INTERNATIONAL STANDARD

ISO/IEC 9072-1

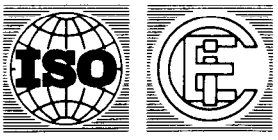
First edition
1989-11-15

Information processing systems — Text communication — Remote Operations —

Part 1 :
Model, notation and service definition

Systèmes de traitement de l'information — Communication de texte — Opérations à distance —

Partie 1 : Modèle, notation et définition du service



Reference number
ISO/IEC 9072-1 : 1989 (E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Contents

Page

Foreword iii

Introduction iv

1 Scope 1

2 Normative references 1

3 Definitions 1

4 Abbreviations 3

5 Conventions 3

6 Remote Operations Model 4

7 Overview of notation and service 8

8 Relationship with other ASEs and lower layer services 9

9 Remote Operations notation 10

10 Service definition 15

11 Mapping of notation on service 21

12 Sequencing information 23

Annexes

A Notation supporting the specification of Application-service-elemnts and application-contexts 26

B Guidelines for application protocol designers on the use of ROSE 30

© ISO/IEC 1989
All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.
ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Foreword

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

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for approval before their acceptance as International Standards. They are approved in accordance with procedures requiring at least 75 % approval by the national bodies voting.

International Standard ISO/IEC 9072-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology.*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Introduction

This part of ISO/IEC 9072 defines a notation and the services provided by an application-service-element - the Remote Operations Service Element (ROSE) - to support interactive applications in a distributed open systems environment. This part of ISO 9072 is one of a set of International Standards defining sets of application-service-elements commonly used by a number of applications.

Interactions between entities of a distributed application are modeled as Remote Operations, and defined using a Remote Operations notation. A Remote Operation is requested by one entity; the other entity attempts to perform the Remote Operation and then reports the outcome of the attempt. Remote Operations are supported by the ROSE.

This part of ISO/IEC 9072 is technically aligned with CCITT Recommendation X.219.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Information processing systems -Text communication - Remote Operations - Part 1: Model, notation and service definition

1 Scope

This part of ISO/IEC 9072 defines emote Operation (RO-) notation for defining t... ervices provided to interactive applications. This part of ISO/IEC 9072 also defines the services provided by the Remote Operation Service Element (ROSE) services. The ROSE services are provided by the use of the ROSE protocol (part 2 of ISO/IEC 9072) in conjunction with the Association Control Service Element (ACSE) services (ISO 8649) and the ACSE protocol (ISO 8650), optionally the Reliable Transfer Service Element (RTSE) services (ISO/IEC 9066-1) and the RTSE protocol (ISO/IEC 9066-2), and the presentation-service (ISO 8822).

No requirement is made for conformance to this part of ISO/IEC 9072.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9072. At the time of publication, the editions were valid. All Standards are subject to revision, and parties to agreement based on this part of ISO/IEC 9072 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of ISO and IEC maintain Registers of currently valid International Standards.

ISO 7498: 1984, *Information processing systems - Open Systems Interconnection - Basic Reference Model.*

ISO/TR 8509: 1987, *Information processing systems - Open Systems Interconnection - Service Conventions.*

ISO 8649: 1988, *Information processing systems - Open Systems Interconnection - Service definition for the Association Control Service Element.*

ISO 8650: 1988, *Information processing systems - Open Systems Interconnection - Protocol specification for the Association Control Service Element.*

ISO 8822: 1988, *Information processing systems - Open Systems Interconnection - Connection oriented presentation service definition.*

ISO 8824: 1987, *Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).*

ISO 8825: 1987, *Information processing systems - Open Systems Interconnection - Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1).*

ISO/IEC 9066-1: 1989, *Information processing systems - Text communication - Reliable Transfer - Part 1: Model and service definition.*

ISO/IEC 9066-2: 1989, *Information processing systems - Text communication - Reliable Transfer - Part 2: Protocol specification.*

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

3 Definitions

3.1 Reference Model definitions

This part of ISO/IEC 9072 is based on the concepts developed in ISO 7498 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 Layer;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

h) presentation-service;

i) presentation-connection;

j) session-service;

k) session-connection

l) transfer syntax; and

m) user-element.

3.2 Service conventions definitions

This part of ISO/IEC 9072 makes use of the following terms defined in ISO/TR 8509:

a) service-provider;

b) service-user;

c) confirmed service;

d) non-confirmed service;

e) provider-initiated service;

f) service-primitive; primitive;

g) request (primitive);

h) indication (primitive);

i) response (primitive); and

j) confirm (primitive).

3.3 Presentation service definitions

This part of ISO/IEC 9072 makes use of the following terms defined in ISO 8822:

a) abstract syntax;

b) abstract syntax name;

c) transfer syntax name;

d) presentation context.

3.4 Association control definitions

This part of ISO/IEC 9072 makes use of the following terms defined in ISO 8649:

a) application-association; association;

b) application context;

c) Association Control Service Element.

3.5 Reliable Transfer definitions

This part of ISO/IEC 9072 makes use of the following terms defined in ISO/IEC 9066-1:

a) Reliable Transfer Service Element.

3.6 ROSE definitions

For the purpose of this part of ISO/IEC 9072 the following definitions apply:

3.6.1 association-initiating-application-entity; association-initiator: The application-entity that initiates the application-association.

3.6.2 association-responding-application-entity; association-responder: The application-entity that responds to the initiation of an application-association by another AE.

3.6.3 invoking-application-entity; invoker: The application-entity that invokes the Remote Operation.

3.6.4 performing-application-entity; performer: The application-entity that performs a Remote Operation invoked by the other application-entity.

3.6.5 requestor: The part of an application-entity that issues a request primitive for a particular ROSE service.

3.6.6 acceptor: The part of an application-entity that receives the indication primitive for a particular ROSE service.

3.6.7 linked-operations: A set of operations formed by one parent-operation and one or more child-operations.

3.6.8 parent-operation: An operation during the execution of which the performer may invoke linked child-operations to be performed by the invoker of the parent-operation.

3.6.9 child-operation: An operation which might be invoked by the performer of the linked parent-operation during the execution of the parent-operation, and which is performed by the invoker of the parent-operation.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.6.10 Remote Operations:

(1) A concept and notation supporting the specification of interactive communication between application-entities. This includes the Remote Operation Service Element and the mapping of the notation onto the service primitives of used application-service-elements.

(2) The set of bind-operations, unbind-operations and operations.

3.6.11 RO-notation: The notation used for the specification of Remote Operations, defined in this part of ISO/IEC 9072.

3.6.12 ACSE-user: The application-specific function that performs the mapping of the bind-operation and unbind-operation of the RO-notation onto ACSE.

3.6.13 Remote Operation Service Element: The application-service-element defined in this part of ISO/IEC 9072.

3.6.14 ROSE-provider: The provider of the Remote Operations Service Element services.

3.6.15 ROSE-user: The application-specific function that performs the mapping of the operations and errors of the RO-notation onto ROSE.

3.6.16 RTSE-user: The application-specific function that performs the mapping of the bind-operation and unbind-operation of the RO-notation onto RTSE.

3.6.17 operation-interface: The interface within an application entity between the user element and the application service elements, defined as a set of application service element services (Remote Operations) available to the user element in RO-notation.

4 Abbreviations

AE	application-entity
ACSE	Association Control Service Element
ASE	application-service-element
APDU	application-protocol-data-unit
OSI	Open Systems Interconnection
RO (or ROS)	Remote Operations
ROSE	Remote Operations Service Element
RT (or RTS)	Reliable Transfer
RTSE	Reliable Transfer Service Element

5 Conventions

This part of ISO/IEC 9072 defines services for the ROSE following the descriptive conventions defined in ISO/TR 8509. In clause 10, the definition of each ROSE service includes a table that lists the parameters of its primitives. For a given primitive, the presence of each parameter is described by one of the following values.

blank	not applicable
M	mandatory
U	user option
C	conditional
O	presence is an ROSE service-provider option

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6 Remote Operations Model

In the OSI environment, communication between application processes is represented in terms of communication between a pair of application entities (AEs) using the presentation service. Communication between some application-entities are inherently interactive. Typically, one entity requests that a particular operation be performed; the other entity attempts to perform the operation and then reports the outcome of the attempt. This clause introduces the concept of Remote Operations as a vehicle for supporting interactive applications.

The generic structure of an operation is an elementary request/reply interaction. Operations are carried out within the context of an application-association.

Figure 1 models this view.

Operations invoked by one AE (the invoker) are performed by the other AE (the performer). Operations may be classified according to whether the performer of an operation is expected to report its outcome:

- in case of success or failure (a result reply is returned if the operation is successful, an error reply is returned if the operation is unsuccessful);
- in case of failure only (no reply is returned if the operation is successful, an error reply is returned if the operation is unsuccessful);
- in case of success only (a result reply is returned if the operation is successful, no reply is returned if the operation is unsuccessful);
- or not at all (neither a result nor an error reply is returned, whether the operation was successful or not).

Operations may also be classified according to two possible operation modes: synchronous, in which the invoker requires a reply from the performer before invoking another operation; and asynchronous, in which the invoker may continue to invoke further operations without awaiting a reply.

The following Operation Classes are defined:

Operation Class 1: Synchronous, reporting success or failure (result or error).

Operation Class 2: Asynchronous, reporting success or failure (result or error).

Operation Class 3: Asynchronous, reporting failure (error) only, if any.

Operation Class 4: Asynchronous, reporting success (result) only.

Operation Class 5: Asynchronous, outcome not reported.

The Operation Class of each operation has to be agreed between application entities (e.g. in an Application Protocol International Standard).

In some cases it is useful to group operations into a set of linked-operations which is formed by one parent-operation and one or more child-operations. The performer of the parent-operation may invoke none, one, or more child-operations

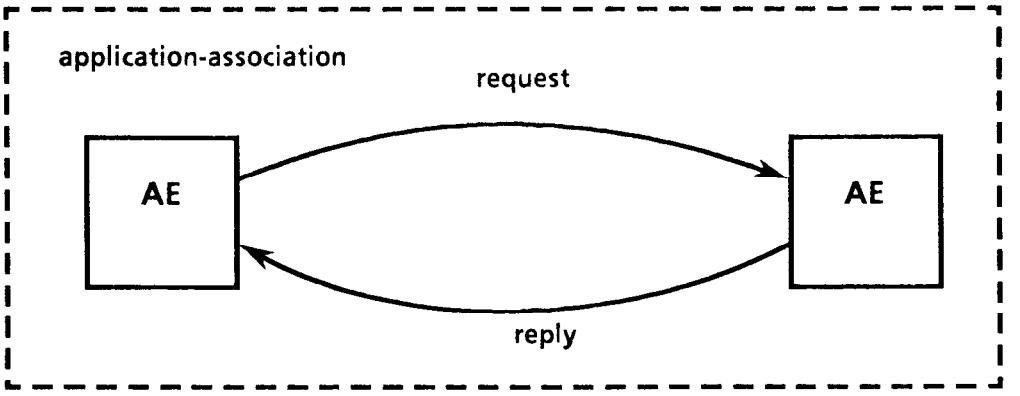


Figure 1 - Remote Operations Model

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

during the execution of the parent-operation. The invoker of the parent-operation is the performer of the child-operations. A child-operation may be a parent-operation of another set of linked-operations in a recursive manner. Figure 2 models this concept.

