INTERNATIONAL STANDARD

ISO/IEC 9596

First edition
1990-05-15

Information technology — Open Systems Interconnection — Common management information protocol specification

Technologies de l'information — Interconnexion de systèmes ouverts — Spécification du protocole commun d'information de gestion

Reference number
ISO 9596 : 1990 (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 9596 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Annex A forms an integral part of this International Standard. Annexes B and C are for information only.

Information technology — Open Systems Interconnection — Common management information protocol specification

1 Scope

This International Standard specifies a protocol which is used by application layer entities to exchange management information.

This International Standard specifies

— procedures for the transmission of management information between application entities;

— the abstract syntax of the Common Management Information Protocol and the associated encoding rules to be applied;

— procedures for the correct interpretation of protocol control information;

— the conformance requirements to be met by implementation of this International Standard.

This International Standard does not specify

— the structure or meaning of the management information that is transmitted by means of CMIP;

— the manner in which management is accomplished as a result of CMIP exchanges;

— the interactions which result in the use of CMIP.

2 Normative references

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

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

ISO/IEC 7498-4 : 1989, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework.*

ISO 8326 : 1987, *Information processing systems - Open Systems Interconnection - Basic connection oriented session service definition.*

ISO 8649 : 1987, *Information processing systems - Open Systems Interconnection - Service definition for the Association Control Service Element.*

ISO 8650 : 1987, *Information processing systems - Open Systems Interconnection - Protocol specification for the Association Control Service Element.*

ISO 8822 : 1987, *Information processing systems - Open Systems Interconnection - Connection oriented presentation service definition.*

ISO 8823 : 1987, *Information processing systems - Open Systems Interconnection - Connection oriented presentation protocol specification.*

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 9072-1 : 1989, *Information processing systems - Text Communication - Remote Operations - Part 1: Model, Notation and Service Definition.*

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

ISO/IEC 9595 : 1989, *Information technology - Open Systems Interconnection - Common Management Information Service Definition.*

3 Definitions

For the purposes of this International Standard, the following definitions apply.

3.1 Basic Reference Model definitions

This International Standard makes use of the following terms defined in ISO 7498.

a) application-service-element;
b) application-process;
c) real open system;
d) systems-management.

3.2 Management Framework definitions

This International Standard makes use of the following terms defined in ISO/IEC 7498-4.

a) managed object;
b) management information;
c) management information base;
d) systems management application-entity.

3.3 Remote Operations definitions

This International Standard makes use of the following terms defined in ISO/IEC 9072-1.

a) association-initiator;
b) association-responder;
c) linked-operations;
d) Remote Operations;
e) Remote Operation Service Element.

3.4 CMIS definitions

This International Standard makes use of the following terms defined in ISO/IEC 9595.

a) attribute;
b) common management information service element;
c) common management information services;
d) CMISE-service-provider;
e) CMISE-service-user;
f) invoker;
g) invoking CMISE-service-user;
h) performer;
i) performing CMISE-service-user.

3.5 ACSE definitions

This International Standard makes use of the following terms defined in ISO 8649.

a) application context;
b) application-association;
c) association.

3.6 Presentation definitions

This International Standard makes use of the following terms defined in ISO 8822.

a) abstract syntax;
b) transfer syntax.

4 Symbols and abbreviations

ACSE Association Control Service Element

APDU Application Protocol Data Unit

ASE Application Service Element

ASN.1 Abstract Syntax Notation One

CMIP Common Management Information Protocol

CMIPM Common Management Information Protocol Machine

CMIS Common Management Information Service

CMISE Common Management Information Service Element

DCS defined context set

PCI protocol control information

PDU protocol data unit

PICS protocol implementation conformance statement

RO remote operations

ROSE Remote Operations Service Element

SMAE systems management application-entity

5 Overview

The Common Management Information Protocol (CMIP) specifies protocol elements that may be used to provide the operation and notification services described in ISO/IEC 9595, which defines the Common Management Information Services (CMIS).

5.1 Service provided

The protocol specified in this International Standard supports the services defined in ISO/IEC 9595. These services are summarized in table 1.

Table 1 —
Common Management Information Services

Service	Type
M-EVENT-REPORT	confirmed/non-confirmed
M-GET	confirmed
M-SET	confirmed/non-confirmed
M-ACTION	confirmed/non-confirmed
M-CREATE	confirmed
M-DELETE	confirmed

5.2 Underlying services

This International Standard uses the RO-INVOKE, RO-RESULT, RO-ERROR and RO-REJECT-U services of the Remote Operations Service Element (ROSE) defined in ISO/IEC 9072-1. ROSE assumes the use of the presentation service defined in ISO 8822. The confirmed operations of CMIP are class 2 (asynchronous) or class 1 (synchronous) as required by the application. The unconfirmed operations of CMIP are Class 5 (synchronous, outcome not reported). CMIP uses association class 3.

If the extended service functional unit is successfully negotiated, ROSEapdus may be mapped on to presentation services other than the P-DATA service.

NOTE — For example, it may be necessary to modify the presentation defined context set (DCS) when the CMIP operation is sent to the peer CMISE-service-user. In this case, the ROSE APDU which carries the CMIP operation will be mapped onto the P-ALTER-CONTEXT service which is also used to perform the changes to the DCS.

Details of which other presentation services are required and how they are used, are described in the description of the application context in use on the association.

5.2.1 Service assumed from the ACSE

This International Standard assumes the use of the A-ASSOCIATE, A-RELEASE, A-ABORT, and A-P-ABORT services of the Association Control Service Element.

5.2.2 Service assumed from the presentation layer

ISO/IEC 9072-2 assumes the use of the P-DATA service of the presentation layer for the transfer of the RO-INVOKE, RO-RESULT, RO-ERROR and RO-REJECT PDUs.

5.3 Management information definitions

This International Standard defines the abstract syntax of the Common Management Information Protocol. Attributes specific to a particular managed object are specified by the International Standard which defines that object.

6 Elements of procedure

This clause provides definition for the procedural elements of the CMIP. The procedures define the transfer of CMIP PDUs whose structure, coding and relationship with the CMIS service primitives is specified in clause 7.

The Common Management Information Protocol Machine (CMIPM) accepts CMIS request and response service primitives, and issues CMIP PDUs initiating specific elements of procedure as specified in this clause.

A CMIPM shall accept any well-formed CMIP PDU, and pass it to the performing CMISE-service-user for processing, by means of CMIS indication and confirmation service primitives. If the received PDU is not well formed or does not contain a supported notification or operation, a PDU is returned indicating that the received PDU has been rejected.

The procedures indicate only how to interpret the various fields in the CMIP PDU, not what an invoking CMISE-service-user should do with the information it requests nor how a performing CMISE-service-user should process the invocation.

6.1 Association establishment

The establishment of an association involves two CMISE-service-users, one that is the association-initiator and one that is the association-responder.

A CMISE-service-user may initiate an association establishment by using the A-ASSOCIATE service of ISO 8649.

The application context specifies, among other things, the rules required for the coordination of initialisation information corresponding to different ASEs. The association rules for CMISE are specified in annex A.

6.2 Remote operations

6.2.1 RO elements of procedure

The CMIP elements of procedure rely on the following underlying Remote Operations elements of procedure

a) invocation;
b) return-result;
c) return-error;
d) user-reject;
e) provider-reject.

These elements of procedure are described fully in ISO/IEC 9072-2.

