INTERNATIONAL STANDARD

ISO/IEC 13712-1

First edition
1995-09-15

Information technology — Remote Operations: Concepts, model and notation

Technologies de l'information — Opérations à distance: Concepts, modèle et notation

Reference number
ISO/IEC 13712-1:1995(E)

CONTENTS

Foreword

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

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

International Standard ISO/IEC 13712-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 21, *Open systems interconnection, data management and open distributed processing*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.880.

This part of ISO/IEC 13712 is a partial revision of ISO/IEC 9072-1:1989 and ISO/IEC 9072-2:1989.

ISO/IEC 13712 consists of the following parts, under the general title *Information technology — Remote Operations*:

— *Part 1: Concepts, model and notation*

— *Part 2: OSI realizations — Remote Operations Service Element (ROSE) service definition*

— *Part 3: OSI realizations — Remote Operations Service Element (ROSE) protocol specification*

Annex A forms an integral part of this part of ISO/IEC 13712. Annexes B to D are for information only.

Introduction

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

The concepts of ROS are abstract, and may be realized in many ways. For example, objects whose interactions employ ROS concepts may be separated by a software interface or by an OSI network.

This Recommendation | International Standard describes the concepts and model of ROS. It uses ASN.1 to specify information object classes corresponding to the fundamental concepts of ROS, such as operation and error. This in turn provides a notation so that designers can specify particular instances of those classes, e.g. particular operations and errors.

This Recommendation | International Standard provides a generic set of PDUs which can be used in realizing the ROS concepts between objects remote from one another. These PDUs are used in the OSI realization of ROS, which are specified in the companion Recommendations | International Standards to this one.

This Recommendation | International Standard also provides a number of definitions of general utility to designers of ROS-based applications.

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

Annexes B, C and D do not form an integral part of this Recommendation | International Standard.

This page intentionally left blank

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – REMOTE OPERATIONS: CONCEPTS, MODEL AND NOTATION

1 Scope

This Recommendation | International Standard specifies the Remote Operations Service (ROS) using the Abstract Syntax Notation (ASN.1) to define information object classes corresponding to the fundamental concepts of ROS. This, in turn, provides the notation that will allow application designers to specify particular instances of these classes.

This Recommendation | International Standard also provides a collection of definitions for specifying the generic protocol between objects that communicate using ROS concepts. These definitions are used in the companion Recommendations | International Standards to this one to provide the protocol data units, the service primitives and the application context definitions used in the OSI realization of ROS.

A number of definitions of general utility to designers of ROS-based applications is also provided.

No requirement is made for conformance to this Recommendation | International Standard.

2 Normative references

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

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (1994) | ISO/IEC 8824-4:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*
- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The basic model.*
- ITU-T Recommendation X.881 (1994) | ISO/IEC 13712-2:1995, *Information technology – Remote Operations: OSI realizations – Remote Operations Service Element (ROSE) service definition.*
- ITU-T Recommendation X.882 (1994) | ISO/IEC 13712-3:1995, *Information technology – Remote Operations: OSI realizations – Remote Operations Service Element (ROSE) protocol specification.*

2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.219 (1988), *Remote Operations: Model, notation and service definition.*

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

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

2.3 Additional references

– CCITT Recommendation X.407 (1988), *Message handling systems: Abstract service definition conventions.*

3 Definitions

3.1 OSI reference model definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.200 | ISO/IEC 7498-1:

a) abstract syntax;

b) protocol data unit;

c) quality of service.

3.2 ASN.1 definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.680 | ISO/IEC 8824-1:

a) (data) type;

b) (data) value.

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.681 | ISO/IEC 8824-2:

a) field;

b) (information) object;

c) (information) object class;

d) (information) object set.

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.682 | ISO/IEC 8824-3:

a) constraint;

b) exception value.

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.683 | ISO/IEC 8824-4:

– parameterized.

3.3 ROS definitions

This Recommendation | International Standard defines the following terms:

3.3.1 argument: A data value accompanying the invocation of an operation.

3.3.2 association: A relationship between a pair of objects, serving as the context for the invocation and perfomance of operations.

3.3.3 association contract: A specification of the roles of a pair of communicating objects who may have an association with each other.

3.3.4 asymmetrical: Describing an operation package (or association contract), where the sets of operations which the two parties are capable of performing differ.

3.3.5 connection package: A specification of the roles of a pair of communicating objects in the dynamic establishment and release of associations between them.

3.3.6 contract: A set of requirements on one or more objects prescribing a collective behaviour.

3.3.7 error: A report of the unsuccessful performance of an operation.

3.3.8 linked operation: An operation invoked during the performance of another operation by the (latter's) performer and intended to be performed by the (latter's) invoker.

3.3.9 object: A model of (possibly a self-contained part of) a system, characterized by its initial state and its behaviour arising from external interactions over well-defined interfaces.

3.3.10 operation: A function that one object (the invoker) can request of another (the performer).

3.3.11 operation package: A collection of related operations used to specify the roles of a pair of communicating objects, each operation being invokable by one or both objects of the pair and performable by the partner.

3.3.12 parameter (of an error): A data value which may accompany the report of an error.

3.3.13 result: A data value which may accompany the report of the successful performance of an operation.

3.3.14 ROS-object: An object whose interactions with other objects are described using ROS concepts.

3.3.15 symmetrical: Describing an operation package (or assocation contract) in which both parties are capable of performing the same set of operations.

3.3.16 synchronous: A characteristic of an operation that, once invoked, its invoker cannot invoke another synchronous operation (with the same intended performer) until the outcome has been reported.

4 Abbreviations

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

ASN.1	Abstract Syntax Notation One
PDU	Protocol Data Unit
QOS	Quality of Service
RO (or ROS)	Remote Operations

5 Conventions

This Recommendation | International Standard employs ASN.1 to define:

a) Information object classes corresponding to the ROS concepts – These also provide notation with which designers of ROS applications can specify particular instances of those classes.

b) Particular information objects of those classes.

c) The PDUs of the generic ROS protocol.

d) Data types needed in these definitions.

Many of these definitions are parameterized, so that their users must supply actual parameters in order to complete them.

6 ROS model

Remote operations (ROS) is a paradigm for interactive communication between objects. Objects whose interactions are described and specified using ROS are **ROS-objects**. The basic interaction involved is the invocation of an operation by one ROS-object (the invoker) and its performance by another (the performer).

Completion of the performance of the operation (whether successfully or unsuccessfully) may lead to the return, by the performer to the invoker, of a report of its outcome. This is illustrated in Figure 1.

A report of the successful completion of an operation is a **result**; a report of unsuccessful completion an **error**.

During the performance of an operation, the performer can invoke **linked operations**, intended to be performed by the invoker of the original operation.

For correct interworking, certain properties of the operation must be known by both invoker and performer. The properties include:

whether reports are to be returned, and if so, which ones;

the types of the values, if any, to be conveyed with invocations of the operation or returns from it;

which operations, if any, can be linked to it;

the code value to be used to distinguish this operation from the others that might be invoked.

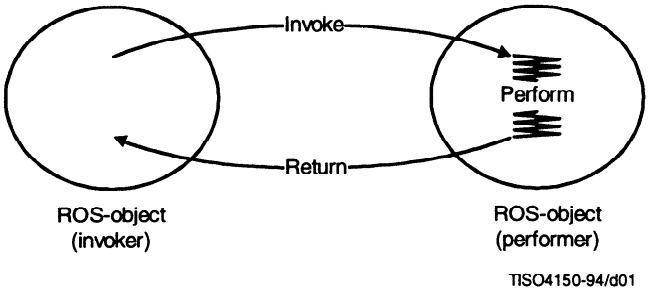

Figure 1 – Invocation, performance, and return of an operation

The interworking capabilities of (pairs of) ROS-objects of some **ROS-object class** are defined in terms of sets of related operations called **operation packages**. A package may be **symmetrical**, in which case it is defined by a single set of operations which each ROS-object in the pair can invoke (of the other). Alternatively, a package may be **asymmetrical**, in which case it is defined by two sets of operations, those which one ROS-object of the pair can invoke and those which the other can invoke. For the purpose of defining an asymmetrical package, the two ROS-objects are (arbitrarily) labelled the **consumer** and **supplier** respectively.

NOTE 1 – While these labels are, in general, arbitrary, it will often be the case that an intuitive assignment can be made, given that frequently in such a pair of objects one will be supplying a service which the other consumes.

A pair of ROS-objects must have an association between them to serve as a context for the invocation and performance of operations. Each such association is governed by an **association contract**. A contract is specified in terms of the set of packages which (collectively) determine the operations which can be invoked during the association. If the contract specification includes one or more asymmetrical packages then the contract is itself asymmetrical. For the purpose of specifying an asymmetrical association contract, the two ROS-objects which establish an association with each other are labelled the **initiator** and **responder**.

An association may be brought into and out of existence by "off-line" means. Alternatively, an association may be established and released dynamically. One option, described in this Recommendation | International Standard, by which an association may be established and released dynamically is, respectively, through the invocation and performance of special **bind** and **unbind** operations. The contract for associations of the latter variety includes a **connection package**, which includes the particular bind and unbind operations to be used.

NOTE 2 – The mechanism for the establishment and release of associations may also be done by other means described in other Recommendations | International Standards.

An association requires a relationship between the objects, the existence of which corresponds to the mutual agreement of the objects to the terms of some association contract.

NOTE 3 – This specification does not address the means by which these relationships are established or terminated.

In the foregoing, the only objects seen to be involved in an operation have been the invoker and performer. However, in general, the invoker and performer of an operation are not directly attached to one another, but are connected by some medium through which invocations and returns can be conveyed. This expanded view is illustrated in Figure 2.

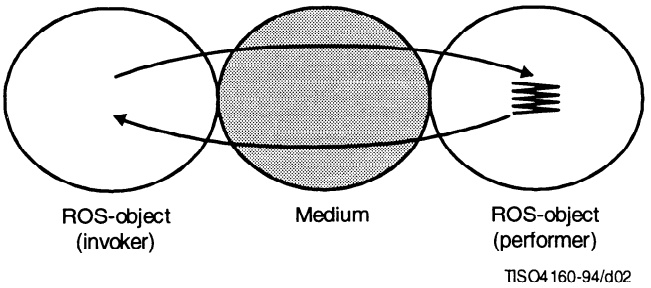

Figure 2 – Expanded view

The medium may introduce delay and the possibility of failure or inaccuracy both in the conveyance of invocations and returns, and in the establishment, release, and maintenance of associations. It may also introduce the possibility of the security of the association and its operations being threatened. The extent of these (together with other factors) are described as the quality-of-service (QOS).

Association contracts can now be seen as having three parties, the third party being the medium. The medium's obligation under the contract is to satisfy the QOS requirements.

NOTE 4 – In the future, target and minimum acceptable QOS requirements may form part of the specification of operations, operation packages and of the association contract itself directly. Different aspects of QOS are applicable to these different levels of specification.

7 Realization of ROS

A **realization** of ROS involves the definition of a suitable medium to convey invocations and returns between ROS-objects. Such a medium may, for example, comprise:

a) a message-passing or procedure calling capability allowing the invoker and performer of an operation to be implemented in separately developed software modules within a single computer system;

b) a communications capability, allowing the invoker and performer of an operation to be implemented in separate computer systems.

A realization may be general-purpose, in which case it can be employed to support any association contract. Others are special-purpose, accommodating only some particular contract(s).

Figure 3 depicts an approach to realizing ROS by communications means which is likely to be widely used.

Figure 3 – Approach to communications realization of ROS

In this approach, the medium is composed of **stub** objects, one for each ROS-object, and an information transfer object. The stub object associated with each ROS-object appears to play the role of the partner ROS-object. However, it does not actually invoke or perform any operations, merely transforming invocations and returns into PDUs and vice versa, as appropriate. These PDUs are exchanged between the stubs by means of the information transfer object.

Thus, to invoke an operation, the invoker invokes the operation of the associated stub, which forms a PDU describing the invocation. The stub uses the information transfer capability to transfer the PDU to the other stub. The latter stub interprets the PDU and then invokes the appropriate operation of the ROS-object with which it is associated, the performer. When the operation has been performed, the performer conveys any return to its associated stub, which forms a PDU describing it. The stub then, uses the information transfer capability to transfer the PDU to the other stub. The latter stub interprets the PDU and then reports the return to the invoker.

Clause 9 defines a collection of suitable PDUs.

Various information transfer capabilities can be used in a ROS realization of this kind. Of particular importance are the information transfer capabilities of OSI. The pair of companion Recommendations | International Standards, ITU-T Rec. X.881 | ISO/IEC 13712-2 and ITU-T Rec. X.882 | ISO/IEC 13712 -3, describe a number of such realizations.

8 ROS concepts

8.1 Introduction

8.1.1 This clause defines the information object classes corresponding to the basic concepts of ROS, specifying the characteristics that objects of such classes have. The following information object classes are defined:

- `OPERATION` (describing operations);
- `ERROR` (describing errors);
- `OPERATION-PACKAGE` (describing operation packages);
- `CONNECTION-PACKAGE` (describing connection packages);
- `CONTRACT` (describing association contracts);
- `ROS-OBJECT-CLASS` (describing ROS-object classes).

8.1.2 The information object classes are defined using ASN.1. This provides notation which is available to designers of ROS applications for specifying particular instances of those classes. Designers are encouraged, but not obliged, to use this specification approach. If some other approach is employed, the resulting specification shall include or reference a description of how a valid use of the notation provided could be derived.

NOTE – A number of existing specifications employ the ASN.1 MACRO notation (which were defined in previous versions of this Recommendation | International Standard: see CCITT Rec. X.219 | ISO/IEC 9072-1) to specify operations, errors, and other classes of information object relevant to ROS. Annex C describes how the use of these macros can be transformed onto the notation provided. These macros should not be used for new applications.

8.2 Operation

8.2.1 An operation is a function that one object (the invoker) can request of another (the performer). The information object class `OPERATION`, to which all operations belong, is specified as follows, the various fields being described in 8.2.2 to 8.2.13:


```
OPERATION ::= CLASS
{
        &ArgumentType                   OPTIONAL,
        &argumentTypeOptional           BOOLEAN OPTIONAL,
        &returnResult                   BOOLEAN DEFAULT TRUE,
        &ResultType                     OPTIONAL,
        &resultTypeOptional             BOOLEAN OPTIONAL,
        &Errors                         ERROR OPTIONAL,
        &Linked                         OPERATION OPTIONAL,
        &synchronous                    BOOLEAN DEFAULT FALSE,
        &alwaysReturns                  BOOLEAN DEFAULT TRUE,
        &InvokePriority                 Priority OPTIONAL,
        &ResultPriority                 Priority OPTIONAL,
        &operationCode                  Code UNIQUE OPTIONAL
}
WITH SYNTAX
{
        [ARGUMENT                       &ArgumentType   [OPTIONAL   &argumentTypeOptional]]
        [RESULT                         &ResultType     [OPTIONAL   &resultTypeOptional]]
        [RETURN RESULT                  &returnResult]
        [ERRORS                         &Errors]
        [LINKED                         &Linked]
        [SYNCHRONOUS                    &synchronous]
        [ALWAYS RESPONDS                &alwaysReturns]
        [INVOKE PRIORITY                &InvokePriority]
        [RESULT-PRIORITY                &ResultPriority]
        [CODE                           &operationCode]
}
```

8.2.2 The `&ArgumentType` field specifies the data type of the argument of the operation. If in some operation the field is omitted, then that operation takes no argument value.

8.2.3 The `&argumentTypeOptional` field, which can be present only if the `&ArgumentType` field is present, specifies if the data type of the operation argument may optionally be omitted. If this field is absent or takes the value `FALSE`, the value of the `&ArgumentType` cannot be omitted from the `Invoke{}` PDU (see 9.3).

8.2.4 The `&returnResult` field specifies whether a result is returned in the event that the operation is performed successfully, taking the value `TRUE` if it is, and `FALSE` otherwise.

8.2.5 The `&ResultType` field specifies the data type of the value returned with the result of the operation. If it is omitted, then the operation returns no result value. It shall be omitted if the `&returnResult` field is `FALSE`.

8.2.6 The `&resultTypeOptional` field, which can be present only if the `&ResultType` field is present, specifies if the data type of the value returned as the result of performing the operation may optionally be omitted. If this field is absent or takes the value `FALSE`, the value of the `&ResultType` cannot be omitted from the `ReturnResult{}` PDU (see 9.4).

8.2.7 The `&Errors` field specifies a set of errors, any one of which may be returned to report an unsuccessful performance of the operation. If this field is omitted, then unsuccessful performance of the operation is either not possible or not reported.

8.2.8 The `&alwaysReturns` field specifies whether the outcome of performing the operation is always returned, taking the value `TRUE` if it is and `FALSE` otherwise. If this field is set to `TRUE`, at least one of the `&returnResult` or `&Errors` field must be present.

8.2.9 The `&Linked` field, if present, specifies a set of operations, any of which may be invoked as linked operations during the performance of the operation. If this field is omitted, then no operations may be linked to an invocation of this one.

8.2.10 The `&synchronous` field specifies whether or not the operation is synchronous, taking the value `TRUE` if it is, and `FALSE` otherwise. If the `&returnResult` field is `FALSE`, this field shall also take the value `FALSE`. When the `&synchronous` field is set to `TRUE`, it implies that no other *synchronous* operation may be invoked by this side until this operation has returned.

NOTE – The combination of the `&alwaysReturns` and the `&synchronous` fields replaces the earlier concept of "operation classes" defined in CCITT Rec. X.219 | ISO/IEC 9072-1.

8.2.11 The `&InvokePriority` field specifies the permitted `Priority` (see 8.9) levels at which this operation can be invoked.

8.2.12 The `&ResultPriority` field specifies the permitted `Priority` (see 8.9) levels at which the result of this operation can be returned. If the `&returnResult` field is `FALSE`, this field shall be omitted.

8.2.13 The `&operationCode` field, if present, specifies the `Code` value (see 8.8) which is used to identify this operation, e.g. when it is to be invoked.

NOTE – An operation which does not have an `&operationCode` cannot be invoked using the `Invoke{}` PDU (see 9.3). In practice, therefore, all operations should have `&operationCodes` assigned except when intended for use in some special circumstances, e.g. as a bind operation.

8.3 Error

8.3.1 An error is a report of the unsuccessful performance of an operation. The information object class `ERROR`, to which all errors belong, is specified as follows, the various fields being described in 8.3.2 to 8.3.5:

```
ERROR ::= CLASS
{
        &ParameterType              OPTIONAL,
        &parameterTypeOptional      BOOLEAN OPTIONAL,
        &ErrorPriority              Priority OPTIONAL,
        &errorCode                  Code UNIQUE OPTIONAL
}
WITH SYNTAX
{
        [PARAMETER                  &ParameterType  [OPTIONAL  &parameterTypeOptional]]
        [PRIORITY                   &ErrorPriority]
        [CODE                       &errorCode]
}
```

8.3.2 The `&ParameterType` field specifies the data type of the parameter of the error. If in some error the field is omitted, then that error takes no parameter value.