An application-association defines the relationship between a pair of AEs, and is formed by the exchange of application-protocol-control-information through the use of presentation-services. The AE that initiates an application-association is called the association-initiating AE, or the association-initiator, while the AE that responds to the initiation of an application-association by another AE is called the association-responding AE, or the association-responder. Only the association-initiating AE may release an established application-association.

Application-associations are classified by which application-entity is allowed to invoke operations:

Association Class 1: Only the association-initiating application entity can invoke operations.

Association Class 2: Only the association-responding application entity can invoke operations.

Association Class 3: Both the association-initiating and the association-responding application entities can invoke operations.

Linked-operations require Association Class 3.

The Association Class has to be agreed between application-entities (e.g. in an Application Protocol International Standard).

The functionality of an AE is factored into one user-element and a set of application-service-elements (ASEs). Each ASE may itself be factored into a set of (more primitive) ASEs. The interaction between AEs is described in terms of their use of ASEs.

The specific combination of a user-element and the set of ASEs which comprise an AE defines the application-context.

Figure 3 illustrates an example of an application-context involving the Remote Operations Service Element (ROSE). Note that this figure is not meant to imply that the application is symmetric. Interactive applications are often inherently asymmetric, that is, either one or both AEs may be permitted to invoke operations, and the operations that either AE may invoke may be different. The rules governing which AE may invoke operations, and which operations an AE may invoke, is defined using the RO-notation in an Application Protocol International Standard, and determines the application-context.

The set of ASEs available to the user element of the AE at the operation-interface is defined using

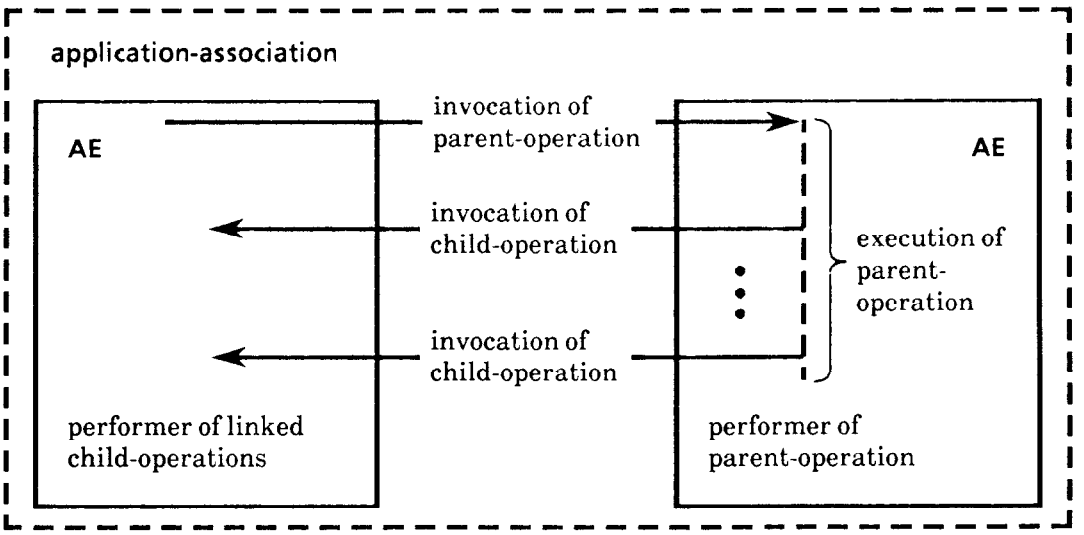


Figure 2 - Linked-operations

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

the Remote Operations (RO-) notation. The RO-notation is based on the macro concept defined in ISO 8824. The complexity of a particular set of ASEs is dependent upon the needs of the application, and is not limited by the Remote Operations concept.

An important characteristic of Remote Operations is that they provide applications with independence from OSI communication services. Since the notation is based on established object-oriented programming principles, automatic tools can be developed to bind Remote Operations into the execution environment of applications.

The ASEs available to the user-element require communication over an application-association. The control of that application-association (establishment, release, abort) is performed either by the Association Control Service Element (ACSE) defined in ISO 8649, or the Reliable Transfer Service Element defined in ISO/IEC 9066-1 and the Association Control Service Element (ACSE). Communication over the application-association is performed by the Remote Operations Service Element (ROSE) defined in this part of ISO/IEC 9072.

An application-specific function performs the mapping of the operations available to the user-element onto either the ACSE services, or the RTSE services; and the ROSE services. The mapping is defined in this part of ISO/IEC 9072. The function that performs the mapping of the operations onto the ACSE services, or the RTSE services, and the ROSE services is said to be the user of ACSE, RTSE and ROSE, or the ACSE-user, the RTSE-user, and the ROSE-user.

If the RTSE is included in the application-context, the mapping function is an RTSE-user and a ROSE-user, the ROSE is an RTSE-user, the RTSE is an ACSE-user and a presentation service-user, and the ACSE is a presentation service-user.

If the RTSE is excluded from the application-context, the mapping function is an ACSE-user and a ROSE-user, the ROSE is a presentation service-user, and the ACSE is a presentation service-user.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

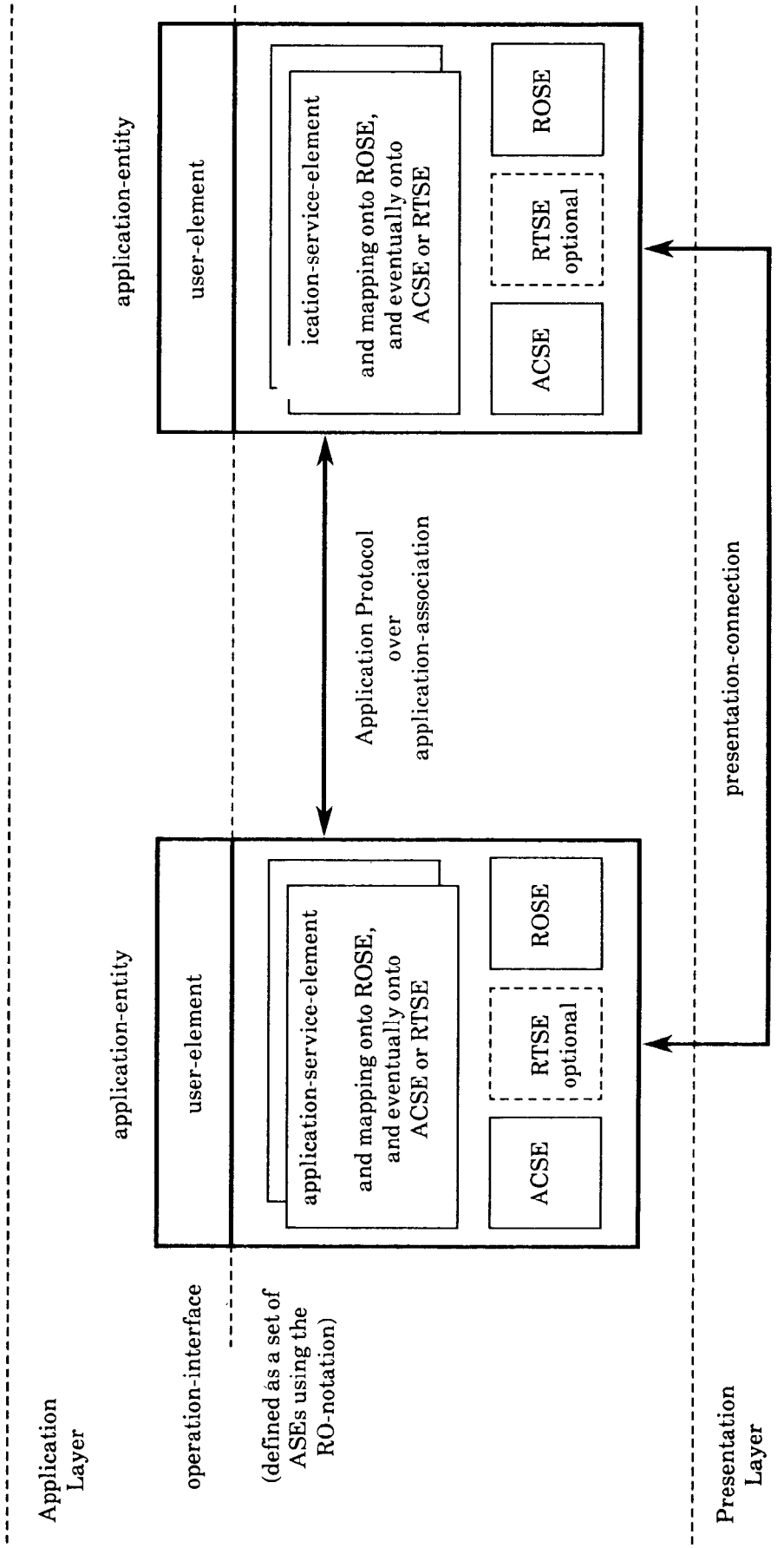


Figure 3 - Model of an application context involving Remote Operations

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7 Overview of notation and service

7.1 Notation Overview

This part of ISO/IEC 9072 defines the RO-notation for the specification of an application-context and the related abstract syntax component of the presentation context.

The functionality of an application-context is provided to the user-element by means of Remote Operations and errors which form the operation-interface.

The following types of Remote Operations form an operation interface:

- a bind-operation to establish an application-association;
- a set of operations and, for each operation, a list of error (negative reply) situations;
- an unbind-operation to release an application-association.

The abstract syntax notation of ISO 8824 is used for the definition of the following macros:

a) BIND;

b) UNBIND;

c) OPERATION; and

d) ERROR.

These macros provide both a type notation and a value notation for Remote Operations and errors.

The type notation of the BIND macro enables the specification of a bind-operation type and the types for user data values (if any) to be exchanged in the establishment phase of an application-association. The value notation of the BIND macro enables the specification of user data values (if any) to be exchanged in the establishment phase of an application-association. The type notation of the UNBIND macro enables the specification of an unbind-operation type and types for user data values (if any) to be exchanged in the release phase of an application-association. The value notation of the UNBIND macro enables the specification of user data values (if any) to be exchanged in the release phase of an application-association.

The type notation of the OPERATION macro enables the specification of an operation and user data types to be exchanged for a request and a positive reply. In addition , the type notation enables the specification of a list of valid negative reply situations. If the operation is a parent-operation, the type notation enables the specification of the list of linked child-operations. The value notation of the OPERATION macro enables the specification of the identifier of an operation.

The type notation of the ERROR macro enables the specification of user data types to be exchanged in a negative reply situation. The value notation of the ERROR macro enables the specification of the identifier of an error.

Additional macros supporting the notation for the specification of application-service-elements and application contexts are defined in annex A.

7.2 Service overview

This part of ISO/IEC 9072 defines the following ROSE services:

a) RO-INVOKE

b) RO-RESULT

c) RO-ERROR

d) RO-REJECT-U

e) RO-REJECT-P

The RO-INVOKE service enables an invoking AE to request an operation to be performed by the performing AE.