6.2.2 RO-Reject problem parameters

The RO-Reject problem parameters are mapped or processed as follows

6.2.2.1 RO-Reject-User.Invoke-problem mapping to CMIS error codes is as follows

Table 2 —
Mapping RO-Reject-User.Invoke-problem to CMISE error codes

RO-REJECT parameter	CMISE error code
duplicate-invocation	duplicate invocation
mistyped-argument	mistyped argument
resource-limitation	resource limitation
unrecognized-operation	unrecognized operation

Other Invoke-problem parameters are a local matter.

6.2.2.2 Other RO-Reject parameters will be handled as a local matter.

6.3 Event reporting procedure

6.3.1 Invocation

The event reporting procedures are initiated by the M-EVENT-REPORT request primitive.

On receipt of the M-EVENT-REPORT request primitive, the CMIPM shall

a) in the confirmed mode, construct an APDU requesting the m-EventReport-Confirmed operation, otherwise, construct an APDU requesting the m-EventReport operation;

b) send the APDU using the RO-INVOKE procedure.

6.3.2 Receipt

On receipt of an APDU requesting either the m-EventReport or m-EventReport-Confirmed operation, the CMIPM shall, if the APDU is well formed, issue an M-EVENT-REPORT indication primitive to the CMISE-service-user with the mode parameter indicating whether or not confirmation is requested, otherwise, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.3.3 Response

In the confirmed mode, the CMIPM shall accept an M-EVENT-REPORT response primitive and shall

a) construct an APDU confirming the M-EVENT-REPORT notification;

b) if the parameters in the M-EVENT-REPORT response primitive indicate that the notification was accepted, send the APDU using the RO-RESULT procedure, otherwise, send the APDU using the RO-ERROR procedure.

6.3.4 Receipt of response

On receipt of an APDU responding to an M-EVENT-REPORT notification, the CMIPM shall, if the APDU is well formed, issue an M-EVENT-REPORT confirmation primitive to the CMISE-service-user, thus completing the notification

procedure, otherwise, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.4 Get procedure

6.4.1 Invocation

The Get procedures are initiated by the M-GET request primitive.

On receipt of the M-GET request primitive, the CMIPM shall

a) construct an APDU requesting the m-Get operation;

b) send the APDU using the RO-INVOKE procedure.

6.4.2 Receipt

On receipt of an APDU requesting the m-Get operation, the CMIPM shall, if the APDU is well formed, issue an M-GET indication primitive to the CMISE-service-user, otherwise, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.4.3 Response

The CMIPM shall

a) accept zero or more M-GET response primitives containing a linked-ID followed by a single M-GET response primitive without a linked-ID;

b) for each M-GET response primitive containing a linked-ID the CMIPM shall

— construct an APDU requesting the m-Linked-Reply operation with LinkedReplyArgument set appropriately as either getListError, getResult or processingFailure;

— send each APDU using the RO-INVOKE procedure.

c) for the M-GET response primitive not containing a linked-ID the CMIPM shall

— construct an APDU confirming the m-Get operation;

— if the parameters in the M-GET response primitive indicate that the operation was performed correctly, send the APDU using the RO-RESULT procedure. If the parameters in the M-GET response primitive indicate that the operation was performed with partial success or was not performed because of an error, send the APDU using the RO-ERROR procedure.

6.4.4 Receipt of response

On receipt of an APDU responding to an m-Get operation, the CMIPM shall

a) if the APDU included a linked-ID and is well formed, issue an M-GET confirm primitive to the CMISE-service-user;

b) if the APDU is the last response (i.e. not containing a linked-ID) and is well formed, issue an M-GET confirmation primitive to the CMISE-service-user, thus completing the M-GET procedure;

c) if the APDU is not well formed, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.5 Set procedure

6.5.1 Invocation

The Set procedures are initiated by the M-SET request primitive.

On receipt of the M-SET request primitive, the CMIPM shall

a) in the confirmed mode, construct an APDU requesting the m-Set-Confirmed operation, otherwise, construct an APDU requesting the m-Set operation;

b) send the APDU using the RO-INVOKE procedure.

6.5.2 Receipt

On receipt of an APDU requesting the m-Set or m-Set-Confirmed operation, the CMIPM shall, if the APDU is well formed, issue an M-SET indication primitive to the CMISE-service-user, with the mode parameter indicating whether or not confirmation is requested, otherwise, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.5.3 Response

In the confirmed mode, the CMIPM shall

a) accept zero or more M-SET response primitives containing a linked-ID followed by a single M-SET response primitive without a linked-ID;

b) for each M-SET response primitive containing a linked-ID the CMIPM shall

— construct an APDU requesting the m-Linked-Reply operation with LinkedReplyArgument set appropriately as either setListError, setResult or processingFailure;

— send each APDU using the RO-INVOKE procedure.

c) for the M-SET response primitive not containing a linked-ID the CMIPM shall

— construct an APDU confirming the m-Set operation;

— if the parameters in the M-SET response primitive indicate that the operation was performed correctly, send the APDU using the RO-RESULT procedure. If the parameters in the M-SET response primitive indicate that the operation was performed with partial success or was not performed because of an error, send the APDU using the RO-ERROR procedure.

6.5.4 Receipt of response

On receipt of an APDU responding to an m-Set-Confirmed operation, the CMIPM shall

a) if the APDU included a linked-ID and is well formed, issue an M-SET confirm primitive to the CMISE-service-user;

b) if the APDU is the last response (i.e. not containing a linked-ID) and is well formed, issue an M-SET confirmation

primitive to the CMISE-service-user, thus completing the M-SET procedure;

c) if the APDU is not well formed, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.6 Action procedure

6.6.1 Invocation

The Action procedures are initiated by the M-ACTION request primitive.

On receipt of the M-ACTION request primitive, the CMIPM shall

a) in the confirmed mode, construct an APDU requesting the m-Action-Confirmed operation otherwise, construct an APDU requesting the m-Action operation;

b) send the APDU using the RO-INVOKE procedure.

6.6.2 Receipt

On receipt of an APDU requesting the m-Action or m-Action-Confirmed operation, the CMIPM shall, if the APDU is well formed, issue an M-ACTION indication primitive to the CMISE-service-user, with the mode parameter indicating whether or not confirmation is requested, otherwise, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.6.3 Response

In the confirmed mode, the CMIPM shall

a) accept zero or more M-ACTION response primitives containing a linked-ID followed by a single M-ACTION response primitive without a linked-ID;

b) for each M-ACTION response primitive containing a linked-ID the CMIPM shall

— construct an APDU requesting the m-Linked-Reply operation with LinkedReplyArgument set appropriately as either actionError, actionResult or processingFailure;

— send each APDU using the RO-INVOKE procedure.

c) for the M-ACTION response primitive not containing a linked-ID the CMIPM shall

— construct an APDU confirming the m-Action operation;

— if the parameters in the M-ACTION response primitive indicate that the operation was performed correctly, send the APDU using the RO-RESULT procedure, otherwise, send the APDU using the RO-ERROR procedure.

6.6.4 Receipt of response

On receipt of an APDU responding to an m-Action-Confirmed operation, the CMIPM shall

a) if the APDU included a linked-ID and is well formed, issue an M-ACTION confirm primitive to the CMISE-service-user;

b)if the APDU is the last response (i.e. not containing a linked-ID) and is well formed, issue an M-ACTION confirmation primitive to the CMISE-service-user, thus completing the M-ACTION procedure;

c) if the APDU is not well formed, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.7 Create procedure

6.7.1 Invocation

The Create procedures are initiated by the M-CREATE request primitive.

