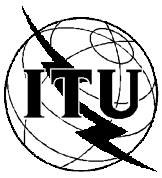

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

X.745

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(11/93)

DATA NETWORKS AND
OPEN SYSTEMS COMMUNICATIONS

OSI MANAGEMENT

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – SYSTEMS MANAGEMENT: TEST MANAGEMENT FUNCTION

ITU-T Recommendation X.745

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.745 was approved on 16th of November 1993. The identical text is also published as ISO/IEC International Standard 10164-12.

NOTE

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

© ITU 1994

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the ITU.

CONTENTS

		Page
1	Scope	1
2	Normative references	1
3	Definitions	3
4	Abbreviations	5
5	Conventions	5
6	Requirements	6
7	Model for the test function	6
8	Generic definitions	14
9	Service definitions	20
10	Systems management functional units	26
11	Protocol	27
12	Relationships with other functions	32
13	Conformance	32
Annex A – Definition of management information		34
Annex B – Examples		46

Summary

This Recommendation | International Standard specifies a model and managed objects for the invocation of tests on remote resources. Tests may be controlled and subject to monitoring, suspension and resumption during application or uncontrolled and where results are returned until complete.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – SYSTEMS MANAGEMENT: TEST MANAGEMENT FUNCTION

1 Scope

This Recommendation | International Standard defines a Systems Management Function that may be used by an application process in a centralized or decentralized management environment to interact for the purpose of systems management, as defined by CCITT Rec. X.700 | ISO/IEC 7498-4. This Recommendation | International Standard defines a function which consists of generic definitions, services and functional units. This function is positioned in the application layer of CCITT Rec. X.200 | ISO/IEC 7498 and is defined according to the model provided by ISO 9545. The role of systems management functions is described by CCITT Rec. X.701 | ISO/IEC 10040.

This Recommendation | International Standard

- establishes user requirements for this Recommendation | International Standard;
- establishes a test model that relates the service and generic definitions provided by this function to user requirements;
- defines generic object classes, attribute types, action types, notification types, packages and parameters documented in accordance with CCITT Rec. X.722 | ISO/IEC 10165-4;
- specifies compliance requirements placed on other standards that make use of these generic definitions;
- defines the service provided by the function;
- specifies the protocol that is necessary in order to provide the service;
- defines the relationship between this service and SMI operations and notifications;
- specifies the abstract syntax necessary to identify and negotiate the functional unit in protocol;
- defines relationships with other systems management functions;
- specifies conformance requirements to be met by implementations of this Recommendation | International Standard.

This Recommendation | International Standard does not

- define the nature of any implementation intended to provide the Test management function;
- specify the manner in which management is accomplished by the user of the Test management function;
- define the nature of any interactions which result in the use of the Test management function;
- specify the services necessary for the establishment, normal and abnormal release of a management association;
- specify the details of specific tests or test categories;
- specify a framework or methodology for conformance tests.

2 Normative references

The following Recommendations | International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations | Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent editions of the Recommendations | Standards listed 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.210 (1993) | ISO/IEC 10731:1993, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*.
- ISO/TR 8509:1987, *Information processing systems – Open Systems Interconnection – Service conventions*.
- CCITT Recommendation X.701 (1992) | ISO/IEC 10040:1992, *Information technology – Open Systems Interconnection – Systems management overview*.
- CCITT Recommendation X.731 (1992) | ISO/IEC 10164-2:1992, *Information technology – Open Systems Interconnection – Systems Management: State management function*.
- CCITT Recommendation X.732 (1992) | ISO/IEC 10164-3:1992, *Information technology – Open Systems Interconnection – Systems Management: Attributes for representing relationships*.
- CCITT Recommendation X.734 | ISO/IEC 10164-5:1992, *Information technology – Open Systems Interconnection – Systems Management: Event report management function*.
- CCITT Recommendation X.720 (1992) | ISO/IEC 10165-1:1992, *Information technology – Open Systems Interconnection – Structure of Management information: Management information model*.
- CCITT Recommendation X.721 (1992) | ISO/IEC 10165-2:1992, *Information technology – Open Systems Interconnection – Structure of management information: Definition of management information*.
- CCITT Recommendation X.722 (1992) | ISO/IEC 10165-4:1992, *Information technology – Open Systems Interconnection – Structure of management information: Guidelines for the definition of managed objects*.

2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.200 (1988), *Reference Model of Open Systems Interconnection for CCITT Applications*.

  ISO 7498:1984, *Information processing systems – Open Systems Interconnection – Basic Reference Model*.
- CCITT Recommendation X.208 (1988), *Specification of Abstract Syntax Notation One (ASN.1)*.

  ISO/IEC 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*.
- CCITT Recommendation X.209 (1988), *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*.

  ISO 8825:1990, *Information technology – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*.
- CCITT Recommendation X.700, *Management framework definition for Open Systems Interconnection (OSI) for CCITT applications*.

  ISO/IEC 7498-4:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management framework*.
- CCITT Recommendation X.710 (1991), *Common Management Information Service Definition for CCITT Applications*.

  ISO/IEC 9595:1991, *Information technology – Open Systems Interconnection – Common management information service definition*.
- CCITT Recommendation X.290 (1991), *OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT Applications – General Concepts*.

  ISO/IEC 9646-1:1991, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 1: General concepts*.

2.3 Additional references

- ISO/IEC 9545:1989, *Information processing systems – Open Systems Interconnection – Application Layer structure*.

3 Definitions

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

3.1 Basic reference model definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.200 | ISO 7498:

a) open system;

b) systems management.

3.2 Management framework definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.700 | ISO/IEC 7498-4:

a) managed object;

b) systems management application-entity.

3.3 CMIS definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.710 | ISO/IEC 9595:

a) attribute;

b) Common Management Information Service Element;

c) Common Management Information Service.

3.4 Remote Operations definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.219 | ISO/IEC 9072-1:

a) invoker;

b) performer.

3.5 Systems management overview definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.701 | ISO/IEC 10040:

a) agent;

b) agent role;

c) dependent conformance;

d) general conformance;

e) generic definitions;

f) managed object class;

g) managed (open) system;

h) manager;

i) manager role;

j) MIS-User;

k) notification;

l) notification type;

m) systems management application protocol;

n) systems management functional unit.

3.6 Management information model definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.720 | ISO/IEC 10165-1:

a) attribute type;

b) containment hierarchy.

3.7 Guidelines for the definition of managed objects definitions

This Recommendation | International Standard makes use of the following term defined in CCITT Rec. X.722 | ISO/IEC 10165-4:

template.

3.8 Event report management function definitions

This Recommendation | International Standard makes use of the following term defined in CCITT Rec. X.734 | ISO/IEC 10164-5:

event forwarding discriminator.

3.9 OSI conformance testing definitions

This Recommendation | International Standard makes use of the following term defined in CCITT Rec. X.290 | ISO/IEC 9646-1:

system conformance statement.

3.10 Additional definitions

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

3.10.1 associated objects (AOs): Managed objects, distinct from MORT(s), TO(s) and managed objects with TARR functionality, that are involved in a test.

3.10.2 controlled test: A test for which one or more test objects having the same value for the Test invocation identifier are created for the purpose of monitoring and control. Test results are provided by notifications emitted by the test objects or are made available as attribute values of the test objects.

3.10.3 intrusive test: A statement made with respect to a test invocation if service user disruption will or may occur as a result of the test.

3.10.4 managed object referring to test (MORT) (plural MORTs, one or more MORT(s)): A managed object which is used to refer to the functionalities that are being tested.

3.10.5 non-intrusive test: A statement made with respect to a test invocation if no service user disruption will occur as a result of the test.

3.10.6 solicited reporting: The reporting of test results by use of TO attributes.

3.10.7 test action request receiver (TARR): A term used to identify the ability of a managed object to act upon a test request. In addition to those required as a TARR, the managed object may also have other attributes, operations and notifications.

A managed object with TARR functionality may create (and subsequently delete) an instance of a particular TO class (as the result of receiving a controlled test request), representing a specific test invocation.

3.10.8 test category: One or more tests which share a common purpose and similar characteristics. A test category documents the additional behaviour and templates necessary to realize a test.

3.10.9 test conductor: A manager that issues test operations.

3.10.10 test execution: That phase of a test during which the test outcome is being determined.

3.10.11 test invocation: A specific instance of test, from the time of initiation to termination.

3.10.12 test object (TO): A managed object that exists only for a controlled test invocation and which has attributes, operations and notifications that pertain to that instance of test.

3.10.13 test object execution: That phase of testing which starts with a transition to the testing state and ends with a transition from the testing state. A TO may exhibit one or more such executions as a result of one test request.

3.10.14 test operations: management operations concerned with tests.

3.10.15 test performer: An agent which receives test operations.

3.10.16 test request: An individual request issued by a test conductor to a test performer, in order to initiate one or more test invocations.

3.10.17 test results: Information generated during test execution and made available to an open system.

3.10.18 test session: A set of test invocations.

3.10.19 uncontrolled test: A test which is not subject to monitoring or control. Test results are provided in one or more replies to the test request.

3.10.20 unsolicited reporting: The reporting of test results by use of test result notifications emitted by a TO.

4 Abbreviations

AO	Associated Object
ASE	Application Service Element
CMIS	Common Management Information Service
Conf	Confirm
DN	Distinguished Name
Id	Identifier
Ind	Indication
MAPDU	Management Application Protocol Data Unit
MORT	Managed Object Referring to Test
PDU	Protocol Data Unit
Req	Request
RDN	Relative Distinguished Name
Rsp	Response
SMAE	Systems Management Application Entity
SMAPM	Systems Management Application Protocol Machine
TARR	Test Action Request Receiver
TO	Test Object

5 Conventions

This Recommendation | International Standard defines services following the descriptive conventions defined in ITU-T Rec. X.210 | ISO/IEC 10731. In clause 9, the definition of each service includes a table that lists the parameters of its primitives. For a given primitive, the presence of each parameter is described by one of the following values:

M The parameter is mandatory.

(=) The value of the parameter is equal to the value of the parameter in the column to the left.

U The use of the parameter is a service-user option.

– The parameter is not present in the interaction described by the primitive concerned.

C The parameter is conditional. The condition(s) are defined by the text which describes the parameter.

P Subject to the constraints imposed on the parameter by CCITT Rec. X.701 | ISO/IEC 9595.

NOTE – The parameters which are marked "P" in service tables of this Recommendation | International Standard are mapped directly onto the corresponding parameters of the CMIS service primitive, without changing the semantics or syntax of the parameters. The remaining parameters are used to construct an MAPDU.

6 Requirements

This Recommendation | International Standard is intended to satisfy requirements for the remote control of tests and provides a framework for the specification of tests which exercise resources included in an open system. The requirement for the same test functionality may originate from differing higher level areas such as fault or performance management. For example a particular test may be used to generate information that may be of use either in the verification of correct functionality, diagnosis of a fault or in the generation of performance statistics.

A test is the operation and monitoring of open systems, or parts thereof, within an environment designed to elicit information regarding the functionality and/or the performance of the subject system(s).

Each test may involve creating the environment for the test, control and monitoring of the test operation, and reassertion of the normal environment. Control of a test includes the need to suspend, resume and terminate tests. Each test will require a unique identification so that, for example, data generated by the test can be tracked.

In some cases, there is a requirement to specify tests that may be suspended, resumed and terminated when pre-defined conditions are encountered.

Features of the systems environment which may require alteration for testing are:

- the connections to other open systems;
- the configuration of the subject systems;
- the workloads requested of the subject systems.

In some cases, there is a requirement for scheduling tests. The scheduling of tests must be considered in both a periodic and an aperiodic way. For such tests there is also a requirement to allow modification of the schedule. There is also, in some cases, a requirement for allowing the test to be performed at a time convenient to the system which is to perform the test.

A test may need to be specified such that it becomes active when a preset condition exists (i.e. a threshold is crossed) or when a specific event is detected.

Requirements exist to create more complex tests from simpler ones. For example, to provide the result of many subordinate tests in a single result, or to sequence tests to efficiently diagnose a fault in an entity with a large number of components.

It may be necessary to perform a number of individual tests which together fulfill a specific requirement. In such cases, it is necessary to correlate the results of each test in order to formulate an outcome. There may also be a need for global uniqueness.

This function is seen as being applicable to different test methodologies, for example, loopback tests which configure a resource in such a way that the data sent is then received; fault injection tests in which errors are deliberately introduced in order to verify that such errors are handled properly; or self-tests which simply provide a pass-fail indication.

NOTE – There is a requirement for the test conductor to request the priority of a given test. This Recommendation | International Standard does not meet this requirement.

7 Model for the test function

This clause provides terminology concerned with tests; it defines aspects of tests that are independent of the category of test and identifies aspects of tests that shall be defined for each category of test.

7.1 Environment

The execution of a test involves two or more application processes (MIS-Users). The simplest tests involve just two, a managing process that initiates the test, the *test conductor*, and an agent process that executes the test, the *test performer*. The test performer is requested to perform the test by the test conductor. (Annex B shows examples of tests involving more than two MIS-Users).

Figure 1 illustrates a single instance of a test invocation, showing only a subset of possible protocol exchanges and does not show managed objects. For simplicity, this and succeeding figures do not show all protocol responses, for example where a confirmed event report service is used.

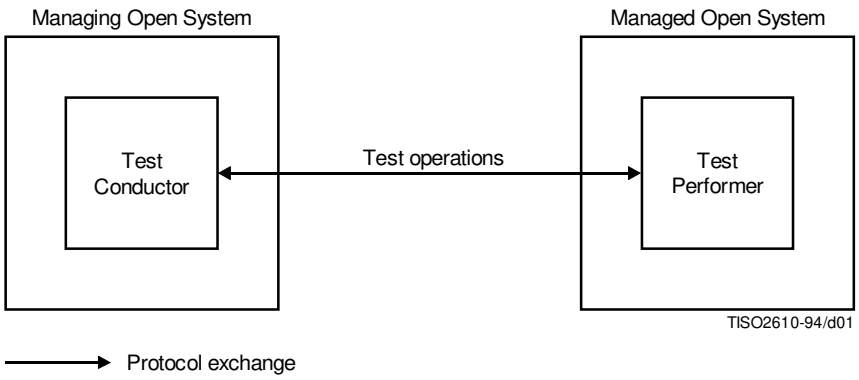

Figure 1 – Test Model

A test request is directed to a managed object, being managed by the test performer, which has functionality to receive and respond to such requests. Such functionality is called the test action request receiver functionality (TARR functionality). Managed objects which refer to functionalities that are the subjects of tests, MORTs, are identified in test requests. Each test shall involve one or more MORT. For any test, the TARR may be part of the functionality of either a MORT or another managed object. For example, the TARR functionality may be part of a managed object which exists expressly for the purpose of receiving test requests.

The execution of the test may include the use of (N)-Layer protocols, (N)-Layer Management protocols or other system specific management mechanisms, as well as Systems Management protocols. Deductions regarding the operability of the resources represented by the MORT(s) may be made if the test results are known.