The RO-RESULT service enables the performing AE to return the positive reply of a successfully performed operation to the invoking AE.

The RO-ERROR service enables the performing AE to return the negative reply of an unsuccessfully performed operation to the invoking AE.

The RO-REJECT-U service enables one AE to reject the request or reply of the other AE if the ROSE-user has detected a problem.

The RO-REJECT-P service enables the ROSE-user to be informed about a problem detected by the ROSE-provider.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3 Mapping of notation on to services

Note that the function that performs the mapping of the OPERATION macros and ERROR macros of the RO-notation onto ROSE services is said to be the ROSE-user. While the function that performs the mapping of the BIND and UNBIND macros of the RO-notation onto ACSE services or RTSE services respectively is said to be the ACSE-user or RTSE-user respectively.

The specification of the mapping of the RO-notation onto the used services of ACSE, RTSE, and ROSE is given in clause 11. Therefore International Standards using the RO-notation for the protocol specification need not to specify the mapping onto these used services.

8 Relationship with other ASEs and lower layer services

8.1 Other application-service-elements

The ROSE is intended to be used with other ASEs in order to support specific interactive information processing tasks. Therefore it is expected that the ROSE will be included in a large number of application-context specifications.

The collection of the ROSE and other ASEs included in an application context are required to use the facilities of the presentation-service in a co-ordinated manner among themselves.

The ROSE requires an existing application-association controlled by ACSE.

For some application context specifications a Reliable Transfer Service Element (RTSE) is included.

An ROSE-user protocol specification uses the RO-notation. It defines one or more abstract syntaxes and provides unique abstract syntax names of type object identifier for each abstract syntax.

If a named abstract syntax specifies operations and errors, the ROSE APDUs defined in ISO/IEC 9072-2 are included in that named abstract syntax. If multiple named abstract syntaxes are defined for operations and errors, the ROSE APDUs are included in each named abstract syntax.

If a named abstract syntax specifies a bind-operation, the APDUs specified by the value notation of the BIND macro are included in that named abstract syntax. If the RTSE is included in the application context, the APDUs for the bind-operation share a single named abstract syntax with the RTSE APDUs defined in ISO/IEC 9066-2.

If a named abstract syntax specifies an unbind-operation, the APDUs specified by the value notation of the UNBIND macro are included in that named abstract syntax.

The APDUs resulting from the specification of a bind-operation, an unbind-operation, operations and errors and the RTSE APDUs may share a single named abstract syntax.

8.2 Presentation-service

If an application context including RTSE and ROSE is defined, ROSE services do not use the presentation-service.

If an application context including ROSE but excluding RTSE is defined, the ROSE services require access to the P-DATA service and require the use of the duplex functional unit of the presentation-service. The ROSE services neither use, nor constrain the use of, any other presentation service.

A named abstract syntax associated with a compatible transfer syntax (negotiated by the Presentation Layer) constitutes a presentation context.

The object identifier value {joint-iso-ccitt asn1(1) basic-encoding(1)} specified in ISO 8825 may be used as a transfer syntax name. In this case the ROSE-user protocol specification need not to name and specify a transfer syntax.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9 Remote Operations notation

9.1 General

The notation used in this part of ISO/IEC 9072 is defined as follows:

- the data syntax notation and macro notation are defined in ISO 8824;
- the Remote Operation macros are defined in 9.2 of this part of ISO/IEC 9072.

A bind-operation defines where an object binding (establishment of an application-association) begins. If such a binding is established, operations may be invoked. An unbind-operation defines where an object binding is released.

An interactive protocol is specified using the Remote Operation and error data types. This clause defines those types. It also explains the notational definitions of a particular Remote Operation, and of the particular errors it can report. The notation is defined by means of the macro facility defined in ISO 8824. This macro definition allows a generalized specification of the mapping onto various execution environments.

The macros enabling the specification of bind-operations, unbind-operations, operations and errors are listed in figure 4.

9.2 Specification of bind-operations

A single data value, the argument of the bind-operation, may accompany the request to establish the application-association. Some bind-operations report their outcome, whether success (i.e the normal outcome) or failure (i.e. the exceptional outcome). Other bind-operations report their outcome only if they fail, and still others never at all. A single data value, the result of the bind-operation, may accompany the positive response. A single data value, the bind-error of the bind-operation, may accompany the negative response.

The notation for a bind-operation type is the keyword **BIND**, optionally followed by the keyword **ARGUMENT** and the type of the bind-operation's argument, the reference name optionally assigned to it, and the nature of the operation's outcome reporting (if any). If the bind-operation reports success, the keyword **RESULT** and the type of its result and the reference name optionally assigned to it are specified. If the bind-operation reports failure, the keyword **BIND-ERROR** and the type of the error-information it reports and the reference name optionally assigned to it are specified.

The value notation for a bind-operation is either an argument value, or a result value or an error value. The value notation for an argument value (if any) is the key word **ARGUMENT** followed by a value of the argument type. The value notation for a result value (if any) is the key work **RESULT** followed by a value of the result type. The value notation for an error value (if any) is the key word **ERROR** followed by a value of the error type.

9.3 Specification of unbind-operations

A single data value, the argument of the unbind-operation, may accompany the request to release the application-association. Some unbind-operations report their outcome, whether success (i.e the normal outcome) or failure (i.e. the exceptional outcome). Other unbind-operations report their outcome only if they fail, and still others never at all. A single data value, the result of the unbind-operation, or a single data value, the unbind-error of the unbind-operation, may accompany the response.

The notation for an unbind-operation type is the keyword **UNBIND**, optionally followed by the keyword **ARGUMENT** and the type of the unbind-operation's argument, the reference name optionally assigned to it, and the nature of the unbind-operation's outcome reporting (if any). If the unbind-operation reports success, the keyword **RESULT** and the type of its result and the reference name optionally assigned to it are specified. If the Unbind-operation reports failure, the keyword **UNBIND-ERROR** and the type of the error-information it reports and the reference name optionally assigned to it are specified.

The value notation for an unbind-operation is either an argument value, or a result value or an error value. The value notation for an argument value (if any) is the keyword **ARGUMENT** followed by a value of the argument type. The value notation for a result value (if any) is the keyword **RESULT** followed by a value of the result type. The value notation for an error value (if any) is the keyword **ERROR** followed by a value of the error type.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
Remote-Operation-Notation { joint-iso-ccitt remote-operations(4) notation (0) }
DEFINITIONS ::=
BEGIN

EXPORTS BIND, UNBIND, OPERATION, ERROR;

-- macro definition for bind-operations

BIND MACRO ::=
BEGIN

TYPE NOTATION          ::= Argument Result Error
VALUE NOTATION         ::= Argument-value | Result-value | Error-value

Argument        ::= empty | "ARGUM       Name type (Argument-type)
                          -- Expects any ASN.1 type and assigns it to the variable Argument
                          -- type
Result          ::= empty | "RESULT" Name type (Result-type)
                          -- Expects any ASN .1 type and assigns it to the variable Result-type
Error           ::= empty | "BIND-ERROR" Name type (Error-type)
                          -- Expects any ASN.1 type and assigns it to the variable Error-type
Name                ::= empty | identifier
Argument-value  ::= empty | "ARGUMENT" value (Arg-value Argument-type)
                          -- Expects a value for the type in Argument-type, and assigns it to the
                          -- variable Arg-value
                          <VALUE [16] EXPLICIT Argument-type ::= Arg-value>
                          -- Returns the final value as explicitly tagged type
Result-value    ::= empty | "RESULT" value (Res-value Result-type)
                          -- Expects a value for the type in Result-type and assigns it to the
                          -- variable Res-value
                          <VALUE [17] EXPLICIT Result-type ::= Res-value>
                          -- Returns the final value as explicitly tagged type
Error-value     ::= empty | "ERROR" value (Err-value Error-type)
                          -- Expects a value for the type in Error-type, and assigns it to the
                          -- variable Err-value
                          <VALUE [18] EXPLICIT Error-type ::= Err-value>
                          -- Returns the final value as explicitly tagged type
END

-- Remote Operations Notation continued
```

Figure 4 (Part 1 of 3) - Formal definition of Remote Operations data types

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
-- Remote Operations Notation continued
-- macro definition for unbind-operations

UNBIND MACRO ::=

BEGIN

TYPE NOTATION          ::= Argument Result Errors
VALUE NOTATION         ::= Argument-value | Result-value | Error-value

Argument        ::= empty | "ARGUMENT" Name type (Argument-type)
                        -- E.    s any ASN.1 type and assigns it to the variable Argument-
                        -- type
Result          ::= empty | "RESULT" Name type (Result-type)
                        -- Expects any ASN .1 type and assigns it to the variable Result-type
Error           ::= empty | "UNBIND-ERROR" Name type (Error-type)
                        -- Expects any ASN.1 type and assigns it to the variable Error-type
Name            ::= empty | identifier
Argument-value  ::= empty | "ARGUMENT" value (Arg-value Argument-type)
                        -- Expects a value for the type in Argument-type, and assigns it to the
                        -- variable Arg-value
                        <VALUE [19] EXPLICIT Argument-type ::= Arg-value>
                        -- Returns the final value as explicitly tagged type
Result-value    ::= empty | "RESULT" value (Res-value Result-type)
                        -- Expects a value for the type in Result-type and assigns it to the
                        -- variable Res-value
                        <VALUE [20] EXPLICIT Result-type ::= Res-value>
                        -- Returns the final value as explicitly tagged type
Error-value     ::= empty | "ERROR" value (Err-value Error-type)
                        -- Expects a value for the type in Error-type, and assigns it to the
                        -- variable Err-value
                        <VALUE [21] EXPLICIT Error-type ::= Err-value>
                        -- Returns the final value as explicitly tagged type
END

-- Remote Operations Notation continued
```

Figure 4 (Part 2 of 3) - Formal definition of Remote Operations data types

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.4 Specification of operations

A data value of type operation represents the identifier for an operation that a ROSE-user in one open system may request to be performed by a peer ROSE-user in another open system. A single data value, the argument of the operation, may accompany the request. Some operations report their outcome, whether success (i.e the normal outcome) or failure (i.e. the exceptional outcome). Other operations report their outcome only if they fail, and still others never at all. A single data value, the result of the operation, accompanies a report of success; a report of failure id ies the exceptional condition that was encount

The notation for an operation type is the keyword **OPERATION**, optionally followed by the keyword **ARGUMENT** and the type of the operation's argument, the reference name optionally assigned to it, and the nature of the operation's outcome reporting (if any). If the operation reports success, the keyword **RESULT** and optionally the type of its result and the reference name optionally assigned to it are specified. If the operation reports failure, the keyword **ERRORS** and the reference names of the error values or error types it reports are specified. If the operation is the parent-operation of a set of linked-operations, the keyword **LINKED-OPERATIONS** and the reference names of the linked child-operation values or child-operation types are specified. The reference to error values or child-operation values is preferred, however the references to types shall be used if the values are defined elsewhere (see 9.6).

