. (See 7.2.3.)

f) Sending the UnbindErrorPDU is optional if the **&parameterTypeOptional** field in the definition of the **unbind** error is set to **TRUE**. (See 7.2.3.)

Table A.1 b) (TABLE) – ROPM not involving a connection package

	STA05	STA06
RO-INVreq	Invoke STA05 TRANSreq	
RO-RESreq	RetRes STA05 TRANSreq	
RO-ERRreq	RetErr STA05 TRANSreq	
RO-RJUreq	RejU STA05 TRANSreq	
Invoke	RO-INVind STA05	
RetRes	RO-RESind STA05	
RetErr	RO-ERRind STA05	
RejU	RO-RJUind STA05	
RejP	RO-RJPind STA05	
APDUua	p1: STA05 ¬p1: ABTreq STA06	
ABTind	STA06	STA06

Table A.2 (IN) – Association state – ACSE realization

Abbreviated name	Source	Name and description
ESTreq	binding	*ESTABLISH* request
ESTrsp+	binding	*ESTABLISH* response (accepted)
ESTrsp–	binding	*ESTABLISH* response (rejected)
RELreq	binding	*RELEASE* request
RELrsp+	binding	*RELEASE* response (accepted)
RELrsp–	binding	*RELEASE* response (rejected)
ABTreq	binding	*ABORT* request
A-ASSind	ACSE	A-ASSOCIATE indication
A-ASScnf+	ACSE	A-ASSOCIATE confirm (accepted)
A-ASScnf–	ACSE	A-ASSOCIATE confirm (rejected)
A-RELind	ACSE	A-RELEASE indication
A-RELcnf+	ACSE	A-RELEASE confirm (affirmative/normal)
A-RELcnf–	ACSE	A-RELEASE confirm (affirmative/not finished or negative)
A-ABTind	ACSE	A-ABORT indication or A-P-ABORT indication

Table A.2 (ST) – Association state – ACSE realization

Abbreviated name	Name and description
AA01	unassociated
AA02	associated
AA03	association pending
AA04	release pending
AA05	release collision

Table A.2 (OUT) – Association state – ACSE realization

Abbreviated name	Target	Name and description
ESTind	binding	*ESTABLISH* indication
ESTcnf+	binding	*ESTABLISH* confirm (accepted)
ESTcnf–	binding	*ESTABLISH* confirm (rejected)
RELind	binding	*RELEASE* indication
RELcnf+	binding	*RELEASE* confirm (accepted)
RELcnf–	binding	*RELEASE* confirm (rejected)
ABTind	binding	*ABORT* indication
A-ASSreq	ACSE	A-ASSOCIATE request
A-ASSrsp+	ACSE	A-ASSOCIATE response (accepted)
A-ASSrsp–	ACSE	A-ASSOCIATE response (rejected)
A-RELreq	ACSE	A-RELEASE request
A-RELrsp+	ACSE	A-RELEASE response (affirmative/normal)
A-RELrsp–	ACSE	A-RELEASE response (affirmative/not finished or negative)
A-ABTreq	ACSE	A-ABORT request
start	transfer	start the transfer state machine
stop	transfer	stop the transfer state machine

Table A.2 (PR) – Association state – ACSE realization

Code	Name and description
p1	initiator
p2	result is “rejected-not-released”

Table A.2 (TABLE) – Association state – ACSE realization

	AA01	AA02	AA03	AA04	AA05
ESTreq	A-ASSreq start AA03				
ESTrsp+			A-ASSrsp+ AA02		
ESTrsp–			A-ASSrsp– stop AA01		
RELreq		A-RELreq AA04			
RELrsp+				A-RELrsp+ stop AA01	A-RELrsp+ p1: AA05 ¬p1: AA01
RELrsp–				A-RELrsp– p2: AA02 ¬p2: AA01	A-RELrsp– p1: AA05 ¬p1: AA01
ABTreq		A-ABTreq stop AA01	A-ABTreq stop AA01	A-ABTreq stop AA01	
A-ASSind	ESTind start AA03				
A-ASScnf+			ESTcnf+ AA02		
A-ASScnf–			ESTcnf– stop AA01		
A-RELind		RELind AA04		RELind stop AA05	
A-RELcnf+				RELcnf+ stop AA01	RELcnf+ p1: AA01 ¬p1: AA05
A-RELcnf–				RELcnf– p2: AA02 ¬p2: AA01	RELcnf– p1: AA01 ¬p1: AA05
A-ABTind		ABTind stop AA01	ABTind stop AA01	ABTind stop AA01	ABTind AA01