On receipt of the M-CREATE request primitive, the CMIPM shall

a) construct an APDU requesting the m-Create operation;

b) send the APDU using the RO-INVOKE procedure.

6.7.2 Receipt

On receipt of an APDU requesting the m-Create operation, the CMIPM shall, if the APDU is well formed, issue an M-CREATE indication primitive to the CMISE-service-user, otherwise, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.7.3 Response

The CMIPM shall accept an M-CREATE response primitive and shall

a) construct an APDU confirming the m-Create operation;

b) if the parameters in the M-CREATE response primitive indicate that the operation was performed correctly, send the APDU using the RO-RESULT procedure, otherwise, send the APDU using the RO-ERROR procedure.

6.7.4 Receipt of response

On receipt of an APDU responding to an m-Create operation, the CMIPM shall, if the APDU is well formed, issue an M-CREATE confirmation primitive to the CMISE-service-user, thus completing the M-CREATE procedure, otherwise, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.8 Delete procedure

6.8.1 Invocation

The Delete procedures are initiated by the M-DELETE request primitive.

On receipt of the M-DELETE request primitive, the CMIPM shall

a) construct an APDU requesting the m-Delete operation;

b) send the APDU using the RO-INVOKE procedure.

6.8.2 Receipt

On receipt of an APDU requesting the m-Delete operation, the CMIPM shall, if the APDU is well formed, issue an M-DELETE indication primitive to the CMISE-service-user, otherwise, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.8.3 Response

The CMIPM shall

a) accept zero or more M-DELETE response primitives containing a linked-ID followed by a single M-DELETE response primitive without a linked-ID;

b) for each M-DELETE response primitive containing a linked-ID the CMIPM shall

— construct an APDU requesting the m-Linked-Reply operation with LinkedReplyArgument set appropriately as either deleteError, deleteResult or processingFailure;

—send each APDU using the RO-INVOKE procedure.

c) for the M-DELETE response primitive not containing a linked-ID the CMIPM shall

— construct an APDU confirming the m-Delete operation;

— if the parameters in the M-DELETE response primitive indicate that the operation was performed correctly, send the APDU using the RO-RESULT procedure, otherwise, send the APDU using the RO-ERROR procedure.

6.8.4 Receipt of response

On receipt of an APDU responding to an m-Delete operation, the CMIPM shall

a) if the APDU included a linked-ID and is well formed, issue an M-DELETE confirm primitive to the CMISE-service-user;

b) if the APDU is the last response (i.e. not containing a linked-ID) and is well formed, issue an M-DELETE confirmation primitive to the CMISE-service-user, thus completing the M-DELETE procedure;

c) if the APDU is not well formed, construct an APDU containing notification of the error and send it using the RO-REJECT-U procedure.

6.9 Association orderly release

Either CMISE-service-user may initiate an orderly release of the association by using the A-RELEASE service of ISO 8649.

NOTE — This specification is different from the ROSE use of the BIND operation in which only the association-initiator may use the A-RELEASE procedure.

6.10 Association abrupt release

Either CMISE-service-user may initiate an abrupt release of the association using the A-ABORT service of ISO 8649.

The CMISE-service-provider may initiate an abrupt release of the association using the A-P-ABORT service of ISO 8649.

7 Abstract syntax

This clause specifies the abstract syntax for the CMIP PDUs.

7.1 Conventions

The abstract syntax is defined using the notation specified in ISO 8824. The ASN.1 MACRO productions used or referenced by this International Standard do not exercise the ambiguous aspects of the grammar.

For each of the CMISE service parameters which is to be transferred by a CMIP PDU, there is a PDU field (an ASN.1 NamedType) with the same name as the corresponding service parameter (see ISO/IEC 9595), except for the differences required by the use of ASN.1, which are that blanks between words are removed and the first letter of the following word is capitalized, e.g. "managed object class" becomes "managedObjectClass". To make some of the names shorter, some words are abbreviated as follows

ack = acknowledgement
arg = argument
id = identifier
info = information
sync = synchronization

7.2 Correspondence between CMISE primitives and CMIP operations

Table 3 — Correspondence between CMISE primitives and CMIP operations

CMIS primitive	Mode	Linked-ID	CMIP operation
M-EVENT-REPORT req/ind	non-confirmed	not applicable	m-EventReport
M-EVENT-REPORT req/ind	confirmed	not applicable	m-EventReport-Confirmed
M-EVENT-REPORT rsp/conf	not applicable	not applicable	m-EventReport-Confirmed
M-GET req/ind	confirmed	not applicable	m-Get
M-GET rsp/conf	not applicable	absent	m-Get
M-GET rsp/conf	not applicable	present	m-Linked-Reply
M-SET req/ind	non-confirmed	not applicable	m-Set
M-SET req/ind	confirmed	not applicable	m-Set-Confirmed
M-SET rsp/conf	not applicable	absent	m-Set-Confirmed
M-SET rsp/conf	not applicable	present	m-Linked-Reply
M-ACTION req/ind	non-confirmed	not applicable	m-Action
M-ACTION req/ind	confirmed	not applicable	m-Action-Confirmed
M-ACTION rsp/conf	not applicable	absent	m-Action-Confirmed
M-ACTION rsp/conf	not applicable	present	m-Linked-Reply
M-CREATE req/ind	confirmed	not applicable	m-Create
M-CREATE rsp/conf	not applicable	not applicable	m-Create
M-DELETE req/ind	confirmed	not applicable	m-Delete
M-DELETE rsp/conf	not applicable	absent	m-Delete
M-DELETE rsp/conf	not applicable	present	m-Linked-Reply

NOTE — The mapping from the OPERATION macro to ROSE is as defined in ISO/IEC 9072-1.

7.3 ACSE user data

The ACSE protocol (ISO 8650) is described using ASN.1. The "user information" is defined using the EXTERNAL data type.

7.3.1 A-ASSOCIATE user data

The encoding of the CMIP user information to be passed to A-ASSOCIATE in the "user information" parameter is defined as follows

```
CMIP-A-ASSOCIATE-Information {joint-iso-ccitt ms(9) cmip(1) version1(1) aAssociateUserInfo(1)}
DEFINITIONS ::= BEGIN

FunctionalUnits ::= BIT STRING { multipleObjectSelection    (0),
                                 filter                     (1),
                                 multipleReply              (2),
                                 extendedService            (3) }

-- Functional unit i is supported if and only if bit i is one.

-- information carried in user-information parameter of A-ASSOCIATE

CMIPUserInfo ::= SEQUENCE { protocolVersion    [0] IMPLICIT ProtocolVersion DEFAULT { version1 },
                            functionalUnits    [1] IMPLICIT FunctionalUnits DEFAULT { },
                            accessControl      [2] EXTERNAL OPTIONAL,
                            userInfo           [3] EXTERNAL OPTIONAL }

ProtocolVersion ::= BIT STRING { version1 (0) }

END
```

The encoding of other "user information" supplied by the CMISE-service user is not defined by this International Standard.

7.3.2 A-ABORT user data

The encoding of the CMIP user information to be passed to A-ABORT in the "user information" parameter is defined as follows

```
CMIP-A-ABORT-Information {joint-iso-ccitt ms(9) cmip(1) version1(1) aAbortUserInfo(2)}
DEFINITIONS ::= BEGIN

-- information carried in user-information parameter of A-ABORT

CMIPAbortInfo ::= SEQUENCE { abortSource     [0] IMPLICIT CMIPAbortSource,
                             userInfo        [1] EXTERNAL OPTIONAL }

CMIPAbortSource ::= ENUMERATED { cmiseServiceUser          (0),
                                 cmiseServiceProvider      (1) }

END
```

The encoding of other "user information" supplied by the CMISE-service user is not defined by this International Standard.

7.4 CMIP data units

The protocol is described in terms of Common Management Information Protocol Data Units exchanged between the peer CMISEs. The PDUs are specified using ASN.1 and the Remote Operations Protocol OPERATION and ERROR external macros defined in ISO/IEC 9072-1.

```
-- Common Management Information Protocol (CMIP)

CMIP-1 {joint-iso-ccitt ms(9) cmip(1) version1(1) protocol(3)}
DEFINITIONS ::= BEGIN

-- Remote Operations definitions

IMPORTS OPERATION, ERROR FROM Remote-Operation-Notation {joint-iso-ccitt remoteOperations(4) notation(0)}

-- Directory Service definitions

DistinguishedName, RDNSequence FROM InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework(1)};

-- CMISE operations

-- in the following operations, the argument type is mandatory in the corresponding ROSE APDU

-- Action operations (M-ACTION)

m-Action OPERATION
  ARGUMENT ActionArgument
  ::= localValue 6

m-Action-Confirmed OPERATION
  ARGUMENT      ActionArgument
  RESULT        ActionResult
  -- this result is conditional; for conditions see ISO/IEC 9595 subclause 8.3.3.2.9
  ERRORS {      accessDenied, classInstanceConflict, complexityLimitation, invalidScope, invalidArgumentValue,
                invalidFilter, noSuchAction, noSuchArgument, noSuchObjectClass, noSuchObjectInstance,
                processingFailure, syncNotSupported }
  LINKED {      m-Linked-Reply }
  ::= localValue 7

-- Create operation (M-CREATE)

m-Create OPERATION
  ARGUMENT      CreateArgument
  RESULT        CreateResult
  -- this result is conditional; for conditions see ISO/IEC 9595 subclause 8.3.4.1.3
  ERRORS {      accessDenied, classInstanceConflict, duplicateManagedObjectInstance, invalidAttributeValue,
                invalidObjectInstance, missingAttributeValue, noSuchAttribute, noSuchObjectClass,
                noSuchObjectInstance, noSuchReferenceObject, processingFailure }
  ::= localValue 8
```

```
-- Delete operation (M-DELETE)

m-Delete OPERATION
  ARGUMENT      DeleteArgument
  RESULT        DeleteResult
  -- this result is conditional; for conditions see ISO/IEC 9595 subclause 8.3.5.2.8
  ERRORS {      accessDenied, classInstanceConflict, complexityLimitation, invalidFilter, invalidScope,
                noSuchObjectClass, noSuchObjectInstance, processingFailure, syncNotSupported }
  LINKED {      m-Linked-Reply }
  ::= localValue 9

-- Event Reporting operations (M-EVENT-REPORT)

m-EventReport OPERATION
  ARGUMENT EventReportArgument
  ::= localValue 0

m-EventReport-Confirmed OPERATION
  ARGUMENT      EventReportArgument
  RESULT        EventReportResult -- optional
  ERRORS {      invalidArgumentValue, noSuchArgument, noSuchEventType, noSuchObjectClass,
                noSuchObjectInstance, processingFailure }
  ::= localValue 1

-- Get operation (M-GET)

m-Get OPERATION
  ARGUMENT      GetArgument
  RESULT        GetResult
  -- this result is conditional; for conditions see ISO/IEC 9595 subclause 8.3.1.2.8
  ERRORS {      accessDenied, classInstanceConflict, complexityLimitation, getListError, invalidFilter,
                invalidScope, noSuchObjectClass, noSuchObjectInstance, processingFailure, syncNotSupported }
  LINKED {      m-Linked-Reply }
  ::= localValue 3

-- Linked operation to M-GET, M-SET (Confirmed), M-ACTION (Confirmed), and M-DELETE

m-Linked-Reply OPERATION
  ARGUMENT      LinkedReplyArgument
  ::= localValue 2

-- Set operations (M-SET)

m-Set OPERATION
  ARGUMENT SetArgument
  ::= localValue 4

m-Set-Confirmed OPERATION
  ARGUMENT      SetArgument
  RESULT        SetResult
  -- this result is conditional; for conditions see ISO/IEC 9595 subclause 8.3.2.2.9
  ERRORS {      accessDenied, classInstanceConflict, complexityLimitation, invalidFilter, invalidScope,
                noSuchObjectClass, noSuchObjectInstance, processingFailure, setListError, syncNotSupported }
  LINKED {      m-Linked-Reply }
  ::= localValue 5

-- CMIS error definitions

-- in the following errors, unless otherwise indicated, the parameter type is mandatory in the corresponding ROSE APDU

accessDenied ERROR
  ::= localValue 2

classInstanceConflict ERROR
  PARAMETER BaseManagedObjectId
  ::= localValue 19
```

```
complexityLimitation ERROR
  PARAMETER ComplexityLimitation -- optional
  ::= localValue 20

duplicateManagedObjectInstance ERROR
  PARAMETER ObjectInstance
  ::= localValue 11

getListError ERROR
  PARAMETER GetListError
  ::= localValue 7

invalidArgumentValue ERROR
  PARAMETER InvalidArgumentValue
  ::= localValue 15

invalidAttributeValue ERROR
  PARAMETER Attribute
  ::= localValue 6

invalidFilter ERROR
  PARAMETER CMISFilter
  ::= localValue 4

invalidScope ERROR
  PARAMETER Scope
  ::= localValue 16

invalidObjectInstance ERROR
  PARAMETER ObjectInstance
  ::= localValue 17

missingAttributeValue ERROR
  PARAMETER SET OF AttributeId
  ::= localValue 18

noSuchAction ERROR
  PARAMETER NoSuchAction
  ::= localValue 9

noSuchArgument ERROR
  PARAMETER NoSuchArgument
  ::= localValue 14

noSuchAttribute ERROR
  PARAMETER AttributeId
  ::= localValue 5

noSuchEventType ERROR
  PARAMETER NoSuchEventType
  ::= localValue 13

noSuchObjectClass ERROR
  PARAMETER ObjectClass
  ::= localValue 0

noSuchObjectInstance ERROR
  PARAMETER ObjectInstance
  ::= localValue 1

noSuchReferenceObject ERROR
  PARAMETER ObjectInstance
  ::= localValue 12

processingFailure ERROR
  PARAMETER ProcessingFailure -- optional
  ::= localValue 10
```

```
setListError ERROR
  PARAMETER SetListError
  ::= localValue 8

syncNotSupported ERROR
  PARAMETER CMISSync
  ::= localValue 3

-- Supporting type definitions.

AccessControl ::= EXTERNAL

ActionArgument ::= SEQUENCE { COMPONENTS OF   BaseManagedObjectId,
                              accessControl   [5] AccessControl OPTIONAL,
                              synchronization [6] IMPLICIT CMISSync DEFAULT bestEffort,
                              scope           [7] Scope DEFAULT baseObject,
                              filter          CMISFilter DEFAULT and {},
                              actionInfo      [12] IMPLICIT ActionInfo }

ActionError ::= SEQUENCE { managedObjectClass    ObjectClass OPTIONAL,
                           managedObjectInstance ObjectInstance OPTIONAL,
                           currentTime           [5] IMPLICIT GeneralizedTime OPTIONAL,
                           actionErrorInfo       [6] ActionErrorInfo }

ActionErrorInfo ::= SEQUENCE { errorStatus  ENUMERATED { accessDenied       (2),
                                                         noSuchAction       (9),
                                                         noSuchArgument     (14),
                                                         invalidArgumentValue (15) },
                               errorInfo    CHOICE { actionType     ActionTypeId,
                                                     actionArgument [0] NoSuchArgument,
                                                     argumentValue  [1] InvalidArgumentValue } }

ActionInfo ::= SEQUENCE { actionType    ActionTypeId,
                          actionInfoArg [4] ANY DEFINED BY actionType OPTIONAL }

ActionReply ::= SEQUENCE { actionType     ActionTypeId,
                           actionReplyInfo [4] ANY DEFINED BY actionType }

ActionResult ::= SEQUENCE { managedObjectClass    ObjectClass OPTIONAL,
                            managedObjectInstance ObjectInstance OPTIONAL,
                            currentTime           [5] IMPLICIT GeneralizedTime OPTIONAL,
                            actionReply           [6] IMPLICIT ActionReply OPTIONAL }

ActionTypeId ::= CHOICE { globalForm [2] IMPLICIT OBJECT IDENTIFIER,
                          localForm  [3] IMPLICIT INTEGER }
  -- This International Standard does not allocate any values for localForm
  -- where this alternative is used, the permissible values for the integers and their meanings
  -- shall be defined as part of the application context in which they are used

Attribute ::= SEQUENCE { attributeId    AttributeId,
                         attributeValue ANY DEFINED BY attributeId }

AttributeError ::= SEQUENCE { errorStatus  ENUMERATED { accessDenied       (2),
                                                        noSuchAttribute    (5),
                                                        invalidAttributeValue (6) },
                              attributeId  AttributeId,
                              attributeValue ANY DEFINED BY attributeId }

AttributeId ::= CHOICE { globalForm [0] IMPLICIT OBJECT IDENTIFIER,
                         localForm  [1] IMPLICIT INTEGER }
  -- This International Standard does not allocate any values for localForm
  -- where this alternative is used, the permissible values for the integers and their meanings
  -- shall be defined as part of the application context in which they are used

AttributeIdError ::= SEQUENCE { errorStatus  ENUMERATED { accessDenied    (2),
                                                          noSuchAttribute (5) },
                                attributeId  AttributeId }
```

```
BaseManagedObjectId ::= SEQUENCE { baseManagedObjectClass      ObjectClass,
                                   baseManagedObjectInstance   ObjectInstance }

CMISFilter ::= CHOICE { item    [8] FilterItem,
                        and     [9] IMPLICIT SET OF CMISFilter,
                        or      [10] IMPLICIT SET OF CMISFilter,
                        not     [11] CMISFilter }

CMISSync ::= ENUMERATED { bestEffort    (0),
                          atomic        (1) }

ComplexityLimitation ::= SET { scope    [0] Scope OPTIONAL,
                               filter   [1] CMISFilter OPTIONAL,
                               sync     [2] CMISSync OPTIONAL }

CreateArgument ::= SEQUENCE {
  managedObjectClass        ObjectClass,
                            CHOICE { managedObjectInstance    ObjectInstance,
                                     superiorObjectInstance   [8] ObjectInstance } OPTIONAL,
  accessControl             [5] AccessControl OPTIONAL,
  referenceObjectInstance   [6] ObjectInstance OPTIONAL,
  attributeList             [7] IMPLICIT SET OF Attribute OPTIONAL }

CreateResult ::= SEQUENCE { managedObjectClass      ObjectClass OPTIONAL,
                            managedObjectInstance   ObjectInstance OPTIONAL,
                                -- shall be returned if omitted from CreateArgument
                            currentTime             [5] IMPLICIT GeneralizedTime OPTIONAL,
                            attributeList           [6] IMPLICIT SET OF Attribute OPTIONAL }

DeleteArgument ::= SEQUENCE { COMPONENTS OF   BaseManagedObjectId,
                              accessControl   [5] AccessControl OPTIONAL,
                              synchronization [6] IMPLICIT CMISSync DEFAULT bestEffort,
                              scope           [7] Scope DEFAULT baseObject,
                              filter          CMISFilter DEFAULT and { } }

DeleteError ::= SEQUENCE { managedObjectClass      ObjectClass OPTIONAL,
                           managedObjectInstance   ObjectInstance OPTIONAL,
                           currentTime             [5] IMPLICIT GeneralizedTime OPTIONAL,
                           deleteErrorInfo         [6] ENUMERATED { accessDenied (2) } }

DeleteResult ::= SEQUENCE { managedObjectClass      ObjectClass OPTIONAL,
                            managedObjectInstance   ObjectInstance OPTIONAL,
                            currentTime             [5] IMPLICIT GeneralizedTime OPTIONAL }

EventReply ::= SEQUENCE { eventType         EventTypeId,
                          eventReplyInfo    [8] ANY DEFINED BY eventType OPTIONAL }

EventReportArgument ::= SEQUENCE { managedObjectClass      ObjectClass,
                                   managedObjectInstance   ObjectInstance,
                                   eventTime               [5] IMPLICIT GeneralizedTime OPTIONAL,
                                   eventType               EventTypeId,
                                   eventInfo               [8] ANY DEFINED BY eventType OPTIONAL }

EventReportResult ::= SEQUENCE { managedObjectClass      ObjectClass OPTIONAL,
                                 managedObjectInstance   ObjectInstance OPTIONAL,
                                 currentTime             [5] IMPLICIT GeneralizedTime OPTIONAL,
                                 eventReply              EventReply OPTIONAL }

EventTypeId ::= CHOICE { globalForm    [6] IMPLICIT OBJECT IDENTIFIER,
                         localForm     [7] IMPLICIT INTEGER }
  -- This International Standard does not allocate any values for localForm
  -- where this alternative is used, the permissible values for the integers and their meanings
  -- shall be defined as part of the application context in which they are used
```

```
FilterItem ::= CHOICE {
  equality                  [0] IMPLICIT Attribute,
  substrings                [1] IMPLICIT SEQUENCE OF CHOICE {
                                initialString   [0] IMPLICIT SEQUENCE {
                                                    attributeId     AttributeId,
                                                    string          ANY DEFINED BY attributeId},
                                anyString       [1] IMPLICIT SEQUENCE {
                                                    attributeId     AttributeId,
                                                    string          ANY DEFINED BY attributeId},
                                finalString     [2] IMPLICIT SEQUENCE {
                                                    attributeId     AttributeId,
                                                    string          ANY DEFINED BY attributeId} },
  greaterOrEqual            [2] IMPLICIT Attribute, -- asserted value ≥ attribute value
  lessOrEqual               [3] IMPLICIT Attribute, -- asserted value ≤ attribute value
  present                   [4] AttributeId,
  subsetOf                  [5] IMPLICIT Attribute, -- asserted value is a subset of attribute value
  supersetOf                [6] IMPLICIT Attribute, -- asserted value is a superset of attribute value
  nonNullSetIntersection    [7] IMPLICIT Attribute }

GetArgument ::= SEQUENCE { COMPONENTS OF    BaseManagedObjectId,
                           accessControl    [5] AccessControl OPTIONAL,
                           synchronization  [6] IMPLICIT CMISSync DEFAULT bestEffort,
                           scope            [7] Scope DEFAULT baseObject,
                           filter           CMISFilter DEFAULT and {},
                           attributeIdList  [12] IMPLICIT SET OF AttributeId OPTIONAL }

GetInfoStatus ::= CHOICE { attributeIdError  [0] IMPLICIT AttributeIdError,
                           attribute         [1] IMPLICIT Attribute }

GetListError ::= SEQUENCE { managedObjectClass      ObjectClass OPTIONAL,
                            managedObjectInstance   ObjectInstance OPTIONAL,
                            currentTime             [5] IMPLICIT GeneralizedTime OPTIONAL,
                            getInfoList             [6] IMPLICIT SET OF GetInfoStatus }

GetResult ::= SEQUENCE { managedObjectClass      ObjectClass OPTIONAL,
                         managedObjectInstance   ObjectInstance OPTIONAL,
                         currentTime             [5] IMPLICIT GeneralizedTime OPTIONAL,
                         attributeList           [6] IMPLICIT SET OF Attribute OPTIONAL }

InvalidArgumentValue ::= CHOICE { actionValue   [0] IMPLICIT ActionInfo,
                                  eventValue    [1] IMPLICIT SEQUENCE {
                                                    eventType   EventTypeId,
                                                    eventInfo   [8] ANY DEFINED BY eventType OPTIONAL } }

LinkedReplyArgument ::= CHOICE { getResult          [0] IMPLICIT GetResult,
                                 getListError       [1] IMPLICIT GetListError,
                                 setResult          [2] IMPLICIT SetResult,
                                 setListError       [3] IMPLICIT SetListError,
                                 actionResult       [4] IMPLICIT ActionResult,
                                 processingFailure  [5] IMPLICIT ProcessingFailure,
                                 deleteResult       [6] IMPLICIT DeleteResult,
                                 actionError        [7] IMPLICIT ActionError,
                                 deleteError        [8] IMPLICIT DeleteError }

NoSuchAction ::= SEQUENCE { managedObjectClass   ObjectClass,
                            actionType           ActionTypeId }

NoSuchArgument ::= CHOICE { actionId    [0] IMPLICIT SEQUENCE {
                                              managedObjectClass   ObjectClass OPTIONAL,
                                              actionType           ActionTypeId },
                            eventId     [1] IMPLICIT SEQUENCE {
                                              managedObjectClass   ObjectClass OPTIONAL,
                                              eventType            EventTypeId } }

NoSuchEventType ::= SEQUENCE { managedObjectClass   ObjectClass,
                               eventType            EventTypeId }
```

```
ObjectClass ::= CHOICE { globalForm      [0] IMPLICIT OBJECT IDENTIFIER,
                         localForm       [1] IMPLICIT INTEGER }
  -- This International Standard does not allocate any values for localForm.
  -- where this alternative is used, the permissible values for the integers and their meanings
  -- shall be defined as part of the application context in which they are used

ObjectInstance ::= CHOICE { distinguishedName          [2] IMPLICIT DistinguishedName,
                            nonSpecificForm            [3] IMPLICIT OCTET STRING,
                            localDistinguishedName     [4] IMPLICIT RDNSequence }
  -- localDistinguishedName is that portion of the distinguished name that is necessary to unambiguously
  -- identify the managed object within the context of communication between the open systems

ProcessingFailure ::= SEQUENCE { managedObjectClass      ObjectClass,
                                 managedObjectInstance   ObjectInstance OPTIONAL,
                                 specificErrorInfo       [5] ANY DEFINED BY managedObjectClass }

Scope ::= CHOICE { INTEGER { baseObject          (0),
                             firstLevelOnly      (1),
                             wholeSubtree        (2) },
                   individualLevels    [1] IMPLICIT INTEGER,   -- POSITIVE integer indicates the level to be selected
                   baseToNthLevel      [2] IMPLICIT INTEGER }  -- POSITIVE integer N indicates that the range of levels
                                                               -- (0 - N) is to be selected
  -- with individualLevels and baseToNthLevel, a value of 0 has the same semantics as baseObject
  -- with individualLevels, a value of 1 has the same semantics as firstLevelOnly

SetArgument ::= SEQUENCE { COMPONENTS OF  BaseManagedObjectId,
                           accessControl      [5] AccessControl OPTIONAL,
                           synchronization    [6] IMPLICIT CMISSync DEFAULT bestEffort,
                           scope              [7] Scope DEFAULT baseObject,
                           filter             CMISFilter DEFAULT and { },
                           attributeList      [12] IMPLICIT SET OF Attribute }

SetInfoStatus ::= CHOICE { attributeError    [0] IMPLICIT AttributeError,
                           attribute         [1] IMPLICIT Attribute }

SetListError ::= SEQUENCE { managedObjectClass      ObjectClass OPTIONAL,
                            managedObjectInstance   ObjectInstance OPTIONAL,
                            currentTime             [5] IMPLICIT GeneralizedTime OPTIONAL,
                            setInfoList             [6] IMPLICIT SET OF SetInfoStatus }

SetResult ::= SEQUENCE { managedObjectClass      ObjectClass OPTIONAL,
                         managedObjectInstance   ObjectInstance OPTIONAL,
                         currentTime             [5] IMPLICIT GeneralizedTime OPTIONAL,
                         attributeList           [6] IMPLICIT SET OF Attribute OPTIONAL }

END -- End of CMIP syntax definitions
```

7.5 Definition of abstract syntax for CMIP

This International Standard assigns the ASN.1 object identifier value

{joint-iso-ccitt ms(9) cmip(1) version1(1) abstractSyntax(4)}

as an abstract syntax name for the set of presentation data values, each of which is either a value of the ASN.1 type

Remote-Operations-APDUs.ROSEapdus

as defined in ISO/IEC 9072-2 with the argument component filled according to the definitions in CMIP-1, or a value of one of the ASN.1 types

— CMIP-A-ASSOCIATE-Information.CMIPUserInfo;

— CMIP-A-ABORT-Information.CMIPAbortInfo.

The corresponding ASN.1 object descriptor value shall be

"CMIP-PCI".

This abstract syntax is defined to include all data types resolved by the ANY DEFINED BY X productions, in which X is of type OBJECT IDENTIFIER.

The ASN.1 object identifier and object descriptor values

{joint-iso-ccitt asn1(1) basic-encoding(1)} and
"Basic Encoding of single ASN.1 type"

(assigned to an object in ISO 8825) can be used as a transfer syntax name with this abstract syntax.

8 Conformance

A system claiming to implement the procedures specified in this standard shall comply with the requirements in 8.1 to 8.3.

8.1 Static requirements

The system shall

a) support the kernel functional unit defined in ISO/IEC 9595, and the facilities implied by that functional unit;

b) support the transfer syntax derived from the encoding rules specified in ISO 8825 and named

{joint-iso-ccitt asn1(1) basic-encoding(1)}

for the purpose of generating and interpreting CMISE protocol information as defined by the abstract syntax

"CMIP-PCI"

for the functional units supported;

c) support the ACSE protocol defined in ISO 8650, to establish and to release an association;

d) support the rules specified in annex A in any application context that includes CMISE as one of the ASEs;

e) support association class 3 of the ROSE protocol defined in ISO/IEC 9072-2;

f) support the multiple reply functional unit if the multiple object selction functional unit is supported.

8.2 Dynamic requirements

The system shall

a) follow the procedures relevant to each functional unit that the system claims to implement;

b) when used, verify the optional security parameters defined in the CMIP PDUs;

c) when the extended service functional unit is supported, support the presentation protocol defined in ISO/IEC 8823, as required by the application context;

d) when scoping is provided, support the multiple reply functional unit.

8.3 PICS requirements

The following shall be stated by the implementer when defining the PICS corresponding to its implementation

a) which management information functional units, as defined in ISO/IEC 9595, are supported by the real open system on which the SMAE resides;

b) which abstract syntaxes for management information are supported by the real open system on which the SMAE resides;

c) which optional parameters are supported by the PDUs belonging to the supported functional units;

d) the types and ranges of values for all the parameters supported.

NOTE — These requirements may be supplemented by further requirements defined in other management standards.

A PICS proforma is provided in 8.4 below.

8.4 PICS proforma

• ISO/IEC 9596 details

CMIP protocol version number	

• ISO/IEC 9596 Addenda implemented (if any)

• Defect report numbers implemented (if any)

ISO/IEC 9596	

• Abstract syntaxes supported

Date of statement yy-mm-dd	

• Implementation details

Implementation Supplier	
Implementation Name Implementation Version Number	
Machine Name Machine Version Number	
Operating System Name Operating System Version Number	
Other Operating System and hardware claimed	
System Name (if different)	

In the declarations for functional units and parameters supported, the following conventions are used

SR = ISO/IEC 9596 Standard Requirement

"m" for mandatory,
"o" for optional.

IV = Implementation Value (depending on the location)

"y" for yes,
"n" for no.

• Functional units supported

Functional unit name	SR	IV
kernel	m	
multipleObjectSelection	o	
filter	o	
multipleReply	o	
extendedService	o	

• Parameters supported

M-ACTION	SR	IV
invoker support	m	
performer support	m	

Parameters	SR	IV	types and ranges supported
Invoke identifier	m		
Mode	m		
Base object class	m		
Base object instance	m		
Managed object class	m		
Managed object instance	m		
Access control	m		
Action type	m		
Action information	m		
Current time	m		
Action reply	m		
Errors	m		
Linked identifier	o		
Scope	o		
Filter	o		
Synchronization	o		

M-CREATE	SR	IV
invoker support	m	
performer support	m	

Parameters	SR	IV	types and ranges supported
Invoke identifier	m		
Managed object class	m		
Managed object instance	m		
Superior object instance	m		
Access control	m		
Reference object instance	m		
Attribute list	m		
Current time	m		
Errors	m		

M-DELETE	SR	IV
invoker support	m	
performer support	m	

Parameters	SR	IV	types and ranges supported
Invoke identifier	m		
Base object class	m		
Base object instance	m		
Access control	m		
Managed object class	m		
Managed object instance	m		
Current time	m		
Errors	m		
Linked identifier	o		
Scope	o		
Filter	o		
Synchronization	o		

M-EVENT-REPORT	SR	IV	
invoker support	m		
performer support	m		
Parameters	**SR**	**IV**	**types and ranges supported**
Invoke identifier	m		
Mode	m		
Managed object class	m		
Managed object instance	m		
Event type	m		
Event time	m		
Event information	m		
Current time	m		
Event reply	m		
Errors	m		

M-GET	SR	IV	
invoker support	m		
performer support	m		
Parameters	**SR**	**IV**	**types and ranges supported**
Invoke identifier	m		
Base object class	m		
Base object instance	m		
Access control	m		
Attribute identifier list	m		
Managed object class	m		
Managed object instance	m		
Current time	m		
Attribute list	m		
Errors	m		
Linked identifier	o		
Scope	o		
Filter	o		
Synchronization	o		

M-SET	SR	IV	
invoker support	m		
performer support	m		
Parameters	**SR**	**IV**	**types and ranges supported**
Invoke identifier	m		
Mode	m		
Base object class	m		
Base object instance	m		
Access control	m		
Managed object class	m		
Managed object instance	m		
Attribute list	m		
Current time	m		
Errors	m		
Linked identifier	o		
Scope	o		
Filter	o		
Synchronization	o		

Annex A

(normative)

Association rules for CMISE

A.1 ACSE, session and presentation requirements

A.1.1 CMISE requires the kernel presentation functional unit as defined in ISO 8822.

A.1.2 CMISE requires the kernel and full duplex session functional units as defined in ISO 8326.

A.1.3 CMISE requires the normal mode of ACSE and presentation services as defined in ISO 8649 and ISO 8822.

A.2 Association initialisation rules

A.2.1 Request

The CMISE-service-user that initiates the association establishment shall provide the A-ASSOCIATE user information defined by ISO/IEC 9595. The CMIP user information shall be made available to the CMIPM which shall

a) construct CMIPUserInfo from the information supplied;

b) set the protocol version parameter within CMIPUserInfo by setting the bit corresponding to each version supported;

c) include CMIPUserInfo as a separate EXTERNAL in the user information parameter of the A-ASSOCIATE request primitive;

d) wait for the user information specific to CMIS to be returned in the A-ASSOCIATE confirm primitive.

A.2.2 Indication

On receipt of an A-ASSOCIATE indication primitive, the CMIPUserInfo parameter shall be made available to the CMIPM which shall

a) check that at least one of the proposed protocol version can be supported;

b) verify that the optional access control parameter is valid and that the association-initiator has sufficient privileges to establish an association which uses CMISE;

c) if any of the checks fail, the association shall be rejected by setting the reason for failure parameter in the A-ASSOCIATE response primitive to "rejected by responder (permanent)". The association is not established and that instance of the CMIPM shall cease to exist;

d) if the above checks succeed, the CMIPUserInfo shall be made available to the CMISE-service-user and the CMIPM shall wait for the response from the CMISE-service-user.

A.2.3 Response

The A-ASSOCIATE response primitive indicating "accepted" or "rejected" shall be made available to the CMIPM which shall

a) construct CMIPUserInfo required for the response. The CMIPUserInfo shall include the version parameter indicating all versions of CMIP that are supported;

b) include CMIPUserInfo as a separate EXTERNAL in the user information parameter of the A-ASSOCIATE response primitive;

c) if the association response indicates "accepted", the protocol version agreed to is the version corresponding to the highest number supported by both CMIPMs. The CMIPM shall then be ready to accept CMISE indication primitives;

d) if the association response indicates "rejected", that instance of the CMIPM shall cease to exist.

A.2.4 Confirmation

On receipt of the A-ASSOCIATE confirmation primitive, the CMIPUserInfo parameter shall be made available to the CMIPM which shall

a) if the association confirmation indicates success, the association is established. The functional units agreed to correspond to those for which both CMISE-service-users indicated support and the protocol version is the highest version number supported by both CMIPMs;

b) if the association confirmation indicates failure, the association is not established and that instance of the CMIPM shall cease to exist.

A.3 Association release rules

Either CMISE-service-user may initiate an association release.

A.3.1 Request

On receipt of a request for association release, the necessary A-RELEASE parameters shall be made available to the CMIPM which shall cease to accept service requests and wait for the confirmation of the release of the association.

A.3.2 Indication

On receipt of an A-RELEASE indication primitive, the necessary A-RELEASE indication parameters shall be made available to the responding CMIPM which shall wait for the association release response.

A.3.3 Response

On receipt of an association release response from the responding CMISE-service-user, the necessary A-RELEASE response parameters shall be made available to the responding CMIPM. Thereafter, that instance of the CMIPM shall cease to exist.