The notation for an operation value is the operation's identifier. If a locally unique identifier (local value) is sufficient, the identifier is of type **INTEGER**. If a globally unique identifier (global value) is required to allow the unique identification of operations used in several abstract syntaxes, the identifier is of type **OBJECT IDENTIFIER**.

Child-operations and errors referenced by a specific operation shall share a single named abstract syntax (see 8.1) with that operation, if the child-operations or errors are identified by local values. The use of global values is not restricted.

9.5 Specification of errors

A data value of type error represents the identifier for an exception condition that a ROSE-user in one open system may report to a peer ROSE-user in another open system, where the exception condition is reporting an exceptional outcome of a previously requested operation. A single data value, the parameter of the error, may accompany the report.

The notation for an error type is the keyword **ERROR**, optionally followed by the keyword **PARAMETER** and the type of the error's parameter and the reference name optionally assigned to it.

The notation for an error value is the error's identifier. If a locally unique identifier (local value) is sufficient, the identifier is of type **INTEGER**. If a globally unique identifier (global value) is required to allow the unique identification of errors used in several abstract syntaxes, the identifier is of type **OBJECT IDENTIFIER**.

9.6 Export and import of operations and errors

Operation values and error values have to be unique within a named abstract syntax. If operations and errors are specified in several ASN.1 modules and are imported to a module specifying a specific named abstract syntax, one of the following rules apply:

a) If local values are used and exported, it is in the responsibility of the designer of the importing module to ensure uniqueness.

b) A module may specify and export operation types and error types. The operation values and error values are assigned in the module importing the types. A single value shall be assigned for each operation type or error type.

c) If global values are assigned and exported, uniqueness is ensured.

However different named abstract syntaxes might be used for conflicting local values.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
-- Remote Operations Notation continued
-- macro definition for operations

OPERATION MACRO ::=
BEGIN
TYPE NOTATION       ::= Argument Result Errors LinkedOperations
VALUE NOTATION      ::= value (VALUE CHOICE{
                                    localValue    INTEGER,
                                    globalValue   OBJECT IDENTIFIER})

Argument            ::= "ARGUMENT" NamedType | empty

Result              ::= "RE!       ResultType | empty

ResultType          ::= NamedType | empty

Errors              ::= "ERRORS" "{" ErrorNames "}" | empty

LinkedOperations    ::= "LINKED" "{" LinkedOperationNames "}" | empty

ErrorNames          ::= ErrorList | empty

ErrorList           ::= Error | ErrorList "," Error

Error               ::= value ( ERROR )    -- shall reference an error value
                      | type     -- shall reference an error type if no error value is specified

LinkedOperationNames ::= OperationList | empty

OperationList       ::= Operation | OperationList "," Operation

Operation           ::= value ( OPERATION )    -- shall reference an operation value
                      | type     -- shall reference an operation type if no operation value is
                                 -- specified

NamedType           ::= identifier type | type

END

-- macro definition for operations errors

ERROR MACRO ::=
BEGIN
TYPE NOTATION       ::= Parameter
VALUE NOTATION      ::= value (VALUE CHOICE{
                                    localValue    INTEGER,
                                    globalValue   OBJECT IDENTIFIER})