Table A.3 (IN) – Association state – RTSE realization

Abbreviated name	Source	Name and description
ESTreq	binding	*ESTABLISH* request
ESTrsp+	binding	*ESTABLISH* response (accepted)
ESTrsp–	binding	*ESTABLISH* response (rejected)
RELreq	binding	*RELEASE* request
RELrsp+	binding	*RELEASE* response (accepted)
RELrsp–	binding	*RELEASE* response (rejected)
ABTreq	binding	*ABORT* request
RT-OPNind	RTSE	RT-OPEN indication
RT-OPNcnf+	RTSE	RT-OPEN confirm (accepted)
RT-OPNcnf–	RTSE	RT-OPEN confirm (rejected)
RT-CLSind	RTSE	RT-CLOSE indication
RT-CLScnf+	RTSE	RT-CLOSE confirm (normal)
RT-CLScnf–	RTSE	RT-CLOSE confirm (not finished)
RT-ABTind	RTSE	RT-U-ABORT indication or RT-P-ABORT indication

Table A.3 (ST) – Association state – RTSE realization

Abbreviated name	Name and description
AR01	unassociated
AR02	associated
AR03	association pending
AR04	release pending

Table A.3 (OUT) – Association state – RTSE realization

Abbreviated name	Target	Name and description
ESTind	binding	*ESTABLISH* indication
ESTcnf+	binding	*ESTABLISH* confirm (accepted)
ESTcnf–	binding	*ESTABLISH* confirm (rejected)
RELind	binding	*RELEASE* indication
RELcnf+	binding	*RELEASE* confirm (accepted)
RELcnf–	binding	*RELEASE* confirm (rejected)
ABTind	binding	*ABORT* indication
RT-OPNreq	RTSE	RT-OPEN request
RT-OPNrsp+	RTSE	RT-OPEN response (accepted)
RT-OPNrsp–	RTSE	RT-OPEN response (rejected)
RT-CLSreq	RTSE	RT-CLOSE request
RT-CLSrsp+	RTSE	RT-CLOSE response (normal)
RT-CLSrsp–	RTSE	RT-CLOSE response (not finished)
RT-ABTreq	RTSE	RT-U-ABORT request
start	transfer	start the transfer state machine
stop	transfer	stop the transfer state machine

Table A.3 (TABLE) – Association state – RTSE realization

	AR01	AR02	AR03	AR04
ESTreq	RT-OPNreq AR03			
ESTrsp+			RT-OPNrsp+ start AR02	
ESTrsp–			RT-OPNrsp– AR01	
RELreq		RT-CLSreq AR04		
RELrsp+				RT-CLSrsp+ stop AR01
RELrsp–				RT-CLSrsp– AR02
ABTreq (Note)		RT-ABTreq stop AR01	RT-ABTreq stop AR01	RT-ABTreq stop AR01
RT-OPNind	ESTind AR03			
RT-OPNcnf+			ESTcnf+ start AR02	
RT-OPNcnf–			ESTcnf– AR01	
RT-CLSind		RELind AR04		
RT-CLScnf+				RELcnf+ stop AR01
RT-CLScnf–				RELcnf– AR02
RT-ABTind		ABTind stop AR01	ABTind stop AR01	ABTind stop AR01
NOTE – ABTreq is not supported in the CCITT Recommendation X.410 (1984) mode.				

Table A.4 (IN) – Transfer state – P-DATA realization

Abbreviated name	Source	Name and description
TRANSreq	operating	*TRANSFER* request
PDV	presentation / ACSE	PDV carried on P-DATA indication / A-ASSOCIATE or A-RELEASE indication or confirmation.
start	association	start the transfer state machine
stop	association	stop the transfer state machine

Table A.4 (ST) – Transfer state – P-DATA realization

Abbreviated name	Name and description
TP01	inactive
TP02	active

Table A.4 (OUT) – Transfer state – P-DATA realization

Abbreviated name	Target	Name and description
TRANSind	operating	*TRANSFER* indication
PDV	presentation / ACSE	PDV carried on P-DATA request / A-ASSOCIATE or A-RELEASE request or response.

Table A.4 (TABLE) – Transfer state – P-DATA realization

	TP01	TP02
TRANSreq		PDV TP02
PDV		TRANSind TP02
start	TP02	
stop		TP01

Table A.5 (IN) – Transfer state – RT-TRANSFER realization

Abbreviated name	Source	Name and description
TRANSreq	operating	*TRANSFER* request
RT-TRind	RTSE	RT-TRANSFER indication
RT-TRcnf+	RTSE	positive RT-TRANSFER confirm
RT-TRcnf–	RTSE	negative RT-TRANSFER confirm
RT-TPind	RTSE	RT-TURN-PLEASE indication
RT-TGind	RTSE	RT-TURN-GIVE indication
start	association	start the transfer state machine
stop	association	stop the transfer state machine

Table A.5 (ST) – Transfer state – RT-TRANSFER realization

Abbreviated name	Name and description
TR01	inactive
TR02	active, token assigned, no transfer
TR03	active, token assigned, transfer in progress
TR04	active, token not assigned, no transfer
TR05	active, token not assigned, transfer required

Table A.5 (OUT) – Transfer state – RT-TRANSFER realization

Abbreviated name	Target	Name and description
TRANSind	operating	*TRANSFER* indication
RT-TRreq	RTSE	RT-TRANSFER request
RT-TPreq	RTSE	RT-TURN-PLEASE request
RT-TGreq	RTSE	RT-TURN-GIVE request

Table A.5 (PR) – Transfer state – RT-TRANSFER realization

Code	Name and description
p2	Token initially assigned to ROPM-TR

Table A.5 (TABLE) – Transfer state – RT-TRANSFER realization

	TR01	TR02	TR03	TR04	TR05
start	p2: TR02 ¬ p2: TR04				
TRANS req		RT-TR req TR03		RT-TP req TR05	
RT-TR cnf+			TR02		
RT-TR cnf–			TRANSind TR02		
RT-TR ind				TRANS ind TR04	TRANS ind TR05
RT-TP ind		RT-TG req TR04	TR03		
RT-TG ind				TR02	RT-TR req TR03
stop		TR01	TRANSind TR01	TR01	TRANSind TR01
NOTE – TRANS ind in the rows RT-TRcnf– and “stop”contain a Reject APDU (provider reject).					

Annex B

ASN.1 Modules

(This annex forms an integral part of this Recommendation | International Standard)

```
Remote-Operations-Realizations {joint-iso-itu-t remote-operations(4) realizations(9) version1(0)}
DEFINITIONS ::=
BEGIN
-- exports everything
IMPORTS REALIZATION FROM Remote-Operations-Information-Objects-extensions {joint-iso-itu-t
remote-operations(4) informationObjects-extensions(8) version1(0)};

RealizationParameter ::= SEQUENCE
{
        realization-type   ENUMERATED {association-service(0), transfer-service(1)},
        concatenation      BOOLEAN DEFAULT FALSE
}

acse REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(association-service)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) association-realizations(10)
acse-without-concatenation(0)}
 }

acse-with-concatenation REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type (association-service),
                                      concatenation (TRUE)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) association-realizations(10) acse-with-concatenation(1)}
 }

association-by-RTSE REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(association-service)})
IDENTIFIED BY {joint-iso-itu-t association-realizations(10) association-by-rtse(2)}
 }

pData REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(transfer-service)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) transfer-realizations(11) pData-without-concatenation(0)}
 }

pData-with-concatenation REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(transfer-service),
                                      concatenation(TRUE)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) transfer-realizations(11) pData-with-concatenation(0)}
 }

transfer-by-RTSE REALIZATION ::=
 {
RealizationParameter (WITH COMPONENTS{realization-type(transfer-service)})
IDENTIFIED BY {joint-iso-itu-t remote-operations(4) transfer-realizations(11) rTSE-transfer(0)}
 }


END -- end of the OSI realizations module
```

```
Remote-Operations-Abstract-Syntaxes {joint-iso-itu-t remote-operations(4) remote-operations-abstract-syntaxes(12)
version1(0)}
DEFINITIONS ::=
BEGIN
-- exports everything
IMPORTS OPERATION-PACKAGE FROM Remote-Operations-Information-Objects {joint-iso-itu-t remote-operations(4)
informationObjects(5) version1(0)}
        InvokeId, noInvokeId, ROS{}, Bind{}, Unbind{} FROM Remote-Operations-Generic-ROS-PDUs {joint-iso-itu-t
remote-operations(4) generic-ROS-PDUs(6) version1(0)}
        ACSE-apdu FROM ACSE-1 {joint-iso-ccitt association-control(2) modules(0) apdus(0) version1(1)} RTSE-apdus
        FROM Reliable Transfer-APDUs {joint-iso-ccitt reliable-transfer(3) apdus(0)}
        combine{}, AllOperations{}, ConsumerPerforms{}, SupplierPerforms{} FROM Remote-Operations-Useful-
Definitions {joint-iso-itu-t remote-operations(4) useful-definitions(7) version1(0)}
            APPLICATION-CONTEXT FROM Remote-Operations-Information-Objects-extensions {joint-iso-ccitt
remote-operations(4) informationObjects-extensions(8) version1(0)};

acse-abstract-syntax ABSTRACT-SYNTAX ::=
{
        ACSE-apdu IDENTIFIED BY {joint-iso-ccitt association-control(2) abstract-syntax(1) apdus(0) version1(1)}
}

rtse-abstract-syntax ABSTRACT-SYNTAX ::=
{
        RTSE-apdus IDENTIFIED BY {joint-iso-ccitt reliable-transfer(3) apdus(0)}
}

AllValues{APPLICATION-CONTEXT:ac} ::= CHOICE {
        bind{ac.&associationContract.&connection.&bind},
        unbind{ac.&associationContract.&connection.&unbind},
        ros-SingleAs
        {
            {ROSEInvokeIds},
            combine
            {
                {
                    ac.&associationContract.&OperationsOf
                |   ac.&associationContract.&InitiatorConsumerOf
                |   ac.&associationContract.&ResponderConsumerOf
                },
                {...},
                {}
            }
        }                                                                   }
ROS-SingleAS {InvokeId:ROSEInvokeIds, OPERATION-PACKAGE:package} ::=
        ROS{{ROSEInvokeIds}, {AllOperations{package}}, {AllOperations{package}}}

ROS-ConsumerAS {InvokeId:ROSEInvokeIds, OPERATION-PACKAGE:package} ::=
        ROS{{ROSEInvokeIds}, {ConsumerPerforms{package}},{SupplierPerforms{package}}}

ROS-SupplierAS {InvokeId:ROSEInvokeIds, OPERATION-PACKAGE:package} ::=
        ROS{{ROSEInvokeIds}, {SupplierPerforms{package}},{ConsumerPerforms{package}}}

ROSEInvokeIds InvokeId ::= {ALL EXCEPT noInvokeId}
END -- end of the remote-operations-abstract-syntaxes module
```

Annex C

Guidelines for the use of the notation

(This annex does not form an integral part of this Recommendation | International Standard)

The following is a consolidated example of the use of the notation defined in this Recommendation | International Standard.

This example is that of a simple client-server protocol. The client can invoke two asynchronous operations, an interrogation operation **get** and a modification operation **set**, of the server. These are defined as:

```
get OPERATION ::=
  {
  ARGUMENT      GetArgument
  RESULT        GetResult
  ERRORS        {get-error | general-error}
  CODE          local:1
  }

set OPERATION ::=
  {
  ARGUMENT      SetArgument
  RESULT        SetResult
  ERRORS        {set-error | general-error}
  CODE          local:2
  }
```

The errors these operations might report are **general-error, get-error** and **set-error**, which are defined below:

```
general-error ERROR ::=
  {
  PARAMETER     GeneralParameter
  CODE          local:1
  }

get-error ERROR ::=
  {
  PARAMETER     GetParameter
  CODE          local:2
  }

set-error ERROR ::=
  {
  PARAMETER     SetParameter
  CODE          local:3
  }
```

Next, two (operation) packages are defined, one involving the interrogation operation **get** and the other involving the modification operation **set**. These are:

```
interrogation-package OPERATION-PACKAGE ::=
  {
  CONSUMER INVOKES  {get}
  ID                global:{--some object identifier value--}
  }

modification-package OPERATION-PACKAGE ::=
  {
  CONSUMER INVOKES  {set}
  ID                global:{--some object identifier value--}
  }
```

These operations will be invoked over a dynamically established application association. In this example, the association will be established through the invocation of a bind operation, **client-bind**:

```
client-bind OPERATION ::=
 {
 ARGUMENT      BindArgument
 RESULT        BindResult
 ERROR         {bind-error}
 }

bind-error ERROR ::=
 {
 PARAMETER     BindErrorParameter
 }
```

In this example, no information is exchanged while unbinding; so the default **emptyUnbind** operation (see 10.3 of ITU-T Rec. X.880 | ISO/IEC 13712-1) is used to release the association.