A test is *uncontrolled* or *controlled*. The general test behaviour is documented by a test category.

7.1.1 Uncontrolled tests

An uncontrolled test is one which is not subject to monitoring or control. Test results are provided in one or more replies to the test request. For uncontrolled tests, the span of test invocation is from the time of the initiation request to the time at which the final response is returned. An uncontrolled test is modeled by using MORT(s) and a managed object with TARR functionality. Figure 2 depicts an example of an uncontrolled test.

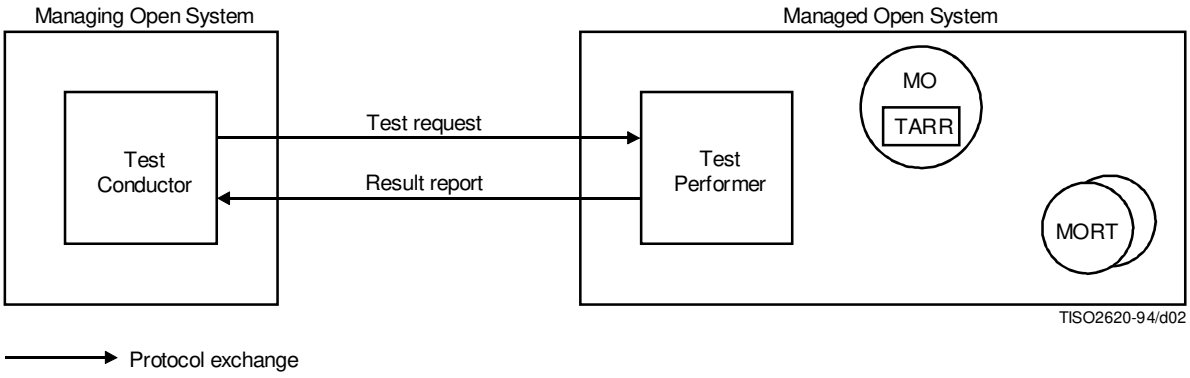

Figure 2 – Example of an uncontrolled test

7.1.2 Controlled tests

A controlled test is one for which one or more test objects, TO(s), having the same value for the test invocation identifier attribute are created for the purpose of monitoring and control. The TO(s) of a controlled test are instantiated as a consequence of a single test request. Results of the test are made available as attribute values of the TO(s) and/or

are returned via notifications issued by the TO(s). The test result notifications may be sent to any open system as event reports by instantiating an appropriate Event forwarding discriminator in the test performer as defined in CCITT Rec. X.734 | ISO/IEC 10164-5. The test results may also be stored as test result records by instantiating an appropriate log in the test performer as defined in CCITT Rec. X.735 | ISO/IEC 10164-6.

A controlled test is modeled by using TO(s), MORT(s), and a managed object with TARR functionality. Figure 3 depicts an example of a controlled test.

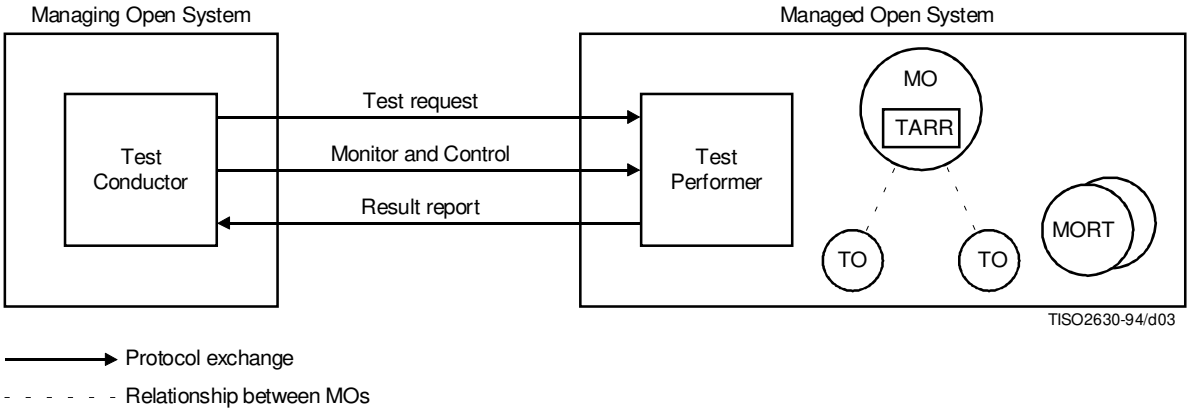

Figure 3 – Example of a controlled test

TO(s) hold information pertaining to the test. A single test request may create any number of TO(s). Relationships may exist between TO(s) and the behaviour of one TO may depend upon the behaviour of other TO(s). The name of the TO may be assigned by either the test conductor or the test performer. The test performer assigns the test invocation identifier which identifies the test invocation.

Requests to suspend, resume or terminate a controlled test are directed to the object with TARR functionality. The affected TO(s) are identified using either a test invocation identifier or a test session identifier. Requests to abort a test, if permitted, may be directed to the TO(s). The specification of the test may include the conditions under which it will be executed, suspended, resumed and terminated. These conditions may be controlled by either a scheduling mechanism, the status of a MORT or detection of a specific event.

7.2 Functional model

7.2.1 Test initiation

The request by the test conductor to the test performer to initiate the test(s) specifies the test input information and may include the following:

- the identification of MORT(s);
- test category specific input information;
- a timeout period;
- the identity of one or more associated objects;
- a test session identifier.

For controlled tests, a test conductor may also:

- indicate whether one or more tests are being requested;
- supply the classes and, optionally, names of required TO(s);
- include information about initial attribute values for the TO(s).

For uncontrolled tests, a response to a successful test request will contain test results. For controlled tests, a successful response will contain the distinguished names of the TO(s), their test invocation identifier(s), and optionally, initial attribute values. A failure response will contain a failure indication and information pertaining to the failure.

The TOs created as a result of a controlled test request may be independent tests or part of one related test. The Controlled test request type parameter is used to indicate whether the TOs are independent or related. An independent test request initiates multiple (one or more) tests each comprised of a single TO. A related test request initiates a single test comprised of more than one related TO. A single controlled test request may initiate one related test, or one or more independent tests. A controlled test with only one TO is always an independent test.

7.2.2 Test scheduling

The test conductor may provide a time window in which it would like a TO to execute. If the test performer can schedule TO execution within this time window it will return a successful confirmation and may provide an actual (or expected) start and stop time for the TO, as measured or predicted by the test performer. The test performer may report any changes in the actual start and stop times. If the actual stop time becomes greater than the requested stop time or the actual start time is prior to the requested start time the TO shall cease execution (if executing) and emit a scheduling conflict notification.

The test conductor may provide a requested time window for the TO to be in the Testing state. The start time is the earliest time at which the test conductor wishes a TO to enter the Testing state and the stop time is the latest time at which the test conductor wishes a TO to leave the Testing state. This information is in the Requested window package.

The test conductor may provide a time for the TO to enter the Initializing state. This time can be specified as an absolute time or relative to the time the TO will enter the Testing state. This information is in the Initializing time package.

The test performer may provide the actual start and stop time for a TO. This information can not be directly changed by the test conductor. The actual start time is the time at which the test performer expects a TO to start execution or the time at which a TO has started execution. The actual stop time is the time at which the test performer expects a TO to end execution or the time when the TO ended execution. This information is defined in an Actual test time package.

Figure 4 shows examples of the scheduling model. Example 1 in Figure 4 illustrates the concept of the Request window. Example 2 of Figure 4 illustrates the use of a Timeout period for a test.

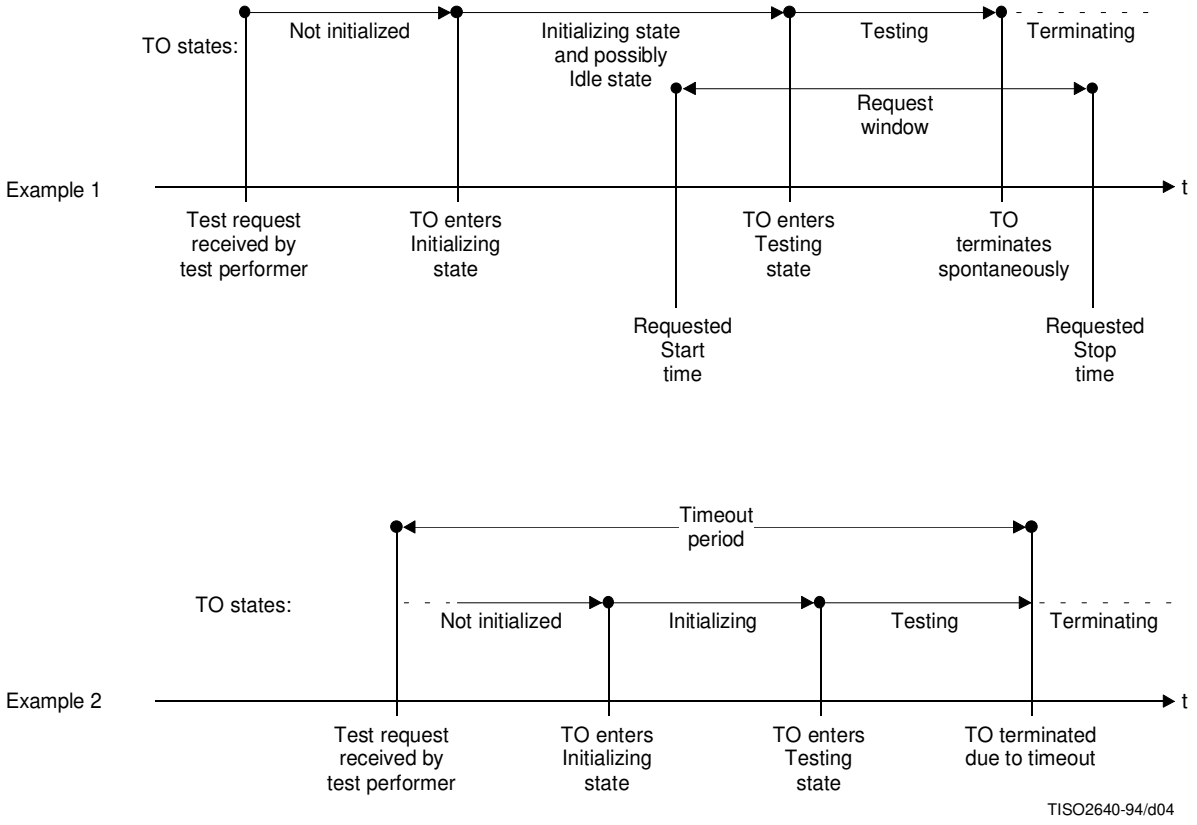

Figure 4 – Test scheduling examples

7.2.3 Reporting of test results

For uncontrolled tests, the results of the test are reported in one or more confirmations to the test request. The final confirmation indicates that the test was completed and shall contain the test outcome parameter. The test outcome parameter shall not be present in confirmations other than the final one. This parameter may take one of the following values: pass, fail, inconclusive, timed-out or premature termination. The interpretation of this parameter is contingent on the type of test that was requested. In addition, confirmations may contain information pertinent to the type of test. If the test outcome indicates fail, the confirmation primitive may contain parameters indicating the nature of the problem and proposed repair actions.

For controlled tests, the results may be requested of the TO (solicited) by the test conductor, or results may be emitted as notifications (unsolicited) from the TO.

In the case of unsolicited reporting the results are provided by one or more notifications from a TO. A result notification shall contain the test invocation identifier of the test. The test session identifier shall also be present if present in the test request. The TO shall indicate that it is sending no more reports for an execution of a test by including the test outcome parameter in a report. Result reports may contain any other information in accord with the specification of the notifications for the TO.

The test session id may be used for correlation of test results. The algorithm by which correlation is accomplished is outside the scope of this Recommendation | International Standard.

A TO may be defined to hold test results as attribute values instead of or in addition to emitting notifications. In this case, the test conductor may directly retrieve this information by reading the relevant attributes.

Irrespective of whether the test results are solicited or unsolicited, test outcome information may only be reliable on completion of a test execution.

7.2.4 Test suspension and resumption

Only a controlled test may be suspended or resumed by a test suspend/resume request directed to the object which received the test request. Successfully suspending or resuming a test implies that all the TO(s) of the test are suspended/resumed. Tests are identified in suspend/resume requests using either a test invocation id or a test session id.

If a test session id is provided in the suspend/resume request, test invocations identified by the session shall be suspended/resumed in a best effort fashion.

If a test invocation id is provided in the suspend/resume request, all TO(s) identified by the test shall be suspended/resumed, otherwise an error shall be returned.

In the response to the suspend/resume request, the list of all test invocations that have been suspended/resumed as a result of this request shall be returned.

When a test is suspended all the TO(s) for the test are set to the suspended state. The TO definitions determine if the normal operating state of the MORT(s) is restored during suspension. When a test is resumed the TO definition determines at what point in the test lifecycle the test will be resumed. The TO test states are returned in the confirmation of the resumption request.

7.2.5 Test termination

A test may terminate spontaneously or by request. Both uncontrolled and controlled tests may terminate spontaneously, either upon the completion of the test or upon abnormal conditions (including scheduling conflicts for controlled tests). Only controlled tests may be terminated by a test terminate request or aborted by deleting all TO(s) related by the same test invocation identifier.

Spontaneous termination occurs upon the fulfillment of predefined criteria or a fault situation. These predefined criteria and some specific fault situations shall be specified by the test category or TO class. For uncontrolled tests, the final test results or test failure response shall be returned.

A controlled test may be terminated by a test terminate request directed to the object which received the test request. Successfully terminating a test implies that all the TO(s) of the test are terminated. Tests are identified in termination requests using either a test invocation id or a test session id.

If a test session id is provided in the termination request, test invocations identified by the session shall be terminated in a best effort fashion.

If a test invocation id is provided in the termination request, all TO(s) identified by the test shall be terminated otherwise an error shall be returned.

In the response to the termination request, the list of all test invocations that have been terminated as a result of this request shall be returned.

When a test is terminated the TO(s) of the test will execute a termination sequence which may include issuing test result reports and performing any necessary cleanup including ending the test activity of MORT(s) and Associated object(s). If a result report is issued and the test outcome has not been concluded then the test outcome shall indicate a value of premature termination. The temporal order in which the termination sequence(s) are carried out is system specific and not defined by the test.

A controlled test may be aborted by deleting all the TO(s) with that test invocation identifier. Test objects accepting deletion requests shall not emit any further test result reports.

7.3 Information

7.3.1 Test invocation identifier

All controlled tests, whether independent or related, shall have a unique identifier called a test invocation id. The test invocation ids are returned in the reponse to the test request. All TOs belonging to the same test shall have the same test invocation id. For a related test request, the same unique test invocation id is assigned to all TOs in the related test. For an independent test request, each TO represents a single test. In this case, the test invocation id may be the DN of the TO.

The Test request controlled response returns the test invocation id for each test initiated by the request. For a related test request, only one test invocation id is returned. Also returned is the identifiers of the instantiated TOs followed optionally with their attribute list. For an independent test request, a test invocation id is returned for each instantiated TO, optionally followed by the TO Name and/or its attribute list.

The test invocation id attribute is mandatory in the TO object class.

7.3.2 Test session identifier

A test session is a set of test invocations. A test session identifier identifies a test session. If used, the test session identifier shall be assigned by the test conductor and provided in the test request. The test conductor may use test session identifiers at its discretion. If present in the test request, the same test session identifier shall also be present in any TO(s) instantiated and any result notifications emitted by the TO(s) for the test.

7.3.3 Test states

TO(s) are required to support a subset of the state attributes defined in CCITT Rec. X.722 | ISO/IEC 10164-2 in order to support queries about the current state of a test.

A TO may exhibit seven distinct states:

a) **Not initialized** – The TO is enabled but waiting to transit to the Initializing state. This transition may be controlled by the Initializing time attribute of the Initializing time package.

b) **Idle** – The TO has not yet entered the testing state as the test start criterion has not yet been met. The transition to the testing state may be controlled by the Actual start time attribute of the Actual test time package.

c) **Initializing** – The test environment is being prepared for testing. When the test environment has successfully been prepared, the TO proceeds to either the Idle state or the testing state depending on whether the test start criteria have been met. A test may be in this state for a significant length of time if, for example, it is required to wait for managed objects to be put into a specific administrative or operational state.

d) **Suspended** – A TO may be suspended by a Suspend request. In this state it is not active but the TO(s) attributes are visible, i.e. read attributes may be accessed and write attributes may be modified. For example, while a test is suspended result attributes may be accessed or scheduling parameters may be changed. The test may be resumed, as determined by the TO behaviour, from the point at which it was suspended by a Resume request.

e) **Testing** – The testing state reflects that phase of the TO during which the testing algorithms and measurements are taking place.

f) **Terminating** – The test environment is being taken down. This may include the activities that are necessary to restore test resources or MORT(s) to their pre-test condition. A test environment may take an appreciable amount of time to be taken down.

g) **Disabled** – The TO is disabled when it becomes totally inoperable due to failure conditions. This may be temporary or may lead to abnormal test termination.

Some classes of TO(s) exhibit a subset of these states, however all TOs shall at least support the testing state. For example, a test which requires no particular testing environment might not need any set up time and may not exhibit the initializing state nor the terminating state. Likewise, a test which does not support scheduling would not exhibit the idle state. Table 1 specifies the mapping between the states defined above and the attributes defined in CCITT Rec. X.722 | ISO/IEC 10164-2. The Control status attribute is necessary only if the TO supports the suspended state. The availability status attribute is necessary only if scheduling is supported.

Table 1 – Test state mapping

Test state	Operational state	Procedural status	Control status	Availability Status
Not initialized	Enabled	Not initialized	Not suspended	Off-duty
Idle	Enabled	(empty) or reporting	Not suspended	Off-duty
Initializing	Enabled	Initializing	Not suspended	(empty) or Off-duty
Testing	Enabled	(empty) or reporting	Not suspended	(empty)
Terminating	Enabled	Terminating (may also be reporting)	Not suspended	(empty)
Suspended	Enabled	(any value)	Suspended	(any value)
Disabled	Disabled	(any value)	(any value)	(any value)

Figure 5 shows an example of a state transition diagram for a TO that supports scheduling. Only a single TO execution is shown.

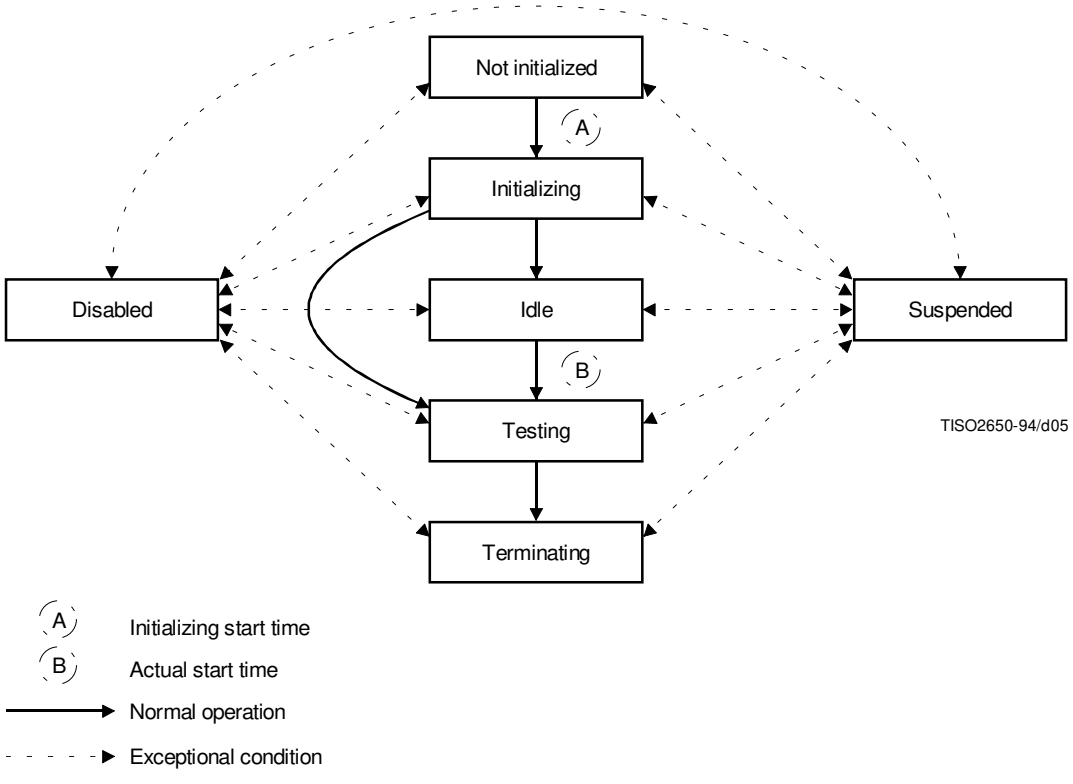

Figure 5 – Test state transition diagram

7.3.4 Test steps

A TO may have a test step package which consists of a test step attribute and a possible qualifier for a step. The test step attribute can reflect steps within each TO test state. These attributes are read only. The test steps, when used, will be defined by the TO class. The qualifier provides textual information about the test steps such as enabling conditions ('awaiting external event' or 'awaiting management input'), a procedural statement ('stop on error', 'loop on error' or 'loop at end') or status information ('fault detected'). This allows a test conductor to monitor the progress of the test and any special conditions.

7.4 Specification of test information

This form documents the management information which, taken together, specify a test. Controlled and uncontrolled tests may be in the same test category but are documented separately.

Items which do not apply to the test category being documented may be omitted.

7.4.1 Test category name

The name of this test category

7.4.2 Test category purpose

A description of the test and possible reasons for invoking it.

7.4.3 MORT requirements

A description of the MORTs to which this test is applicable. (This may include a list of managed objects).

7.4.4 Associated object requirements

A description of the Associated objects to which this test is applicable. (This may include a list of managed objects).

7.4.5 TARR requirements

Requirements upon the managed object with TARR functionality.

Subclauses 7.4.5.1 through 7.4.5.5 may be iterated for each test request service that is applicable to this test category.

7.4.5.1 Test request service type

The name of the test request action template. (Indicate the Controlled/Uncontrolled package to be used).

7.4.5.2 Specific errors

List each specific error parameter template which specifies an error response for this test and service.

7.4.5.3 Test category information parameter

The names of the parameter templates which specify the possible values for this parameter.

7.4.5.4 Additional information parameter

The names of the parameter templates which specify the possible values for this parameter (uncontrolled only).

7.4.5.5 Test control

List each of the possible control services, directed to the TARR managed object, applicable to the category (controlled only).

7.4.5.5.1 Test control specific errors

List each specific error parameter template which specifies an error response for this service.

7.4.6 TO requirements

List each TO class involved in the test (controlled only).

7.4.6.1 TO class

The name of the managed obect class template.

7.4.6.2 Additional information parameter

The names of the parameter templates which specify the values for this parameter in any test result notifications for the TO.

7.4.7 Comments

Pertinent information concerning the test. For example, intrusive behaviour, relationships among the objects involved in the test, correlation of request and response parameters, or additional semantics for test parameters.

8 Generic definitions

8.1 Generic attribute types

This clause specifies a set of generic attributes and/or parameters which are suitable for inclusion in the definition of TO classes, test result notifications or as test invocation parameters. The templates for these definitions are provided in Annex A.

8.1.1 Associated objects

This attribute or parameter, when present, identifies the managed object instance(s) which represents another resource involved in the test. Additional information related to the associated object may also be present. For example, in a connectivity test, the Associated object specifies the entity to which the resource represented by the MORT can establish a connection.

8.1.2 MORT(s)

This attribute or parameter, when present, identifies the managed object instance(s) which identifies the resource being or to be tested. This parameter is mandatory in all notifications concerning the test invocation when the parameter is also used in the test initiation request.

8.1.3 Test invocation identifier

This attribute or parameter shall be present to identify a test as specified in 7.3.1. The syntax is composed of either the identifier of the object with TARR functionality which received the test request and an integer or the DN of a TO which comprises an independent test. In the case of a related test request the former syntax shall be used. For independent tests the latter syntax may be used. It is a mandatory attribute of all TO(s).

8.1.4 Test object id

This attribute uniquely identifies a TO. The attribute for naming a TO will be a Test object id, used as an RDN component, and may be assigned either by the test conductor or the test performer.

8.1.5 Test outcome

This attribute or parameter, when present, provides a standardised view of a test result. This parameter may appear in a test results notification or it may be present as part of an uncontrolled test response. The conditions of its use are specified in 7.2.3.

This attribute may take on five values in order to categorize the outcome of a test:

- Pass;
- Fail;
- Inconclusive;
- Timed-out;
- Premature termination.

An outcome of Pass generally indicates that the test executed correctly and no problems were found. An outcome of Fail generally indicates that the test executed correctly and a problem was found. The Inconclusive outcome indicates that no Pass or Fail outcome has been determined. An outcome of Timed-out indicates that test execution ceased because of a time-out or scheduling conflict. An outcome of Premature termination indicates that test execution never started or ceased prematurely, either spontaneously or by request.

For a particular test category the definer may specify further the meaning of each outcome and the conditions under which it may occur.

8.1.6 Test session identifier

This attribute or parameter, when present, identifies a test session as specified in 7.3.2. It consists of an integer and optionally an Object Identifier or Distinguished Name. All test performers shall support all components of the Test session identifier.

8.1.7 Timeout period

This parameter or attribute, when present, defines the maximum amount of time that a test may be instantiated.

8.2 Generic packages

8.2.1 Actual test time package

This package defines two attributes for a test performer to indicate the time during which a test will run. It consists of an actual start time (absolute time, default unknown) and an actual stop time (absolute time, default is unknown).

8.2.2 Initializing Time Package

This package defines an attribute to indicate the time the TO enters the initializing state. It consists of an Initializing time which can be an absolute time or relative to the time the TO will enter the testing state.

8.2.3 Requested window package

This package defines two attributes which allow a test conductor to control the time window within which a TO shall execute. This consists of a Start time and an End time (defaults to NULL meaning forever). If the Start time is not provided in the test request then the attribute will be populated with the value of the current time. If the Start time is specified to be later than the End time the TO shall emit a scheduling conflict notification.

8.2.4 Test step package

This package defines two attributes which identifies a test step within one of the defined test states. It consists of a step identifier together with an optional set of qualifying information concerning that step.

8.3 Generic notifications

The set of generic notifications, parameters and semantics defined by this Recommendation | International Standard provide the detail for the following general parameters of the M-EVENT-REPORT service as defined by CCITT Rec. X.710 | ISO/IEC 9595:

- event type;
- event information;
- event reply.

All notifications are potential entries in a systems management log and this Recommendation | International Standard defines a managed object class for this purpose. CCITT Rec. X.721 | ISO/IEC 10165-2 defines a generic Event log record object class from which all entries are derived, the additional information being specified by the event information and event reply parameters.

8.3.1 Event type

This parameter identifies the notification as being either

- information pertaining to a test, the Test result type parameter is defined for this purpose;
- a scheduling conflict notification, the Scheduling conflict report type is defined for this purpose.

8.3.2 Event information

The following parameters defined in this Recommendation | International Standard constitute the notification specific information:

- Actual start time;
- Actual stop time;
- Associated objects;
- End time;
- MORT(s);
- Start time;
- Test outcome;
- Test invocation identifier;
- Test session identifier.

In addition the following parameters defined in CCITT Rec. X.733 | ISO/IEC 10164-4 are also utilized:

- Additional information;
- Additional text;
- Correlated notifications;
- Monitored attributes;
- Notification identifier;
- Proposed repair actions.

8.4 Generic actions

The set of generic action parameters and semantics defined by this Recommendation | International Standard provide the detail for the following general parameters of the M-ACTION service as defined by CCITT Rec. X.710 | ISO/IEC 9595:

- action type;
- action information;
- action reply.

8.4.1 Action type

This parameter identifies the action as being:

- a request to initiate a controlled test, the Test request controlled type is defined for this purpose;
- a request to initiate an uncontrolled test, the Test request uncontrolled type is defined for this purpose;
- a request to suspend or resume a test, the Test Suspend/resume type is defined for this purpose;
- a request to terminate a test, the Test terminate type is defined for this purpose.

8.4.2 Action information

The following parameters may be carried as action information.

8.4.2.1 Controlled test request type

This parameter identifies the request as pertaining to either a single test composed of related TO(s) (a related test) or multiple tests each comprising a single TO (independent tests).

8.4.2.2 Indicated tests

This parameter indicates the tests that are the subject of a control request. It consists of either a Test session identifier or a set of Test invocation identifiers.

8.4.2.3 Suspend/resume choice

This parameter identifies whether the indicated test(s) are to be suspended or resumed.

8.4.2.4 Test category information

This optional parameter indicates the test specific information associated with a test. The syntax and semantics of the information is defined by a GDMO parameter template as documented by the test category form. Test specific information may include input information needed by the test performer to execute a test, but which is not required to be held in any TO attributes.

This parameter may be present in controlled test requests and shall be present in uncontrolled test requests.

8.4.2.5 Test object list

The test object list parameter, when present, specifies the TO(s) which are to be created as a result of a controlled test request. For each TO in the list it includes the TO class, and may include a TO instance name as well as TO attribute initialization information. The ordering of TO information is significant because it may be used to correlate TO information returned in either a positive or negative response to the request. This ordering does not imply the ordering of TO creation, as this is a local matter.

8.4.2.6 To be tested MORT(s)

This attribute or parameter, when present, identifies the managed object instance(s) which represents the resource to be tested. The MORT(s) shall be identified explicitly or implicitly in the test request. The MORT may be implicitly identified by being the object with TARR functionality or MORT(s) may be provided in the TO(s) initial attribute list. When MORT(s) are explicitly specified one of two forms may be used, either a list of individual MORT(s) or by use of a scoping and filtering mechanism.

8.4.2.7 Other information

The following parameters defined in this Recommendation | International Standard are utilized:

- Associated objects;
- Timeout period;
- Test invocation identifier;
- Test session identifier.

8.4.3 Action reply

The following parameters may be carried in the action reply.

8.4.3.1 Independent test response

The Independent test response parameter, when present, indicates that all of the TOs in the test request were instantiated and the Controlled test request type parameter was specified as independent. The parameter returns information about the TOs that have been successfully instantiated, including their test invocation identifiers and, if specified by the TO behaviour, the values of TO attributes.

8.4.3.2 Related test response

The Related test response parameter, when present, indicates that all of TOs specified in the test request were instantiated. This parameter is used if the Controlled test request type parameter was specified as related. The parameter returns the test invocation identifier for the test, the names of the TOs that have been instantiated and, if specified by the TO behaviour, the values of TO attributes. The information about the TOs is in the same order as in the test request.

8.4.3.3 Test object response list

The test object response list, when present, conveys the names of the TO(s) created as a result of a controlled test request. The TO information shall be in the same sequence as given in the test request to allow the test performer to correlate the request and response.

8.4.3.4 Test request controlled response

This parameter shall be present in a positive controlled test request response. If the Controlled request type was specified as independent then the results are conveyed using the Independent test response list parameter, otherwise the Related test response parameter is used.

8.4.3.5 Test request uncontrolled response

This parameter shall be present in a positive uncontrolled test request response. It conveys information about the result of the uncontrolled test. It consists of information pertaining to the MORT(s) identified in the test request.

8.4.3.6 Test suspend/resume result

The Test suspend/resume result parameter, when present, indicates that the managed object with TARR functionality was able to suspend or resume all of the requested tests. For each such test, the test invocation identifier and TO states are specified.

8.4.3.7 Other information

The following parameters defined in this Recommendation | International Standard are utilized:

- Associated objects;
- Timeout period;
- MORT(s);
- Test invocation identifier;
- Test state.

The following parameters defined in CCITT Rec. X.733 | ISO/IEC 10164-4 are utilized:

- Additional information;
- Additional text;
- Proposed repair actions.

8.4.4 Specific errors

The following parameters specify the information conveyed when an error occurs. The test request failure parameter classifies the error.

8.4.4.1 Test request failure

This parameter is included in the failure confirmation to a test request, as described in 7.2, in order to provide information about the failure. The remainder of this clause lists the possible specific errors that shall be used.

8.4.4.2 Associated object not available error

The Associated object not available error parameter indicates that one or more of the specified Associated object(s) were not in the correct state for the test. The parameter syntax identifies the Associated object(s) that are not in the appropriate state.

8.4.4.3 Independent test invocation error

The Independent test invocation error parameter indicates that one or more of the TOs requested in the test request could not be instantiated and the Controlled test request type parameter was specified as independent. The parameter returns information about the TOs that have been successfully instantiated and the reason why TOs have failed to be instantiated. The information about the TOs is in the same order as in the test request.

8.4.4.4 Invalid test operation error

The Invalid test operation error parameter indicates that the test operation requested for the managed object class is not valid. The parameter syntax identifies the object identifier of the invalid test operation.

8.4.4.5 Mistyped test category information error

The Mistyped test category information error parameter indicates that the test category information in the test request was not valid. The parameter syntax identifies the object identifier of the invalid information.

8.4.4.6 MORT not available error

The MORT not available error parameter indicates that one or more of the MORT(s) are not in a state in which they may be tested. The parameter syntax identifies the MORT(s) that are not in an appropriate state for testing.

8.4.4.7 No such associated object error

The No such associated object error parameter indicates that one or more of the specified Associated object(s) do not exist. The parameter syntax identifies the Associated object(s) that do not exist.

8.4.4.8 No such MORT error

The No such MORT error parameter indicates that the object with TARR functionality did not recognize one or more of the MORTS(s) specified in the To be tested MORTs parameter of the test request. The parameter syntax identifies the MORT(s) that were not recognized.

8.4.4.9 No such test invocation id error

The No such test invocation id error parameter indicates that a test with the specified Test invocation identifier does not exist. The parameter syntax identifies the Test invocation identifier.

8.4.4.10 No such test session id error

The No such test session id error parameter indicates that a test with the specified Test session identifier does not exist under control of the object with TARR functionality. The parameter syntax identifies the Test session identifier.

8.4.4.11 Related TO error

The Related test object error parameter indicates that the test could not be initiated because one or more of the TOs required for the test could not be created and the Controlled test request type parameter was specified as related. The parameter indicates the TOs that could have been successfully instantiated and specifies why TOs were unable to be instantiated. The information about the TOs is in the same order as in the test request.

8.4.4.12 Test suspend/resume error

The Test suspend/resume error parameter, if present, indicates that the object with TARR functionality was unable to suspend or resume one or more of the specified tests. For those tests for which the operation failed the test invocation identifier is specified together with the TO states. For those tests for which the operation was successful the test invocation identifier and the TO states are also given.

8.4.4.13 Test terminate error

The Test terminate error, when present, indicates that the managed object with TARR functionality was unable to terminate one or more of the specified tests. This parameter specifies the test invocation identifiers of each test that was not terminated as well as the test invocation identifiers of those that were successfully terminated.

8.5 Managed objects

8.5.1 Scheduling conflict record

A Scheduling conflict record is a managed object class derived from the Event log record object class defined in CCITT Rec. X.721 | ISO/IEC 10165-2. The Scheduling conflict record managed object class represents information stored in logs as a result of receiving an event report where the event type is that defined in 9.10.

8.5.2 Test action performer

A Test action performer is a support managed object class with TARR functionality as defined in this Recommendation | International Standard. This class may be used to request either controlled or uncontrolled tests. This Recommendation | International Standard does not mandate use of this managed object class.

8.5.2.1 Supported uncontrolled tests

This attribute may be used to identify the specific uncontrolled tests which are supported by a managed object with TARR functionality. The value of the attribute is the set of object identifiers each of which is a valid value for the Test category information field in an uncontrolled test request directed to this managed object.

8.5.2.2 Supported TO classes

This attribute may be used to identify TO classes supported by a managed object with TARR functionality.

8.5.2.3 Test action performer id

This attribute may be used to identify a test action performer managed object.

8.5.3 Test object

A test object managed object class is defined in this Recommendation | International Standard so that more specific test objects may be defined by specialization.

8.5.4 Test results record

A Test results record is a managed object class derived from the Event log record object class defined in CCITT Rec. X.721 | ISO/IEC 10165-2. The Test results record object class represents information stored in logs as a result of receiving an event report where the event type is that defined in 9.9.

NOTE – The type of test results that are required to be logged also include uncontrolled test results and directly retrieving test results.

8.6 Imported generic definitions

The following parameters are also utilized. These parameters are defined in CCITT Rec. X.733 | ISO/IEC 10164-4:

- Additional text;
- Additional information.
- Correlated notifications;
- Monitored attributes;
- Notification identifier;
- Proposed repair actions.

8.7 Compliance

Managed object class definitions support the functions defined in this Recommendation | International Standard by incorporating the appropriate specification of attributes, actions and notifications through reference to the templates defined in Annex A and in CCITT Rec. X.721 | ISO/IEC 10165-2. The reference mechanism is defined in CCITT Rec. X.722 | ISO/IEC 10165-4.

9 Service definitions

9.1 Introduction

This clause defines services for requesting test initiation, reporting test results, suspending and resuming tests, for terminating tests and for reporting schedule conflicts.

9.2 Retrieving TO attributes

The PT-GET service defined in CCITT Rec. X.730 | ISO/IEC 10164-1 may be used to retrieve any of the readable attributes of a TO.

9.3 Modifying TO attributes

The PT-SET service defined in CCITT Rec. X.730 | ISO/IEC 10164-1 may be used to set any of the settable attributes of a TO.

9.4 Aborting controlled tests

The PT-DELETE service defined in CCITT Rec. X.730 | ISO/IEC 10164-1 may be used to allow one open system to abort a controlled test in another open system. Where available it is recommended that the Scope parameter include all TOs with the same Test invocation identifier and that the Filter parameter select TOs with the same Test invocation identifier.

If an abort request results in the deletion of some, but not all, TOs with the same test invocation identifier, the test becomes indeterminate. In such a case, this Recommendation | International Standard does not attach any semantics to any subsequent protocol exchanges involving the surviving TOs.

9.5 Test request controlled service

The Test request controlled service allows a manager (the test conductor) to request that another open system (the managed system) initiate a controlled test. Table 2 lists the parameters for this service.

The Test request controlled service uses the parameters defined in clause 8 in addition to the general M-ACTION service parameters defined in CCITT Rec. X.710 | ISO/IEC 9595.

NOTE – TO instance names shall not be supplied in the Test object list parameter if scoping and filtering is being used to select multiple TARRs.

The Test request controlled response parameter shall be present in a positive response, otherwise the Errors parameter shall be present. In a positive response the Independent test response list or the Related test response list parameter shall be present as determined by the value of the Controlled test request type parameter. In either case, the TO attribute list parameter shall contain the initial values of just those attributes specified to be returned by the TO behaviour.

Table 2 – Test request controlled parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Linked identifier	–	P
Mode	P	–
Base object class	P	–
Base object instance	P	–
Scope	P	–
Filter	P	–
Managed object class	–	P
Managed object instance	–	P
Access Control	P	–
Synchronization	P	–
Test request controlled type	M	C(=)
Test request controlled info	M	–
Test category information	U	–
Controlled test request type	M	–
Test session identifier	U	–
To be tested MORT(s)	U	–
Associated objects	U	–
Timeout period	U	–
Test object list	M	–
TO class	M	–
TO instance	U	–
Reference TO instance	U	–
Initial Attribute list	U	–
Current time	–	P
Test request controlled response	–	C
Independent test response	–	C
Test invocation identifier	–	M
TO instance	–	C
TO attribute list	–	C
Related test response	–	C
Test invocation identifier	–	M
Test object response list	–	M
TO instance	–	C
TO attribute	–	C
Errors	–	C

9.6 Test request uncontrolled service

The Test request uncontrolled service allows a manager (the test conductor) to request that another open system (the managed system) initiate and carry out an uncontrolled test. Table 3 lists the parameters for this service.

The Test request uncontrolled service uses the parameters defined in clause 8 in addition to the general M-ACTION service parameters defined in CCITT Rec. X.710 | ISO/IEC 9595.

The Test response data list parameter shall be present in a positive response, otherwise an Errors parameter shall be present.

Table 3 – Test request uncontrolled parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Linked identifier	–	P
Mode	P	–
Base object class	P	–
Base object instance	P	–
Scope	P	–
Filter	P	–
Managed object class	–	P
Managed object instance	–	P
Access Control	P	–
Synchronization	P	–
Test request uncontrolled type	M	C(=)
Test request uncontrolled info.	M	–
Test category information	M	–
Test session identifier	U	C(=)
Timeout period	U	–
Associated objects	U	U
To be tested MORT(s)	U	–
Current time	–	P
Test request uncontrolled response	–	C
Test outcome	–	C
MORT(s)	–	U
Proposed repair actions	–	U
Additional text	–	U
Additional information	–	U
Errors	–	C

9.7 Test suspend/resume service

The Test suspend/resume service allows a manager (the test conductor) to request that another open system (the managed system) suspend or resume a test or test session. This service is applicable only to controlled tests. Table 4 lists the parameters for this service.

The Test suspend/resume service uses the parameters defined in clause 8 in addition to the general M-ACTION service parameters defined in CCITT Rec. X.710 | ISO/IEC 9595.

The Test suspend/resume result parameter shall be present in a positive response, otherwise the Errors parameter shall be present.

In a response to a suspend request the Test state parameter shall indicate the test state of an affected TO before the test was suspended. In a response to a resume request, the test state parameter shall indicate the test state of affected TOs after resumption.

Table 4 – Test suspend/resume parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Linked identifier	–	P
Mode	P	–
Base object class	P	–
Base object instance	P	–
Scope	P	–
Filter	P	–
Managed object class	–	P
Managed object instance	–	P
Access Control	P	–
Synchronization	P	–
Test suspend/resume type	M	C(=)
Test suspend/resume info	M	–
Indicated tests	M	–
Suspend/resume choice	M	–
Current time	–	P
Test suspend/resume result	–	C
Test invocation identifier	–	M
Test objects state	–	M
TO instance	–	C
Test state	–	M
Errors	–	C

9.8 Test terminate service

The Test terminate service allows a manager (the test conductor) to request that another open system (the managed system) terminate a test or test session. This service is applicable only to controlled tests. Table 5 lists the parameters for this service.

The Test terminate service uses the parameters defined in clause 8 in addition to the general M-ACTION service parameters defined in CCITT Rec. X.710 | ISO/IEC 9595.

The Test terminate result parameter shall be present in a positive response, otherwise the Errors parameter shall be present.

Table 5 – Test terminate parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Linked identifier	–	P
Mode	P	–
Base object class	P	–
Base object instance	P	–
Scope	P	–
Filter	P	–
Managed object class	–	P
Managed object instance	–	P
Access Control	P	–
Synchronization	P	–
Test terminate type	M	C(=)
Test terminate info.	M	–
Indicated tests	M	–
Current time	–	P
Test terminate result	–	C
Test invocation identifier	–	M
Errors	–	C

9.9 Test result service

The Test result service allows one open system (the managed system) to report controlled test results. Table 6 lists the parameters for this service.

The Test result service uses the parameters defined in clause 8 in addition to the general M-EVENT-REPORT service parameters defined in CCITT Rec. X.710 | ISO/IEC 9595.

The Test invocation identifier shall be present if the TO was created as part of a related test, that is the Controlled test request type indicated a related test when the test was initiated.

The Test session identifier shall be present if it was supplied by the test conductor when the test was initiated.

The Test outcome parameter shall be present if this is the last report for this execution of the test for this TO.

Table 6 – Test result parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Mode	P	–
Managed object class	P	–
Managed object instance	P	–
Test result type	M	C(=)
Event time	P	–
Test result info		
Test invocation identifier	C	–
Test session identifier	C	–
Test outcome	C	–
MORT(s)	U	–
Associated objects	U	–
Monitored attributes	U	–
Proposed repair actions	U	–
Additional text	U	–
Additional information	U	–
Notification identifier	U	–
Correlated notifications	U	–
Current time	–	P
Event reply	–	–
Errors	–	C

9.10 Scheduling conflict service

The Scheduling conflict service allows one open system (the managed system) to report a test schedule conflict. Table 7 lists the parameters for this service.

The Scheduling conflict service uses the parameters defined in clause 8 in addition to the general M-EVENT-REPORT service parameters defined in CCITT Rec. X.710 | ISO/IEC 9595.

The Test session identifier shall be present if it was supplied by the test conductor when the test was initiated.

Table 7 – Scheduling conflict parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	P	P
Mode	P	–
Managed object class	P	–
Managed object instance	P	–
Scheduling conflict report type	M	C(=)
Event time	P	–
Scheduling conflict info		
Test invocation identifier	M	–
Test session identifier	C	–
Start time	M	–
End time	M	–
Actual start time	M	–
Actual stop time	M	–
Additional text	U	–
Additional information	U	–
Notification identifier	U	–
Correlated notifications	U	–
Current time	–	P
Event reply	–	–
Errors	–	C

10 Systems management functional units

Two functional units are defined in this Recommendation | International Standard for the management of tests:

a) uncontrolled test management functional unit;

b) controlled test management functional unit.

The uncontrolled test management functional unit requires support of the uncontrolled test request service. The controlled test management functional unit requires support of the PT-GET, PT-SET, PT-DELETE, Test request controlled, Test result, Test suspend/resume, Test terminate and Scheduling conflict services.

11 Protocol

11.1 Elements of procedure

11.1.1 Test request controlled procedure

11.1.1.1 Manager role

11.1.1.1.1 Invocation

The test request controlled procedures are initiated by the the test request controlled primitive. On receipt of a test request controlled primitive, the SMAPM shall construct an MAPDU and issue a CMIS M-ACTION request service primitive with parameters derived from the test request controlled primitive. The confirmed mode shall be used.

11.1.1.1.2 Receipt of response

On receipt of a CMIS M-ACTION confirm service primitive containing an MAPDU responding to a test request controlled operation, the SMAPM shall issue a test request controlled confirmation primitive to the test request controlled service user with parameters derived from the CMIS M-ACTION confirm service primitive, thus completing the test request controlled procedure.

NOTE – The SMAPM shall ignore all errors in the received MAPDU. The test request controlled service user may ignore such errors, or abort the association as a consequence of such errors.

11.1.1.2 Agent role

11.1.1.2.1 Receipt of request

On receipt of a CMIS M-ACTION indication service primitive containing an MAPDU requesting the test request controlled service, the SMAPM shall, if the MAPDU is well formed, issue a test request controlled indication primitive to the test request controlled service user with parameters derived from the CMIS M-ACTION indication service primitive. Otherwise, the SMAPM shall construct an appropriate MAPDU indicating the error, and shall issue a CMIS M-ACTION response service primitive with an error parameter present.

11.1.1.2.2 Response

The SMAPM shall accept a test request controlled response primitive and shall construct an MAPDU confirming the operation and issue a CMIS M-ACTION response service primitive with parameters derived from the test request controlled response primitive.

11.1.2 Test request uncontrolled procedure

11.1.2.1 Manager role

11.1.2.1.1 Invocation

The test request uncontrolled procedures are initiated by the test request uncontrolled primitive. On receipt of a test request uncontrolled primitive, the SMAPM shall construct an MAPDU and issue a CMIS M-ACTION request service primitive with parameters derived from the test request uncontrolled primitive. The confirmed mode shall be used.

11.1.2.1.2 Receipt of response

On receipt of a CMIS M-ACTION confirm service primitive containing an MAPDU responding to a test request uncontrolled operation, the SMAPM shall issue a test request uncontrolled confirmation primitive to the test request uncontrolled service user with parameters derived from the CMIS M-ACTION confirm service primitive, thus completing the test request uncontrolled procedure.

NOTE – The SMAPM shall ignore all errors in the received MAPDU. The test request uncontrolled service user may ignore such errors, or abort the association as a consequence of such errors.

11.1.2.2 Agent role

11.1.2.2.1 Receipt of request

On receipt of a CMIS M-ACTION indication service primitive containing an MAPDU requesting the test request uncontrolled service, the SMAPM shall, if the MAPDU is well formed, issue a test request uncontrolled indication primitive to the test request uncontrolled service user with parameters derived from the CMIS M-ACTION indication service primitive. Otherwise, the SMAPM shall construct an appropriate MAPDU indicating the error, and shall issue a CMIS M-ACTION response service primitive with an error parameter present.

11.1.2.2.2 Response

The SMAPM shall accept a test request uncontrolled response primitive and shall construct an MAPDU confirming the operation and issue a CMIS M-ACTION response service primitive with parameters derived from the test request uncontrolled response primitive.

11.1.3 Test suspend-resume procedure

11.1.3.1 Manager role

11.1.3.1.1 Invocation

The test suspend/resume procedures are initiated by the test suspend/resume request primitive. On receipt of a test suspend/resume request primitive, the SMAPM shall construct an MAPDU and issue a CMIS M-ACTION request service primitive with parameters derived from the test suspend/resume request primitive. The confirmed mode shall be used.

11.1.3.1.2 Receipt of response

On receipt of a CMIS M-ACTION confirm service primitive containing an MAPDU responding to a test suspend/resume operation, the SMAPM shall issue a test suspend/resume confirmation primitive to the test suspend/resume service user with parameters derived from the CMIS M-ACTION confirm service primitive, thus completing the test suspend/resume procedure.

NOTE – The SMAPM shall ignore all errors in the received MAPDU. The test suspend/resume service user may ignore such errors, or abort the association as a consequence of such errors.

11.1.3.2 Agent role

11.1.3.2.1 Receipt of request

On receipt of a CMIS M-ACTION indication service primitive containing an MAPDU requesting the test suspend/resume service, the SMAPM shall, if the MAPDU is well formed, issue a test suspend/resume indication primitive to the test suspend/resume service user with parameters derived from the CMIS M-ACTION indication service primitive. Otherwise, the SMAPM shall construct an appropriate MAPDU indicating the error, and shall issue a CMIS M-ACTION response service primitive with an error parameter present.

11.1.3.2.2 Response

The SMAPM shall accept a test suspend/resume response primitive and shall construct an MAPDU confirming the operation and issue a CMIS M-ACTION response service primitive with parameters derived from the test suspend/resume response primitive.

11.1.4 Test terminate procedure

11.1.4.1 Manager role

11.1.4.1.1 Invocation

The test terminate procedures are initiated by the test terminate request primitive. On receipt of a test terminate request primitive, the SMAPM shall construct an MAPDU and issue a CMIS M-ACTION request service primitive with parameters derived from the test terminate request primitive. The confirmed mode shall be used.

11.1.4.1.2 Receipt of response

On receipt of a CMIS M-ACTION confirm service primitive containing an MAPDU responding to a test terminate operation, the SMAPM shall issue a test terminate confirmation primitive to the test terminate service user with parameters derived from the CMIS M-ACTION confirm service primitive, thus completing the test terminate procedure.

NOTE – The SMAPM shall ignore all errors in the received MAPDU. The test terminate service user may ignore such errors, or abort the association as a consequence of such errors.

11.1.4.2 Agent role

11.1.4.2.1 Receipt of request

On receipt of a CMIS M-ACTION indication service primitive containing an MAPDU requesting the test terminate service, the SMAPM shall, if the MAPDU is well formed, issue a test terminate indication primitive to the test terminate service user with parameters derived from the CMIS M-ACTION indication service primitive. Otherwise, the SMAPM shall construct an appropriate MAPDU indicating the error, and shall issue a CMIS M-ACTION response service primitive with an error parameter present.

11.1.4.2.2 Response

The SMAPM shall accept a test terminate response primitive and shall construct an MAPDU confirming the operation and issue a CMIS M-ACTION response service primitive with parameters derived from the test terminate response primitive.

11.1.5 Test results procedure

11.1.5.1 Agent role

11.1.5.1.1 Invocation

The test results procedures are initiated by the test results request primitive. On receipt of a test results request primitive, the SMAPM shall construct an MAPDU and issue a CMIS M-EVENT-REPORT request service primitive with parameters derived from the test results request primitive. In the non-confirmed mode, the procedure in 11.1.5.1.2 does not apply.

11.1.5.1.2 Receipt of response

On receipt of a CMIS M-EVENT-REPORT confirm service primitive containing an MAPDU responding to a test results notification, the SMAPM shall issue a test results confirmation primitive to the test results service user with parameters derived from the CMIS M-EVENT-REPORT confirm service primitive, thus completing the test results procedure.

NOTE – The SMAPM shall ignore all errors in the received MAPDU. The test results service user may ignore such errors, or abort the association as a consequence of such errors.

11.1.5.2 Manager role

11.1.5.2.1 Receipt of request

On receipt of a CMIS M-EVENT-REPORT indication service primitive containing an MAPDU requesting the test results service, the SMAPM shall, if the MAPDU is well formed, issue a test results indication primitive to the test results service user with parameters derived from the CMIS M-EVENT-REPORT indication service primitive. Otherwise, the SMAPM shall, in the confirmed mode, construct an appropriate MAPDU indicating the error, and shall issue a CMIS M-EVENT-REPORT response service primitive with an error parameter present. In the non-confirmed mode, the procedure in 11.1.5.2.2 does not apply.

11.1.5.2.2 Response

In the confirmed mode, the SMAPM shall accept a test results response primitive and shall construct an MAPDU confirming the notification and issue a CMIS M-EVENT-REPORT response service primitive with parameters derived from the test results response primitive.

11.1.6 Scheduling conflict procedure

11.1.6.1 Agent role

11.1.6.1.1 Invocation

The scheduling conflict procedures are initiated by the scheduling conflict request primitive. On receipt of a scheduling conflict request primitive, the SMAPM shall construct an MAPDU and issue a CMIS M-EVENT-REPORT request service primitive with parameters derived from the scheduling conflict request primitive. In the non-confirmed mode, the procedure in 11.1.6.1.2 does not apply.

11.1.6.1.2 Receipt of response

On receipt of a CMIS M-EVENT-REPORT confirm service primitive containing an MAPDU responding to a scheduling conflict notification, the SMAPM shall issue a scheduling conflict confirmation primitive to the scheduling

conflict service user with parameters derived from the CMIS M-EVENT-REPORT confirm service primitive, thus completing the scheduling conflict procedure.

NOTE – The SMAPM shall ignore all errors in the received MAPDU. The scheduling conflict service user may ignore such errors, or abort the association as a consequence of such errors.

11.1.6.2 Manager role

11.1.6.2.1 Receipt of request

On receipt of a CMIS M-EVENT-REPORT indication service primitive containing an MAPDU requesting the scheduling conflict service, the SMAPM shall, if the MAPDU is well formed, issue a scheduling conflict indication primitive to the scheduling conflict service user with parameters derived from the CMIS M-EVENT-REPORT indication service primitive. Otherwise, the SMAPM shall, in the confirmed mode, construct an appropriate MAPDU indicating the error, and shall issue a CMIS M-EVENT-REPORT response service primitive with an error parameter present. In the non-confirmed mode, the procedure in 11.1.6.2.2 does not apply.

11.1.6.2.2 Response

In the confirmed mode, the SMAPM shall accept a scheduling conflict response primitive and shall construct an MAPDU confirming notification and issue a CMIS M-EVENT-REPORT response service primitive with parameters derived from the scheduling conflict response primitive.

11.2 Abstract syntax

11.2.1 Objects

This Recommendation | International Standard references the following support objects whose ASN.1 value notation is specified in Annex A:

a) schedulingConflictRecord;

b) testActionPerformer;

c) testObject;

d) testResultsRecord.

11.2.2 Packages

This Recommendation | International Standard references the following package definitions whose ASN.1 value notation is specified in Annex A:

a) actualTestTimePackage;

b) associatedObjectsPackage;

c) controlledTestRequestPackage;

d) initializingTimePackage;

e) mORTsPackage;

f) requestedWindowPackage;

g) supportedTOClassesPackage;

h) supportedUncontrolledTestsPackage;

i) testActionPerformerPackage;

j) testInvocationPackage;

k) testObjectPackage;

l) testOutcomePackage;

m) testResultPackage;

n) testSessionPackage;

o) testStepsPackage;

p) testSuspendResumePackage;

q) testTerminatePackage;

r) tOControlStatusPackage;

s) uncontrolledTestRequestPackage.

11.2.3 Attributes

This Recommendation | International Standard references the following specific management attributes, the abstract syntax for which is specified in Annex A:

a) actualStartTime;

b) actualStopTime;

c) associatedObjects;

d) endTime;

e) initializingTime;

f) mORTS;

g) supportedTOClasses

h) supportedUncontrolledTests;

i) testActionPerformerId;

j) testInvocationId;

k) testObjectId;

l) testOutcome;

m) testSessionId;

n) testStep;

o) testStepQualifier;

p) timeoutPeriod.

11.2.4 Notifications

This Recommendation | International Standard references the following specific notification types, the abstract syntax for which is specified in Annex A:

a) schedulingConflictNotification;

b) testResultNotification.

11.2.5 Actions

This Recommendation | International Standard references the following specific action types, the abstract syntax for which is specified in Annex A:

a) testRequestControlledAction;

b) testRequestUncontrolledAction;

c) testSuspendResumeAction;

d) testTerminateAction.

11.3 Negotiation of functional units

This Recommendation | International Standard assigns the following object identifier

{ joint-iso-ccitt ms(9) function(2) part12(12) functionalUnitPackage(1) }

as a value of the ASN.1 type FunctionalUnitPackageId defined in CCITT Rec. X.701 | ISO/IEC 10040 for negotiating the following functional units:

0 uncontrolled test management functional unit;

1 controlled test management functional unit,

where the number identifies the bit position assigned to the functional unit, and the name references the functional unit as defined in clause 10.

Within the Systems management application context, the mechanism for negotiating the test management functional unit is described by CCITT Rec. X.701 | ISO/IEC 10040.

NOTE – The requirement to negotiate functional units is specified by the application context.

12 Relationships with other functions

The test results notification defined in this Recommendation | International Standard may be logged using the services provided by CCITT Rec. X.735 | ISO/IEC 10164-6.

MORT(s), TO(s) and Associated objects involved in a test invocation may show changes of state and exhibit test-related state qualifier values as defined in CCITT Rec. X.731 | ISO/IEC 10164-2.

Initiation, termination, suspension and resumption of the TEST-RESULT service is permitted using the services defined in CCITT Rec. X.734 | ISO/IEC 10164-5, operating upon instances of the event forwarding discriminator.

TO(s) may be deleted using object management services provided in CCITT Rec. X.730 | ISO/IEC 10164-1. The object creation notification and object deletion notification services provided in CCITT Rec. X.730 | ISO/IEC 10164-1 may be used when a TO is instantiated or deleted.

Figure 6 illustrates these relationships graphically.

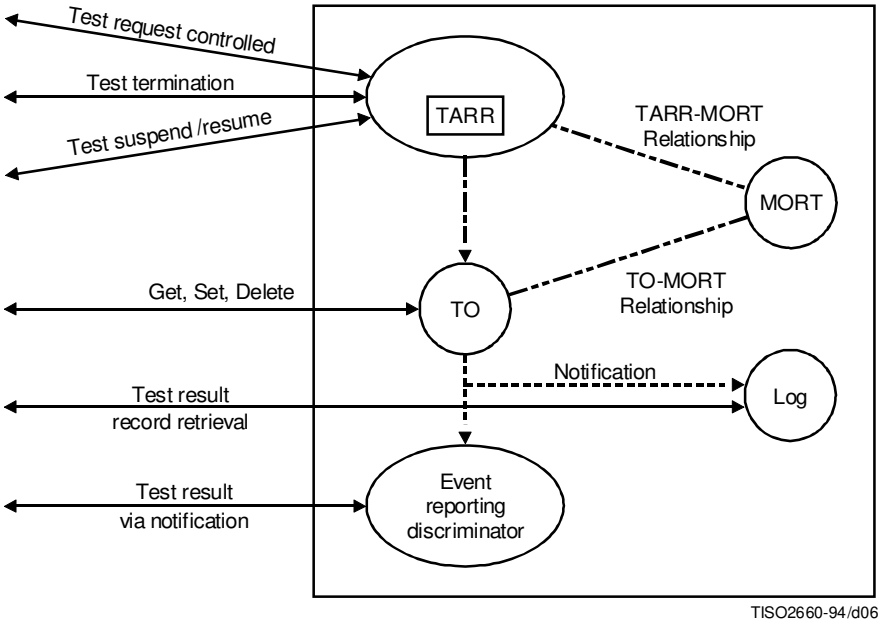

Figure 6 – Relationships with other functions

13 Conformance

There are two conformance classes: general conformance class and dependent conformance class. A system claiming to implement the elements of procedure for the systems management services defined in this Recommendation | International Standard shall comply with the requirements for either the general or the dependent conformance class as defined in the following clauses. The supplier of the implementation shall state the class to which conformance is claimed.

NOTE – The use of the two terms “general conformance class” and “dependent conformance class”, is under review. However, this standard continues to use these terms in order to be consistent with CCITT Rec. X.710 | ISO/IEC 10040 and other standards under the general title *Information technology – Open Systems Interconnection – Systems Management.* When the review has been completed, it is intended to clarify and/or correct this conformance clause together with the related clauses in those other Recommendations | International Standards.

13.1 General conformance class requirements

A system claiming general conformance shall support this function for all managed object classes that import the management information defined in this Recommendation | International Standard.

NOTE – This is applicable to any subclass of the management support object classes defined in this Recommendation | International Standard.

13.1.1 Static conformance

The system shall

a) support the role of manager or agent or both with respect to the uncontrolled and/or controlled test management functional units;

b) support the transfer syntax derived from the encoding rules specified in CCITT Rec. X.209 | ISO/IEC 8825 and named { joint-iso-ccitt asn1(1) basic encoding(1) }, for the purpose of generating and interpreting the MAPDUs defined by the abstract data types referenced in 11.2.3, 11.2.4 and 11.2.5;

c) when acting in the agent role with respect to the uncontrolled test management functional unit, support an instance of a managed object class with an Uncontrolled test request receiver package; and

d) when acting in the agent role with respect to the controlled test management functional unit, support one or more instances of the Test object class or any of its subclasses.

13.1.2 Dynamic conformance

The system shall, in the role(s) for which conformance is claimed,

a) support the elements of procedure defined in 11.1;

b) support the elements of procedure defined in CCITT REC. X.730 | ISO/IEC 10164-1 for the PT-GET, PT-SET and PT-DELETE services.

13.2 Dependent conformance class requirements

13.2.1 Static conformance

The system shall

a) support the transfer syntax derived from the encoding rules specified in CCITT Rec. X.209 | ISO/IEC 8825 and named { joint-iso-ccitt asn1(1) basic encoding(1) }, for the purpose of generating and interpreting the MAPDUs, defined by the abstract data types referenced in 11.2.3, 11.2.4 and 11.2.5, as required by a referencing specification;

b) when acting in the agent role with respect to the uncontrolled test management functional unit, support an instance of a managed object class with a Test request uncontrolled package; or

c) when acting in the agent role with respect to the controlled test management functional unit, support one or more instances of the Test object class or all subclasses.

13.2.2 Dynamic conformance

The system shall support the elements of procedure defined in, and referenced by, this Recommendation | International Standard as required by a referencing specification.

13.3 Conformance to support managed object definitions

The Test action performer objects and TOs supported by the open system shall comply with the behaviour specified in clause 8 and the syntax specified in Annex A.

Annex A

Definition of management information

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

A.1 Generic Object Classes

A.1.1 Scheduling conflict record object class

```
schedulingConflictRecord MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X.721 | ISO/IEC 10165-2:1992": eventLogRecord;
        CHARACTERIZED BY
            schedulingConflictRecordPackage PACKAGE
            BEHAVIOUR schedulingConflictRecordBehaviour BEHAVIOUR DEFINED AS
                "This managed object is used to represent information logged as a result of receiving a
                Scheduling conflict event report.";;
            ATTRIBUTES
                testInvocationId    GET,
                "Rec. X.721 | ISO/IEC 10165-2:1992": startTime      GET,
                endTime             GET,
                actualStartTime     GET,
                actualStopTime      GET;;;
            CONDITIONAL PACKAGES
                testSessionPackage PRESENT IF
                    "a Test session identifier was present in the event report.";
    REGISTERED AS { part12MObjectClass 1 };
```

A.1.2 Test action performer object class

```
testActionPerformer MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X.721 | ISO/IEC 10165-2:1992": top;
    CHARACTERIZED BY testActionPerformerPackage;
    CONDITIONAL PACKAGES
        uncontrolledTestRequestPackage PRESENT IF
            "this functionality is supported. This package and/or the controlledTestRequest package shall
            be present.",
        controlledTestRequestPackage PRESENT IF
            "this functionality is supported. This package and/or the uncontrolledTestRequest package
            shall be present.",
        testSuspendResumePackage PRESENT IF
            "the controlledTestRequestPackage is present and this functionality is supported.",
        testTerminatePackage PRESENT IF
            "the controlledTestRequestPackage is present and this functionality is supported.",
        supportedTOClassesPackage PRESENT IF
            "the controlledTestRequestPackage is present.",
        supportedUncontrolledTestsPackage PRESENT IF
            "the uncontrolledTestRequestPackage is present.";
REGISTERED AS { part12MObjectClass 2 };
```

A.1.3 TO object class

```
testObject MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X.721 | ISO/IEC 10165-2:1992": top;
    CHARACTERIZED BY testObjectPackage;
    CONDITIONAL PACKAGES
        testOutcomePackage PRESENT IF
            "test results are to be retrieved by the Test conductor.",
        testSessionPackage PRESENT IF
            "a Test session identifier was present in the test request.",
        testResultPackage PRESENT IF
            "the test object is required to emit test result notifications.",
        associatedObjectsPackage PRESENT IF
            "Associated objects were specified in the test request.",
        mORTsPackage PRESENT IF
            "MORT(s) are identified by the test object.",
```

```
            tOControlStatusPackage PRESENT IF
                "the TO may exhibit the suspended state as specified in 7.3.3",
            "Rec. X.721 | ISO/IEC 10165-2:1992": availabilityStatusPackage PRESENT IF
                "scheduling is supported as defined in 7.2.2 or if the TO may exhibit the Idle state
                as specified in 7.3.3",
            requestedWindowPackage PRESENT IF
                "scheduling is supported as defined in 7.2.2 and the test conductor may control the time at
                which the test is to be executed.",
            actualTestTimePackage PRESENT IF
                "scheduling is supported as defined in 7.2.2 and the test performer may schedule the time at
                which the test is to be executed.";
REGISTERED AS { part12MObjectClass 3 };
```

A.1.4 Test results record object class

```
testResultsRecord MANAGED OBJECT CLASS
        DERIVED FROM "Rec. X.721 | ISO/IEC 10165-2:1992": eventLogRecord;
        CHARACTERIZED BY
            testResultsRecordPackage PACKAGE
            BEHAVIOUR testResultsRecordBehaviour BEHAVIOUR DEFINED AS
                "This managed object is used to represent information logged as a result of receiving a Test
                results event report.";;;;
            CONDITIONAL PACKAGES
                testInvocationIdPackage PRESENT IF
                    "a Test invocation identifier was present in the event report.",
                testSessionPackage PRESENT IF
                    "a Test session identifier was present in the event report.",
                testOutcomePackage PRESENT IF
                    "a Test outcome parameter was present in the event report.",
                mORTsPackage PRESENT IF
                    "a MORT(s) parameter was present in the event report.",
                associatedObjectsPackage PRESENT IF
                    "an Associated objects parameter was present in the event report.",
                monitoredAttributesPackage PRESENT IF
                    "a Monitored attributes parameter was present in the event report.",
                proposedRepairActionsPackage PRESENT IF
                    "a Proposed repair actions parameter was present in the event report.";
REGISTERED AS { part12MObjectClass 4 };
```

A.2 Package definitions

```
actualTestTimePackage PACKAGE
        BEHAVIOUR actualTestTimePackageBehaviour BEHAVIOUR
            DEFINED AS "as specified in 7.2.2 and 8.2.1.";;
        ATTRIBUTES actualStartTime     DEFAULT VALUE Test-ASN1Module.defaultActualStartTime GET,
                   actualStopTime      DEFAULT VALUE Test-ASN1Module.defaultActualStopTime GET;
REGISTERED AS { part12Package 1 };

associatedObjectsPackage PACKAGE
        BEHAVIOUR associatedObjectsBehaviour BEHAVIOUR
            DEFINED AS "as specified in 8.1.1.";;
        ATTRIBUTES associatedObjects GET;
REGISTERED AS { part12Package 2 };

controlledTestRequestPackage PACKAGE
        BEHAVIOUR controlledTestRequestBehaviour BEHAVIOUR
            DEFINED AS "indicates that the managed object supports the Test request controlled action and
            behaviour as specified in clauses 7 and 9.5.";;
        ACTIONS testRequestControlledActionn o S u c h M O R T       m O R T N o t A v a i l a b l e
                                            mistypedTestCategoryInformation noSuchAssociatedObject
                                            associatedObjectNotAvailable independentTestInvocationError
                                            relatedTOError;
REGISTERED AS { part12Package 3 };
```

```
initializingTimePackage PACKAGE
        BEHAVIOUR initializingTimePackageBehaviour       BEHAVIOUR
            DEFINED AS “as specified in 7.2.2 and 8.2.2”;;
        ATTRIBUTES initializingTime     GET;
REGISTERED AS { part12Package 4 };

mORTsPackage PACKAGE
        BEHAVIOUR mORTsBehaviour BEHAVIOUR
            DEFINED AS “as defined in 8.1.2.”;;
        ATTRIBUTES mORTs GET;
REGISTERED AS { part12Package 5 };

requestedWindowPackage PACKAGE
        BEHAVIOUR  requestedWindowBehaviour BEHAVIOUR
            DEFINED AS “as specified in 7.2.2 and 8.2.3”;;
        ATTRIBUTES “Rec. X.721 | ISO/IEC 10165-2:1992”: startTime      DEFAULT VALUE DERIVED
                                                                      RULES requestedWindowBehaviour
                                                                      GET-REPLACE,
                   endTime     DEFAULT VALUE Test-ASN1Module.defaultEndTime        GET-REPLACE
                               REPLACE-WITH-DEFAULT;
REGISTERED AS { part12Package 6 };

supportedTOClassesPackage PACKAGE
        BEHAVIOUR supportedTOClassesBehaviour BEHAVIOUR
            DEFINED AS “as defined in 8.5.2.2.”;;
        ATTRIBUTES supportedTOClasses GET;
REGISTERED AS { part12Package 7 };

supportedUncontrolledTestsPackage PACKAGE
        BEHAVIOUR supportedUncontrolledTestsBehaviour BEHAVIOUR
            DEFINED AS “as defined in 8.5.2.1.”;;
        ATTRIBUTES supportedUncontrolledTests GET;
REGISTERED AS { part12Package 8 };

testActionPerformerPackage PACKAGE
        BEHAVIOUR testActionPerformerBehaviour BEHAVIOUR
            DEFINED AS “as specified in 8.5.2.”;;
        ATTRIBUTES testActionPerformerId            GET;
REGISTERED AS { part12Package 9 };

testInvocationIdPackage PACKAGE
        BEHAVIOUR testInvocationIdPackageBehaviour BEHAVIOUR
            DEFINED AS “as specified in 7.3.1.”;;
        ATTRIBUTES testInvocationId    GET;
REGISTERED AS { part12Package 10 };

testObjectPackage PACKAGE
        BEHAVIOUR testObjectBehaviour BEHAVIOUR
            DEFINED AS “as defined in clause 7. The behaviour of the Operational state and Procedural status
            attributes in this context are specified in 7.3.3.”;;
        ATTRIBUTES      testObjectId                                          GET,
                        testInvocationId                                      GET,
                        “Rec. X.721 | ISO/IEC 10165-2:1992”: operationalState GET,
                        “Rec. X.721 | ISO/IEC 10165-2:1992”: proceduralStatus GET;
REGISTERED AS { part12Package 11 };

testOutcomePackage PACKAGE
        BEHAVIOUR testOutcomeBehaviour BEHAVIOUR
            DEFINED AS “as specified in 8.1.5.”;;
        ATTRIBUTES testOutcome        GET;
REGISTERED AS { part12Package 12 };

testResultPackage PACKAGE
            NOTIFICATIONS testResultNotification;
REGISTERED AS { part12Package 13 };
```

```
testSessionPackage PACKAGE
        BEHAVIOUR testSessionBehaviour BEHAVIOUR
            DEFINED AS "as defined in 7.3.2 and 8.1.6.";;
        ATTRIBUTES testSessionId        GET;
REGISTERED AS { part12Package 14 };

testStepsPackage PACKAGE
        BEHAVIOUR testStepsBehaviour BEHAVIOUR
            DEFINED AS "as defined in 7.3.4 and 8.2.4.";;
        ATTRIBUTES testStep             GET,
                   testStepQualifier    GET;
REGISTERED AS { part12Package 15 };

testSuspendResumePackage PACKAGE
        BEHAVIOUR testSuspendResumePackageBehaviour BEHAVIOUR
            DEFINED AS "as specified in 7.2.4 and 9.7.";;
        ACTIONS
            testSuspendResumeAction invalidTestOperation    noSuchTestInvocationId
                                    noSuchTestSessionId     testSuspendResumeError;
REGISTERED AS { part12Package 16 };

testTerminatePackage PACKAGE
        BEHAVIOUR testTerminatePackageBehaviour BEHAVIOUR
            DEFINED AS "as specified in 7.2.5 and 9.8.";;
        ACTIONS
            testTerminateAction     invalidTestOperation    noSuchTestInvocationId
                                    noSuchTestSessionId     testTerminateError;
REGISTERED AS { part12Package 17 };

tOControlStatusPackage PACKAGE
        BEHAVIOUR tOControlStatusBehaviour BEHAVIOUR
            DEFINED AS "as specified in 7.3.3.";;
        ATTRIBUTES "Rec. X.721 | ISO/IEC 10165-2:1992":controlStatus GET;
REGISTERED AS { part12Package 18 };

uncontrolledTestRequestPackage PACKAGE
        BEHAVIOUR uncontrolledTestRequestBehaviour BEHAVIOUR
            DEFINED AS "indicates that the managed object supports the Test request uncontrolled action and
            behaviour as specified in clause 7 and in 9.6.";;
        ACTIONS testRequestUncontrolledAction   noSuchMORT  mORTNotAvailable
                                                mistypedTestCategoryInformation
                                                noSuchAssociatedObject
                                                associatedObjectNotAvailable;
REGISTERED AS { part12Package 19 };
```