Parameter           ::= "PARAMETER " NamedType | empty

NamedType           ::= identifier type | type

END


END         -- end of Remote Operations Notation
```

Figure 4 (Part 3 of 3) - Formal definition of Remote Operations data types

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10 Service definition

The ROSE services are listed in table 1.

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

Identification of the named abstract syntax in use is assumed for all ROSE services, however this is a local matter and outside the scope of this part of ISO/IEC 9072.

10.1 RO-INVOKE service

The RO-INVOKE service is used by one ROSE-user (the invoker) to cause the invocation of an operation to be performed by the other ROSE-user (the performer). This service is an non-confirmed service.

The related service structure consists of two service-primitives, as illustrated in figure 5.

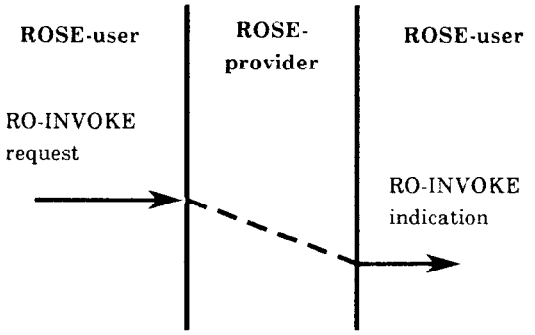


Figure 5 - RO-INVOKE service-primitives

10.1.1 RO-INVOKE parameters

Table 2 lists the RO-INVOKE service parameters.

10.1.1.1 Operation-value

This parameter is the identifier of the operation to be invoked. The value has to be agreed between the ROSE-users. This parameter has to be supplied by the requestor of the service.

Table 2 - RO-INVOKE parameters

Parameter name	Req	Ind
Operation-value	M	M(=)
Operation-class	U	
Argument	U	C(=)
Invoke-ID	M	M(=)
Linked-ID	U	C(=)
Priority	U	

10.1.1.2 Operation-class

This parameter defines whether a synchronous or an asynchronous reply is expected and the nature of the expected reply, i.e. result and/or error or none (see clause 6). This parameter has to be supplied by the requestor of the service. This parameter is used solely to optimize the turn management (see 8.1.1 of part 2 of ISO/IEC 9072).

10.1.1.3 Argument

This parameter is the argument of the invoked operation. The type has to be agreed between the ROSE-users. This parameter has to be supplied by the requestor of the service.

10.1.1.4 Invoke-ID

This parameter identifies the request of a RO-INVOKE service and is used to correlate this request with the corresponding replies (RO-RESULT, RO-ERROR, RO-REJECT-U, and RO-REJECT-P services) or the invocation of linked child-operations (RO-INVOKE). This parameter has to be supplied by the requestor of the service.

This parameter distinguishes several requests of the service the requestor may have in progress (asynchronous operations). The requestor may begin to reuse Invoke-ID values whenever it chooses, subject to the constraint that it may not reuse an Invoke-ID value that was previously assigned to a request of the service for which it expects, but has not yet received, a reply or the invocation of a linked child-operation.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The ROSE-user to which an RO-INVOKE indication is issued, assumes that an Invoke-ID value violating the above rule is a duplicate; and therefore, it does not perform the invoked operation. Instead, it rejects the duplicate invocation.

If Operation Classes 3, 4 or 5 are used, the requestor of this service may reuse an Invoke-ID value after a reasonably long period of time, or if the reply is carried by other means (e.g. result of a have-you-finished operation).

In some application contexts peer R -users may communicate Invoke-ID values. To support this the type of the Invoke-ID parameter is exported by the module defining the abstract syntax of Remote Operations in clause 9 of part 2 of ISO/IEC 9072.

10.1.1.5 Linked-ID

If this parameter is present, the invoked operation is a child-operation and the parameter identifies the invocation of the linked parent-operation. This parameter has to be supplied by the requestor of the service. The value is that of the invoke-ID parameter of the RO-INVOKE indication primitive of the parent operation.

10.1.1.6 Priority

This parameter defines the priority assigned to the transfer of the corresponding APDU with respect to the other APDUs to be exchanged between the AEs. The lower the value, the higher the priority. If several APDUs with the same priority are awaiting transfer, they are transferred "first in, first out".

NOTES

1 The Priority parameter has an effect in the case of a two-way alternate association in that it prioritizes the sending of APDUs, and may be used to determine when to request the Turn to send APDUs. The Priority parameter may also have a local effect in the case of a two-way simultaneous association.

2 The Priority of a reply (RO-RESULT, RO-ERROR, and RO-REJECT-U) should normally be higher (lower in value) than the priority of the corresponding invocation.

10.2 RO-RESULT service

The RO-RESULT service is used by a ROSE-user to reply to a previous RO-INVOKE indication in the case of a successfully performed operation. This service is an non-confirmed service.

The related service structure consists of two service-primitives, as illustrated in figure 6.

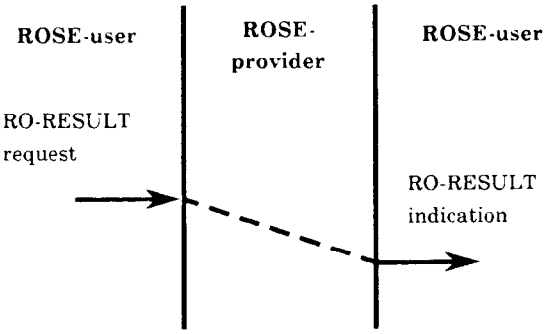


Figure 6 - RO-RESULT service-primitives

10.2.1 RO-RESULT parameters

Table 3 lists the RO-RESULT service parameters.

Table 3 - RO-RESULT parameters

Parameter name	Req	Ind
Operation-value	U	C(=)
Result	U	C(=)
Invoke-ID	M	M(=)
Priority	U	

10.2.1.1 Operation-value

This parameter is the identifier of an invoked and successfully performed operation. This parameter has to be supplied by the requestor of the service. The value is that of the corresponding RO-INVOKE indication primitive. This parameter shall be present only if the Result parameter is present.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.2.1.2 Result

This parameter is the result of an invoked and successfully performed operation. The type has to be agreed between the ROSE users. This parameter has to be supplied by the requestor of the service.

10.2.1.3 Invoke-ID

This parameter identifies the corresponding invocation (see 10.1.1.4). This parameter has to be supplied by the requestor of the service. The value is that of the corresponding RO VOKE indication primitive.

10.2.1.4 Priority

This parameter defines the priority assigned to the transfer of the corresponding APDU (see 10.1.1.6).

10.3 RO-ERROR service

The RO-ERROR service is used by a ROSE-user to reply to a previous RO-INVOKE indication in the case of an unsuccessfully performed operation. This service is an non-confirmed service.

The related service structure consists of two service-primitives as illustrated in Figure 7.

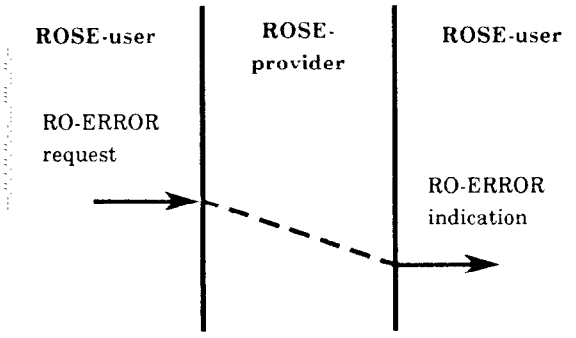


Figure 7 - RO-ERROR service-primitives

10.3.1 RO-ERROR parameters

Table 4 lists the RO-ERROR service parameters.

Table 4 - RO-ERROR parameters

Parameter name	Req	Ind
Error-value	M	M(=)
Error-parameter	U	C(=)
Invoke-ID	M	M(=)
Priority	U	

10.3.1.1 Error-value

This parameter identifies the error that occurred during execution of the operation. The value has to be agreed between the ROSE-users. This parameter has to be supplied by the requestor of the service.

10.3.1.2 Error-parameter

This parameter provides additional information about the error. The type (if any) has to be agreed between the ROSE-users. This parameter has to be supplied by the requestor of the service.

10.3.1.3 Invoke-ID

This parameter identifies the corresponding invocation (see 10.1.1.4). This parameter has to be supplied by the requestor of the service. The value is that of the corresponding RO-INVOKE indication primitive.

10.3.1.4 Priority

This parameter defines the priority assigned to the transfer of the corresponding APDU (see 10.1.1.6).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.4 RO-REJECT-U

The RO-REJECT-U service is used by a ROSE-user to reject a request (RO-INVOKE indication) of the other ROSE-user if it has detected a problem. The RO-REJECT-U service may also be used by a ROSE-user to reject a reply (RO-RESULT indication, RO-ERROR indication) from the other ROSE-user. However, to avoid violating the sequencing rules of other ASEs in some application contexts, a ROSE-user may choose not to use the RO-REJECT-U service to reject replies. This service is an non-confirmed service.

The related service structure consists of two service-primitives, as illustrated in figure 8.

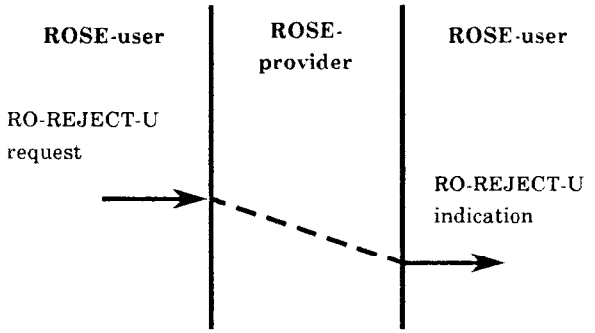


Figure 8 - RO-REJECT-U service-primitives

10.4.1 RO-REJECT-U parameters

Table 5 lists the RO-REJECT-U service parameters.

Table 5 - RO-REJECT-U parameters

Parameter name	Req	Ind
Reject-reason	M	M(=)
Invoke-ID	M	M(=)
Priority	U	

10.4.1.1 Reject-reason

This parameter specifies the reason for rejection as follows:

a) Invoke-problem: user-reject of an RO-INVOKE indication primitive with values:

- duplicate-invocation: signifies that the invoke-ID parameter violates the assignment rules of 10.1.1.4;
- unrecognized-operation: signifies that the operation is not one of those agreed between the ROSE-users;
- mistyped-argument: signifies that the type of the operation argument supplied is not that agreed between the ROSE-users
- resource-limitation: the performing ROSE-user is not able to perform the invoked operation due to resource limitation;
- initiator-releasing: the association-initiator is not willing to perform the invoked operation because it is about to attempt to release the application-association;
- unrecognized-linked-ID: signifies that there is no operation in progress with an invoke-ID equal to the specified linked-ID;
- linked-response-unexpected: signifies that the invoked operation referred to by the linked-ID is not a parent-operation
- unexpected-child-operation: signifies that the invoked child-operation is not one that the invoked parent-operation referred to by the linked-ID allows.

b) Return-result-problem: user-reject of an RO-RESULT indication primitive with values:

- unrecognized-invocation: signifies that no operation with the specified invoke-ID is in progress;
- result-response-unexpected: signifies that the invoked operation does not report a result;
- mistyped-result: signifies that the type of the result parameter supplied is not that agreed between the ROSE-users.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) Return-error-problem: user-reject of an RO-ERROR indication primitive with values:

- unrecognized-invocation: signifies that no operation with the specified Invoke-ID is in progress;
- error-response-unexpected: signifies that the invoked operation does not report failure;
- unrecognized-error: signifies that the reported error is not one of those agreed between the ROSE-users;
- unexpected-error: signifies that the reported error is not one that the invoked operation may report;
- mistyped-parameter: signifies that the type of the error parameter supplied is not that agreed between the ROSE-user.

This parameter has to be supplied by the requestor of the service.

10.4.1.2 Invoke-ID

This parameter identifies the corresponding invocation (see 10.1.1.4). This parameter has to be supplied by the requestor of the service. The value is that of the rejected RO-INVOKE indication, RO-RESULT indication, or RO-ERROR indication primitive.

10.4.1.3 Priority

This parameter defines the priority assigned to the transfer of the corresponding APDU (see 10.1.1.6).

10.5 RO-REJECT-P

The RO-REJECT-P service is used to advise a ROSE-user of a problem detected by the ROSE-provider. This service is a provider-initiated service.

The related service structure consists of a single service-primitive, as illustrated in figure 9.

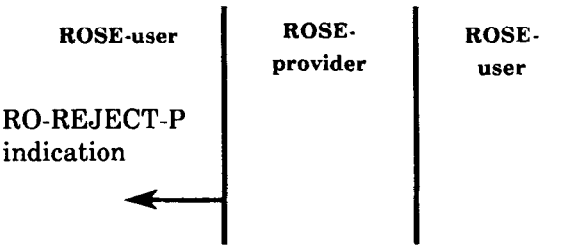


Figure 9 - RO-REJECT-P service-primitive

10.5.1 RO-REJECT-P parameters

Table 6 lists the RO-REJECT-P service parameters.

Table 6 - RO-REJECT-P parameters

Parameter name	Ind
Invoke-ID	O
Returned-parameters	O
Reject-reason	O

10.5.1.1 Invoke-ID

This parameter identifies the corresponding invocation (see 10.1.1.4). This parameter is supplied by the ROSE-provider. The value is that of the rejected RO-INVOKE request, RO-RESULT request, RO-ERROR request or RO-REJECT-U request primitives. This parameter may be omitted if an invoke-ID is not available.

10.5.1.2 Returned-parameters

This parameter contains the parameters of the RO-INVOKE request, RO-RESULT request, RO-ERROR request or RO-REJECT-U request primitives, if the corresponding APDU could not be transferred by the ROSE-provider. This parameter and the parameter reject-reason are mutually exclusive.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.5.1.3 Reject-reason

This parameter specifies the reason for rejection as follows:

d) General-problem: provider-reject of an APDU with values:

- unrecognized-APDU: signifies that the type of the APDU, as evidenced by its type identifier, is not one of the four defined by ISO/IEC 9072-2;
- mistyped-APDU: signifies that the structure of the APDU does not conform to ISO/IEC 9072-2;
- badly-structured-APDU: signifies that the structure of the APDU does not conform to the standard notation and encoding, defined in ISO 8824 and ISO 8825.

This parameter is supplied by the ROSE-provider. This parameter and the parameter returned-parameters are mutually exclusive.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11 Mapping of notation on service

11.1 Application context and operations

This clause describes how an application-context is specified by means of the notation provided by the macros defined in clause 9.

Such an application-context specification consists of

a) a bind-operation specified by means of the BIND macro, and

b) an unbind-operation specified eans of the UNBIND macro, and

c) a set of operations specified by means of the OPERATION macro, and

d) a set of errors related to operations and specified by means of the ERROR macro.

The Remote Operations (i.e. bind-operation, unbind-operation and operations) may be invoked by the user-element.

An association-initiating user-element establishes an application-association by invoking a bind-operation. If the application-association is established, operations may be invoked by the user-element. When the association-initiating user-element wishes to release the application-association, it invokes an unbind-operation.

11.2 Mapping of Remote Operations on ACSE services, RTSE services, and ROSE services

The bind-operation and the unbind-operation are mapped either on ACSE services, or the RTSE services.

The operations are mapped on the ROSE services.

11.2.1 Mapping on ACSE services

The bind-operation is mapped on the A-ASSOCIATE service and the unbind-operation mapped on the A-RELEASE service.

11.2.1.1 Mapping of a bind-operation

A bind-operation is mapped on the A-ASSOCIATE service.

11.2.1.1.1 Invocation of a bind-operation

The invocation of a bind-operation is mapped on the A-ASSOCIATE request and A-ASSOCIATE indication service primitives.

The argument value of the bind-operation is mapped on the user information parameter of the service primitives.

11.2.1.1.2 Reply of a bind-operation

The reply of a bind-operation is mapped on the A-ASSOCIATE response and A-ASSOCIATE confirm service primitives.

If the bind-operation was successfully performed, the result parameter of the service primitives is "accepted", and the result value of the bind-operation is mapped on the user information parameter of the service primitives.

If the bind-operation was not successfully performed, the result parameter value of the service primitives is "rejected (permanent)", and the error value of the bind-operation is mapped on the user information parameter of the service primitives.

11.2.1.2 Mapping of an unbind-operation

An unbind-operation is mapped on the A-RELEASE service.

11.2.1.2.1 Invocation of an unbind-operation

The invocation of an unbind-operation is mapped on the A-RELEASE request and the A-RELEASE indication service primitives.

The argument value of the unbind-operation is mapped on the user information parameter of the service primitives. The reason parameter value of the service primitives is "normal".

11.2.1.2.2 Reply of an unbind-operation

The reply of an unbind-operation is mapped on the A-RELEASE response and A-RELEASE confirm service primitives.