A.3.4 Confirmation

On receipt of an A-RELEASE confirm primitive, the necessary A-RELEASE confirm parameters shall be made available to the initiating CMIPM. Thereafter, that instance of the CMIPM shall cease to exist.

A.4 Association abort rules

Either CMISE-service-user may initiate an abrupt termination of the association.

On the basis of local information, if the ability of the underlying services to convey unlimited user information by A-ABORT does not exist, the CMIPAbortInfo parameter may not be included in the A-ABORT service primitives.

A.4.1 A-ABORT request

On receipt of a request to abort the association, the necessary A-ABORT request parameters including the A-ABORT user information defined by ISO/IEC 9595 shall be made available to the CMIPM which shall

a) construct CMIPAbortInfo from the information supplied;

b) set the abort source parameter within CMIPUserInfo to CMISE-service-user;

c) include CMIPAbortInfo as a separate field in the user information parameter of the A-ABORT request primitive;

d) thereafter, that instance of the CMIPM shall cease to exist.

A.4.2 A-ABORT Indication

On receipt of an A-ABORT indication primitive, the necessary A-ABORT indication parameters including CMIPAbortInfo shall be made available to the CMIPM. Thereafter, that instance of the CMIPM shall cease to exist.

A.4.3 A-P-ABORT Indication

On receipt of an A-P-ABORT indication primitive, the necessary A-P-ABORT indication parameters shall be made available to the CMIPM. Thereafter, that instance of the CMIPM shall cease to exist.

A.4.4 CMIP protocol error

On detecting a protocol error, the CMIPM shall

a) construct CMIPAbortInfo with the abort source parameter set to CMISE-service-provider;

b) indicate to the CMISE-service-user that a protocol error has occurred;

c) include CMIPAbortInfo as a separate field in the user information parameter of the A-ABORT request primitive;

d) thereafter, that instance of the CMIPM shall cease to exist.

Annex B

(informative)

Expanded ASN.1 syntax

This annex describes how the OPERATION and ERROR macros of ISO/IEC 9072-1 are expanded into ASN.1 data types and subtypes.

If any inconsistencies exist between these definitions and the definitions in clause 7, then the definitions in clause 7 take precedence.

```
-- Common Management Information Protocol (CMIP)

CMIP-1 {joint-iso-ccitt ms(9) cmip(1) version1(1) protocol(3)}
DEFINITIONS ::= BEGIN

-- Remote Operations definitions

IMPORTS OPERATION, ERROR FROM Remote-Operation-Notation {joint-iso-ccitt remoteOperations(4) notation(0)}

-- Directory Service definitions

DistinguishedName, RDNSequence FROM InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework(1)};

-- CMISE operations

ROSEapdus ::= CHOICE { roiv-apdu     [1] IMPLICIT ROIVapdu,
                       rors-apdu     [2] IMPLICIT RORSapdu,
                       roer-apdu     [3] IMPLICIT ROERapdu,
                       rorj-apdu     [4] IMPLICIT RORJapdu }

ROIVapdu ::= SEQUENCE { invokeID         InvokeIDType,
                        linked-ID        [0] IMPLICIT InvokeIDType OPTIONAL,
                        operation-value  OPERATION,
                        argument         ANY DEFINED BY operation-value OPTIONAL }

RORSapdu ::= SEQUENCE { invokeID         InvokeIDType,
                        SEQUENCE { operation-value  OPERATION,
                                   result           ANY DEFINED BY operation-value } OPTIONAL }

ROERapdu ::= SEQUENCE { invokeID     InvokeIDType,
                        error-value  ERROR,
                        parameter    ANY DEFINED BY error-value OPTIONAL }

RORJapdu ::= SEQUENCE { invokeID     CHOICE { InvokeIDType,
                                              NULL },
                        problem      CHOICE { [0] IMPLICIT GeneralProblem,
                                              [1] IMPLICIT InvokeProblem,
                                              [2] IMPLICIT ReturnResultProblem,
                                              [3] IMPLICIT ReturnErrorProblem } }

InvokeIdType ::= INTEGER

-- The use of the GeneralProblem, ReturnResultProblem, and ReturnErrorProblem codes are a local issue.

GeneralProblem ::= INTEGER { unrecognisedAPDU      (0), -- ROSE-provider detected
                             mistypedAPDU          (1),
                             badlyStructuredAPDU   (2) }

InvokeProblem ::= INTEGER { duplicateInvocation       (0), -- ROSE-user detected
                            unrecognisedOperation     (1),
                            mistypedArgument          (2),
                            resourceLimitation        (3),
                            initiatorReleasing        (4),
                            unrecognisedLinkedID      (5),
                            linkedResponseUnexpected  (6),
                            unexpectedChildOperation  (7) }
```

```
ReturnResultProblem ::= INTEGER { unrecognisedInvocation      (0), -- ROSE-user detected
                                  resultResponseUnexpected    (1),
                                  mistypedResult              (2) }

ReturnErrorProblem ::= INTEGER { unrecognisedInvocation       (0), -- ROSE-user detected
                                 errorResponseUnexpected      (1),
                                 unrecognisedError            (2),
                                 unexpectedError              (3),
                                 mistypedParameter            (4) }

-- This part of the ASN.1 specification provides a definition of the InvokeProblem subtype used by CMIP.

InvokeProblem-CMIPUser ::= InvokeProblem ( duplicateInvocation      |
                                           unrecognisedOperation    |
                                           mistypedArgument         |
                                           resourceLimitation )

-- This part of the ASN.1 specification provides a definition of ROIVapdu and RORSapdu subtypes used by CMIP.
-- The subtypes of the ROIVapdu define the allowed values of the operation-value and argument defined by that
-- operation-value for all CMIP notifications and operations. The subtypes of the RORSapdu define the allowed
-- values of the operation-value and result defined by that operation-value for all CMIP notifications and operations.

m-Action OPERATION ::= localValue 6

ROIV-m-Action ::= ROIVapdu (WITH COMPONENTS
                        { invokeID          PRESENT,
                          linked-ID         ABSENT,
                          operation-value   (m-Action),
                          argument          (INCLUDES ActionArgument) } )

m-Action-Confirmed OPERATION ::= localValue 7

ROIV-m-Action-Confirmed ::= ROIVapdu (WITH COMPONENTS
                        { invokeID          PRESENT,
                          linked-ID         ABSENT,
                          operation-value   (m-Action-Confirmed),
                          argument          (INCLUDES ActionArgument) } )

RORS-m-Action-Confirmed ::= RORSapdu (WITH COMPONENTS
                        { ... ,
                          invokeID          PRESENT,
                          -- result sequence -- (WITH COMPONENTS
                                { operation-value   (m-Action-Confirmed),
                                  result            (INCLUDES ActionResult) } )
                          -- required only if there is a single reply to the ROIV-m-Action-Confirmed ROIVapdu
                          -- and data is to be returned in the RORSapdu
                          } )

m-Create OPERATION ::= localValue 8

ROIV-m-Create ::= ROIVapdu (WITH COMPONENTS
                        { invokeID          PRESENT,
                          linked-ID         ABSENT,
                          operation-value   (m-Create),
                          argument          (INCLUDES CreateArgument) } )

RORS-m-Create ::= RORSapdu (WITH COMPONENTS
                        { ... ,
                          invokeID          PRESENT,
                          -- result sequence -- (WITH COMPONENTS
                                { operation-value   (m-Create),
                                  result            (INCLUDES CreateResult) } )
                          } )
```