A.3 Attribute definitions

A.3.1 Actual start time

```
actualStartTime ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.ActualStartTime;
        MATCHES FOR EQUALITY, ORDERING;
REGISTERED AS { part12AttributeId 1 };
```

A.3.2 Actual stop time

```
actualStopTime ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.ActualStopTime;
        MATCHES FOR EQUALITY, ORDERING;
REGISTERED AS { part12AttributeId 2 };
```

A.3.3 Associated objects

```
associatedObjects ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.AssociatedObjects;
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 3 };
```

A.3.4 End time

```
endTime ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.EndTime;
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 4 };
```

A.3.5 Initializing time

```
initializingTime ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.InitializingTime;
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 5 };
```

A.3.6 MORT(s)

```
mORTs ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.MORTs;
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 6 };
```

A.3.7 Supported TO classes

```
supportedTOClasses ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.SupportedTOClasses;
        MATCHES FOR SET-COMPARISON, SET-INTERSECTION;
REGISTERED AS { part12AttributeId 7 };
```

A.3.8 Supported uncontrolled Tests

```
supportedUncontrolledTests ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.SupportedUncontrolledTests;
        MATCHES FOR SET-COMPARISON, SET-INTERSECTION;
REGISTERED AS { part12AttributeId 8 };
```

A.3.9 Test action performer id

```
testActionPerformerId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.TestActionPerformerId;
        MATCHES FOR SET-COMPARISON, SET-INTERSECTION;
REGISTERED AS { part12AttributeId 9 };
```

A.3.10 Test invocation identifier

```
testInvocationId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.TestInvocationId;
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 10 };
```

A.3.11 Test object id

```
testObjectId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.TestObjectId
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 11 };
```

A.3.12 Test outcome

```
testOutcome ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.TestOutcome;
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 12 };
```

A.3.13 Test session identifier

```
testSessionId ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.TestSessionId;
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 13 };
```

A.3.14 Test step

```
testStep ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.TestStep;
        MATCHES FOR EQUALITY, ORDERING;
REGISTERED AS { part12AttributeId 14 };
```

A.3.15 Test step qualifier

```
testStepQualifier ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.TestStepQualifier;
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 15 };
```

A.3.16 Timeout period

```
timeoutPeriod ATTRIBUTE
        WITH ATTRIBUTE SYNTAX Test-ASN1Module.TimeoutPeriod;
        MATCHES FOR EQUALITY;
REGISTERED AS { part12AttributeId 16 };
```

A.4 Action definitions

```
testRequestControlledAction ACTION
        BEHAVIOUR                       testRequestControlledActionBehaviour;
        MODE CONFIRMED;
        WITH INFORMATION SYNTAX         Test-ASN1Module.TestRequestControlledInfo;
        WITH REPLY SYNTAX               Test-ASN1Module.TestRequestControlledResult;
REGISTERED AS { part12Action 1 };

testRequestControlledActionBehaviour BEHAVIOUR DEFINED AS
        "When an action request as defined by 9.5 is received a controlled test shall be invoked as defined in
        clause 7. The information associated with the action request and specific behaviour of the test shall
        depend upon the Test category. The names of the TOs created for the test shall be returned in the
        response if no such names were supplied in the request.";

testSuspendResumeAction ACTION
        BEHAVIOUR                       testSuspendResumeBehaviour;
        MODE CONFIRMED;
        WITH INFORMATION SYNTAX         Test-ASN1Module.TestSuspendResumeInfo;
        WITH REPLY SYNTAX               Test-ASN1Module.TestSuspendResumeResult;
REGISTERED AS { part12Action 2 };

testSuspendResumeBehaviour BEHAVIOUR DEFINED AS
        "When an action request as defined by 9.7 is received the test or tests specified by the request shall be
        suspended or resumed according to the Suspend/resume choice parameter. In the response to a
        suspend request the Test state parameter shall indicate the state of the TO before the test was
        suspended. In the response to a resume request the Test state parameter shall indicate the state at
        which the TO is being resumed.";

testTerminateAction ACTION
        BEHAVIOUR                       testTerminateBehaviour;
        MODE CONFIRMED;
        WITH INFORMATION SYNTAX         Test-ASN1Module.TestTerminateInfo;
        WITH REPLY SYNTAX               Test-ASN1Module.TestTerminateResult;
REGISTERED AS { part12Action 3 };

testTerminateBehaviour BEHAVIOUR DEFINED AS
        "When an action request as defined by 9.8 is received the test or tests specified by the request shall be
        terminated. Result reports pertaining to associated TO(s) may be returned before the TO(s) are deleted
        as a side effect.";
```

```
testRequestUncontrolledAction ACTION
        BEHAVIOUR                       testRequestControlledActionBehaviour;
        MODE CONFIRMED;
        WITH INFORMATION SYNTAX         Test-ASN1Module.TestRequestUncontrolledInfo;
        WITH REPLY SYNTAX               Test-ASN1Module.TestRequestUncontrolledResult;
REGISTERED AS { part12Action 4 };

testRequestUncontrolledActionBehaviour BEHAVIOUR DEFINED AS
        "When an action request as defined by 9.6 is received an uncontrolled test shall be invoked as defined in
        clause 7. The information associated with the action request and specific behaviour of the test shall
        depend upon the Test category.";
```

A.5 Notification definitions

```
schedulingConflictNotification NOTIFICATION
        BEHAVIOUR schedulingConflictBehaviour;
        WITH INFORMATION SYNTAX Test-ASN1Module.SchedulingConflictInfo
        AND ATTRIBUTE IDS
            testInvocationId        testInvocationId,
            testSessionId           testSessionId,
            startTime               "Rec. X.721 | ISO/IEC 10165-2:1992": startTime,
            endTime                 endTime,
            actualStartTime         actualStartTime,
            actualStopTime          actualStopTime,
            additionalText          "Rec. X.721 | ISO/IEC 10165-2:1992": additionalText,
            additionalInformation       "Rec. X.721 | ISO/IEC 10165-2:1992": additionalInformation;
REGISTERED AS { part12Notification 1 };

schedulingConflictBehaviour BEHAVIOUR DEFINED AS
        "This notification type is used to report scheduling conflicts as defined in 7.2.2 and 9.10.";

testResultNotification NOTIFICATION
        BEHAVIOUR testResultBehaviour;
        WITH INFORMATION SYNTAX Test-ASN1Module.TestResultInfo
        AND ATTRIBUTE IDS
            testInvocationId        testInvocationId,
            testSessionId           testSessionId,
            testOutcome             testOutcome,
            mORTs                   mORTs,
            associatedObjects       associatedObjects,
            monitoredAttributes     "Rec. X.721 | ISO/IEC 10165-2:1992": monitoredAttributes,
            proposedRepairActions   "Rec. X.721 | ISO/IEC 10165-2:1992": proposedRepairActions,
            additionalText          "Rec. X.721 | ISO/IEC 10165-2:1992": additionalText,
            additionalInformation       "Rec. X.721 | ISO/IEC 10165-2:1992": additionalInformation,
            notificationIdentifier  "Rec. X.721 | ISO/IEC 10165-2:1992": notificationIdentifier,
            correlatedNotifications "Rec. X.721 | ISO/IEC 10165-2:1992": correlatedNotfications;
REGISTERED AS { part12Notification 2 };

testResultBehaviour BEHAVIOUR DEFINED AS
        "This notification type is used to report test results as specified in 7.2.3 and 9.9.";
```

A.6 Specific Error definitions

```
associatedObjectNotAvailable PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.AssociatedObjectNotAvailable;
        BEHAVIOUR               associatedObjectNotAvailableBehaviour BEHAVIOUR
            DEFINED AS "as specified in 8.4.4.2";;
REGISTERED AS { part12Parameter 1 };

independentTestInvocationError PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.IndependentTestInvocationError;
        BEHAVIOUR               independentTestInvocationErrorBehaviour BEHAVIOUR
            DEFINED AS "as defined in 8.4.4.3";;
REGISTERED AS { part12Parameter 2 };
```

```
invalidTestOperation PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.InvalidTestOperation;
        BEHAVIOUR               invalidTestOperationBehaviour BEHAVIOUR
            DEFINED AS "as specified in 8.4.4.4";;
REGISTERED AS { part12Parameter 3 };

mistypedTestCategoryInformation PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.MistypedTestCategoryInformation;
        BEHAVIOUR               mistypedTestCategoryInformationBehaviour BEHAVIOUR
            DEFINED AS "as defined in 8.4.4.5";;
REGISTERED AS { part12Parameter 4 };

mORTNotAvailable PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.MORTNotAvailable;
        BEHAVIOUR               mORTNotAvailable BEHAVIOUR
            DEFINED AS "as specified in 8.4.4.6";;
REGISTERED AS { part12Parameter 5 };

noSuchAssociatedObject PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.NoSuchAssociatedObject;
        BEHAVIOUR               noSuchAssociatedObjectBehaviour BEHAVIOUR
            DEFINED AS "as defined in 8.4.4.7";;
REGISTERED AS { part12Parameter 6 };

noSuchMORT PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.NoSuchMORT;
        BEHAVIOUR               noSuchMORTBehaviour BEHAVIOUR
            DEFINED AS "as specified in 8.4.4.8";;
REGISTERED AS { part12Parameter 7 };

noSuchTestInvocationId PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.NoSuchTestInvocationId;
        BEHAVIOUR               noSuchTestInvocationIdBehaviour BEHAVIOUR
            DEFINED AS "as defined in 8.4.4.9";;
REGISTERED AS { part12Parameter 8 };

noSuchTestSessionId PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.NoSuchTestSessionId;
        BEHAVIOUR               noSuchTestSessionIdBehaviour BEHAVIOUR
            DEFINED AS "as specified in 8.4.4.10";;
REGISTERED AS { part12Parameter 9 };

relatedTOError PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.RelatedTOError;
        BEHAVIOUR               relatedTOErrorBehaviour BEHAVIOUR
            DEFINED AS "as defined in 8.4.4.11";;
REGISTERED AS { part12Parameter 10 };

testSuspendResumeError PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.TestSuspendResumeError;
        BEHAVIOUR               testSuspendResumeErrorBehaviour BEHAVIOUR
            DEFINED AS "as specified in 8.4.4.12";;
REGISTERED AS { part12Parameter 11 };

testTerminateError PARAMETER
        CONTEXT SPECIFIC-ERROR ;
        WITH SYNTAX             Test-ASN1Module.TestTerminateError;
        BEHAVIOUR               testTerminateErrorBehaviour BEHAVIOUR
            DEFINED AS "as defined in 8.4.4.13";;
REGISTERED AS { part12Parameter 12 };
```

A.7 Abstract Syntax definitions

```
Test-ASN1Module { joint-iso-ccitt ms(9) function(2) part12(12) asn1Module(2) 0 }
DEFINITIONS IMPLICIT TAGS
::=     BEGIN
--      EXPORTS everything
        IMPORTS
            DistinguishedName
        FROM InformationFramework
            { joint-iso-ccitt ds(5) modules(1) informationFramework(1) }

            CMISFilter, Attribute, AttributeId, ObjectInstance, Scope
        FROM CMIP-1
            { joint-iso-ccitt ms(9) cmip(1) version1(1) protocol(3) }

              StopTime, StartTime, CorrelatedNotifications, NotificationIdentifier, MonitoredAttributes,
              AdditionalInformation, AdditionalText, ProposedRepairActions, AttributeList, AvailabilityStatus,
              ControlStatus, ProceduralStatus, OperationalState
        FROM Attribute-ASN1Module
            { joint-iso-ccitt ms(9) smi(3) part2(2) asn1Module(2) 1 };

part12Action            OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part12(12) action(9) }
part12AttributeId       OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part12(12) attribute(7) }
part12MObjectClass          OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part12(12)
mObjectClass(3) }
part12Notification      OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part12(12) notification(10) }
part12Package           OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part12(12) package(4) }
part12Parameter         OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part12(12) parameter(5) }
```

A.7.1 Attribute definitions

```
ActualStartTime ::= CHOICE {
        unknown             NULL,
        actualStart         GeneralizedTime }

ActualStopTime ::= CHOICE {
        unknown             NULL,
        actualStop          GeneralizedTime }

AssociatedObjects ::= SET OF SEQUENCE {
        associatedObject        ObjectInstance,
        associatedObjectInfo    SEQUENCE {
            associatedObjectInfoId      OBJECT IDENTIFIER,
            associatedObjectInform      ANY DEFINED BY associatedObjectInfoId } OPTIONAL }

defaultActualStartTime ActualStartTime ::= unknown:NULL
-- The Actual start time remains unknown until the test performer has a real knowledge of the actual or expected time.

defaultEndTime EndTime ::= continual: NULL

EndTime ::=     CHOICE {
                specific    GeneralizedTime,
                relative    Timespec,
                continual   NULL }

InitializingTime ::= CHOICE {
                actualTime      GeneralizedTime,
                relativeTime    Timespec }
-- Timespec value forever [0] means NOW

InvocationId ::= SEQUENCE {
        tARRName    DistinguishedName,
        testId      INTEGER }

MORTs ::= SET OF ObjectInstance

SupportedTOClasses ::= SET OF OBJECT IDENTIFIER

SupportedUncontrolledTests ::= SET OF OBJECT IDENTIFIER

TestObjectId ::= INTEGER
```

```
TestStep ::= INTEGER

TestStepQualifier ::= GraphicString

TestOutcome ::= INTEGER {
        inconclusive(0),
        pass(1),
        fail(2),
        timed-out(3),
        premature-termination(4) }

Timespec ::= CHOICE {
        forever     [22]  NULL,
        hours       [23]  INTEGER,
        minutes     [24]  INTEGER,
        seconds     [25]  INTEGER,
        millisecs   [26]  INTEGER,
        microsecs   [27]  INTEGER,
        nanosecs    [28]  INTEGER }

TimeoutPeriod ::= Timespec

TestInvocationId ::= CHOICE {
        tOName       [0] ObjectInstance,
        invocationId [1] InvocationId }

TestSessionId ::= SEQUENCE {
        localId              INTEGER,
        globalRef    [1]     CHOICE { dnGlobRef DistinguishedName, oidGlobRef OBJECT IDENTIFIER }
                             OPTIONAL }
```

A.7.2 Action Types

```
TestRequestControlledInfo ::= SEQUENCE {
        controlledTestRequestType     [1]    ControlledTestRequestType,
        testCategoryInformation       [2]    TestCategoryInformation OPTIONAL,
        testSessionId                 [3]    TestSessionId OPTIONAL,
        toBeTestedMORTs                      ToBeTestedMORTs OPTIONAL,
        associatedObjects             [5]    AssociatedObjects OPTIONAL,
        timeoutPeriod                        TimeoutPeriod OPTIONAL,
        testObjectList                [7]    TestObjectList }

TestObjectList ::= SEQUENCE OF SEQUENCE {
        tOClass                              OBJECT IDENTIFIER,
        tOInstance                [1]        ObjectInstance OPTIONAL,
        referenceTOInstance       [2]        ObjectInstance OPTIONAL,
        initialAttributeList      [3]        AttributeList OPTIONAL }
        -- initialAttributeList overrides any referenceTOInstance attribute values

ControlledTestRequestType ::= ENUMERATED { independent(0), related(1) }

ToBeTestedMORTs ::= CHOICE {
        normalForm  [29] SET OF ObjectInstance,
        scopedSet   [30] SEQUENCE {
            base            ObjectInstance,
            mORTsScope      Scope DEFAULT baseObject,
            mORTsFilter     CMISFilter DEFAULT "and:{  }" } }

TestRequestControlledResponse ::= CHOICE {
        independentTestResponseList   [0]    SEQUENCE OF IndependentTestResponse,
        relatedTestResponse           [1]    RelatedTestResponse }

IndependentTestResponse ::= SEQUENCE {
        testInvocationId    TestInvocationId,
        tOName              TOName OPTIONAL,
        tOAttributeList     AttributeList OPTIONAL}
```

```
RelatedTestResponse ::= SEQUENCE {
        testInvocationId            TestInvocationId,
        testObjectResponseList      SEQUENCE OF TestObjectResponse }

TestObjectResponse ::= SEQUENCE {
        tOName              TOName,
        tOAttributeList     AttributeList OPTIONAL }

TOName ::= CHOICE {
        conductorProvidedName  NULL,
        performerProvidedName  ObjectInstance }

TestRequestUncontrolledInfo ::= SEQUENCE {
        testCategoryInformation  [1]    TestCategoryInformation,
        testSessionId                   [2]    TestSessionId OPTIONAL,
        timeoutPeriod                   TimeoutPeriod OPTIONAL,
        associatedObjects        [4]    AssociatedObjects OPTIONAL,
        toBeTestedMORTs                 ToBeTestedMORTs OPTIONAL }

TestRequestUncontrolledResult ::= SEQUENCE {
        testOutcome                     [0] TestOutcome OPTIONAL,
        mORTs                           [1] SET OF ObjectInstance OPTIONAL,
        proposedRepairActions           [2] ProposedRepairActions OPTIONAL,
        additionalText                  [3] AdditionalText OPTIONAL,
        additionalInformation           [4] AdditionalInformation OPTIONAL }

TestSuspendResumeInfo ::= SEQUENCE {
        indicatedTests              IndicatedTests,
        suspendResumeChoice         SuspendResumeChoice }

SuspendResumeChoice ::= ENUMERATED { suspend(0), resume(1) }

IndicatedTests ::= CHOICE {
        testSessionId               [0]    TestSessionId,
        testInvocationId    [1]     SET OF TestInvocationId }

TestSuspendResumeElement ::= SEQUENCE {
        testInvocationId    [0]     TestInvocationId,
        tOsStates           [1]     SET OF TOsState }

TestSuspendResumeResult ::= SET OF TestSuspendResumeElement

TOsState ::= SEQUENCE {
        tOInstance    ObjectInstance OPTIONAL,
        -- tOInstance is optional if Test Invocation Id is a tOName --,
        testState     TestState }

TestTerminateInfo ::= IndicatedTests

TestTerminateResult ::= SET OF TestInvocationId

TestState ::= SEQUENCE {
        operationalState    OperationalState,
        proceduralStatus    ProceduralStatus,
        controlStatus       ControlStatus OPTIONAL,
        availabilityStatus  AvailabilityStatus OPTIONAL }
```

A.7.3 Notification Types

```
TestResultInfo ::= SEQUENCE {
        testInvocationId            [0]    TestInvocationId OPTIONAL,
        testSessionId                      [1]    TestSessionId OPTIONAL,
        testOutcome                 [2]    TestOutcome OPTIONAL,
        mORTs                       [3]    MORTs OPTIONAL,
        associatedObjects           [4]    AssociatedObjects OPTIONAL,
        monitoredAttributes         [5]    MonitoredAttributes OPTIONAL,
        proposedRepairActions       [6]    ProposedRepairActions OPTIONAL,
        additionalText              [7]    AdditionalText OPTIONAL,
        additionalInformation       [8]    AdditionalInformation OPTIONAL,
        notificationIdentifier      [9]    NotificationIdentifier OPTIONAL,
        correlatedNotifications     [10]   CorrelatedNotifications OPTIONAL }
```

```
SchedulingConflictInfo ::= SEQUENCE {
        testInvocationId                TestInvocationId,
        testSessionId                   [1]     TestSessionId OPTIONAL,
        startTime                       StartTime,
        endTime                         EndTime,
        actualStartTime                 ActualStartTime,
        actualStopTime                  ActualStopTime,
        additionalText          [6]     AdditionalText OPTIONAL,
        additionalInformation   [7]     AdditionalInformation OPTIONAL }
```

A.7.4 Parameter Types

```
NoSuchMORT ::= SET OF ObjectInstance

MORTNotAvailable ::= SET OF ObjectInstance

MistypedTestCategoryInformation ::= OBJECT IDENTIFIER

NoSuchTestInvocationId ::= TestInvocationId

NoSuchTestSessionId ::= TestSessionId

InvalidTestOperation ::= OBJECT IDENTIFIER

NoSuchAssociatedObject ::= SET OF ObjectInstance

AssociatedObjectNotAvailable ::= SET OF ObjectInstance

IndependentTestInvocationError ::= SEQUENCE OF CHOICE {
        testInstanceCreated     [0]     IndependentTestResponse,
        tONotCreated                            TONotCreated }

RelatedTOError ::= SEQUENCE OF CHOICE {
        ableToCreateTO          NULL,
        unableToCreateTO        TONotCreated }

TONotCreated ::= CHOICE {
        reason  ENUMERATED {
                        invalidTOclass(0),
                        duplicateTOInstance(1),
                        invalidMORTClass(2),
                        invalidAssociatedObjectClass(3),
                        mORTNotAvailable(4),
                        associatedObjectNotAvailable(5) },
        inappropriateAttributeValue     AttributeList }

TestSuspendResumeError ::= SET OF CHOICE {
        testSuspendResumeSuccess    [0]     TestSuspendResumeElement,
        testSuspendResumeFailure    [1]     TestSuspendResumeElement }

TestTerminateError ::= SET OF CHOICE {
        testTerminateSuccess    [0]     TestInvocationId,
        testTerminateFailure    [1]     TestInvocationId }
END
```

Annex B

Examples

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

This informative annex contains rationale, examples, and protocol sequences which clarify the model of clause 7 and the ways in which the model may be used in order to fulfill differing requirements.

B.1 Model for compound tests

While each of the test invocations may be performed separately it is possible to use the tests in combination in order to fulfill a user requirement. This is known as a compound test.

A test conductor may request a number of test performers to initiate tests concurrently. In this way a test conductor may collect information from several different systems concerning the particular set of circumstances under investigation.

A test performer may need to involve one or more systems in performing the test requested by the test conductor. In this case, an application process in the system containing the test performer may act as a subsidiary test conductor and request (subsidiary) test performers in other systems to initiate tests. Such a test performer may need the cooperation of yet further systems to perform the requested test, in which case a test conductor in the same system as the subsidiary test performer may request other test performers to conduct tests and so on.

B.2 Model examples

The following figures and text describe ways in which the test model may be used.

B.2.1 Recursive use of the Test model

In order to fulfill a user requirement, it is possible and may be desirable, for the test performer to cause further instances of the model to be created. This concept is illustrated in Figure B.1.

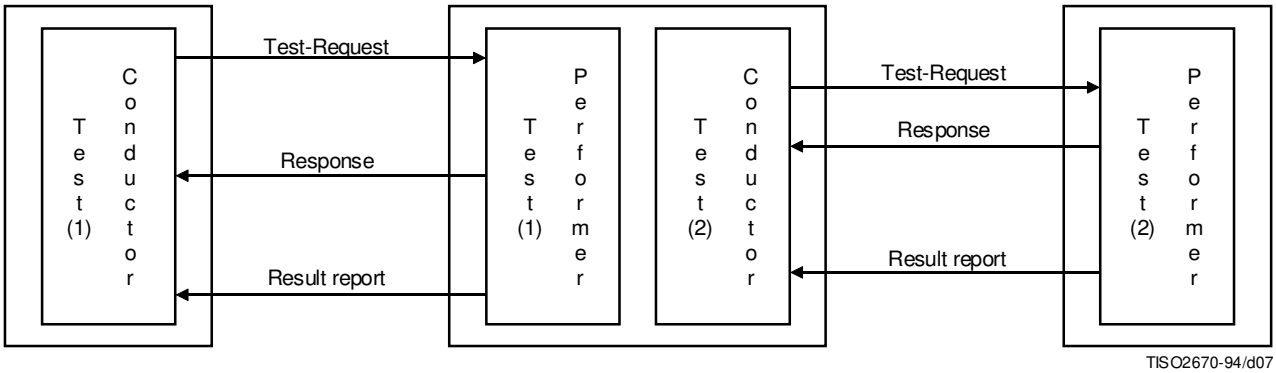

Figure B.1 – Recursive use of the Test model

B.2.2 Parallel use of the Test model

In order to fulfill a user requirement it is possible and may be desirable, for a test conductor to execute multiple, concurrent tests. This is illustrated in Figure B.2.

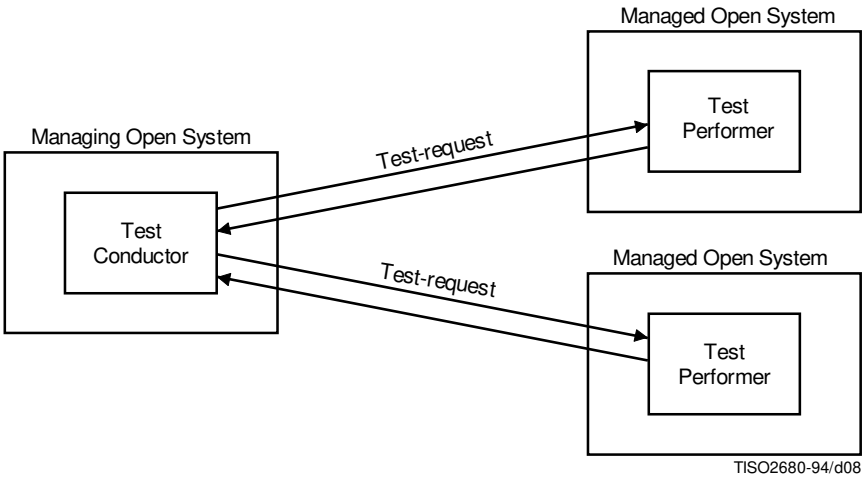

Figure B.2 – Parallel use of the Test model

B.3 Protocol sequences

The following figures and text describe the basic protocol exchanges dictated by the model considering the differing options of: uncontrolled or controlled tests, solicited or unsolicited reporting, and implicit or explicit termination.

B.3.1 Solicited reporting and explicit termination

Figure B.3 shows an example of a controlled test which uses solicited reporting and explicit termination. The PT-GET service is used to retrieve the test results during any test state. However, if the results are retrieved during the initiating state, the information may be unreliable.

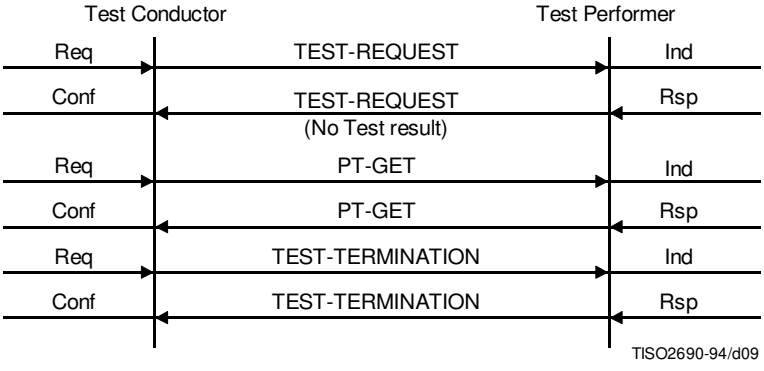

Figure B.3 – Solicited reporting and explicit termination

B.3.2 Unsolicited reporting and explicit termination

Figure B.4 shows an example of a controlled test which uses unsolicited reporting of results and explicit termination.

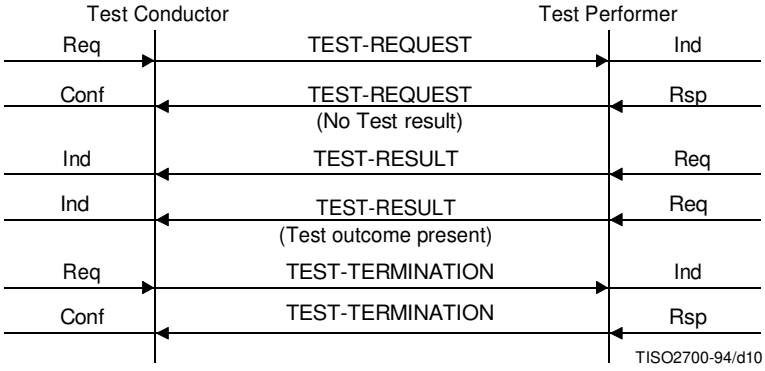

Figure B.4 – Unsolicited reporting and explicit termination

B.3.3 Solicited reporting and implicit termination

Figure B.5 shows an example of an uncontrolled test which uses solicited reporting and implicit termination.

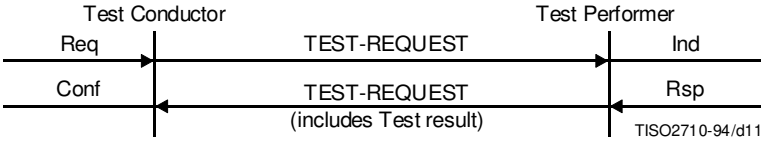

Figure B.5 – Solicited reporting and implicit termination

B.3.4 Unsolicited reporting and implicit termination

Figure B.6 shows an example of a controlled test which uses unsolicited reporting and implicit termination.

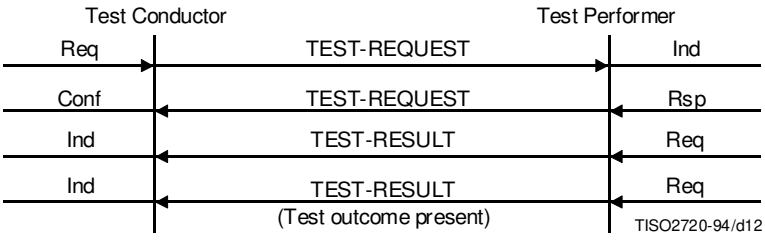

Figure B.6 – Unsolicited reporting and implicit termination