If the unbind-operation was successfully performed, the reason parameter value of the service primitives is "normal", the result value of the unbind-operation is mapped on the user information parameter of the service primitives, and the result parameter of the service primitives

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

is "affirmative".

If the unbind-operation was not successfully performed, the reason parameter value of the service primitives is "not finished," the error value of the unbind-operation is mapped on the user information parameter of the service primitives, and the result parameter of the service primitives is "affirmative".

11.2.2 Mapping on RTSE services

The bind-operation is mapped on the RT-OPEN service and the unbind-operation ma on the RT-CLOSE service.

11.2.2.1 Mapping of a bind-operation

A bind-operation is mapped on the RT-OPEN service.

11.2.2.1.1 Invocation of a bind-operation

The invocation of a bind-operation is mapped on the RT-OPEN request and RT-OPEN indication service primitives.

The argument value of the bind-operation is mapped on the user-data parameter of the service primitives. The dialogue-mode parameter value is "two-way-alternate".

11.2.2.1.2 Reply of a bind-operation

The reply of a bind-operation is mapped on the RT-OPEN response and RT-OPEN confirm service primitives.

If the bind-operation was successfully performed, the result parameter of the service primitives is "accepted", and the result value of the bind-operation is mapped on the user-data parameter of the service primitives.

If the bind-operation was not successfully performed, the result parameter value of the service primitives is "rejected (permanent)", and the error value of the bind-operation is mapped on the user-data parameter of the service primitives.

11.2.2.2 Mapping of an unbind-operation

An unbind-operation is mapped on the RT-CLOSE service.

11.2.2.2.1 Invocation of an unbind-operation

The invocation of an unbind-operation is mapped on the RT-CLOSE request and the RT-CLOSE indication service primitives.

The argument value of the unbind-operation is mapped on the user-data parameter of the service primitives. The reason parameter value of the service primitives is "normal".

11.2.2.2.2 Reply of an unbind-operation

The reply of an unbind-operation is mapped on the RT-CLOSE response and RT-CLOSE confirm service primitives.

If the unbind-operation was successfully performed, the reason parameter value of the service primitives is "normal", and the result value of the unbind-operation is mapped on the user-data parameter of the service primitives.

If the unbind-operation was not successfully performed, the reason parameter value of the service primitives is "not finished", and the error value of the unbind-operation is mapped on the user-data parameter of the service primitives.

11.2.3 Mapping on ROSE services

An operation is mapped on the ROSE services.

11.2.3.1 Invocation of an operation

The invocation of an operation is mapped on the RO-INVOKE service.

The value assigned to the operation is mapped on the operation-value parameter of that service. The value of the Named-Type in the ARGUMENT clause of the OPERATION macro is mapped on the argument parameter of that service.

11.2.3.2 Reply of an operation

If an operation was successfully performed, the reply is mapped on the RO-RESULT service.

The value of the Named-Type in the RESULT clause of the OPERATION macro is mapped on the result parameter of that service.

If an operation was not successfully performed, the reply is mapped on the RO-ERROR service.

In this case one of the errors in the Identifier List of Error Names in the ERROR clause of the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

OPERATION macro may be applied. The value assigned to the applied error is mapped on the error-value parameter of that service. The value of the Named-Type in the PARAMETER clause of the ERROR macro of the applied error is mapped on the error-parameter parameter of that service.

12 Sequencing information

This clause defines the interaction among the Remote Operations, and the interaction among the ACSE services and the ROSE services.

12.1 Sequencing information f(emote Operations

12.1.1 Bind-operation

12.1.1.1 Usage restrictions

A bind-operation is not used on an established application-association. A successfully performed bind-operation establishes an application-association.

12.1.1.2 Disrupted Remote Operation

The bind-operation does not disrupt any Remote Operation.

12.1.1.3 Disrupting Remote Operations

There are no disrupting Remote Operations.

12.1.1.4 Collisions

A bind-operation collision results when the user-elements in both AEs simultaneously invoke a bind-operation on each other. In this case two independent application-associations are established.

12.1.2 Unbind-operation

12.1.2.1 Usage restrictions

An unbind-operation is only used on an established application-association. It is only used by the user-element which invoked the bind-operation. It is only used when no replies from Operation Class 1 or 2 operations are outstanding.

The application-association ceases to be established no matter whether the unbind-operation is performed successfully or not.

12.1.2.2 Disrupted Remote Operations

An unbind-operation does not disrupt Remote Operations in the case of Association Class 1 and Operation Class 1 or 2 operations.

In all other cases an unbind-operation may disrupt operations. However if in a specific application-context Operation Classes 3, 4 or 5 and/or Association Class 2 or 3 are used, it is assumed that either the disruption is acceptable or the application-context provides operations to avoid the disruption.

12.1.2.3 Disrupting Remote Operations

There are no disrupting Remote Operations.

12.1.2.4 Collisions

Because only the association-initiator may release the application-association, there is no collision.

12.1.3 Operations

12.1.3.1 Usage restrictions

Operations are only used on an established application-association.

12.1.3.2 Disrupted Remote Operations

Operations do not disrupt any Remote Operations.

12.1.3.3 Disrupting Remote Operations

Operations may be disrupted by an unbind-operation (see 12.1.2.2).

12.1.3.4 Collisions

There are no collisions of operations.

12.1.4 Further sequencing information

Disrupting services are not visible at the operation-interface. However operations may be disrupted by services (see 12.2).

Bind-operations and unbind-operations are disrupted by the A-ABORT, A-P-ABORT, RT-U-ABORT and RT-P-ABORT services.

Operations are disrupted by the A-ABORT, A-P-ABORT, RT-U-ABORT, RT-P-ABORT, RO-REJECT-U and RO-REJECT-P services.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

In addition an operation may be disrupted by the A-RELEASE service. But this reflects only the disruption of an operation by an unbind-operation.The disrupted operations are not considered in the disrupted services of 12.2.

Because all Remote Operations are mapped on services, and disrupting Remote Operations (unbind-operation) are represented by services, no disrupting Remote Operations are considered in the disrupting services subclauses of 12.2.

12.2 Sequencing information for services

12.2.1 ACSE services

The sequencing information for ACSE services is described in ISO 8649. Additional information is provided in this subclause.

12.2.1.1 Disrupted ROSE services

In addition to the disrupted services defined in ISO 8649 all ROSE services except the RO-REJECT-P service are disrupted by the A-ABORT and A-P-ABORT services and may be disrupted by the A-RELEASE service (see 12.2.3.6).

12.2.1.2 Disrupting ROSE services

There are no disrupting ROSE services.

12.2.2 RTSE services

The sequencing information for RTSE services is described in ISO/IEC 9066-1. Additional information is provided in this subclause.

12.2.2.1 Disrupted ROSE services

In addition to the disrupted services defined in ISO/IEC 9066-1 all ROSE services except the RO-REJECT-P service are disrupted by the RT-U-ABORT, RT-P-ABORT and the negative RT-TRANSFER confirm services.

12.2.2.2 Disrupting ROSE services

There are no disrupting ROSE services.

12.2.3 ROSE services

This subclause describes the interaction among the ROSE services. Interactions with ACSE services are described in 12.2.1, and interactions with RTSE services are described in subclause 12.2.2.

12.2.3.1 RO-INVOKE service

12.2.3.1.1 Type of Service

The RO-INVOKE service is an non-confirmed service.

12.2.3.1.2 Usage restriction

The RO-INVOKE service is only used on an established application-association.

12.2.3.1.3 Disrupted services

The RO-INVOKE service does not disrupt any services.

12.2.3.1.4 Disrupting services

The RO-INVOKE service is disrupted by the RO-REJECT-P service.

12.2.3.1.5 Collision

There are no collisions of the RO-INVOKE service.

12.2.3.2 RO-RESULT

12.2.3.2.1 Type of service

The RO-RESULT service is an non-confirmed service.

12.2.3.2.2 Usage restriction

The RO-RESULT service is only used on an established application-association and in reply to an RO-INVOKE service.

12.2.3.2.3 Disrupted services

The RO-RESULT service does not disrupt any services.

12.2.3.2.4 Disrupting services

The RO-RESULT service is disrupted by the RO-REJECT-P service.

12.2.3.2.5 Collisions

There are no collisions of the RO-RESULT service.

12.2.3.3 RO-ERROR

12.2.3.3.1 Type of service

The RO-ERROR service is an non-confirmed service.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.2.3.3.2 Usage restriction

The RO-ERROR service is only used on an established application-association and in reply to an RO-INVOKE service.

12.2.3.3.3 Disrupted services

The RO-ERROR service does not disrupt any service.

12.2.3.3.4 Disrupting services

The RO-ERROR service is disrupted by the RO-REJECT-P service.

12.2.3.3.5 Collisions

There are no collisions of the RO-ERROR service.

12.2.3.4 RO-REJECT-U

12.2.3.4.1 Type of service

The RO-REJECT-U service is an non-confirmed service.

12.2.3.4.2 Usage restriction

The RO-REJECT-U service is only used on an established application-association and in reply to RO-INVOKE, RO-RESULT and RO-ERROR services.

12.2.3.4.3 Disrupted services

The RO-REJECT-U service does not disrupt any service.

12.2.3.4.4 Disrupting services

The RO-REJECT-U service is disrupted by the RO-REJECT-P service.

12.2.3.4.5 Collisions

There are no collisions of the RO-REJECT-U service.

12.2.3.5 RO-REJECT-P

12.2.3.5.1 Type of service

The RO-REJECT-P service is a provider initiated service.

12.2.3.5.2 Usage restriction

Not applicable.

12.2.3.5.3 Disrupted services

The RO-REJECT-P service disrupts all other ROSE services.

12.2.3.5.4 Disrupting services

The RO-REJECT-P service is not disrupted by any other service.

12.2.3.5.5 Collision

If the ACSE service or an RTSE service causes an abort or the release of an application-association, it is a local matter to inform the service-user about outstanding RO-REJECT-P services for Returned-parameters.

12.2.3.6 Additional Sequencing Information

The usage restrictions for unbind-operations (12.1.2.1) and the mapping of unbind-operation on to the A-RELEASE service prevent the disruption of ROSE services, if only Association Class 1 and Operation Classes 1 and 2 are used.

If Association Classes 2 or 3, or Operation Class 3, 4 or 5 are used, the A-RELEASE service may disrupt ROSE services. In this case it is the responsibility of the application-context designer, either to accept this disruption or to provide means (e.g. operations) to prepare for the release of an application-association.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A

(normative)

Notation supporting the specification of application-service-elements and application-contexts

This annex provides a notation supporting the specification of application-service-elements and application contexts which are specified by means of the RO-notation. The RO-notation may be used to specify the bind-operation and the un - operation of an application context. Adc ıally the RO-notation may be used to specify operation types and error types of several application-service-elements (ROSE-user ASEs). If the RO-notation is combined with other notations, other specification tools defined elsewhere might be used.

This annex defines two macros supporting the specification of application-service-elements and application contexts. The formal definition of these macros is shown in figure A.1.

A.1 Application-service-elements

The notation supports the unique identification of an ASE.

If an ASE is an ROSE-user, the notation additionally supports the specification of the characteristics of the ASE. The operation-interface and the protocol of an ROSE-user ASE are specified by a set of operation types and a set of error types.

The protocol specified for a specific ASE may be inherently:

a) symmetric, or

b) asymmetric.

In the symmetric case both ASE-users may invoke the same set of operation types.

In the asymmetric case one ASE-user (called supplier in the context of this annex) provides some information-processing functionality which is used by the peer ASE-user (called consumer in the context of this annex). In this case a specific operation type may be

a) invoked by the consumer, and/or

b) invoked by the supplier

of the information-processing functionality.

NOTE A particular allocation of the terms "supplier" and "consumer" is often intuitive. One might naturally consider a file system, e.g, to be called a supplier and its user to be called a consumer. Strictly speaking, however, the assignment of the two terms is arbitrary.

If an ASE uses the concept of linked-operations, a specific operation type may be invoked as a child-operation. The invoker of the child-operation is the performer of the linked parent-operation. Operation types solely invoked as child-operations shall not be included in the ASE specification, they are listed in the type specification of their parent-operations.

The error types the operations may report are not included in the ASE specification, they are listed in the type specification of the operations.

The set of the remaining operations (operations not solely invoked as child-operations), and who is allowed to invoke this operations may be reflected by a formal notation specifying the ROSE-user ASE.

A.1.1 Specification of an application-service-element

An ASE may be specified by a formal notation supported by the **APPLICATION-SERVICE-ELEMENT** macro (see figure A.1).

The type notation of the **APPLICATION-SERVICE-ELEMENT** macro enables the specification of an ASE. The value notation of the **APPLICATION-SERVICE-ELEMENT** macro enables the specification of an unique identifier for the ASE.

The notation for an ASE type is the keyword **APPLICATION-SERVICE-ELEMENT** optionally followed by the specification of operations.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
Remote-Operations-Notation-extension { joint-iso-ccitt remote-operations(4) notation-extension (2) }
DEFINITIONS ::=
BEGIN

EXPORTS    APPLICATION-SERVICE-ELEMENT, APPLICATION-CONTEXT, aCSE;

IMPORTS    OPERATION, BIND, UNBIND FROM    Remote-Operation-Notation
                                           { joint-iso-ccitt remote-operations(4) notation(0)};

-- macro definition for ASEs

APPLICATION-SERVICE-ELEMENT MACRO ::=
BEGIN

TYPE NOTATION          ::= Symmetr      ConsumerInvokes SupplierInvokes | empty
VALUE NOTATION         ::= value (VALUE OBJECT IDENTIFIER)

SymmetricAse           ::=  "OPERATIONS" "{" OperationList "}"

ConsumerInvokes        ::= "CONSUMER INVOKES" "{" OperationList "}" | empty

SupplierInvokes        ::= "SUPPLIER INVOKES"   "{" OperationList "}" | empty

OperationList          ::= Operation | OperationList "," Operation

Operation              ::=  value (OPERATION)

END

aCSE APPLICATION-SERVICE-ELEMENT ::= { joint-iso-ccitt remote-operations(4) aseID-ACSE (4)}

-- Remote Operations Notation extension continued
```