The **connection-package** that will be used to set up the association is defined as:

```
connection-package CONNECTION-PACKAGE ::=
 {
 BIND                client-bind
 RESPONDER UNBIND    TRUE
 ID                  global:{--some object identifier value--}
 }
```

In this example, the server is allowed to unbind.

Next, the ROS-objects that participate in this interaction are defined together with the (association) contract that they engage in. The contract is defined as:

```
client-server-contract CONTRACT ::=
 {
 CONNECTION                  connection-package
 INITIATOR CONSUMER OF       {interrogation-package|modification-package}
 ID                          global:{--some object identifier value--}
 }
```

The **client** and **server** ROS-objects are defined as:

```
client ROS-OBJECT-CLASS ::=
 {
 INITIATES      {client-server-contract}
 ID             global:{--some object identifier value--}
 }

server ROS-OBJECT-CLASS ::=
 {
 RESPONDS       {client-server-contract}
 ID             global:{--some object identifier value--}
 }
```

The final step is to define the OSI realization of the abstract definitions provided above, This includes the application context definition for this protocol and its abstract syntax.

The abstract syntax of the data values exchanged by this protocol are defined as:

```
client-server-PCI ABSTRACT-SYNTAX ::=
 {
 CLIENT-SERVER-PDUs IDENTIFIED BY  global:{--some object identifier value--}
 }
```

where, using the definition of the generic ROS PDUs from ITU-T Rec. X.880 | ISO/IEC 13712-1:

```
CLIENT-SERVER-PDUs ::= ROS{{CS-InvokeIdSet}, {CS-Invokable}, {CS-Returnable}}

CS-Invokable OPERATION ::= {get | set}
CS-Returnable OPERATION ::= {get | set}
CS-InvokeIdSet ::= INTEGER (-128..127)
```

The application context is defined as:

```
client-server-context APPLICATION-CONTEXT ::=
  {
  CONTRACT                      client-server-contract
  ESTABLISHED BY                acse
  INFORMATION TRANSFER BY       pData
  ABSTRACT SYNTAXES             {acse-abstract-syntax | client-server-PCI}
  ID                            global:{--some object identifier value--}
  }
```

where:

```
acse-abstract-syntax ABSTRACT-SYNTAX ::=
  {
      ACSE-apdu IDENTIFIED BY {joint-iso-ccitt association-control(2) abstract-syntax(1) apdus(0) version1(1)}
  }
```

and

```
pData REALIZATION ::=
  {
  RealizationParameter (WITH COMPONENTS{realization-type(transfer-service)})
  IDENTIFIED BY {joint-iso-itu-t remote-operations(4) transfer-realizations(11) pData-without-concatenation(0)}
   }
```

Annex D

Assignment of object identifier values

(This annex does not form an integral part of this Recommendation | International Standard)

The following object identifier values are assigned in this Recommendation | International Standard:

Reference	Object Identifier Value Object Descriptor Value
Annex C	**{joint-iso-itu-t remote-operations(4) realizations(9) version1(0)}** The above is the identifier of the ASN.1 module defined in this Part which defines the permitted OSI association and transfer realizations. **{joint-iso-itu-t remote-operations(4) remote-operations-abstract-syntaxes(12) version1(0)}** The above is the identifier of the ASN.1 module defined in this Part which defines the abstract syntaxes to represent the APDUs of ROSE.
Clause 8	**{joint-iso-itu-t remote-operations(4) association-realizations(10) acse-without-concatenation(0)}** The above is the object identifier of ACSE association realization information object (without concatenation). **{joint-iso-itu-t remote-operations(4) association-realizations(10) acse-with-concatenation(1)}** The above is the object identifier of the ACSE association realization information object (permitting concatenation). **{joint-iso-itu-t remote-operations(4) association-realizations(10) association-by-rtse(2)}** The above is the object identifier of RTSE association realization information object.
Clause 9.2	**joint-iso-itu-t remote-operations(4) transfer-realizations(11) pData-without-concatenation(0)}** The above is the object identifier of the transfer realization information object when using P-DATA (without concatenation). **{joint-iso-itu-t remote-operations(4) transfer-realizations(11) pData-with-concatenation(1)}** The above is the object identifier of the transfer realization information object when using P-DATA (with concatenation).
Clause 9.3	**{joint-iso-itu-t remote-operations(4) transfer-realizations(11) rTSE-transfer(2)}** The above is the object identifier of the transfer realization information object when using RT-TRANSFER service.