INTERNATIONAL TELECOMMUNICATION UNION

**ITU-T**

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

**Amendment 1**
**X.880**
(11/95)

**DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS**

**OSI APPLICATIONS – REMOTE OPERATIONS**

# INFORMATION TECHNOLOGY – REMOTE OPERATIONS: CONCEPTS, MODEL AND NOTATION

## AMENDMENT 1: BUILT-IN OPERATIONS

**Amendment 1 to**

**ITU-T Recommendation X.880**

(Previously "CCITT Recommendation")

## FOREWORD

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

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

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC. The text of ITU-T Recommendation X.880, Amendment 1, was approved on 21st of November 1995. The identical text is also published as ISO/IEC International Standard 13712-1.

NOTE

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

ITU-T X-SERIES RECOMMENDATIONS

**DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS**

(February 1994)

**ORGANIZATION OF X-SERIES RECOMMENDATIONS**

| Subject area | Recommendation Series |
|---|---|
| PUBLIC DATA NETWORKS | |
| Services and Facilities | X.1-X.19 |
| Interfaces | X.20-X.49 |
| Transmission, Signalling and Switching | X.50-X.89 |
| Network Aspects | X.90-X.149 |
| Maintenance | X.150-X.179 |
| Administrative Arrangements | X.180-X.199 |
| OPEN SYSTEMS INTERCONNECTION | |
| Model and Notation | X.200-X.209 |
| Service Definitions | X.210-X.219 |
| Connection-mode Protocol Specifications | X.220-X.229 |
| Connectionless-mode Protocol Specifications | X.230-X.239 |
| PICS Proformas | X.240-X.259 |
| Protocol Identification | X.260-X.269 |
| Security Protocols | X.270-X.279 |
| Layer Managed Objects | X.280-X.289 |
| Conformance Testing | X.290-X.299 |
| INTERWORKING BETWEEN NETWORKS | |
| General | X.300-X.349 |
| Mobile Data Transmission Systems | X.350-X.369 |
| Management | X.370-X.399 |
| MESSAGE HANDLING SYSTEMS | X.400-X.499 |
| DIRECTORY | X.500-X.599 |
| OSI NETWORKING AND SYSTEM ASPECTS | |
| Networking | X.600-X.649 |
| Naming, Addressing and Registration | X.650-X.679 |
| Abstract Syntax Notation One (ASN.1) | X.680-X.699 |
| OSI MANAGEMENT | X.700-X.799 |
| SECURITY | X.800-X.849 |
| OSI APPLICATIONS | |
| Commitment, Concurrency and Recovery | X.850-X.859 |
| Transaction Processing | X.860-X.879 |
| Remote Operations | X.880-X.899 |
| OPEN DISTRIBUTED PROCESSING | X.900-X.999 |

## CONTENTS

## Summary

This amendment to Rec. X.880 | ISO/IEC 13712-1 provides the definition of three built-in operations – Probe, Acknowledge and Cancel – which are of general utility to designers of ROSE-based applications.

**INTERNATIONAL STANDARD**

**ITU-T RECOMMENDATION**

# INFORMATION TECHNOLOGY – REMOTE OPERATIONS: CONCEPTS, MODEL AND NOTATION

# AMENDMENT 1
# Built-in operations

## 1) Subclause 3.3

*Add the following new definition immediately after 3.3.7:*

"**3.3.8 idempotent**: A characteristic of an operation that it can be invoked repeatedly without changing the state of the performer."

*The definitions which follow definition 3.3.8, should be renumbered accordingly.*

## 2) Subclause 8.2.1

*Add the following field underlined to the OPERATION information object class:*

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

## 3) Subclause 8.2

*Add a new subclause as follows:*

**"8.2.14** The `&idempotent` field specifies whether or not the operation is idempotent, taking the value `TRUE` if it is, and `FALSE` otherwise."

## 4) Subclause 10.1

*Rewrite item a) as follows (with the new text underlined):*

"a) generally useful operations, (`emptyBind, emptyUnbind, no-op,` <u>`probe, acknowledge, cancel`</u>), and their associated errors;"

## 5) Subclause 10.5.1

*Rewrite the no-op OPERATION definition by adding an additional field (underlined) as follows:*

```
no-op OPERATION ::=
{
        IDEMPOTENT              TRUE
        ALWAYS RESPONDS         FALSE
        CODE                    local:-1
}
```

## 6) Subclause 10.5.2

*Rewrite 10.5.2 as follows (with the new text underlined):*

"**10.5.2** The operation <u>is idempotent</u> and does not return."

## 7) Subclauses 10.6 through 10.16

*Renumber 10.6 through 10.16 as 10.12 through 10.22 respectively.*

## 8) Subclauses 10.6 through 10.11

*Add the following new subclauses numbered 10.6 through 10.11:*

### 10.6 Probe

**10.6.1** The `probe` operation enquires about the outcome of a previously invoked operation. It is specified as follows:

```
probe OPERATION ::=
{
        ARGUMENT        SEQUENCE
             {
             invokeId   [0] InvokeId
             }
        RESULT          ENUMERATED{running(0), finished(1), unknown(2), ...}
        IDEMPOTENT      TRUE
        CODE            local:-2
}
```

**10.6.2** There is a single argument, of type `InvokeId`, which identifies the invoked operation being enquired about.

**10.6.3** The request always returns a result, which indicates whether the operation invocation is still `running`, its performance is `finished`, or that it is `unknown`.

NOTE – An invocation may be unknown because it never happened, or because it has been forgotten by the performer.

**10.6.4** The operation is idempotent.

**10.6.5** A `probe` (with a result of `finished`) causes, as a side effect, the retransmission of any return from the invocation concerned, except if the operation was idempotent.

NOTE – This implies that the performer of a non-idempotent operation has to retain the response (result or error) if the probe operation has been included in the operation package.

## 10.7 Acknowledge

**10.7.1** The acknowledge operation acknowledges receipt of the return of some (non-idempotent) operation invocation. It is specified as follows:

```
acknowledge OPERATION ::=
{
          ARGUMENT      InvokeId
          RESULT        ENUMERATED{acknowledged(0), unknown(1), ...}
          IDEMPOTENT    TRUE
          CODE          local:-3
}
```

**10.7.2** There is a single argument, of type `InvokeId`, which identifies the invocation whose return is being acknowledged.

**10.7.3** The request always returns a result, which indicates either that the return is now considered `acknowledged`, or that the operation invocation concerned is `unknown`.

NOTE – An invocation may be unknown because it never happened, or because it has been forgotten by the performer.

**10.7.4** The operation is idempotent.

**10.7.5** This operation must be included in every operation package which includes the `probe` operation.

## 10.8 Probe and Acknowledge

**10.8.1** The `ProbeAndAcknowledge` operation set comprises the two operations suggested by its name, and will frequently both be needed in a package. It is specified as follows:

```
ProbeAndAcknowledge OPERATION ::= {probe | acknowledge}
```

## 10.9 Cancel

**10.9.1** The `cancel` operation requests the premature termination of the performance of an operation. Only operations which include the `cancelled` error (see 10.11) in their `&Errors` field can be cancelled. It is specified as follows:

```
cancel OPERATION ::=
{
          ARGUMENT      InvokeId
          ERRORS        {cancelFailed}
          IDEMPOTENT    TRUE
          CODE          local:-4
}
```

**10.9.2** There is a single argument, of type `InvokeId`, which identifies the invoked operation being cancelled.

**10.9.3** Should the request fail, a `cancelFailed` error (see 10.10) will be returned.

**10.9.4** The operation is idempotent.

## 10.10 Cancel failed

**10.10.1** A `cancelFailed` error reports a problem in performing a `cancel`. It is specified as follows:

```
cancelFailed ERROR ::=
{
        PARAMETER               SET
        {
            problem             [0]  CancelProblem,
            operation           [1]  InvokeId
        }
        CODE                    local:-2
}

CancelProblem ::= ENUMERATED
                {unknownOperation(0), tooLate(1), operationNotCancellable(2), ...}
```

**10.10.2** The various parameters have the meaning as defined in 10.10.2.1 and 10.10.2.2.

**10.10.2.1** The particular `problem` encountered with cancellation is indicated from the following possibilities:

a) `unknownOperation` – This operation invocation has either not happened, or has been forgotten.

b) `tooLate` – The operation has already been performed, or the execution is at a stage that does not permit a cancellation.

c) `operationNotCancellable` – The operation that was invoked was not one of those able to be cancelled.

**10.10.2.2** The identification of the `operation` (invocation) which was to be cancelled.

## 10.11 Cancelled

The `cancelled` error is reported if some operation is cancelled. The error must be included in the `&Errors` field of the affected operation. It is specified as follows:

```
cancelled ERROR ::= {CODE local:-3}
```

## 9) Annex A

*Change the first module reference as follows (with the change underlined):*

**Remote-Operations-Information-Objects {joint-iso-itu-t remote-operations(4) informationObjects(5) version<u>2(1)</u>}**

*Add the following field (underlined) to the OPERATION information object class:*

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

*Change the third module reference as follows (with the change underlined):*

**Remote-Operations-Useful-Definitions {joint-iso-itu-t remote-operations(4) useful-definitions(7) version2(1)}**

*Change the no-op OPERATION definition by adding an additional field (underlined) as follows:*

```
no-op OPERATION ::=
{
        IDEMPOTENT                TRUE
        ALWAYS RESPONDS           FALSE
        CODE                      local:-1
}
```

*Add the following new items to this module:*

```
probe OPERATION ::=
{
        ARGUMENT      SEQUENCE
               {
               invokeId      [0] InvokeId
               }
        RESULT        ENUMERATED{running(0), finished(1), unknown(2), ...}
        IDEMPOTENT    TRUE
        CODE          local:-2
}

acknowledge OPERATION ::=
{
        ARGUMENT      InvokeId
        RESULT        ENUMERATED{acknowledged(0), unknown(1), ...}
        IDEMPOTENT    TRUE
        CODE          local:-3
}

ProbeAndAcknowledge OPERATION ::= {probe | acknowledge}

cancel OPERATION ::=
{
        ARGUMENT      InvokeId
        ERRORS        {cancelFailed}
        IDEMPOTENT    TRUE
        CODE          local:-4
}

cancelFailed ERROR ::=
{
        PARAMETER             SET
        {
               problem               [0]  CancelProblem,
               operation             [1]  InvokeId
        }
        CODE                  local:-2
}

CancelProblem ::= ENUMERATED
                   {unknownOperation(0), tooLate(1), operationNotCancellable(2), ...}

cancelled ERROR ::= {CODE local:-3}
```

## 10) Annex D

*Make the following changes to the table (with the changes underlined):*

| **Clause** | **Object Identifier Value** |
|---|---|
| Annex A | **{joint-iso-itu-t remote-operations(4) informationObjects(5) version<u>2(1)</u>}** |
| | **{joint-iso-itu-t remote-operations(4) useful-definitions(7) version<u>2(1)</u>}** |