Figure A.1 (Part 1 of 2) - Formal definition of ASE and application-context data types

If the protocol specified for the ASE is symmetric, the keyword **OPERATIONS** and the reference names of the operations are specified.

If the protocol specified for the ASE is asymmetric, the keywords **CONSUMER INVOKES** and the reference names of the operations the consumer may invoke, and/or the keywords **SUPPLIER INVOKES** and the reference names of the operations the supplier may invoke, are specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2 Application contexts

In the context of this annex an application context explicitly identifies:

a) a bind-operation,

b) an unbind-operation, and

c) a set of application-service-elements

and requires one or more identified abstract-syntaxes.

The set of application-service-elements contains

a) ASEs not using the RO-notation, i.e the ACSE, optionally the RTSE, and optionally others; and

b) optionally the ROSE and ROSE-user ASEs.

If the application context contains ROSE-user ASEs with an asymmetrical protocol, the notation supports the specification whether the association-initiator or the association-responder is the consumer of such an ASE.

The identification of the bind-operation, the unbind-operation, the application-service-elements, and the abstract-syntaxes may be reflected by a formal notation specifying the application context.

A.2.1 Specification of an application context

An application context may be specified by a formal notation supported by the **APPLICATION-CONTEXT** macro (see figure A.1).

The type notation of the **APPLICATION-CONTEXT** macro enables the specification of the application context. The value notation of the **APPLICATION-CONTEXT** macro enables the specification of an unique identifier for the application context.

The notation for an application context type is the keyword **APPLICATION-CONTEXT**, followed by the keywords **APPLICATION SERVICE ELEMENTS** and the reference names of ASEs not using the RO-notation, followed by the keyword **BIND** and the reference name of the bind-operation type, followed by the keyword **UNBIND** and the reference name of the unbind-operation type, optionally followed by the specification of ASEs using operations, followed by the keywords **ABSTRACT SYNTAXES** and the reference names of the abstract syntaxes.

If the application context contains ROSE-user ASEs, the keywords **REMOTE OPERATIONS** and the reference name of the ROSE, followed by the specification of ASEs with a symmetrical protocol, and/or the specification of ASEs with an asymmetrical protocol are used. The specification of ASEs with a symmetrical protocol is the keywords **OPERATIONS OF** and the reference names of that ASEs. The specification of ASEs with an asymmetrical protocol is the keywords **INITIATOR CONSUMER OF** and the reference names of ASEs the consumer of which is the association-initiator, and/or the keywords **RESPONDER CONSUMER OF** and the reference names of ASEs the consumer of which is the association-responder.

A.2.2 Mapping of notation on to service

The application context identifier and the list of abstract syntax names specified by means of the **APPLICATION-CONTEXT** macro are mapped either on the RT-OPEN services of RTSE, if RTSE is included in the application context; or else on the A-ASSOCIATE services of ACSE.

The application context value is mapped on to the Application Context Name parameter of the RT-OPEN or A-ASSOCIATE services.

The abstract syntax names are mapped on to the Presentation Context Definition List parameter and the Presentation Context Definition Result List of the RT-OPEN or A-ASSOCIATE services.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
-- Remote Operations Notation extension continued

-- macro definition for application contexts

APPLICATION-CONTEXT MACRO ::=
BEGIN

TYPE NOTATION       ::= NonROelements Binding ROelements AbstractSyntaxes
VALUE NOTATION      ::= value (VALUE OBJECT IDENTIFIER)

NonROelements       ::= "APPLICATION SERVICE ELEMENTS" "{" AseList "}"

Binding             ::= "BIND"      ype    -- shall reference a bind-operation type
                        "UNBIND"    ype    -- shall reference an unbind-operation type

ROelements          ::= "REMOTE OPERATIONS" "{" AseID "}" -- identifying ROSE
                        SymmetricAses AsymmetricAses | empty

SymmetricAses       ::= "OPERATIONS OF" "{" AseList "}" | empty

AsymmetricAses      ::= InitiatorConsumerOf ResponderConsumerOf

InitiatorConsumerOf ::= "INITIATOR CONSUMER OF" "{" AseList "}" | empty

ResponderConsumerOf ::= "RESPONDER CONSUMER OF" "{" AseList "}" | empty

AbstractSyntaxes    ::= "ABSTRACT SYNTAXES" "{" AbstractSyntaxList "}"

AseList             ::= AseID | AseList "," AseID

AseID               ::= value ( APPLICATION-SERVICE-ELEMENT)

AbstractSyntaxList  ::= AbstractSyntax | AbstractSyntaxList "," AbstractSyntax

AbstractSyntax      ::= value (OBJECT IDENTIFIER)-- identifying abstract syntax

END

END                     -- end of Remote Operations Notation extension
```

Figure A.1 (Part 2 of 2) - Formal definition of ASE and application-context data types

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
(informative)
Guidelines for application protocol designers on the use of ROSE

This annex provides examples and guidelines for application protocol designers on the use of ROSE.

B.1 Examples for operations and errors

This subclause provides examples for the definition of operations and errors.

B.1.1 Operation classes

This subclause provides examples for the definition of operations of Operation classes 1 to 5.

The operation operationExample12 of Operation classes 1 or 2 (see example below) reports success (result of type ArgumentType12) or failure (errors errorExample1 or errorExample2). The argument of the operation operationExample12 is of type ArgumentType12. The value of the operation operationExample12 is 1.

```
operationExample12   OPERATION
        ARGUMENT     ArgumentType12
        RESULT       ResultType12
        ERROR        {errorExample1, errorExample2}
        ::= 1
```

The operation operationExample3 of Operation Class 3 (see example below) reports failure (error errorExample1) only, if any. The argument of the operation operationExample3 is of type ArgumentType3. The value of the operation operationExample3 is 2.

```
operationExample3    OPERATION
        ARGUMENT     ArgumentType3
        ERROR        {errorExample1}
        ::= 2
```

The operationExample4 operation of Operation Class 4 (see example below) reports success (result of type ResultType4) only. The argument of the operation operationExample4 is of type ArgumentType4. The value of the operation operationExample4 is 3.

```
operationExample4    OPERATION
        ARGUMENT     ArgumentType4
        RESULT       ResultType4
        ::= 3
```

The operations operationExample51 and operationExample52 of Operation Class 5 (see example below) reports no outcome. The argument of the operation operationExample51 is of type ArgumentType4, the operation operationExample52 has no argument. The value of the operation operationExample51 is 4, the value of the operation operationExample52 is 5.

```
operationExample51   OPERATION
    ARGUMENT         ArgumentType4
    ::= 4

operationExample52   OPERATION
    ::= 5
```

B.1.2 Linked-operations

The example below shows the definition of a set of linked-operations consisting of the parent-operation parent-op12 and the child-operations operationExample51 and operationExample52.

```
parent-op12  OPERATION
    ARGUMENT     ArgumentType12
    RESULT       ResultType12
    ERROR        {errorExample1, errorExample2}
    LINKED       { operationExample51,
                   operationExample52 }
    ::= 6
```

B.1.3 Errors

Errors (see errorExample1 and errorExample2 below) reports failure. The parameter of the error errorExample1 is of type ParameterType1, the error errorExample2 has no parameter. The value of the error errorExample1 is 1, the value of the error errorExample1 is 2.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
errorExample1    ERROR
        PARAMETER  ParameterType1
        ::= 1

errorExample1    ERROR
        ::= 2
```

B.2 Examples for bind-operations and unbind-operations

This subclause provides examples for the definition of bind-operations an ıbind-operations.

B.2.1 Bind-operations

Bind-operations are used to establish an application-association.

The request of the bind-operation **BindExample1** to establish an application-association is accompanied by the argument of type **BindArgumentType1**. The positive response to the application-association establishment is accompanied by the result of type **BindResultType1**. The negative response to the application-association establishment is accompanied by the bind-error of type **BindErrorType1**.

```
BindExample1 ::= BIND
        ARGUMENT    BindArgumentType1
        RESULT      BindResultType1
        BIND-ERROR  BindErrorType1
```

The request of the bind-operation **BindExample2** to establish an application-association is accompanied by the argument of type **BindArgumentType1**. The positive response to the application-association establishment is not accompanied by any user data. The negative response to the application-association establishment is accompanied by the bind-error of type **BindErrorType1**.

```
BindExample2 ::= BIND
        ARGUMENT    BindArgumentType1
        BIND-ERROR  BindErrorType1
```

Note that argument, result and bind-error of a bind-operation are optional. Neither the request of the bind-operation **BindExample3** to establish an application-association nor the response to the application-association establishment are accompanied by any user data.

```
BindExample3 ::= BIND
```

B.2.2 Unbind-operations

Unbind-operations are used to release an application-association.

The request of the unbind-operation **UnbindExample1** to release an application-association is accompanied by the argument of type **UnbindArgumentType1**. The response to the application-association release is accompanied either by the result of type **UnbindResultType1** or by the unbind-error of type **UnbindErrorType1**.

```
UnbindExample1 ::=  UNBIND
        ARGUMENT      UnbindArgumentType1
        RESULT        UnbindResultType1
        UNBIND-ERROR  UnbindErrorType1
```

The request of the unbind-operation **UnbindExample2** to release an application-association is accompanied by the argument of type **UnbindArgumentType1**. The response to the application-association release is optionally accompanied by the unbind-error of type **UnbindErrorType1**.

```
UnbindExample2 ::=  UNBIND
        ARGUMENT      UnbindArgumentType1
        UNBIND-ERROR  UnbindErrorType1
```

Note that argument, result and unbind-error of an unbind-operation are optional. Neither the request of the bind-operation **UnbindExample3** to release an application-association nor the response to the application-association release are accompanied by any user data.

```
UnbindExample3 ::=  UNBIND
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.3 Export and import of operations and errors

This clause gives examples of how to export and import operations and errors.

The example below shows how to export operations and errors. The **operation10** and **error10** have local values. **OperationTypeA** and **ErrorTypeA** are types, and specific values have to be assigned in the importing modules. The **operation11** and **error11** have globally unique values.

```
ExportingModule { objectidentifier1 } DEFINITI        =
BEGIN
    EXPORTS operation10, OperationTypeA, operation11,
            error10, ErrorTypeA, error11;
    IMPORTS OPERATION, ERROR,BIND,UNBIND
            FROM    Remote-Operation-Notation
                    { joint-iso-ccitt remote-
                    operations(4) notation(0)};

    operation10  OPERATION
            ARGUMENT   ArgumentType10
            RESULT     ResultType10
            ERROR      {error10}
            ::= 10

    OperationTypeA  ::=
            OPERATION
            ARGUMENT   ArgumentTypeA
            RESULT     ResultTypeA

    operation11      OPERATION
            ARGUMENT   ArgumentType11
            RESULT     ResultType11
            ERROR      {error11}
            ::= { objectidentifier2component 11}

    error10      ERROR
            PARAMETER  ParameterType10
            ::= 10

    ErrorTypeA       ::=
            ERROR
            PARAMETER  ParameterTypeA

    error11      ERROR
            PARAMETER  ParameterType11
            ::= { objectidentifier2component 12}
END
```

The example below shows how to import operations and errors. The **operation10** and **error10** have local values defined in the exporting module. The designer of the importing module is responsible for the uniqueness of these values within the abstract syntax. The operation **operation13** is of type **OperationTypeA** and has the value **13** assigned. The error **error13** is of type **ErrorTypeA** and has the value **13** assigned. The **operation11** and **error11** have globally unique values defined in the exporting module.

```
ImportingModule { objectidentifier3 } DEFINITIONS ::=
BEGIN
    IMPORTS operation10, OperationTypeA, operation11,
            error10, ErrorTypeA, error11
            FROM    ExportingModule {
                    objectidentifier1 };

    operation13  OperationTypeA  ::= 13
    error13  ErrorTypeA          ::= 13

END
```

B.4 Definition of Application-service-elements

This clause gives examples on how to define application-service-elements. The examples refer to the operations and errors defined in the examples of clause B.1.

The application-service-element **element1** includes the operations **operationExample12** and **operationExample3** and the errors **errorExample1** and **errorExample2**. Note, the errors are included indirectly by the definition of the operations. The application-service-element **element1** is symmetric, i.e. both user-elements may invoke the operations **operationExample12** and **operationExample3**.

```
element1 APPLICATION-SERVICE-ELEMENT
        OPERATIONS {operationExample12,
        operationExample3 }
        ::= { objectidentifierOfElement1 }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The application-service-element **element2** includes the operations **operationExample3** and **operationExample4** and the error **errorExample1**. The application-service-element **element2** is asymmetric, i.e. only one user-element (that in the consumer role) may invoke the operations **operationExample3** and **operationExample4**.

```
element2 APPLICATION-SERVICE-ELEMENT
        CONSUMER INVOKES {operationExample3,
        operationExample4 }
        ::= { objectidentifierOfElement2 }
```

The application-service-element **element3** includes the operations **operationExample12** and **operationExample51** and the errors **errorExample1** and **errorExample2**. The application-service-element **element3** is asymmetric, i.e. only one user-element (that in the supplier role) may invoke the operations **operationExample12** and **operationExample51**.

```
element3 APPLICATION-SERVICE-ELEMENT
        SUPPLIER INVOKES {operationExample12,
        operationExample51 }
        ::= { objectidentifierOfElement3 }
```

The application-service-element **element4** includes the operations **parent-op12**, **operationExample51** and **operationExample52** and the errors **errorExample1** and **errorExample2**. Note, the child-operations **operationExample51** and **operationExample52** are included indirectly by the definition of the parent-operation **parent-op12** . The application-service-element **element4** is asymmetric, i.e. only one user-element (that in the consumer role) may invoke the operation **parent-op12** and only the other user-element (that in the supplier role) may invoke the child-operations **operationExample51** and **operationExample52** during the execution of the parent-operation **parent-op12**.

```
element4 APPLICATION-SERVICE-ELEMENT
        CONSUMER INVOKES {parent-op12 }
        ::= { objectidentifierOfElement4 }
```

The application-service-element **element5** includes the same operations and errors as the application-service-element **element4**. The only difference being, that the user-element in the supplier role may invoke the operation **operationExample52** either as a child-operation, or outside a set of linked-operations (as a non-child-operation).

```
element5 APPLICATION-SERVICE-ELEMENT
        CONSUMER INVOKES {parent-op12 }
        SUPPLIER INVOKES {operationExample52}
        ::= { objectidentifierOfElement5 }
```

B.5 Definition of application contexts

This clause gives examples of how to define application contexts for several Association Classes. The examples refer to the application-service-elements defined in the examples of clause B.4, and the bind-operations and unbind-operations defined in the examples of clause B.2.

The application context **context1** includes the application-service-elements ACSE, RTSE, ROSE, and **element2**; the bind-operation **BindExample1** and the unbind-operation **UnbindExample3**. The association-initiator may invoke the operations **operationExample3** and **operationExample4**. The association-responder is not allowed to invoke any operation (Association Class 1).

```
context1 APPLICATION-CONTEXT
    APPLICATION SERVICE ELEMENTS {aCSE, rTSE}
    BIND                          BindExample1
    UNBIND                        UnbindExample3
    REMOTE OPERATIONS             {rOSE}
    INITIATOR CONSUMER OF         {element2}
    ABSTRACT SYNTAXES
        {{joint-iso-ccitt association-control(2)
        abstractSyntax(1) apdus(0) version1(1)},
        objectidentifierOfAbstractSyntax1}
    ::= { objectidentifierOfContext1 }
```

The application context **context2** includes the application-service-elements ACSE, ROSE, **element2**, and **element3**; the bind-operation **BindExample1**; and the unbind-operation **UnbindExample3**. The association-responder may invoke the operations **operationExample3**, **operationExample4**, **operationExample12** and **operationExample51**. The association-initiator is not allowed to invoke any operation (Association Class 2).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
context2 APPLICATION-CONTEXT
    APPLICATION SERVICE ELEMENTS {aCSE}
    BIND                          BindExample1
    UNBIND                        UnbindExample3
    REMOTE OPERATIONS             {rOSE}
    INITIATOR CONSUMER OF         {element3}
    RESPONDER CONSUMER OF         {element2}
    ABSTRACT SYNTAXES
        {{joint-iso-ccitt association-control(2)
        abstractSyntax(1) apdus(0) version1(1)},
        objectidentifierOfAbstractSyntax2}
    ::= { objectidentifierOfContext2 }
```

The application context **context3** includes the application-service-elements ACSE, RTSE, ROSE, and **element2**; the bind-operation **BindExample1**; and the unbind-operation **UnbindExample3**. The association-responder may invoke the operations **operationExample3** and **operationExample4**. The association-initiator is not allowed to invoke any operation (Association Class 2).

```
context3 APPLICATION-CONTEXT
    APPLICATION SERVICE ELEMENTS {aCSE, rTSE}
    BIND                          BindExample1
    UNBIND                        UnbindExample3
    REMOTE OPERATIONS             {rOSE}
    RESPONDER CONSUMER OF         {element2}
    ABSTRACT SYNTAXES
        {objectidentifierOfAbstractSyntax3}
    ::= { objectidentifierOfContext3 }
```

The application context **context4** includes the application-service-elements ACSE, ROSE, and **element2**; the bind-operation **BindExample3**; and the unbind-operation **UnbindExample3**. The application context **context3** is symmetric, i.e. both the association-initiator and the association-responder may invoke the operations **operationExample12** and **operationExample3** (Association Class 3).

```
context4 APPLICATION-CONTEXT
    APPLICATION SERVICE ELEMENTS {aCSE}
    BIND                          BindExample3
    UNBIND                        UnbindExample3
    REMOTE OPERATIONS             {rOSE}
    OPERATIONS OF                 {element1}
    ABSTRACT SYNTAXES
        {objectidentifierOfAbstractSyntax4}
    ::= { objectidentifierOfContext4 }
```

The last two examples assume a single named abstract syntax.

B.6 Releasing application-associations in an orderly way

B.6.1 Introduction

This part of ISO/IEC 9072 defines five Operation Classes based upon the outcome they report, and three Association Classes based upon whether the association-initiator, the association-responder, or both may invoke operations. This subclause defines the rules that ensure orderly release of application-associations and the Operation Classes invoked over it.

B.6.2 Objectives

The rules of this subclause are intended to achieve one of the following two objectives, depending upon the situation:

a) *The Exactly-Once Objective*: Ideally an application entity should be able to count on the invocation of an operation causing that operation to be performed only once.

b) *The At-Most-Once Objective*: In some circumstances the Exactly-Once Objective cannot be achieved. A lesser but still useful objective is that invoking an operation which will cause that operation to be performed once or not at all.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.6.3 Definition of rules

The following **general rules** apply in all circumstances:

G1. The performer shall report the result or error of each confirmed operation over the same application-association by means of which the operation was invoked.

G2. The initiator shall not release the application-association until all operations it invoked have been confirmed.

The following **specific rules** apply in certain circumstances:

S1. Each time it exercises the RO-INVOKE service, the invoker shall supply a different Invoke-ID (unless reattempting an invocation) even across a succession of application-associations. This enables the performer to achieve the At-Most-Once Objective by suppressing duplicates.

S2. If the performer encounters a duplicate Invoke-ID in the RO-INVOKE service, it shall exercise the RO-REJECT-U service with duplicate-invocation as the Reject-reason. This helps to achieve the Exactly-Once Objective.

S3. The association-initiator shall reject any invocations it has not performed before releasing the application-association.

S4. The association-initiator shall respond to any invocations it has performed before releasing the application-association.

B.6.4 Application of rules

The general rules always apply. The specific rules govern application-associations of particular Association Classes, and operations (invoked by means of such associations) of particular Operation Classes, as follows:

a) *Application-associations of Association Class 1*: For operations of Operation Classes 1 and 2, no specific rules apply. For operations of Operation classes 3, 4, and 5, specific rules S1 and S2 apply.

b) *Application-associations of Association classes 2 and 3*: For operations of Operation Classes 1 and 2, specific rules S3 and S4 apply. (Any invocation issued by the association-responder after the association-initiator has issued a release are lost. Response-rejects may be lost as well). For operations of Operation classes 3, 4, and 5, specific rules S1, S2, S3 and S4 apply.

For protocols containing only operations of Operation Classes 1 or 2, the only constraint upon the values of the Invoke-IDs the invoker supplies to the RO-INVOKE service is that they be distinct over the life of the application-association.

Application-entities make Invoke-IDs unique to an invoker and across consecutive application-associations by exchanging presentation-addresses at application-association establishment time and, for each presentation-address, making Invoke-IDs integers that increase monotonically over some reasonable long period of time.

To ensure that an operation of Operation Classes 3, 4, or 5 has been performed exactly once, an application-entity shall elicit the duplicate-invocation Reject-reason by invoking the operation twice (or more) with the same Invoke-ID. Otherwise, the At-Most-Once Objective is all that is assured, suggesting that, on average, the invoker does not care whether an non-confirmed operation is performed.

B.6.5 Observations

Every operation of Operation Classes 1 or 2 is performed exactly once.

The usefulness of non-confirmed operations or conditionally confirmed operations may depend on the specific application. Protocol designers are recommendet to define only operations of Operation Classes 1 or 2 unless some very specialized requirements exist.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

UDC 681.3 : 621.39

Descriptors : data processing, information interchange, network interconnection, text communication, data transmission, data transfer, teleprocessing, models, definitions.

Price based on 35 pages

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale