INTERNATIONAL TELECOMMUNICATION UNION

ITU-T X.737

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(11/95)

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

OSI MANAGEMENT

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – SYSTEMS MANAGEMENT: CONFIDENCE AND DIAGNOSTIC TEST CATEGORIES

ITU-T Recommendation X.737

(Previously "CCITT Recommendation")

FOREWORD

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

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

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC. The text of ITU-T Recommendation X.737 was approved on 21st of November 1995. The identical text is also published as ISO/IEC International Standard 10164-14.

NOTE

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

ITU-T X-SERIES RECOMMENDATIONS

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

(February 1994)

ORGANIZATION OF X-SERIES RECOMMENDATIONS

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

CONTENTS

		Page
1	Scope	1
2	Normative references	1
	2.1 Identical Recommendations | International Standards	1
	2.2 Paired Recommendation | International Standards equivalent in technical content	2
	2.3 Additional references	2
3	Definitions	3
	3.1 Basic reference model definitions	3
	3.2 Management framework definitions	3
	3.3 CMIS definitions	3
	3.4 Systems management overview definitions	3
	3.5 Management information model definitions	3
	3.6 Management ICS proforma definitions	4
	3.7 Test Management definitions	4
	3.8 OSI conformance testing definitions	4
	3.9 Additional definitions	4
4	Symbols and abbreviations	5
5	Conventions	5
6	Requirements	5
7	Model	6
	7.1 Connection test	7
	7.2 Connectivity test	10
	7.3 Data integrity test	12
	7.4 Loopback test	14
	7.5 Protocol integrity test	19
	7.6 Resource boundary test	22
	7.7 Resource self test	25
	7.8 Test infrastructure test	27
8	Generic definitions	29
	8.1 Generic attribute types	29
	8.2 Managed objects	31
	8.3 Imported generic definitions	32
	8.4 Compliance	32
9	Services	32
10	Functional units	32
11	Protocol	33
12	Relationships with other functions	33
13	Conformance	33
	13.1 Static conformance	33
	13.2 Dynamic conformance	33
	13.3 Management implementation conformance statement requirements	33

Page

Annex A – Definition of management information 34
Annex B – MCS proforma 54
Annex C – MICS proforma 55
Annex D – MOCS proforma 56
Annex E – MRCS proforma 57
Annex F – Summary of test categories 58
Annex G – Example TARR Package (for Connection Test) 59
Annex H – Example of use of resource boundary test category 61
Annex I – Example test process for managed communications network 63

Summary

This Recommendation | International Standard specifies a number of generally useful “test categories” such as “connection request”, loop back test and data integrity test. For each of these, a standard approach is used in terms of the components of the test category such as the purpose of the test, the resources, as defined in Recommendation X.745, test environment and content of result report, etc. The “inheritance” relationship among test object classes (representing test categories) is defined and the specifications of these as managed objects, included. This Recommendation | International Standard provides a means for other, notably TMN related groups, to use the same method for test category specification. This Recommendation | International Standard should be considered with the Test Management Function in Recommendation X.745.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – SYSTEMS MANAGEMENT: CONFIDENCE AND DIAGNOSTIC TEST CATEGORIES

1 Scope

This Recommendation | International Standard defines a Systems Management Function that may be used by an application process in a centralised or decentralised 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 ITU-T Rec. X.200 | ISO/IEC 7498-1 and is defined according to the model provided by ITU-T Rec. X.207 | ISO/IEC 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;
- specifies Confidence and diagnostic test categories that relate generic definitions provided by this Recommendation | International Standard to user requirements;
- defines managed objects, packages, attribute types, operation types, and parameters documented in accordance with CCITT Rec. X.722 | ISO/IEC 10165-4;
- specifies compliance requirements placed on other Recommendations | Standards that make use of the Generic Definitions of this Recommendation | International Standard.

This Recommendation | International Standard does not:

- define the nature of any implementation intended to provide Confidence and diagnostic tests;
- specify the manner in which management is accomplished by the user of Confidence and diagnostic tests;
- define the nature of any interactions which result in the use of Confidence and diagnostic tests;
- specify the services necessary for the establishment, normal and abnormal release of a management association.

2 Normative references

The following Recommendations and 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 and 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 edition of the Recommendations and Standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunications Standardization Bureau of the ITU-T maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.200 (1994) |.ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*
- ITU-T Recommendation X.207 (1993) | ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application Layer structure*.

- ITU-T Recommendation X.210 (1993) | ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: Conventions for the definition of OSI services*.
- CCITT Recommendation X.701 (1992) | ISO/IEC 10040:1992, *Information technology – Open Systems Interconnection – Systems management overview*.
- CCITT Recommendation X.720 (1992) | ISO/IEC 10165-1:1993, *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.*
- ITU-T Recommendation X.724 (1993) | ISO/IEC 10165-6:1994, *Information technology – Open Systems Interconnection – Structure of management information: Requirements and guidelines for implementation conformance statement proformas associated with OSI management.*
- ITU-T Recommendation X.745 (1993) | ISO/IEC 10164-12:1994, *Information technology – Open Systems Interconnection – Systems management: Test management function*.

2.2 Paired Recommendation | International Standards equivalent in technical content

- 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).*
- ITU-T Recommendation X.290 (1995), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – General concepts.*

  ISO/IEC 9646-1:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 1: General concepts.*
- ITU-T Recommendation X.291 (1995), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – Abstract test suite specification.*

  ISO/IEC 9646-2:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 2: Abstract Test Suite specification.*
- CCITT Recommendation X.700 (1992), *Management framework 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*.

2.3 Additional references

- CCITT Recommendation O.151 (1992), *Error performance measuring equipment operating at the primary rate and above.*
- CCITT Recommendation O.152 (1992), *Error performance measuring equipment for bit rates of 64 kbit/s and N x 64 kbit/s.*
- ITU-T Recommendation V.32 (1993), *A family of 2-wire, duplex modems operating at data signalling rates of up to 9 600 bit/s for use on the general switched telephone network and on leased telephone type circuits.*

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 ITU-T Rec. X.200 | ISO/IEC 7498-1:

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) management information.

3.3 CMIS definitions

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

– attribute.

3.4 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) generic definitions;

d) managed object class;

e) managed object conformance statement;

f) managed system;

g) management information conformance statement;

h) manager;

i) manager role;

j) MICS proforma;

k) MOCS proforma;

l) notification.

3.5 Management information model definitions

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

– attribute type.

3.6 Management ICS proforma definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.724 | ISO/IEC 10165-6:

a) managed relationship conformance statement;

b) management conformance summary;

c) MCS proforma;

d) MRCS proforma.

3.7 Test Management definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.745 | ISO/IEC 10164-12:

a) Associated Objects (AOs);

b) controlled test;

c) MORT;

d) intrusive test;

e) non-intrusive test;

f) uncontrolled test;

g) Test Action Request Receiver (TARR);

h) test conductor;

i) test invocation;

j) Test Object (TO);

k) test performer;

l) test request;

m) solicited reporting;

n) time-out period;

o) unsolicited reporting.

3.8 OSI conformance testing definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.290 | ISO/IEC 9646-1 and ITU-T Rec. X.291 | ISO/IEC 9646-2.

a) PICS proforma;

b) point of control and observation;

b) protocol implementation conformance statement;

c) system conformance statement.

3.9 Additional definitions

3.9.1 **establishment time**: The time taken to establish a connection between two protocol entities.

3.9.2 **explicit termination**: Test terminations on request from the test conductor.

3.9.3 **implicit reporting**: Results are reported in the form of an event report (i.e. unsolicited reporting) or an action reply.

3.9.4 **implicit termination**: Test termination on completion or upon error.

3.9.5 **loopback timeout**: The length of time that a test performer will wait for the transmitted data to be returned.

3.9.6 **loopback transmission delay time**: The time delay associated with transmitting a pattern to and back from the loopback point.

3.9.7 pattern period time: The amount of time for which each test pattern is transmitted.

3.9.8 start time: Time used for scheduling a test sometime in the future.

3.9.9 test interval time: The time delay between the completion of reception of one test pattern and the start of transmission of the next.

3.9.10 test pattern: The signals or data to be applied to a communications path.

4 Symbols and abbreviations

AO	Associated Object
ASN.1	Abstract Syntax Notation One
LLC	Logical Link Control
MCS	Management Conformance Summary
MICS	Management Information Conformance Statement
MIDS	Management Information Definition Statement
MOCS	Managed Object Conformance Statement
MORT	Managed Object Referring to Test
MRCS	Managed Relationship Conformance Statement
OSI	Open Systems Interconnection
PCO	Point of Control and Observation
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
TARR	Test Action Request Receiver
TO	Test Object
XID	Exchange Identification Frame

5 Conventions

This Recommendation | International Standard does not define any special conventions.

6 Requirements

This Recommendation | International Standard defines a basic set of confidence and diagnostic test categories that is required by the user of a communication system or network to:

- confirm the ability of a specified part of the system or network to perform correctly its allotted function (that is, the tested entity continues to perform according to its design);
- perform testing, following notification or detection of a fault, to further isolate the cause of the problem.

This requirement includes the need to:

- verify connection between two known end points;
- verify that connectivity may be established between two entities within a specified time;
- verify whether two entities can exchange data without any corruption and measure time taken for the exchange;
- verify that data can be sent and received over a communications path within a specified interval of time;
- determine if two entities can conduct proper protocol interactions;
- verify the observable behaviour of an entity at its boundaries;
- verify the ability of an entity to perform its allotted function;
- verify the ability to receive incoming test requests and generate appropriate responses.

NOTE – It is not the objective of this Recommendation | International Standard to define a complete categorization of the test types that may be offered by a particular resource or required for a particular purpose.

7 Model

A test request is directed to a managed object, being managed by the test performer, which has the functionality to receive and respond to such requests. As defined in ITU-T Rec. X.745 | ISO/IEC 10164-12, 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, either explicitly or implicitly, in test requests. Each test involves one or more MORTs. For any test, the TARR may be part of the functionality of either a MORT or another managed object.

A test is uncontrolled or controlled. An uncontrolled test is one which is not subject to monitoring or control, and for which test results are provided in one or more replies to the test request. A controlled test is one for which one or more test TOs are created for the purpose of monitoring and control. The general test behaviour is documented by a test category. Figure 1 provides examples of some configurations.

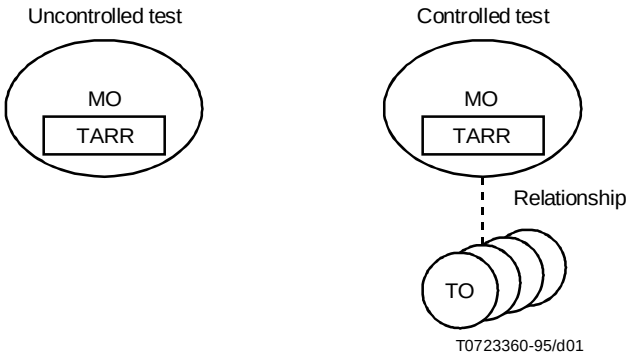

Figure 1 – Managed objects involved in all tests

This clause defines the categories of confidence and diagnostic tests that may be used to investigate the OSI environment. These categories are:

- Connection test, which may be used to verify connection between two known end points.
- Connectivity test, which may be used to verify if connectivity between two entities may be established.
- Data integrity test, which may be used to ascertain two entities can exchange data without any corruption.
- Loopback test, which may be used to verify and measure the time taken for an exchange.
- Protocol integrity test, which may be used to investigate if two communication entities can conduct proper protocol interactions.
- Resource boundary test, which aims to test the behaviour of a resource by observing and controlling the interactions between the resource and its boundaries.
- Resource self test, which may be exercised to investigate the ability of a resource to perform its allotted function.
- Test infrastructure test, which may be used to investigate the capability of a managed system to respond to a test request.

Members of each test category (e.g. the suite of tests offered by a particular managed system) share a common management interface, but will differ in areas such as:

- the function or physical entity that is tested;
- the thoroughness of the test;
- the duration of the test.

For each category, this Recommendation | International Standard defines:

- the purpose of the performed test;
- the resources (MORTs and AOs) that are involved;
- the test environment;
- the controllability (controlled or uncontrolled) of the test;

- the TO class, if controlled test applies;
- other information concerning the test;
- how the test report is delivered (e.g. implicitly or solicited) and the contents of the result report;
- the termination of the test (includes normal test completion, test failure, test abort, test terminate request, test timeout);
- (where appropriate) detailed timing information specific to the test category; and
- the management interface.

NOTE – Test termination is not equivalent to test completion (see ITU-T Rec. X.745 | ISO/IEC 10164-12). Test termination includes: test completion (pass test), test failure or error, test abort request (for controlled tests), test timeout and test terminate request (for controlled tests).

This Recommendation | International Standard also sets out the information required by a user to allow selection between the tests offered by a particular managed system. Typically, this information would be specified by a specialization of a standard test category or communicated by the supplier as a part of the technical documentation of the system, or both (e.g. Recommendation V.32).

For example, a telephone switch may offer functions such as call routing and billing, implemented by hardware and software modules including line cards, processor units, and data storage media (e.g. disc drives). A resource self test may be offered for each of these functions and each of the modules.

Through specialization, a managed object may offer two or more tests of the same internal function/module but for different objectives.

For controlled tests, it is necessary to define TO(s) for each test category. All TO classes are derived from the test object managed object class defined in ITU-T Rec. X.745 | ISO/IEC 10164-12.

These TO classes can be specialized to define more specific tests.

The following subclauses define the aforementioned test categories and their associated test object classes. The inheritance relationships among these TO classes are shown in Figure 2.

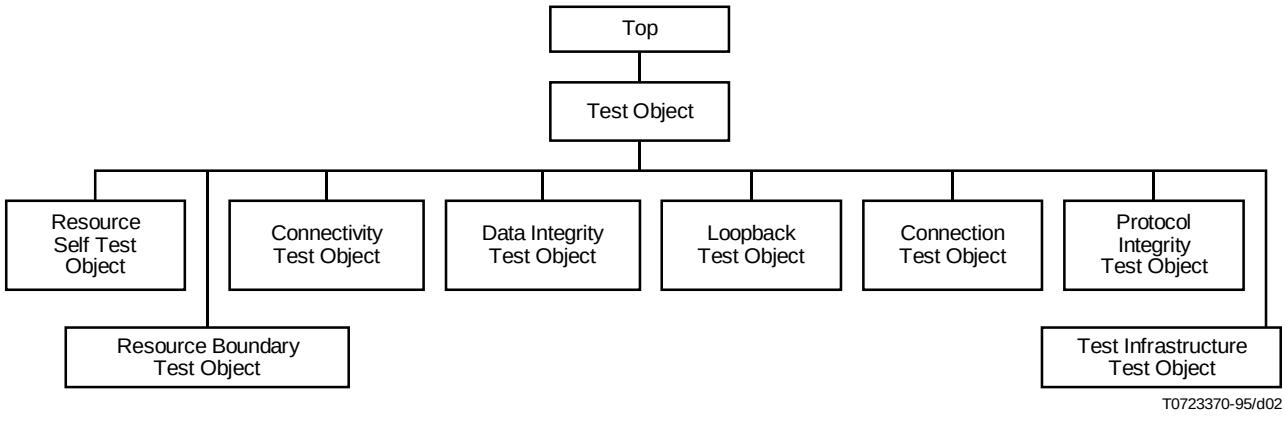

Figure 2 – Inheritance relationships among test categories

7.1 Connection test

7.1.1 Purpose of the test category

The connection test allows investigation of the ability of a communications path (real or virtual) to support a desired service or level of functionality.

The test may be implemented as one or more exercises. The exercises performed are specific to the type of connection concerned and should be sufficient to verify its performance within documented limits.

7.1.2 MORT and AO requirements

The MORT(s) represent the communications path to be tested. Two AOs are defined that reference the resources at the ends of the communications path that drive signals into and receive signals from the communications path. If identities of the MORT(s) or AOs are not specified in the test initiation, then defaults are selected by the managed system that receives the test request. Figure 3 illustrates these managed objects.

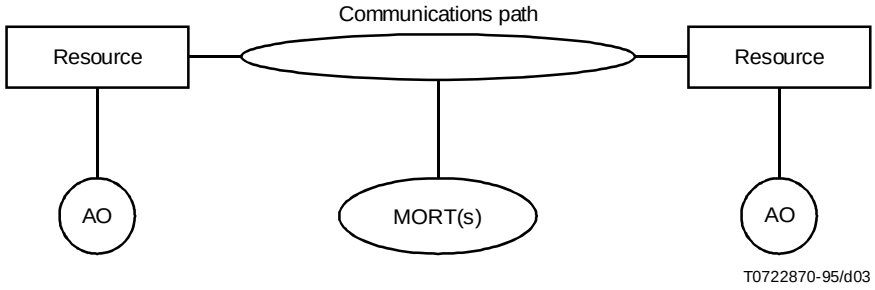

Figure 3 – Connection test test environment

In a case in which a path supports only a single direction of communication, the 'sending' AO will cause a sequence of test patterns or exercises to be applied to the path. The 'receiving' AO will provide a reference to check that the corresponding outputs are correctly received. The precise set of test patterns or exercises to be performed must be identifiable by the AOs that participate in the test (e.g. it will be pre-programmed or built into the two resources).

Note that the individual test patterns or exercises may differ significantly in their character and in the aspect of the path's functionality that is investigated.

In the event that one or more errors are detected, the test may provide diagnostic information in its final report that can be used to assist in fault progression and resolution (e.g. "suspected cable break").

In a case in which a path supports two directions of communication as determined by the test direction in a single test request, exercises of the form described above are performed separately for each direction of communication. The directions of communication will be tested in an implementation-dependent order. Figure 4 provides an example of the transmission of patterns.

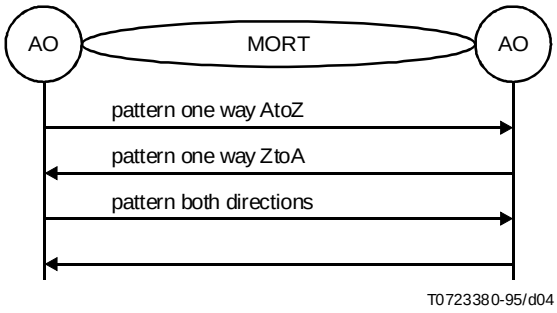

Figure 4 – Example of use of the Connection Test Category

7.1.3 Test environment

The Connection test may be intrusive or non-intrusive to both the MORT(s) and to the AOs.

If the Connection test is intrusive, it will abort under the following conditions:

- if the MORT(s) or AOs have not previously been placed in an appropriate state (e.g. reserved for test);
- if the MORT(s) or AOs cannot be placed in the appropriate state during test initialization.

7.1.4 Uncontrolled/controlled

The test may be modelled either as an uncontrolled test or as a controlled test.

7.1.5 TO requirements

This Recommendation | International Standard defines a connection test object. This TO can be specialized to define more specific TO classes.

7.1.6 Initiation specific to test category

When requesting an uncontrolled test, the time-out period defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 shall be specified.

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following optional parameters defined in ITU-T X.745 | ISO/IEC 10164-12 may also be required:

- identity of the AOs, and if appropriate, additional AO information;
- identity of the MORT;
- time-out period.

The following optional parameters may be needed:

- The test pattern to be used during the test – If this parameter is not present, the test pattern is implementation specific.
- The test direction applied to the MORT – If this parameter is not present, the MORT represents the transmitting direction at the near end of a uni-directional path.
- The test duration applied to the test – If this parameter is not present, the test duration is implementation specific.
- The test-threshold to be used to determine the test outcome – If this parameter is not present, the error threshold is implementation specific.
- The reporting interval time for intermediate test result reporting – If this parameter is not present, there is no intermediate result reporting or the intermediate result reporting time is implementation specific.

7.1.7 Reporting and termination events

The events that trigger implicit intermediate result reporting are:

- completion of the test exercise(s) performed on a single direction of communication;
- expiry of the specified reporting interval time.

The events that trigger implicit result reporting are:

- completion of the test exercise(s) performed on a single direction of communication;
- completion of the test;
- time-out;
- if the test is modelled as a controlled test, reception of a termination request.

The events that trigger test termination are:

- completion of the test;
- failure of the test to complete normally;
- expiry of the time interval during which the test is required to be completed (time-out);
- if the test is modelled as a controlled test, reception of a termination request.

7.1.8 Result report

When the test is modelled as an uncontrolled test, implicit reporting is used by means of one or more action replies. When the test is modelled as a controlled test, solicited reporting is supported by means of a get request; also, conditionally implicit reporting may be used by means of the TEST-RESULT service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 (also known as unsolicited reporting).

In addition to the parameters that are required by ITU-T Rec. X.745 | ISO/IEC 10164-12, the following parameters are required in the result report:

- the identity of the MORT(s), if not specified in the initiation;
- the identities of the AOs, if not specified in the initiation;
- if this is the final result report, the test outcome – chosen as described below.

The following optional parameters may also be returned:

- received test signal;
- error ratio detected through the test;
- test direction made on the MORT;
- test duration time;
- detailed result information for the test (implementation dependent);
- if the outcome is fail, additional diagnostic information to assist in fault progression and resolution;
- other result information specific to the test implementation.

The outcome of the test is determined as follows:

- if all exercises have been completed correctly, the outcome will be Pass;
- if any exercise detected an error, the outcome will be Fail;
- if the test duration exceeded the specified time-out period, the outcome will be Timed-out;
- if the test terminated as a result of a procedural error being encountered or as the result of receipt of a TERMINATION REQUEST, the result will be Premature termination;
- otherwise, the outcome will be Inconclusive.

7.1.9 Test termination

For an uncontrolled test, this test category uses implicit termination.

For a controlled test, both implicit and explicit termination are used. For explicit termination, the TEST-TERMINATION service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 is used.

7.2 Connectivity test

7.2.1 Purpose of the test category

The Connectivity Test is used to verify that connectivity may be established between two entities (represented by a MORT and an AO) within a specified time.

7.2.2 MORT and AO requirements

The test invocation involves two entities, represented by two managed objects (one the MORT and the other the AO), between which connectivity is to be verified. Whilst in the testing state the MORT tries to establish contact with the AO. This may be done in the case of a connection oriented protocol by establishing a connection, or in the case of a connectionless protocol by an exchange of data unit, such as the LLC XID exchange. The time taken between the issuing of the request by the MORT until the receipt of the corresponding confirmation from the AO is measured and called establishment time. A time-out period may be associated with the test. The time-out period may be specified indirectly as the value of a timer attribute provided by the MORT. Figures 5 and 6 provide examples of use of the Connectivitiy test.

7.2.3 Test environment

The connectivity test is intrusive to both the MORT and to the AO. The Connectivity test will abort under the following conditions:

- if the MORT or AO have not previously been placed in an appropriate state (e.g. reserved for test);
- if the MORT or AO cannot be placed in the appropriate state during test initiation.

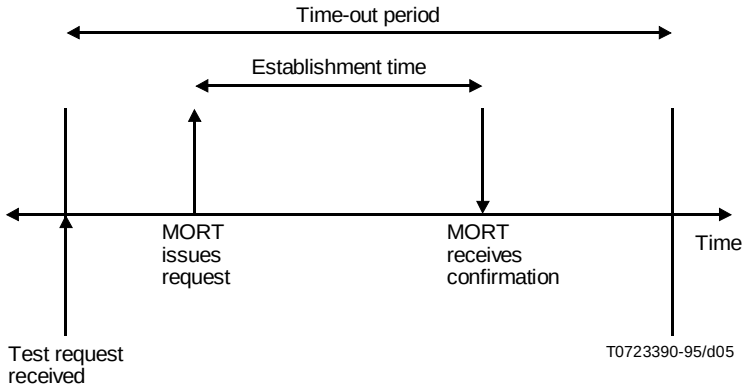

Figure 5 – Connectivity test example 1

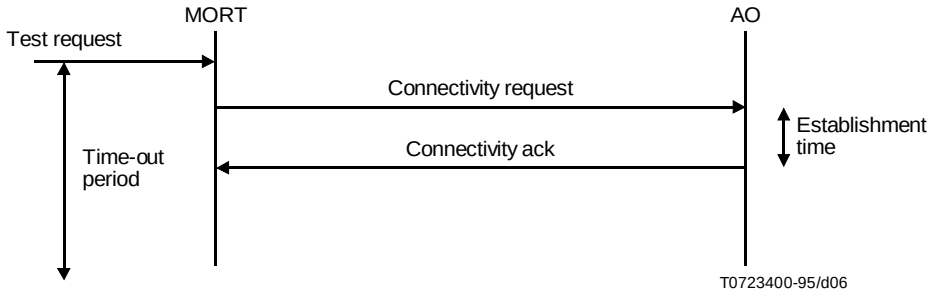

Figure 6 – Connectivity test example 2

7.2.4 Uncontrolled/controlled

A connectivity test may be modelled either as an uncontrolled test or as a controlled test.

7.2.5 TO requirements

This Recommendation | International Standard defines a connectivity test object. This TO can be specialized to define more specific TO classes.

7.2.6 Initiation specific to test category

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following optional parameters defined in ITU-T X.745 | ISO/IEC 10164-12 are also required:

- identity of the AO, and if appropriate, additional AO information;
- identity of the MORT if not implicitly known by the TARR;
- time-out period, if not implicitly known by the MORT.

7.2.7 Reporting and termination events

The events that trigger implicit result reporting and implicit test termination are:

- receipt of confirm from AO prior to expiry of time-out period;
- time-out period expiration.

7.2.8 Result report

When the test is modelled as an uncontrolled test, implicit reporting is used by means of one or more action replies. When the test is modelled as a controlled test, solicited reporting is supported by means of a get request; also, conditionally implicit reporting may be used by means of the TEST-RESULT service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 (also known as unsolicited reporting).

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following parameters are required:

- the identity of the AO, if it was not explicitly specified in the test request;
- identity of the MORT, if it was not explicitly specified in the test request.

The following optional parameters may also be returned:

- if this is the final report, the test outcome chosen as appropriate from ITU-T X.745 | ISO/IEC 10164-12 shall be returned;
- if outcome is Pass, then Establishment time in time-units;
- if outcome is Fail, then the time taken from the issuing of the request by the MORT until the receipt of the corresponding confirmation from the AO, or the termination of the request;
- any additional information specific to the test.

The value of the test outcome to be returned in the final test result report is determined according to the following:

- if positive confirmation is received before the time-out period, the test outcome is Pass;
- if a negative confirmation is received, the outcome is Fail;
- if the confirmation was not received before the expiration of the pre-specified time-out period, the test outcome is Timed-out;
- if the test terminates as a result of a procedural failure, or as a result of receipt of a TERMINATE REQUEST, the test outcome is Premature termination;
- otherwise, the test outcome is Inconclusive.

7.2.9 Test termination

For an uncontrolled test, this test category uses implicit termination.

For a controlled test, both implicit and explicit termination are used. For explicit termination, the TEST-TERMINATION service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 is used.

7.3 Data integrity test

7.3.1 Purpose of the test category

The test ascertains whether two entities can exchange data without any corruption, and measures the time taken for the data exchange.

7.3.2 MORT and AO requirements

While in the testing state the MORT transmits data to an AO. The AO, upon detecting the reception of the data, causes a copy of the data to be reflected back to the MORT. The transfer mechanism used to carry the data may be either connection oriented or connectionless. If it is connection oriented, the MORT must ensure that a connection exists prior to transmitting the data. During the connection establishment, the TO is placed in the initiating test state. While the connection is closing down, the TO is put in the terminating state. Figures 7 and 8 provide examples of use of the data integrity test.

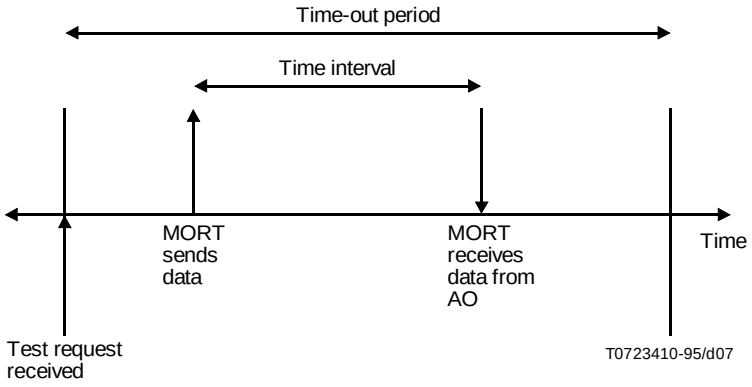

Figure 7 – Data integrity test example 1

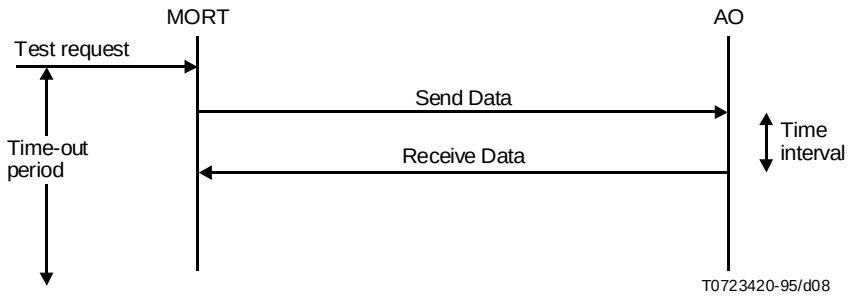

Figure 8 – Data integrity test example 2

7.3.3 Test environment

The Data integrity test is intrusive to both the MORT(s) and to the AO(s). The Data integrity test will abort under the following conditions:

- if the MORT or AOs have not previously been placed in an appropriate state (e.g. reserved for test);
- if the MORTs or AOs cannot be placed in the appropriate state during test initialization.

7.3.4 Uncontrolled/controlled

A data integrity test may be modelled either as an uncontrolled test or as a controlled test.

7.3.5 TO requirements

This Recommendation | International Standard defines a data integrity test object. This TO can be specialized to define more specific TO classes.

7.3.6 Initiation specific to test category

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following optional parameters defined in ITU-T X.745 | ISO/IEC 10164-12 are also required:

- identity of the AO, and if appropriate, additional AO information;
- identity of the MORT if not implicitly known by the TARR;
- time-out period, if not implicitly known by the MORT.

The following optional parameter may be specified:

- The data units for the test. If not provided, the test units are object specific.

7.3.7 Reporting and termination events

The events that trigger implicit result reporting and implicit test termination are:

- receipt of confirmation from AOs prior to expiry of time-out period;
- time-out period expiration.

7.3.8 Result report

When the test is modelled as an uncontrolled test, implicit reporting is used by means of one or more action replies. When the test is modelled as a controlled test, solicited reporting is supported by means of a get request; also, conditionally implicit reporting may be used by means of the TEST-RESULT service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 (also known as unsolicited reporting).

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following parameters are required:

- the identity of the AO, if it is not explicitly specified in the test request;
- identity of the MORT, if it is not explicitly specified in the test request.

The following optional parameters may also be returned:

- If this is the final report, the test outcome chosen as appropriate from ITU-T X.745 | ISO/IEC 10164-12 shall be returned;
- If the test outcome is Pass, the actual time interval for the data exchange is returned;
- If outcome is Fail, then the time taken from the issuing of the request by the MORT until the receipt of the corresponding confirmation from the AO, or the termination of the request;
- If the test failed due to the received data, then the data received. Furthermore, if the data units used in the test were not provided in the test request parameter, the data units shall also be returned;
- Any additional information specific to the test.

The test outcome is returned in the final test result report. The value of the test outcome is determined according to the following:

- if all data units were successfully transmitted and received, the test outcome is Pass;
- if errors were detected in received data, the test outcome is Fail;
- if the test duration exceeds the pre-specified time-out period, the test outcome is TIMEOUT;
- if the test terminates as a result of a procedural failure, or as a result of receipt of a TERMINATE REQUEST, the test outcome is Premature termination;
- otherwise, the test outcome is Inconclusive.

7.3.9 Test termination

For an uncontrolled test, this test category uses implicit termination.

For a controlled test, both implicit and explicit termination are used. For explicit termination, the TEST-TERMINATION service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 is used.

7.4 Loopback test

7.4.1 Purpose of the test category

The Loopback test is used to verify that data may be sent and received over a communications path within a specified loopback time-out period with an acceptable error ratio. A loopback may be implemented in a variety of ways, for example by physical loopback or by echoing data received. Loopbacks may be analogue or digital. Other examples include non-transparent physical loopback, transparent physical loopback, payload physical loopback, echo data back, etc. A loopback test may be performed over a communication path that is connection oriented or connectionless.

7.4.2 MORT and AO requirements

7.4.2.1 Loopback configuration

To allow the invocation of loopback tests, the configuration in which the MORTs and AOs are to be placed must be specified. The MORTs represent the resources that are being tested by the loopback test. However, these MORTs need not necessarily cover the entire communications path as there may be other unidentified managed objects to be tested along the communications path. The AOs references the resources providing the loopback points. This Recommendation | International Standard and the use of these AOs are implementation specific. If AOs are not specified, a manual loopback configuration is assumed.

In some cases, it is desirable to be able to specify that a loopback occurs within a MORT. This may occur when a resource is modelled as a single managed object but is actually complex enough to allow loopbacks to occur in several places within the MORT (e.g. a “path” may be modelled as a single MO but may actually consist of multiple sections each of which could be looped). To allow this to be modelled, AOs are allowed to be present at locations that will test only a part of the MORT.

Figures 9 and 10 show two possible configurations of MORTs and AOs.

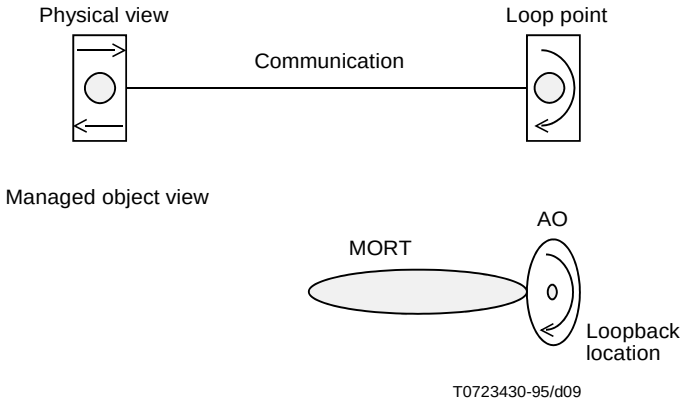

Figure 9 – Loopback configuration example – First case

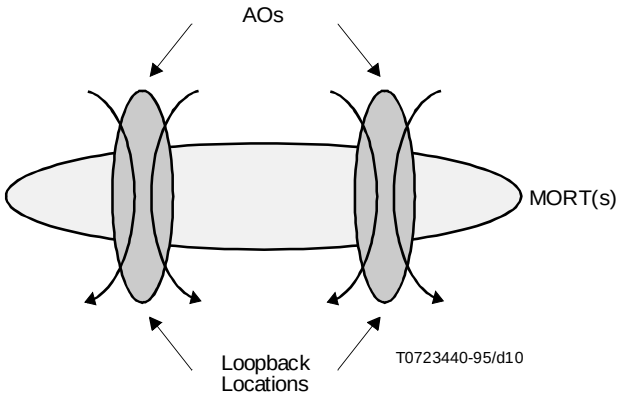

Figure 10 – Loopback configuration example – Second case

The first configuration shows a simple loopback example. The MORT is an object modelling the communication path between the two resources. A single AO models the resource which provides the loopback. The location of the loopback within the AO is indicated. The resource providing the point at which test data is injected is not modelled in this example.

The second configuration represents the capability of invoking a loopback test at different locations within the same MORT. In this case, the manager must specify the AO that is to be used for the loopback test.

The manager may request that multiple loopback tests be performed on the same MORT, or collection of MORTs, by specifying multiple AOs. When this is done the tests are applied, in an unspecified order, from the managed object having TARR functionality to each AO.

7.4.2.2 Test timing terminology

A number of timing parameters is used to control the operation of the loopback test in addition to the time-out period optionally used by all test categories defined in this Recommendation | International Standard. This subclause defines various timing terms, illustrated in Figures 11 and 12, and provides examples of how a test conductor may specify a legitimate and useful loopback test.

Figure 11 shows a test comprising three separate test patterns, with a test result notification being generated after completion of each pattern. Figure 12 shows one long test pattern, with test results being generated at intermediate points during the pattern. Many more configurations can be generated using combinations of these concepts.

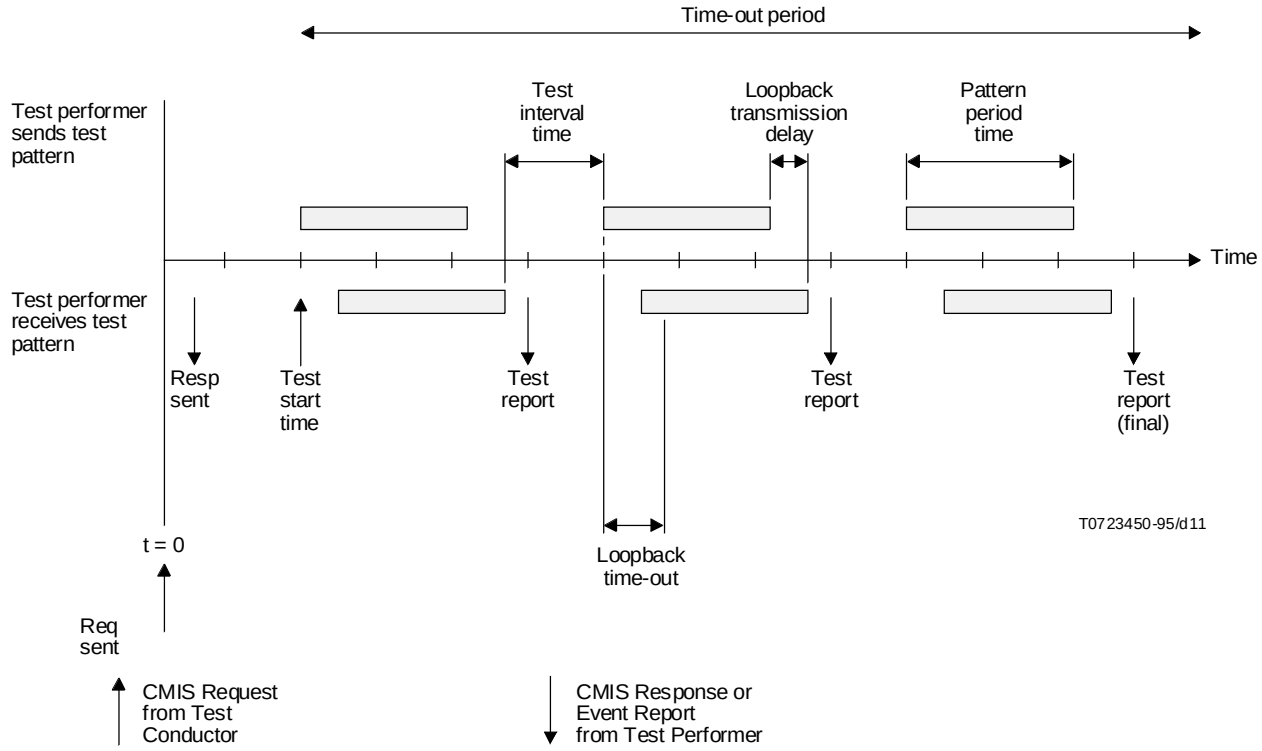

Figure 11 – Timing terminology of a multi-pattern test

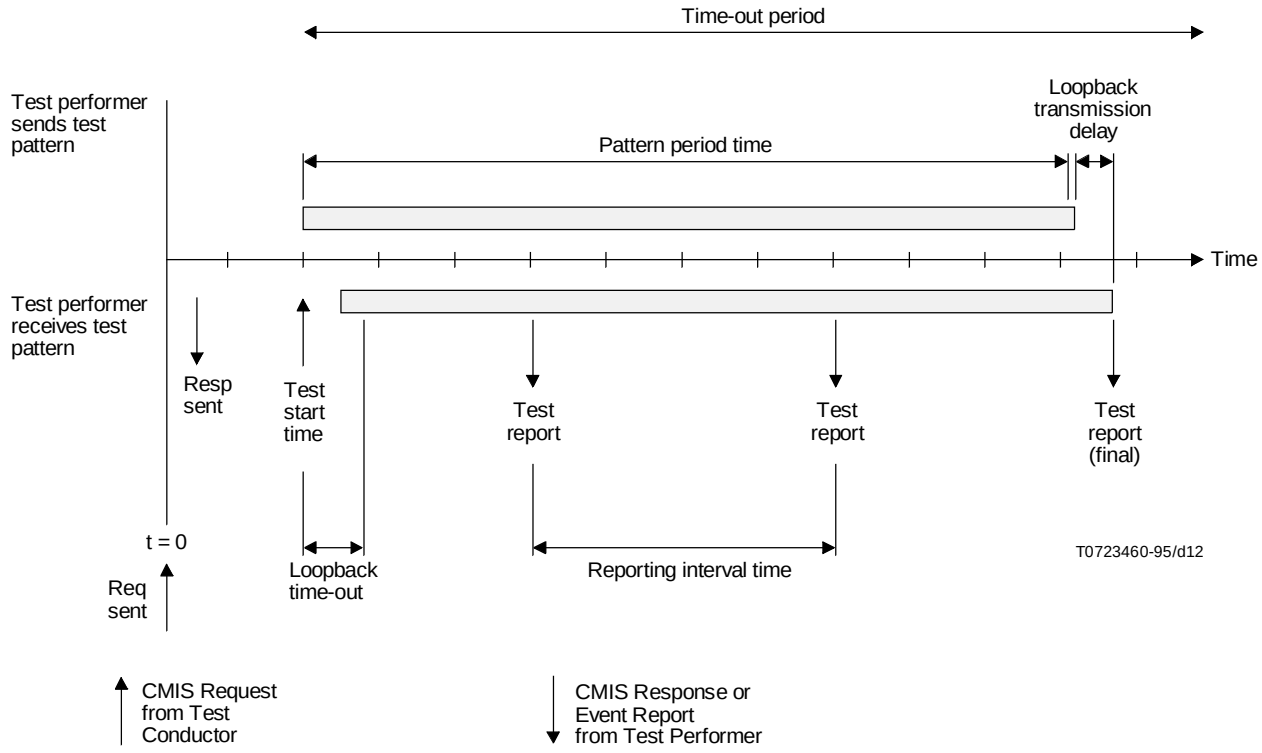

Figure 12 – Timing terminology of a long, single pattern test

The terms used to define the various aspects of test timing are:

- Start time – Used for scheduling the test sometime in the future. The time-out period begins at this time. (End time is not shown as the use of end time and time-out period is both confusing and redundant as the earliest of the two times causes the testOutcome parameter of the TestResult Notification to be Timed-out.)
- Pattern period time – The amount of time for which each test pattern is transmitted. This can either be expressed in terms of actual seconds, or a number of bits/octets of size of packets/blocks (optionally, at a given data ratio).
- Reporting interval time – If not zero (zero signifies no intermediate reporting), this is the time interval between intermediate result reports. As illustrated in Figure 12, intermediate result reports will be sent at the reporting interval following the test start time and at the same time interval thereafter.
- Loopback transmission delay time – The time delay associated with transmitting a pattern to and back from the loopback point. This is a physical measurement of the system under test.
- Loopback timeout – This time-out specifies the length of time that a test performer will wait for the transmitted data to be returned. It is measured from transmission of the test pattern until reception of returned data from the looped-back circuit. If the loopback transmission delay time is longer than the specified value, an immediate failure of the whole test will result and a Fail result will be returned.
- Test interval time – The time delay between the completion of reception of one test pattern and the start of transmission of the next. In Figure 11, for clarity, this is illustrated as quite large, but will be, in many cases, almost instantaneous – that is, transmission of the next test pattern will begin as soon as the previous test pattern had been fully received.
- Time-out period – The time within which the whole test should complete. For a loopback test, this should be greater than the number of patterns multiplied by the pattern period time plus the loopback transmission delay time. End time may be used for the same purpose.

7.4.2.3 Test operation

The test request specifies the MORT (or set of MORTs) that constitutes all or part of the communications path of the test. If the multiple MORTs configuration is not obvious by the normal static configuration information (via naming or other relationship), then the test request should not be allowed to refer to multiple MORTs.

The test request identifier indicates that a loopback test is being requested and may also specify loopback parameters such as data type for the test, test start time, reporting interval, error threshold and test time duration. There may be a set-up time during which:

a) the test environment is being established; and

b) the loopback is automatically or manually set up.

After the set-up time the test will be performed.

Failure to receive the loopback data within the loopback time-out period or error ratio exceeded causes the test outcome to be Fail. At the conclusion of the test, the test outcome will be Pass if all data was sent and received and the data unit error rate did not exceed the error threshold parameter.

For controlled tests, the object with TARR functionality may create TO(s). In the case where one test request specifies multiple loopback points, one TO is used for each loopback test.

The test will execute for a length of time specified by the time-out period parameter conveyed in the test request information or by a prior agreement. When multiple loopbacks are requested, then multiple loopback time-out periods, error thresholds, etc. are specified in the additional information parameter of the test request.

During the test, the test conductor may get intermediate test results. The test outcome is returned in the final test result report.

7.4.3 Test environment

The Loopback test may be intrusive or non-intrusive to both the MORT(s) and to the AO(s).

The Loopback test may not proceed if the MORT cannot be locked. The MORT will be unlocked upon normal termination of the test.

If the Loopback test is intrusive, it will abort under the following conditions:

- if the MORT or AOs have not previously been placed in an appropriate state (e.g. reserved for test);
- if the MORTs or AOs cannot be placed in the appropriate state during test initialization.

7.4.4 Uncontrolled/controlled

This test category may be modelled as a controlled test or an uncontrolled test.

7.4.5 TO requirements

This Recommendation | International Standard defines a loopback test object. This TO can be specialized to define more specific TO classes.

7.4.6 Initiation specific to test category

The following parameters, that are optional in ISO/IEC 10164-12, are required for initiation of a loopback test.

- identity of the AO if not a manual loopback, and if appropriate, additional AO information (e.g. loopback location within AO);
- identity of the MORTs if not implicitly known;
- time-out period if not know by prior agreement or if employing an uncontrolled loopback test.

The following parameters may also be needed:

- The loopback data (test signal sequence) to be used during the test. If this parameter is not present, the test pattern is implementation specific.
- The test start time.
- The time interval between the transmission of test patterns.
- The test result reporting interval time for the intermediate test results.
- The loopback type (e.g. physical loopback, echo, or both). If this parameter is not present, the type is implementation specific.
- The loopback time-out period for the test.
- The loopback error threshold.

7.4.7 Reporting and termination events

The events that trigger implicit result reporting are:

- reception of all data sent prior to expiry of test duration;
- time-out of loopback time-out period value;
- time-out of test time-out period (including set-up time and loopback time-out periods);
- specified time interval of reporting;
- reception of termination request, if a controlled test.

The events which can trigger the implicit termination are:

- completion of the test;
- an error threshold being exceeded;
- loopback time-out period exceeded;
- test time-out period exceeded.

7.4.8 Result report

When the test is modelled as an uncontrolled test, implicit reporting is used by means of one or more action replies. When the test is modelled as a controlled test, solicited reporting is supported by means of a get request; also, conditionally implicit reporting may be used by means of the TEST-RESULT service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 (also known as unsolicited reporting).

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following parameters are required:

- the identity of the AOs;
- if the test passed, the returned looped back data and the loopback error;
- if the test failed, the failure cause (error threshold exceeded or loopback time-out period exceeded);
- the test outcome chosen as appropriate from ITU-T X.745 | ISO/IEC 10164-12 shall be returned, if this is the final report.

The following optional parameters may also be returned:

- identity of the MORTs; if they are not explicitly specified in the test request;
- any additional information specific to the test.

The value of the test outcome is determined according to the following:

- if all test patterns were successfully transmitted and returned and if the error threshold is not exceeded, the test outcome is Pass;
- if the error threshold was exceeded or the loopback time-out period exceeded during any loopback test exercise, the test outcome is Fail;
- if the test duration exceeds the pre-specified time-out period, the test outcome is TIMEOUT;
- if the test terminates as a result of a procedural failure, or as a result of receipt of a TERMINATE REQUEST, the test outcome is Premature termination;
- otherwise, the test outcome is Inconclusive.

7.4.9 Test termination

For an uncontrolled test, this test category uses implicit termination.

For a controlled test, both implicit and explicit termination are used. For explicit termination, the TEST-TERMINATION service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 is used.

7.5 Protocol integrity test

7.5.1 Purpose of the test category

The purpose of the protocol integrity test is to verify if the MORT can conduct proper protocol interactions with a specified AO.

It is not the function of the test to provide logging of any resulting protocol exchanges; such an activity may be provided using a log(s) which stores the notifications that correspond to the transmission/reception of Protocol Elements.

7.5.2 MORT and AO requirements

No precursory exchange is required between the MORT and an AO before the exchange of test Protocol Elements. The MORT emits Protocol Elements according to a specified sequence. A particular Protocol Element may be transmitted upon one of the conditions:

1) the test start time (for the first Protocol Element only);
2) interval time since last transmission; or
3) reception of correct response Protocol Element within time-out period.

The AO(s) merely exhibit normal behaviour in response to the Protocol Elements received.

The test terminates implicitly either when the whole sequence of Protocol Elements has been completed or upon failure to meet condition for continuation of the next Protocol Element in sequence, or explicitly upon an instruction to terminate the test from the test conductor.

Reporting is done implicitly upon termination of the test. The test outcome is returned in the final test result report.

Figure 13 illustrates the managed objects involved in a protocol integrity test and Figure 14 provides an example of a protocol integrity test in which failure has occurred.

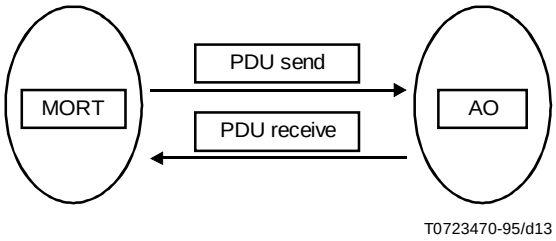

Figure 13 – Protocol integrity test environment

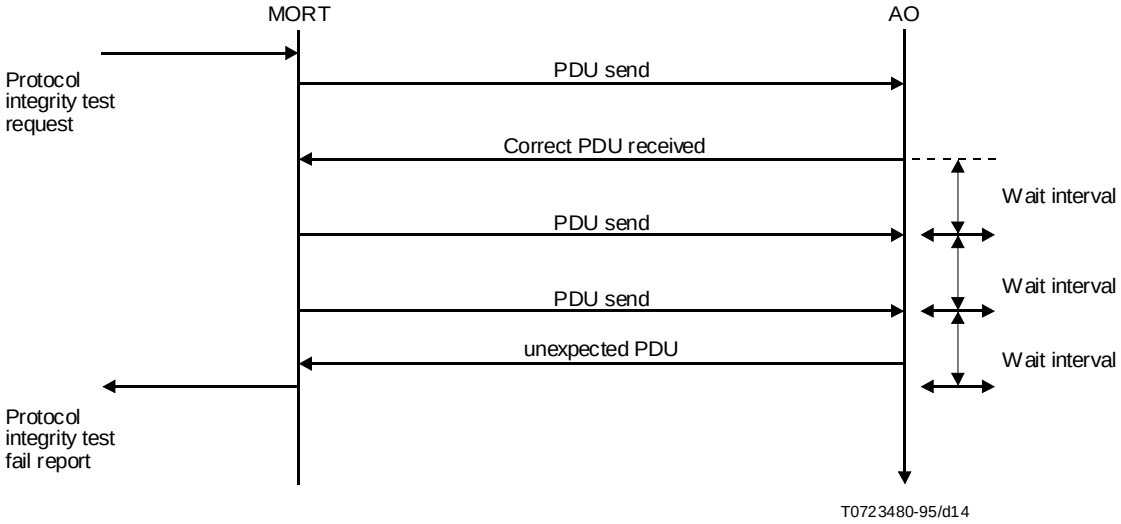

Figure 14 – Example of protocol integrity test

7.5.3 Test environment

The Protocol integrity test is generally non-intrusive to both the MORT(s) and to the AO(s). However, the Protocol integrity test will abort under the following conditions:

- if the MORT or AOs have not previously been placed in an appropriate state (e.g. reserved for test);
- if the MORTs or AOs cannot be placed in the appropriate state during test initialization.

7.5.4 Uncontrolled/controlled

This test category may be modelled either as an uncontrolled test or as a controlled test.

7.5.5 TO requirements

This Recommendation | International Standard defines a protocol integrity test object. This TO can be specialized to define more specific TO classes.

7.5.6 Initiation specific to test category

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following optional parameters defined in ITU-T X.745 | ISO/IEC 10164-12 are also required:

- identity of the AO, and if appropriate, additional AO information;
- identity of the MORT if not implicitly known by the TARR.

The following optional parameters may be specified:

- The test result reporting interval, if the intermediate test result reporting is desired.
- PDU to be transmitted.
- PDU expected to be received in response (optional);
- The condition to be met before the next PDU sent. The condition can be either a fixed time interval or following receipt of expected PDU before a waiting interval.
- A time, to be interpreted as either the time between PDU transmission (if condition is a fixed time interval) or as the time-out period (where proceed on PDU receipt is selected).

When requesting an uncontrolled protocol integrity test, the time-out period parameter defined in ITU-T X.745 | ISO/IEC 10164-12 shall be specified.

7.5.7 Reporting and termination events

The event that triggers implicit reporting of the result is:

- Conditions fulfilled for termination of test.

The events that trigger the termination of the test are:

- reception of termination request, if a controlled test;
- completion of last Protocol Element in sequence;
- failure to meet condition for continuation of the next PDU in sequence.

7.5.8 Result report

When the test is modelled as an uncontrolled test, implicit reporting is used by means of one or more action replies. When the test is modelled as a controlled test, solicited reporting is supported by means of a get request; also, conditionally implicit reporting may be used by means of the TEST-RESULT service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 (also known as unsolicited reporting).

Intermediate test result reports are sent following each Protocol Element tested. The final test result report is sent at termination.

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following parameters are required:

- the identity of the AOs;
- identity of the MORTs, if they are not explicitly specified in the test request.

The following optional parameters may also be returned:

- if this is the final report, the test outcome chosen as appropriate from ITU-T X.745 | ISO/IEC 10164-12 shall be returned;
- any additional information specific to the test;
- if pass, then:
  - the time taken from the initiation of the test until the completion of the test;
- if failed, then:
  - for each PDU expected, whether it was received successfully, whether it was not returned, or whether an incorrect PDU was received;
  - if an incorrect PDU is received, the actual PDU received.

The value of the test outcome is determined according to the following:

- if protocol elements were successfully transmitted and received, the test outcome is Pass;
- if protocol element transmission or reception errors were detected, the test outcome is Fail;
- if the test duration exceeds a pre-specified time-out value, the test outcome is TIMEOUT;
- if the test terminates as a result of a procedural failure, or as a result of receipt of a TERMINATE REQUEST, the test outcome is Premature termination;
- otherwise, the test outcome is Inconclusive.

7.5.9 Test termination

For an uncontrolled test, this test category uses implicit termination.

For a controlled test, both implicit and explicit termination are used. For explicit termination, the TEST-TERMINATION service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 is used.

7.6 Resource boundary test

7.6.1 Purpose of Test Category

A system may consist of many parts or resources. The purpose of this test is to verify the correct behaviour of separate resources internal to the system. The behaviour of the resource is tested by controlling and observing the interactions between the resource and its environment. Points of Control and Observation (PCOs) will be located at the boundaries of the resource. At these points the information exchanged between the resources under test is defined in terms of signals. The behaviour of the resource is tested by inserting test signals at the PCOs. Then, it is verified whether the signals generated by the resource conform to the behaviour specification of the resource.

This test category should be considered for resources that heavily interact with their environment. The resource behaviour can be verified when these interactions are prescribed in terms of protocols, services and interfaces. Annex K gives an example of use of the Resource Boundary test category.

7.6.2 MORT and AO requirements

7.6.2.1 Resource boundary test configuration

The MORTs represent the resources that are being tested by the resource boundary test. A MORT is an identifiable resource. The test can be applied to one or several adjacent resources. A resource can however, recursively contain several resources. When several adjacent resources are tested in one test, it is called multi-resource testing. In the case of multi-resource testing, each resource that is tested is represented by a MORT.

As PCOs are located at the boundaries of the resources, each of them may have connections with several resources. A PCO is represented by an AO. At the PCOs, signals can be observed and inserted. Furthermore, the information stream to adjacent resources can be disconnected. In that case, the signals are not forwarded to the adjacent resources that are not involved with the test. In this way, test data is not handled as user data.

Figure 15 shows possible configurations of MORTs and AOs for Resource Boundary testing.

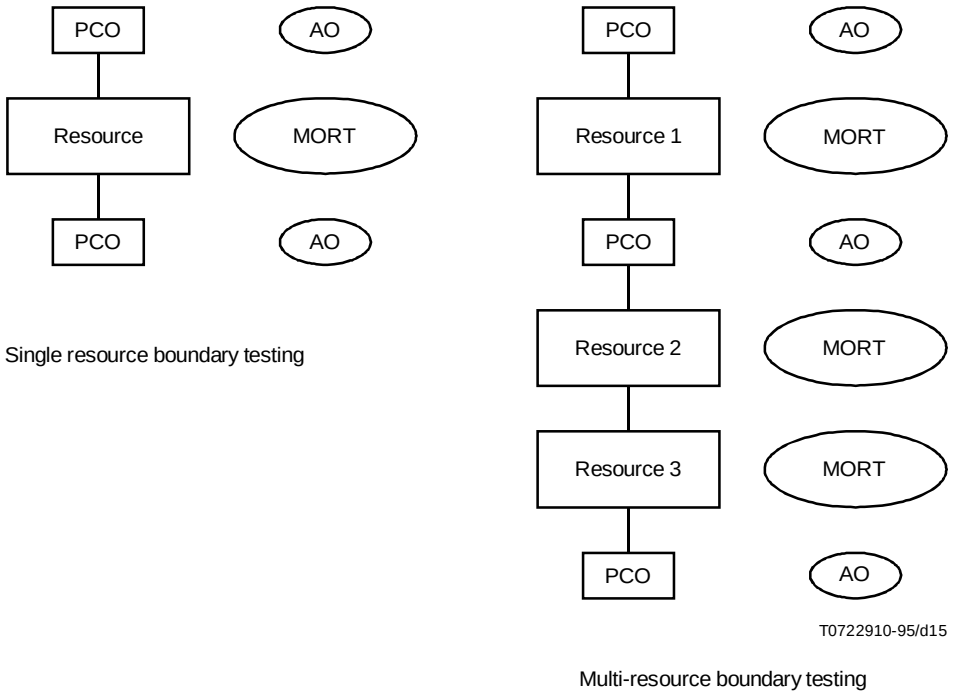

Figure 15 – Possible configurations of MORTs and AOs for Resource Boundary testing

7.6.2.2 Test operation

A resource boundary test is started by means of a test request that specifies the MORTs and the AOs. The test is terminated by means of a test termination. In between, several test events can be executed. Each test event contains one signal that has to be inserted or to be received at a PCO.

The first series of test events that have to be executed can be specified in the test request (when the Test Object is created). Afterwards, the test events to be executed can repeatedly be specified by configuration of the Test Object using the PT-SET service, see 9.3 of ITU-T Rec. X.745 | ISO/IEC 10164-12, the Test Management function.

In the case of non-deterministic behaviour, the sequence of test events will generally contain a single signal that has to be inserted. An expected response is then not specified. Any signal that is received in this case shall immediately be forwarded to the managing system.

Several signals could be listed in the sequence of events when completely determined behaviour of the resource under test is expected. In that case also expected responses in terms of signals are specified. The sequence of events consists then of both send and receive signals. This option enables a faster test execution, and is very suitable when time critical tests are being performed.

Send signals are those signals that have to be inserted in the resource under test via a PCO. Receive signals are generated by the resource under test and are visible at a specified PCO. At one PCO, signals may be received from, as well as inserted in, any of the connected resources. As the received signals may deviate from the expected results, the test course (the interactions between managing system and managed system) is not predefined.

The test performer shall consider the list containing the sequence of events to be active as soon as it starts handling the events. The sequence of events will be handled sequentially while this list is active. The list will remain active until all events are successfully handled or when a failure has occurred.

The handling of events differs for send and receive events. In case of a send event, the test performer shall first await the expiration of an optional timer related to the event. Then it shall insert the signal at the given PCO. In case of a receive event, the test performer shall wait for the receipt of the indicated signal before continuing with the next event. The receipt of another signal than the next one in the list, or the expiration of a wait duration timer related to a receive signal causes a failure. In that case the handling of the list is terminated.

7.6.3 Test environment

The Resource Boundary test may be intrusive or non-intrusive to both the MORT(s) and to the AO(s).

If the Resource Boundary test is intrusive, it will abort under the following conditions:

- if the MORT or AOs have not previously been placed in an appropriate state (e.g. reserved for test);
- if the MORTs or AOs cannot be placed in the appropriate state during test initialization.

7.6.4 Uncontrolled/controlled

This test category is modelled only as a controlled test, because many interactions between the managing system and the managed system are needed when a resource is tested.

7.6.5 TO requirements

This Recommendation | International Standard defines a resource boundary test object. This TO can be specialized to define more specific TO classes.

7.6.6 Initiation specific to test category

All parameters that are mandatory in ITU-T Rec. X.745 | ISO/IEC 10164-12 are also mandatory for this test category. The following parameters, defined as optional in ITU-T Rec. X.745 | ISO/IEC 10164-12, are mandatory for this test category:

- Identity of AO(s).
- For each AO: its state, e.g. to indicate for each information stream whether it will be observed and whether it will be disconnected (using the associatedObjectInform parameter of the AOs ASN.1 Definition).

The following additional information may be supplied in the test request (this information can also be supplied in PT-SET operations on a TO):

- Result Report Indicator (to indicate whether a result report is required when a sequence of test events was passed).
- Sequence of test events each comprising:
  - Event Id;
  - Signal type;
  - Parameters of the signal and their values;
  - Direction of the signal (inserted or received);
  - Identity of the MORT into which the signal is inserted or from which it is received;
  - Identity of the AO where the signal is inserted or received;
  - Wait duration timer value (in the case of a received signal, this timer indicates the time to the receipt of that signal and in the case of an inserted signal, this timer specifies the waiting time before the signal is inserted after the former event was completed).

7.6.7 Reporting and termination events

There are four events that trigger the reporting of intermediate results:

- Case 1 – Completion of last signal in sequence of test events (if indicated by the Result Report Indicator).
- Case 2 – Receipt of a signal at a PCO while no event sequence list is active and therefore no receive signal is to be matched.
- Case 3 – Receipt of a signal at a PCO while another receive signal is to be matched according to the active event sequence list.
- Case 4 – Wait duration timer expired in the case of a receive signal.

There is one event that triggers the explicit termination of the Resource Boundary test and the implicit reporting of the final test results:

- Receipt of termination request, sent by the test conductor.

7.6.8 Result report

The Resource Boundary test category supports both solicited and implicit reporting. Solicited reporting is supported by means of a get request. Implicit reporting is supported by means of the TEST-RESULT service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 (also known as unsolicited reporting).

The test uses implicit reporting, because for Resource Boundary testing, it is important that the responses of the managed system are forwarded to the managing system as soon as possible. There are two reasons for this. In the first place, time critical tests may be performed. Secondly, in the case of unexpected behaviour, the managing system may not know when test results become available.

The information returned in this report depends on the event that triggers the reporting and it relates only to the event that triggers the sending of the report. The following information related to the cases mentioned in 7.7.7 shall be supplied:

In case 1:

- Intermediate Resource Boundary Test outcome (passed).

In case 2:

- Intermediate Resource Boundary Test outcome (unexpected);
- Signal Type of the received signal;
- Signal values of the signal type;
- Identities of the MORTs from which the signal was received;
- Identities of the AOs where the signal was received.

In case 3:

– Intermediate Resource Boundary Test outcome (wrongSignalReceived);

– Event Id of failed test event;

– Signal Type of the received signal;

– Signal values of the signal type;

– Identities of the AOs where the signal was received.

In case 4:

– Intermediate Resource Boundary Test outcome (waitDurationTimerExpired);

– Event Id of failed test event.

The value of the test outcome to be returned in the final result will be Inconclusive.

7.6.9 Test termination

The test uses only explicit termination, because the managing system controls the tests. All intelligence with respect to the tests is contained within the managing system, e.g. the test course is conducted by the managing system. Therefore, the managing system should also decide when a test is terminated.

7.7 Resource self test

7.7.1 Purpose of the test category

Resource self tests (sometimes termed self-test) are used to investigate the ability of a resource (function or physical entities) to perform its allotted function at a given time. The nature of the resource(s) under test and the exercises involved is specific to the containing system.

7.7.2 MORT and AO requirements

Each resource self test involves a single MORT which is the resource under test. The test will cause some of the function inside of the resource be exercised and therefore may cause some of the MORT attributes to change value. Such MORT attributes may be monitored by the test conductor. This test category does not involve any AOs. Figure 16 illustrates the managed objects used by this test category.

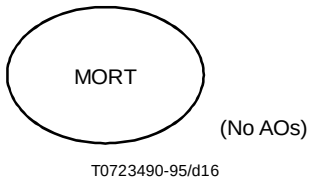

Figure 16 – Example of configuration for Resource Self test

7.7.3 Test environment

Depending on the nature of the test and the MORT, the Resource Self Test may be intrusive or non-intrusive to the MORT.

If the Resource Self test is intrusive, it will abort under the following conditions:

– if the MORT has not previously been placed in an appropriate state (e.g. reserved for test);

– if the MORT cannot be placed in the appropriate state during test initialization.

7.7.4 Uncontrolled/controlled

This test category may be invoked through the uncontrolled test request service or through the controlled test request service.

7.7.5 TO requirements

This Recommendation | International Standard defines a resource self test object. This TO can be specialized to define more specific TO classes.

7.7.6 Initiation specific to test category

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following parameters may also be required:

- Identity of the MORT if not implicitly known (e.g. if object with TARR functionality is the MORT).
- The type of diagnostic test to be run. If the diagnostic type is not included in the test request, the type of diagnostic is specific and known to the MORT.
- Phases of the test to be run.
- The number of iterations for each phase.
- Time-out period.

7.7.7 Reporting and termination events

Resource self test results, when available, may be explicitly retrieved by the test conductor through the resource self test results attribute, or they may be implicitly generated. The events which trigger the implicit reporting are:

- any point during the exercise of the function, typically at the end of appropriate phases, if defined;
- completion of the exercise of the function.

Resource self tests may be explicitly terminated by the test conductor during the life time of the test, or they may be implicitly terminated. The events which trigger the implicit termination are:

- completion of the test;
- failure of the test to complete normally;
- expiry of the time interval during which the test is to be completed.

7.7.8 Result report

When the test is modelled as an uncontrolled test, implicit reporting is used by means of one or more action replies. When the test is modelled as a controlled test, solicited reporting is supported by means of a get request; also, conditionally implicit reporting may be used by means of the TEST-RESULT service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 (also known as unsolicited reporting).

In addition to the parameters that are required by ITU-T X.745 | ISO/IEC 10164-12, the following parameter is required:

- the identity of the MORT, if the MORT was not included as part of the test request.

The following optional parameters may also be returned:

- information specific to the exercises defined;
- where appropriate, additional information that defines as precisely as possible any follow-up actions that should be taken;
- the test outcome chosen as appropriate from ITU-T X.745 | ISO/IEC 10164-12 shall be returned in the final test result.

The value of the test outcome to be returned in the final test result report is determined according to the following:

- if all test procedures were completed successfully, the outcome is Pass;
- if any test procedure detects an error, the outcome is Fail;

– if the test duration exceeds a pre-specified time-out value, the test outcome is TIMEOUT;

– if the test terminates as a result of a procedural failure, or as a result of receipt of a TERMINATE REQUEST, the test outcome is Premature termination;

– otherwise, the test outcome is Inconclusive.

7.7.9 Test termination

For an uncontrolled test, this test category uses implicit termination.

For a controlled test, both implicit and explicit termination are used. For explicit termination, the TEST-TERMINATION service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 is used.

7.8 Test infrastructure test

7.8.1 Purpose of the test category

Test infrastructure tests allow investigation of the ability of a managed open system to initiate tests, return result reports, and (in the case of controlled tests) respond to monitoring and control actions. For example, a test infrastructure test may be performed in advance of initiation of tests of specific resources that involve the managed open system.

The test infrastructure test is a 'null' test whose purpose is solely to investigate the ability of a managed open system to behave as specified in ITU-T Rec. X.745 | ISO/IEC 10164-12 – that is, to receive an incoming test request and generate appropriate responses. No test exercises are performed on resources within or external to the managed open system.

Progress through the test states defined by ITU-T Rec. X.745 | ISO/IEC 10164-12 will occur at a specified interval. If any test offered by the managed open system provides the ability to:

– respond to suspend/resume requests;

– respond to termination requests;

– schedule tests;

– abort tests following specified time-out periods,

then the test infrastructure test should permit verification of these aspects of the implementation of ITU-T Rec. X.745 | ISO/IEC 10164-12 on the managed open system.

NOTE – When using this test category, it is necessary to specify the exact significance of progression through the test states; the "generated appropriate responses", what test states are supported, time interval spent in each state, etc.

7.8.2 MORT and AO requirements and AO requirements

The MORT is implicitly the managed object with TARR functionality. No AOs are involved in this test. Figure 17 summarizes the managed objects involved in this test category.

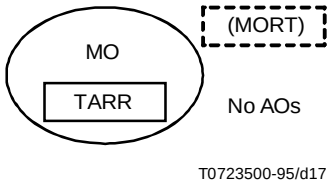

Figure 17 – Example of configuration for Test Infrastructure test

7.8.3 Test environment

The Test Infrastructure test is non-intrusive.

7.8.4 Uncontrolled/controlled

The test may be modelled either as an uncontrolled test or as a controlled test.

7.8.5 TO requirements

This Recommendation | International Standard defines a test infrastructure test object. This TO cannot be specialized.

7.8.6 Initiation specific to test category

When requesting an uncontrolled test, the time-out period defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 shall be specified.

When requesting a controlled test, only those parameters required by ITU-T Rec. X.745 | ISO/IEC 10164-12 need be specified.

Additionally, the time interval required between test state transitions may be specified (defaults to an implementation-specific value if not supplied).

7.8.7 Reporting and termination events

The events that trigger implicit reporting are:

- transition between test states (if provided by the implementation);
- completion of the exercise.

The events that trigger implicit termination are:

- completion of the exercise;
- time-out;
- failure to complete the exercise due to an error being encountered.

7.8.8 Result report

When the test is modelled as an uncontrolled test, implicit reporting is used by means of one or more action replies. When the test is modelled as a controlled test, solicited reporting is supported by means of a get request; also, conditionally implicit reporting may be used by means of the TEST-RESULT service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 (also known as unsolicited reporting).

In addition to the parameters that are required by ITU-T Rec. X.745 | ISO/IEC 10164-12, the following parameter is required in the result report:

- if this is the final result report, the test outcome – chosen as described below.

The following optional parameter may also be returned:

- other information specific to the test implementation.

The outcome of the test is determined as follows:

- if all exercises have been completed correctly, the outcome will be Pass;
- if any exercise detected an error, the outcome will be Fail;
- if the test duration exceeded the specified time-out period, the outcome will be Timed-out;
- if the test terminated as a result of a procedural error being encountered or as the result of receipt of a TERMINATION REQUEST, the result will be Premature termination;
- otherwise, the outcome will be Inconclusive.

7.8.9 Test termination

For an uncontrolled test, this test category uses implicit termination.

For a controlled test, both implicit and explicit termination are used. For explicit termination, the TEST-TERMINATION service defined in ITU-T Rec. X.745 | ISO/IEC 10164-12 is used.

8 Generic definitions

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 Generic attribute types

8.1.1 Connection test results

This attribute contains the connection test results.

8.1.2 Connectivity test results

This attribute contains the connectivity test results.

8.1.3 Connectivity type

This attribute indicates whether the connectivity test is for a connection-oriented protocol, or it is for a connectionless protocol.

8.1.4 Data integrity type

This attribute indicates whether the data integrity test is on a connection-oriented protocol, or it is on a connectionless protocol.

8.1.5 Data integrity results

This attribute contains the test results of a data integrity test. If the test outcome is pass, the test results include the establishment time, and the original data unit.

8.1.6 Data units

This attribute is used to specify the type and quantity of test data units to be sent during the test. This attribute is only used if the test supports manager specification of the type and/or the quantity of data units.

8.1.7 End connection test results

This attribute contains the end connection test results.

8.1.8 Error ratio report type

This attribute contains the form for the measurement of the error ratio. It may be in the form of number of error bits or in the form of percentage of error seconds.

8.1.9 Establishment time

This attribute contains the time taken for establishing connectivity between two entities.

8.1.10 Loopback test results

This attribute contains the test results of a loopback test including the duration of the test, if specified, the data units used in the test, the error ratio, and the error cause, if error was experienced.

8.1.11 Loopback threshold

This attribute specifies the error threshold. Upon the crossing of the threshold, the associated loopback test shall be terminated and the test outcome shall be Fail.

8.1.12 Loopback type

This attribute or parameter is used to indicate the loopback type, e.g. physical loopback, echo, analogue or digital, non-transparent physical loopback, transparent physical loopback, payload physical loopback, echo data back, etc.

8.1.13 PDU reception

This attribute contains the PDU information which is used to compare with the PDU received from a protocol integrity test in order to determine what protocol element should be sent in the next transmission and when the transmission should take place.

8.1.14 PDU sequence

This attribute contains the PDU sequence for a protocol integrity test.

8.1.15 Protocol integrity results

This attribute contains the cause for an unsuccessful protocol integrity test. An unsuccessful protocol integrity test may be caused by a wrong PDU response or by an unacknowledged PDU.

8.1.16 Resource boundary test results

This attribute contains either the details of a failed test and/or the information on the received signal at a PCO. It contains information on a received signal if a signal was received that was not specified as expected in a sequence of events. It contains the details of a failure in the case of a failed test. Two kind of failures may be reported:

- A wrong signal was received – Another signal was received than the signal as specified in the sequence of events.
- The wait duration timer expired – An expected signal was specified, but no signal was received.

In both cases, also the event identifier of the failed test event is reported.

8.1.17 Resource self test results

This attribute contains the results of an resource self test. The result contains the function tested and any additional result related to the function.

8.1.18 Result interval

This attribute specifies the result reporting interval. A new report is generated for every specified time interval after the test is initiated.

8.1.19 Result report indicator

This attribute indicates whether a result report is required when a sequence of test events was passed.

8.1.20 Sequence of events

This attribute specifies a sequence of signals that have to be inserted or received at the specified MORT and AO. This is called a sequence of test events. The sequence of test events may contain a single signal that has to be inserted or received, but also a sequence of events may be specified that consists of send and receive signals. The test performer shall handle the events sequentially. In the case of a receive event, the test performer shall wait for the receipt of the indicated signal before continuing with the next event. It waits until the wait duration timer has expired. If expired, a failure may have occurred within the resource. In the case of a send signal, the wait duration timer specifies the time that is waited upon before the signal is inserted after the former event was completed.

8.1.21 Test conditions

This attribute specifies the conditions under which the resources should be allocated to the test. In particular, it indicates whether the test should initiate if the MORT is busy and whether the user of the MORT can cause the test to abort.

8.1.22 Test signal

This attribute identifies a particular signal type.

8.1.23 Test signal sequence

This attribute contains the data comprising the test traffic for a loopback test.

8.1.24 Test threshold

This attribute specifies the error threshold. Upon the crossing of the threshold, the associated test shall be terminated and the test outcome shall be Fail.

8.1.25 Waiting interval

This attribute, if used, indicates the waiting interval before the transmission of the next protocol element. The interval measures from the time when the last protocol element was emitted.

8.2 Managed objects

8.2.1 Connection test object

This managed object class is derived from the test object managed object class defined in ITU-T Rec. 745 | ISO/IEC 10164-12. An instance of the object class may be instantiated to represent the test characteristics of a connection test.

8.2.2 Connectivity test object

This managed object class is derived from the test object managed object class defined in ITU-T Rec. 745 | ISO/IEC 10164-12. An instance of the object class may be instantiated to represent the test characteristics of a connectivity test. If implicit termination is supported by the managed object with TARR functionality, a connectivity test managed object may be implicitly deleted upon the termination of the test.

8.2.3 Data integrity test object

This managed object class is derived from the test object managed object class defined in ITU-T Rec. 745 | ISO/IEC 10164-12. An instance of the object class may be instantiated to support an instance of the data integrity test. This managed object class represents the test characteristics of a data integrity test. It is created as a result of receiving a controlled test request service as defined in ITU-T Rec. 745 | ISO/IEC 10164-12. If implicit termination is supported by the managed object with TARR functionality, a data integrity test managed object may be implicitly deleted upon the termination of the test.

8.2.4 Resource self test object

This managed object is derived from the test object managed object class defined in ITU-T Rec. 745 | ISO/IEC 10164-12. It may be instantiated to support an instance of the resource self test. This managed object class represents the test characteristics of a resource self test. An instance of the object class may be instantiated as a result of receiving a controlled test request service as defined in ITU-T Rec. 745 | ISO/IEC 10164-12. If implicit termination is supported by the managed object with TARR functionality, a resource self test object may be implicitly deleted upon the termination of the test.

8.2.5 Loopback test object

This managed object is derived from the test object managed object class defined in ITU-T Rec. 745 | ISO/IEC 10164-12. It may be instantiated to support an instance of the loopback test. This managed object class represents the test characteristics of a loopback test. An instance of the object class may be instantiated as a result of receiving a controlled test request service as defined in ITU-T Rec. 745 | ISO/IEC 10164-12. If implicit termination is supported by the managed object with TARR functionality, a loopback test managed object may be implicitly deleted upon the termination of the test.

8.2.6 Protocol integrity test object

This managed object is derived from the test object managed object class defined in ITU-T Rec. 745 | ISO/IEC 10164-12. It may be instantiated to support an instance of the protocol integrity test. This managed object class represents the test characteristics of a protocol integrity test. An instance of the object class may be instantiated as a result of receiving a controlled test request service as defined in ITU-T Rec. 745 | ISO/IEC 10164-12. If implicit termination is supported by the managed object with TARR functionality, a protocol integrity test managed object may be implicitly deleted upon the termination of the test.

8.2.7 Resource boundary test object

This managed object class is derived from the Test Object managed object class defined in ITU-T Rec. X.745 | ISO/IEC 10164-12. It is instantiated to support a resource boundary test. This managed object class represents the management characteristics of a resource boundary test. An instance of the object class may be instantiated as a result of receiving a controlled test request service as defined in ITU-T Rec. X.745 | ISO/IEC 10164-12. As the test result notification package is present, the test results may be reported implicitly. A resource boundary test object may only be explicitly deleted.

8.2.8 Test infrastructure test object

This managed object is derived from the test object managed object class defined in ITU-T Rec. 745 | ISO/IEC 10164-12. It may be instantiated to support an instance of the test infrastructure test. This managed object class represents the test characteristics of a test infrastructure test. An instance of the object class may be instantiated as a result of receiving a controlled test request service as defined in ITU-T Rec. 745 | ISO/IEC 10164-12. If implicit termination is supported by the managed object with TARR functionality, a test infrastructure test managed object may be implicitly deleted upon the termination of the test.

8.3 Imported generic definitions

This Recommendation | International Standard makes use of the following definitions from ITU-T Rec. X.745 | ISO/IEC 10164-12:

- MORTs package;
- AOs package;
- test result notification service;
- time-out period package;
- scheduling conflict package.

NOTE – Any instantiable object classes using the optional AO information should use a parameter template on the associatedObjects attribute, the testRequestControlledAction and/or the testRequestUncontrolledAction as appropriate. The context keyword “associatedObjects.AssociatedObjectInfo” should be used in the parameter template used to register the syntax and object identifier for the ANY DEFINED BY.

8.4 Compliance

Managed object class definitions support the test categories defined in this Recommendation | International Standard by incorporating the appropriate specification of managed object classes, attributes, actions and notifications through reference to the definitions defined in Annex A of this Recommendation | International Standard, Annex A of ITU-T Rec. X.745 | ISO/IEC 10164-12, and in CCITT Rec. X.721 | ISO/IEC 10165-2. The reference mechanism is defined in the CCITT Rec. X.722 | ISO/IEC 10165-4.

9 Services

This Recommendation | International Standard makes use of the services defined in other Systems Management Functions and does not define any additional services.

10 Functional units

This Recommendation | International Standard makes use of the functional units defined in ITU-T X.745 | ISO/IEC 10164-12 and does not define any additional functional units.

11 Protocol

The protocol specification for this Recommendation | International Standard is the same as that of ITU-T Rec. X.745 | ISO/IEC 10164-12.

12 Relationships with other functions

The test categories defined in this Recommendation | International Standard are based on the model defined in ITU-T Rec. 745 | ISO/IEC 10164-12. When solicited reporting of test results is used, this Recommendation | International Standard utilizes the PT-GET services defined in CCITT Rec. X.730 | ISO/IEC 10164-1. When changing attribute values in the TOs, this Recommendation | International Standard utilizes the PT-SET service defined in CCITT Rec. X.730 | ISO/IEC 10164-1.

13 Conformance

Implementations claiming to conform to this Recommendation | International Standard shall comply with the conformance requirements as defined in the following subclauses.

13.1 Static conformance

The implementation shall conform to the requirements of this Recommendation | International Standard in the manager role, the agent role, or both roles. A claim of conformance to at least one role shall be made in Table B.1.

If a claim of conformance is made for support in the manager role, the implementation shall support at least one of the management operations on the attributes or one of the notifications of the managed objects specified by this Recommendation | International Standard. The conformance requirements in the manager role for those management operations are identified in Table B.2 and further tables referenced by Annex B.

If a claim of conformance is made for support in the agent role, the implementation shall support one or more instances of the managed objects described in Table B.3. A claim of conformance in the agent role requires the support of all the mandatory operations and mandatory notifications specified by those management definitions.

The implementation shall support the transfer syntax derived from the encoding rules specified in CCITT Rec. X.209 | ISO/IEC 8825 named {joint-iso-ccitt asn1(1) basicEncoding(1)} for the abstract data types referenced by the definitions for which support is claimed.

13.2 Dynamic conformance

Implementations claiming to conform to this Recommendation | International Standard shall support the elements of procedure and definitions of semantics corresponding to the definitions for which support is claimed.

13.3 Management implementation conformance statement requirements

Any MCS proforma, MOCS proforma, and MRCS proforma which conforms to this Recommendation | International Standard shall be technically identical to the proformas specified in Annexes B, C and D preserving table numbering and the index numbers of items, and differing only in pagination and page headers.

The supplier of an implementation which is claimed to conform to this Recommendation | International Standard shall complete a copy of the Management Conformance Summary (MCS) provided in Annex B as part of the conformance requirements together with any other ICS proformas referenced as applicable from that MCS. An ICS which conforms to this Recommendation | International Standard shall:

- describe an implementation which conforms to this Recommendation | International Standard;
- have been completed in accordance with the instructions for completion given in ITU-T Rec. X.724 | ISO/IEC 10165-6;
- include the information necessary to uniquely identify both the supplier and the implementation.

Annex A

Definition of management information

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

A.1 Test category related management information

For each test category, this subclause summarizes the management information specified for the category.

In the following subclauses for each test category, the parameters associated with the test category information parameter(s) have assigned object identifiers (see A.5) which are used to identify that particular test category in the test request action information, test request action reply (for controlled tests) and test request notification reply (for controlled tests).

A.1.1 Connection test category

a) Test request service type: testRequestControlledAction and/or testRequestUncontrolledAction.

b) Specific errors:

- Test request controlled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable, independentTestInvocationError, relatedTOError.
- Test request uncontrolled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable.
- Test suspend/resume action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testSuspendResumeErrror.
- Test termination action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testTerminateError.

c) Test category information parameter:

- Test request controlled action information parameter: connectionTestInfoParam.
- Test request uncontrolled action information parameter: connectionTestInfoParam.
- Test request notification reply parameter: connectionControlledResultsParam.
- Test request uncontrolled action reply parameter: connectionUncontrolledResultsParam.

d) Test control: Test suspend/resume and Test termination are applicable if controlled test is in use.

e) TO requirements: The controlled test must instantiate an instance of TO.

f) TO class: connectionTestObject.

A.1.2 Connectivity test category

a) Test request service type: testRequestControlledAction and/or testRequestUncontrolledAction.

b) Specific errors:

- Test request controlled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable, independentTestInvocationError, relatedTOError.
- Test request uncontrolled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable.
- Test suspend/resume action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation.
- Test termination action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testTerminateErrror.

c) Test category information parameter:

- Test request controlled action information parameter: connectivityTestInfoParam.
- Test request uncontrolled action information parameter: connectivityTestInfoParam.
- Test request notification reply parameter: connectivityControlledResultsParam, invalidTestOperation, testSuspendResumeErrror.
- Test request uncontrolled action reply parameter: connectivityUncontrolledResultsParam.

d) Test control: Test suspend/resume and Test termination are applicable if controlled test is in use.

e) TO requirements: The controlled test must instantiate an instance of TO.

f) TO class: connectivityTestObject.

A.1.3 Data integrity test category

a) Test request service type: testRequestControlledAction and/or testRequestUncontrolledAction.

b) Specific errors:

- Test request controlled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable, independentTestInvocationError, relatedTOError.
- Test request uncontrolled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable.
- Test suspend/resume action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testSuspendResumeErrror.
- Test termination action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testTerminateErrror.

c) Test category information parameter:

- Test request controlled action information parameter: dataIntegrityTestInfoParam.
- Test request uncontrolled action information parameter: dataIntegrityTestInfoParam.
- Test request notification reply parameter: dataIntegrityControlledResultsParam.
- Test request uncontrolled action reply parameter: dataIntegrityUncontrolledResultsParam.

d) Test control: Test suspend/resume and Test termination are applicable if controlled test is in use.

e) TO requirements: The controlled test must instantiate an instance of TO.

f) TO class: dataIntegrityTestObject.

A.1.4 Loopback test category

a) Test request service type: testRequestControlledAction and/or testRequestUncontrolledAction.

b) Specific errors:

- Test request controlled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable, independentTestInvocationError, relatedTOError.
- Test request uncontrolled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable.
- Test suspend/resume action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation.
- Test termination action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testTerminateErrror.

c) Test category information parameter:

- Test request controlled action information parameter: loopbackTestInfoParam.
- Test request uncontrolled action information parameter: loopbackTestInfoParam.
- Test request notification reply parameter: loopbackControlledResultsParam.
- Test request uncontrolled action reply parameter: loopbackUncontrolledResultsParam.

d) Test control: Test suspend/resume and Test termination are applicable if controlled test is in use.

e) TO requirements: The controlled test must instantiate an instance of TO.

f) TO class: loopbackTestObject.

A.1.5 Protocol integrity test category

a) Test request service type: testRequestControlledAction and/or testRequestUncontrolledAction.

b) Specific errors:

- Test request controlled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable, independentTestInvocationError, relatedTOError.
- Test request uncontrolled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable.
- Test suspend/resume action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testSuspendResumeErrror.
- Test termination action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testTerminateErrror.

c) Test category information parameter:

- Test request controlled action information parameter: protocolIntegrityTestInfoParam.
- Test request uncontrolled action information parameter: protocolIntegrityTestInfoParam.
- Test request notification reply parameter: protocolIntegrityControlledResultsParam.
- Test request uncontrolled action reply parameter: protocolIntegrityUncontrolledResultsParam.

d) Test control: Test suspend/resume and Test termination are applicable if controlled test is in use.

e) TO requirements: The controlled test must instantiate an instance of TO.

f) TO class: protocolIntegrityTestObject.

A.1.6 Resource Boundary test category

a) Test request service type: testRequestControlledAction.

b) Specific Errors:

- Test request controlled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, noSuchAssociatedObject, associatedObjectNotAvailable, independentTestInvocationError, relatedTOError.
- Test suspend/resume action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testSuspendResumeErrror.
- Test termination action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testTerminateErrror.

c) Test category information parameter:

- Test request controlled action information parameter: resourceBoundaryTestInfoParam.

NOTE – No ‘Test request notification reply parameter’ is defined, because no test category specific reply information is needed for the resource boundary test.

d) Test control: Test suspend/resume and Test termination are applicable.

e) TO requirements: The controlled test must instantiate an instance of TO.

f) TO class: resourceBoundaryTestObject.

A.1.7 Resource self test category

a) Test request service type: testRequestControlledAction and/or testRequestUncontrolledAction.

b) Specific errors:

- Test request controlled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, independentTestInvocationError, relatedTOError.

- Test request uncontrolled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId.
- Test suspend/resume action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testSuspendResumeErrror.
- Test termination action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testTerminateErrror.

c) Test category information parameter:

- Test request controlled action information parameter: resourceSelfTestInfoParam.
- Test request uncontrolled action information parameter: resourceSelfTestInfoParam.
- Test request notification reply parameter: resourceSelfControlledResultsParam.
- Test request uncontrolled action reply parameter: resourceSelfUncontrolledResultsParam.

d) Test control: Test suspend/resume and Test termination are applicable if controlled test is in use.

e) TO requirements: The controlled test must instantiate an instance of TO.

f) TO class: resourceSelfTestObject.

A.1.8 Test infrastructure test category

a) Test request service type: testRequestControlledAction and/or testRequestUncontrolledAction.

b) Specific errors:

- Test request controlled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId, independentTestInvocationError, relatedTOError.
- Test request uncontrolled action: noSuchMORT, mORTNotAvailable, mistypedTestRequestInformation, noSuchTestCategoryId.
- Test suspend/resume action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testSuspendResumeErrror.
- Test termination action: noSuchInvocationId, noSuchTestSessionId, invalidTestOperation, testTerminateErrror.

c) Test category information parameter:

- No test category specific parameter is required.

d) Test control: Test suspend/resume and Test termination are applicable if controlled test is in use.

e) TO requirements: The controlled test must instantiate an instance of TO.

f) TO class: "ITU-T Rec. X.745 (1993) | ISO/IEC 10164-12:1994": testObject.

A.2 Generic Object Classes

A.2.1 Connection Test Object

```
connectionTestObject      MANAGED OBJECT CLASS
DERIVED FROM    "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObject;
CHARACTERIZED BY
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":mORTsPackage,
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":associatedObjectsPackage,
        connectionTestPkg
            PACKAGE
                BEHAVIOUR connectionTestBehaviour      BEHAVIOUR
                DEFINED AS “The MORT(s) represent the communications path to be tested. The two associated objects
                            represent the resources at the ends of the communication path that drive signals into and
                            receive signals from the communication path. The test patterns represents the signals or
                            data to be applied to the communication path. If the test is intrusive, the
                            administrativeState and availabilityStatus may need to be supported and to be set to a
                            proper state before the test may be initiated.”;;
            ATTRIBUTES
                testPatterns      GET,
                connectionTestResults      GET;;;
```

```
CONDITIONAL PACKAGES
        timeoutPeriodPkg            PRESENT IF  !The timeout period is specified in the test request.!,
        connectionTestResultPackage PACKAGE NOTIFICATIONS
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testResultNotification connectionControlledResultsParam;;
                                                    PRESENT IF !Unsolicited reporting is supported.!;

REGISTERED AS {part14ObjectClass  ??};
```

A.2.2 Connectivity Test Object

```
connectivityTestObject    MANAGED OBJECT CLASS
DERIVED FROM    "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObject;
CHARACTERIZED BY
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":mORTsPackage,
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":associatedObjectsPackage,
        connectivityTestPkg
            PACKAGE
                BEHAVIOUR connectivityTestBehaviour     BEHAVIOUR
                DEFINED AS "The MORT represents the entity that initiates the establishment of the connection with
                        the associated object. The connectivityTestResults and connectivityType must be included
                        as part of the monitoredAttributes parameter of the testResultNotification. If the
                        timeoutPeriod is present and expires before the test outcome can be concluded, the
                        timeoutPeriod will be returned in the test result with the test outcome set to TIMEOUT.";;
            ATTRIBUTES
                connectivityType            GET,
                connectivityResults         GET,
                establishmentTime           GET;;;
CONDITIONAL PACKAGES
        timeoutPeriodPkg            PRESENT IF !The timeout period is specified in the test request.!,
        connectivityTestResultPackage PACKAGE NOTIFICATIONS
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testResultNotification connectivityControlledResultsParam;;
                                                    PRESENT IF !Unsolicited reporting is supported.!;

REGISTERED AS {part14ObjectClass  ??};
```

A.2.3 Data Integrity Test Object

```
dataIntegrityTestObject  MANAGED OBJECT CLASS
DERIVED FROM    "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObject;
CHARACTERIZED BY
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":mORTsPackage,
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":associatedObjectsPackage,
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":actualTestTimePackage,
        dataIntegrityResultsPkg,
        dataIntegrityTestPkg
            PACKAGE
                BEHAVIOUR dataIntegrityTestBehaviour   BEHAVIOUR
                DEFINED AS "While in the Testing state the MORT transmits data to an Associated object. The
                        Associated object, upon reception of the data, reflects a copy of the data back to the
                        MORT.";;
            ATTRIBUTES
                dataIntegrityType           GET,
                testPatterns                        GET-REPLACE;;;
CONDITIONAL PACKAGES
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":timeoutPeriodPkg        PRESENT IF !The timeout period is
                                                    specified in the test request.!,
        dataIntegrityTestResultPackage PACKAGE NOTIFICATIONS
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testResultNotification dataIntegrityControlledResultsParam;;
                                                    PRESENT IF !Unsolicited reporting is supported.!,
        dataIntegrityThresholdPkg       PRESENT IF !testThreshold is present in the initAttributeList parameter of
                                                    the testRequestControlled request.!;

REGISTERED AS {part14ObjectClass  ??};
```

A.2.4 Loopback Test Object

```
loopbackTestObject MANAGED OBJECT CLASS
DERIVED FROM   "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObject;
CHARACTERIZED BY
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":mORTsPackage,
        loopbackResultsPkg,
        loopbackTestPkg
            PACKAGE
                BEHAVIOUR loopbackTestBehaviour        BEHAVIOUR
                DEFINED AS "The mORTs attribute identifies a part of the communications path that is to be tested.
                        The associatedObjects attribute identifies the loopback point. The startTime and stopTime
                        attributes (when present) indicate the time that the manager desires that the test phase be
                        started and stopped, respectively. The loopbackResults, loopbackType and testPattern
                        attributes must be included as part of the monitoredAttributes parameter of the
                        testResultNotification. The testThreshold should be included if it is present in the loopback
                        test object. The errorRatioReportType attribute indicates in what form the test conductor
                        wants the error ratio be reported, whether in the form of number of error bits or in
                        percentage of error seconds.";;
            ATTRIBUTES
                loopbackType                    GET-REPLACE,
                testPatterns                    GET-REPLACE,
                errorRatioReportType    GET;;;

CONDITIONAL PACKAGES
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":associatedObjectsPackage      PRESENT IF !The AO parameter is
                                                                specified in the test request.!,
        loopbackTestResultPackage PACKAGE NOTIFICATIONS
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testResultNotification loopbackControlledResultsParam;;
                                                                PRESENT IF !Unsolicited reporting is supported.!,
        timeoutPeriodPkg        PRESENT IF !The timeout period is specified in the test request.!,
        testConditionsPkg       PRESENT IF !testConditions is present in the initAttributeList parameter of the
                                testRequestControlled request.!,
        dataUnitsPkg    PRESENT IF !the capability is implemented and parameter is present in the
                                initAttributeList parameter of the testRequestControlled request.!,
        resultIntervalPkgPRESENT IF !the capability is implemented and the parameter is present in the
                                initAttributeList parameter of the testRequestControlled request.!,
        loopbackTimeoutPkg      PRESENT IF !loopbackTimeout is present in the test request of the
                                testRequestControlled request.!,
        loopbackThresholdPkg    PRESENT IF !loopbackThreshold is present in the initAttributeList parameter of the
                                testRequestControlled request.!;

REGISTERED AS {part14ObjectClass ??};
```

A.2.5 Protocol Integrity Test Object

```
protocolIntegrityTestObject             MANAGED OBJECT CLASS
DERIVED FROM   "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObject;
CHARACTERIZED BY
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":mORTsPackage,
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":associatedObjectsPackage,
        "Rec. X.721 (1992) | ISO/IEC 10165-2:1992":actualTestTimePackage,
        protocolIntegrityResultsPackage,
        protocolIntegrityTestPkg
            PACKAGE
                BEHAVIOUR protocolIntegrityTestBehaviour        BEHAVIOUR
                DEFINED AS "The MORT transmits Protocol Elements to an Associated object. The Associated object
                        exhibits its normal behaviour in reaction to the Protocol Elements received. The responses
                        received by the MORT may be compared to the expected responses.";;
        ATTRIBUTES
            pDUSequence         GET-REPLACE;;;
CONDITIONAL PACKAGES
        protocolIntegrityTestResultPackage PACKAGE NOTIFICATIONS
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testResultNotification protocolIntegrityControlledResultsParam;;
                                                                PRESENT IF !Unsolicited reporting is supported.!,
        timeoutPeriodPkg        PRESENT IF !The timeout period is specified in the test request.!,
        waitingIntervalPackage  PRESENT IF !The waitingInterval is specified in the protocolIntegrityTestInfo.!,
```

```
        pDUReceptionPackage     PRESENT IF !The waitingInterval is not specified in the protocolIntegrityTestInfo.!,
        testConditionsPkg       PRESENT IF !testConditions is present in the initAttributeList parameter of the
                                  testRequestControlled request.!;

REGISTERED AS {part14ObjectClass ??};
```

A.2.6 Resource Boundary Test Object

```
resourceBoundaryTestObject          MANAGED OBJECT CLASS
DERIVED FROM   "ITU-T Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObject;
CHARACTERIZED BY
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":mORTsPackage,
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":associatedObjectsPackage,
        resourceBoundaryTestResultPackage PACKAGE NOTIFICATIONS
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testResultNotification
                        resourceBoundaryControlledResultsParam;;
        resourceBoundaryTestPkg
            PACKAGE
                BEHAVIOUR resourceBoundaryTestBehaviour      BEHAVIOUR
                DEFINED AS “The mORT(s) attribute identifies the (part of a) resource(s) that will be tested. The
                        Associated Objects identify the Points of Control and Observation at which the
                        signals of the resource under test are observed and inserted. The signals that are
                        inserted or expected as a response are indicated in the sequenceOfEvents attribute.
                        The resultReportIndicator attribute indicates whether a result report is required when
                        a sequence of test events was passed. The Resource boundary test results are defined by
                        the resourceBoundaryControlledResultsParam parameter of the testResultNotification.”;;
        ATTRIBUTES
            resultReportIndicator               GET-REPLACE,
            sequenceOfEvents                        GET-REPLACE,
            resourceBoundaryTestResults         GET;;;
CONDITIONAL PACKAGES
            timeoutPeriodPkg    PRESENT IF !The timeout period is specified in the test request.!,
            testConditionsPkg   PRESENT IF !testConditions is present in the initAttributeList parameter of the
                                testRequestControlledRequest.!;

REGISTERED AS {part14ObjectClass ??};
```

A.2.7 Resource Self Test Object

```
resourceSelfTestObject   MANAGED OBJECT CLASS
DERIVED FROM   "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObject;
CHARACTERIZED BY
        resourceSelfTestPkg     PACKAGE
            BEHAVIOUR       resourceSelfTestBehaviour  BEHAVIOUR
            DEFINED AS      "Resource self tests (sometimes termed self-test) are used to investigate the ability of a
                            resource (function or physical entities) to perform its allotted function at a given time. The
                            nature of the resource(s) under test and the exercises involved is specific to the containing
                            system.";;
            ATTRIBUTES      resourceSelfTestResults;;
CONDITIONAL PACKAGES
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":mORTsPackage        PRESENT IF MORT id instance other
                                                    than object with TARR functionality.!,
        resourceSelfTestResultPackage PACKAGE NOTIFICATIONS
        "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testResultNotification resourceSelfControlledResultsParam;;
                                                    PRESENT IF !Unsolicited reporting is supported.!,
        timeoutPeriodPkg        PRESENT IF !The timeout period is specified in the test request.!,
        testConditionsPkg       PRESENT IF !testConditions is present in the initAttributeList parameter of the
                                testRequestControlled request.!;

REGISTERED AS {part14ObjectClass ??};
```

A.2.8 Test Infrastructure Test Object

```
testInfrastructureTestObject        MANAGED OBJECT CLASS
DERIVED FROM   "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObject;
CHARACTERIZED BY
        testInfrastructureTestPkg PACKAGE
            BEHAVIOUR       testInfrastructureTestBehaviour     BEHAVIOUR
            DEFINED AS      "The test infrastructure test is a ‘null test’ whose purpose is solely to investigate the ability
                            of a managed open system to behave as specified in ITU-T Rec. X.745 |
```

ISO/IEC 10164-12 – that is, to accept an incoming test request and generate appropriate responses. Not test exercises are performed on resources within or external to the managed open system. If a controlled test is requested, an instance of this class will be created which will exist for the time taken to progress through the test cases defined by ITU-T Rec. X.745 | ISO/IEC 10164-12. During this time, a test conductor may issue control messages (suspend, resume, terminate) to the instance. The time taken to progress through the states can be longer than the timeout value specified thereby deliberately causing the test outcome to be TIMEOUT and the instance to be deleted.";;;

CONDITIONAL PACKAGES

testInfrastructureTestResultPackage PACKAGE NOTIFICATIONS
"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testResultNotification testInfrastructureControlledResultsParam;;
PRESENT IF !Unsolicited reporting is supported.!,
stateTransitionTimeIntervalPkg PRESENT IF !stateTransitionTimeInterval is present in the initAttributeList parameter of the testRequestControlled request.!;

REGISTERED AS {part14ObjectClass ??};

A.3 Package definitions

connectivityThresholdPkg PACKAGE
BEHAVIOUR connectivityThresholdBehaviour BEHAVIOUR
DEFINED AS !The connectivity threshold is either defined explicitly in the test object or it is specified indirectly as an attribute in the MORT. It indicates the maximum acceptable error rate for the test. If the threshold is exceeded, the test is terminated and a fail outcome is returned.!;;
ATTRIBUTES testThreshold GET-REPLACE;

REGISTERED AS {part14Package ??};

DataIntegrityResultsPkg PACKAGE
BEHAVIOUR dataIntegrityResultsBehaviour BEHAVIOUR
DEFINED AS !This attribute contains the test results of a data integrity test.!;;
ATTRIBUTES dataIntegrityResults GET;

REGISTERED AS {part14Package ??};

dataIntegrityThresholdPkg PACKAGE
BEHAVIOUR dataIntegrityResultsBehaviour BEHAVIOUR
DEFINED AS !The testThreshold attribute identifies the maximum Timeout period for the data to be transferred to the associated object and reflected back to the MORT. If the data is returned within this period, the test will terminate indicating that the test outcome is pass. If this threshold is exceeded, it will cause the termination of the data integrity test and will indicate that the test outcome is fail.!;;
ATTRIBUTES testThreshold GET-REPLACE;

REGISTERED AS {part14Package ??};

dataUnitsPkg PACKAGE
BEHAVIOUR dataUnitsBehaviour BEHAVIOUR
DEFINED AS !The type and quantity of test data units to be sent is specified by the dataUnits attribute of the test. This attribute is only used if the test supports manager specification of the type and/or the quantity of data units.!;;
ATTRIBUTES dataUnits GET-REPLACE;

REGISTERED AS {part14Package ??};

loopbackResultsPkg PACKAGE
BEHAVIOUR loopbackResultsBehaviour BEHAVIOUR
DEFINED AS !The loopbackResults attribute contains the test results of a loopback test. The resultTestDuration must be supplied if this result is for a period shorter than the test pattern duration. The meaning of not sending resultTestDuration is that the period of time covered by this result is the whole test pattern duration. The patternType is used to notify the Test Conductor of the type of data this result applies to, and is the same value as the DataUnits attribute. The error cause should be set if the Test Performer is able to determine a reason for the error.!;;
ATTRIBUTES loopbackResults GET;

REGISTERED AS {part14Package ??};

```
loopbackThresholdPkg    PACKAGE
BEHAVIOUR loopbackThresholdBehaviour  BEHAVIOUR
                DEFINED AS      !The testThreshold attribute identifies an explicit error threshold which if crossed causes
                                the termination of the loopback test.!;;
                ATTRIBUTES        testThreshold                 GET-REPLACE;

REGISTERED AS  {part14Package 6};

loopbackTimeoutPkg      PACKAGE
BEHAVIOUR loopbackTimeoutBehaviour      BEHAVIOUR
                DEFINED AS      !The loopback timeout attribute contains the value for the timeout period which measures
                                from the start of transmission of each test pattern to the start of the receipt of the
                                corresponding pattern. When the loopback transmission delay time exceeds the
                                loopackTimeout period, the whole test is terminated and the loopback timeout period must
                                be returned as part of the monitoredAttribute parameter of the testResultNotification and
                                the test outcome is set to Fail. The loopbackTxDelayParam should be returned in all
                                testResultNotifications where the loopback delay time is less than the loopbackTimeout
                                period.!;;
                ATTRIBUTES        loopbackTimeout               GET-REPLACE;
                NOTIFICATIONS     Rec. X.745 (1993) | ISO/IEC 10164-12:1994:
                        testResultNotification loopbackTxDelayParam

REGISTERED AS  {part14Package ??};

pDUReceptionPackage     PACKAGE
BEHAVIOUR pDUReceptionBehaviour         BEHAVIOUR
                DEFINED AS      !The pDUReception attribute, if used, indicates the conditions for protocol element
                                transmission. When the protocol element received by the MORT matches the conditions
                                specified in the pDUReception, the next protocol element is transmitted. The
                                pDUReception attribute should not be used if the waitingInterval attribute is used.!;;
                ATTRIBUTES        pDUReception                  GET-REPLACE;

REGISTERED AS  {part14Package ??};

protocolIntegrityResultsPackage         PACKAGE
                BEHAVIOUR         protocolIntegrityResultsBehaviour;
                ATTRIBUTES        protocolIntegrityResults      GET;

REGISTERED AS  {part14Package ??};

resultIntervalPkg       PACKAGE
BEHAVIOUR resultIntervalBehaviour       BEHAVIOUR
                DEFINED AS      !A "zero" value means that there are no intermediate results. A value other than "zero"
                                specifies the interval (in seconds) between the generation of intermediate results.!;;
                ATTRIBUTES        resultInterval                GET-REPLACE;

REGISTERED AS  {part14Package ??};

stateTransitionTimeIntervalPkg          PACKAGE
BEHAVIOUR stateTransitionTimeIntervalBehaviour  BEHAVIOUR
                DEFINED AS      !The stateTransitionTimeInterval attribute contains the time interval to be used to
                                progress through the test states of a controlled test.!;;
                ATTRIBUTES        stateTransitionTimeInterval           GET-REPLACE;

REGISTERED AS  {part14Package ??};

testConditionsPkg       PACKAGE
BEHAVIOUR testConditionsBehaviour       BEHAVIOUR
                DEFINED AS      !The testConditions attribute specifies under what conditions the resources should be
                                allocated to the test. In particular, it indicates whether the test should initiate if the MORT
                                is busy and whether the user of the MORT can cause the test to abort.!;;
                ATTRIBUTES        testConditions                GET-REPLACE;

REGISTERED AS  {part14Package ??};

timeoutPeriodPkg        PACKAGE
BEHAVIOUR timeoutPeriodBehaviour        BEHAVIOUR
                DEFINED AS      !The timeoutPeriod attribute contains the value for the timeout period, measured from the
                                initiation of the test. When the execution of a test exceeds the timeout period, the test is
                                terminated and the timeout period is returned as part of the test result and the test
                                outcome is set to TIMEOUT.!;;
                ATTRIBUTES        timeoutPeriod                 GET-REPLACE;

REGISTERED AS  {part14Package ??};
```

```
waitingIntervalPackage   PACKAGE
        BEHAVIOUR       waitingIntervalBehaviour;
        ATTRIBUTES      waitingInterval         GET-REPLACE;

REGISTERED AS  {part14Package ??};
```

A.4 Name binding definitions

```
testObject-System-NB     NAME BINDING
        SUBORDINATE OBJECT CLASS
            "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObject AND SUBCLASSES;
        NAMED BY SUPERIOR OBJECT CLASS
            "Rec. X.721 (1992) | ISO/IEC 10165-2:1992":system AND SUBCLASSES;
        WITH ATTRIBUTE      "Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testObjectId;
        DELETE;

REGISTERED AS {part14NameBinding 1};
```

A.5 Parameter definitions

A.5.1 Connection controlled results

```
connectionControlledResultsParam        PARAMETER
        CONTEXT     EVENT-INFO;
        ATTRIBUTE   TestCategories-ASN1Module.ConnectionTestResults;
        BEHAVIOUR   connectionResultsParamBehaviour BEHAVIOUR
            DEFINED AS      !This parameter may be used to convey additional connection test result information that
                            is related to the function being tested.!;;;
```

A.5.2 Connection test info

```
connectionTestInfoParam   PARAMETER
        CONTEXT     ACTION-INFO;
        WITH SYNTAX         TestCategories-ASN.1Module.ConnectionTestInfo;
        BEHAVIOUR   connectionTestInfoParamBehaviour;
            DEFINED AS      !This parameter is used to convey additional connection test information that is related to
                            the function being tested.!;;;
```

A.5.3 Connection uncontrolled results

```
connectionUncontrolledResultsParam      PARAMETER
        CONTEXT     ACTION-REPLY;
        WITH SYNTAX         TestCategories-ASN1Module.ConnectionTestResults;
        BEHAVIOUR   connectionUncontrolledResultsParamBehaviour;

REGISTERED AS {part14Parameter ??};
```

A.5.4 Connectivity controlled results

```
connectivityControlledResultsParam      PARAMETER
        CONTEXT     EVENT-INFO;
        ATTRIBUTE   connectivityResults;
        BEHAVIOUR   connectivityResultsParamBehaviour
            DEFINED AS      !This parameter may be used to convey additional connectivity test result information that
                            is related to the function being tested.!;;;
```

A.5.5 Connectivity uncontrolled results

```
connectivityUncontrolledResultsParam    PARAMETER
        CONTEXT     ACTION-REPLY;
        WITH SYNTAX         TestCategories-ASN1Module.ConnectivityTestResults;
        BEHAVIOUR   connectivityUncontrolledResultsParamBehaviour;
            DEFINED AS      !This parameter is used to convey connection test result information that is related to the
                            function being tested.!;;;

REGISTERED AS {part14Parameter ??};
```

A.5.6 Connectivity test info

```
connectivityTestInfoParam          PARAMETER
        CONTEXT     ACTION-INFO;
        WITH SYNTAX  TestCategories-ASN1Module.ConnectivityTestInfo;
        BEHAVIOUR  connectivityTestInfoParamBehaviour;

REGISTERED AS {part14Parameter ??};
```

A.5.7 Data integrity controlled results

```
dataIntegrityControlledResultsParam          PARAMETER
        CONTEXT     EVENT-INFO;
        ATTRIBUTE   dataIntegrityResults;
        BEHAVIOUR  dataIntegrityControlledResultsParamBehaviour;;
```

A.5.8 Data integrity uncontrolled results

```
dataIntegrityUncontrolledResultsParam        PARAMETER
        CONTEXT     ACTION-REPLY;
        WITH SYNTAX  TestCategories-ASN1Module.DataIntegrityTestResults;
        BEHAVIOUR  dataIntegrityUncontrolledResultsParamBehaviour;

REGISTERED AS {part14Parameter ??};
```

A.5.9 Data integrity test info

```
dataIntegrityTestInfoParam          PARAMETER
        CONTEXT     ACTION-INFO;
        WITH SYNTAX  TestCategories-ASN1Module.DataIntegrityTestInfo;
        BEHAVIOUR  dataIntegrityTestInfoParamBehaviour;

REGISTERED AS {part14Parameter ??};
```

A.5.10 Loopback controlled results

```
loopbackControlledResultsParam     PARAMETER
        CONTEXT     EVENT-INFO;
        ATTRIBUTE   TestCategories-ASN1Module.LoopbackTestResults;
        BEHAVIOUR  loopbackControlledResultsParamBehaviour;;
```

A.5.11 Loopback uncontrolled results

```
loopbackUncontrolledResultsParam          PARAMETER
        CONTEXT     ACTION-REPLY;
        WITH SYNTAX  TestCategories-ASN1Module.LoopbackTestResults;
        BEHAVIOUR  loopbackUncontrolledResultsParamBehaviour;

REGISTERED AS {part14Parameter ??};
```

A.5.12 Loopback test info

```
loopbackTestInfoParam  PARAMETER
        CONTEXT     ACTION-INFO;
        WITH SYNTAX  TestCategories-ASN1Module.LoopbackTestInfo;
        BEHAVIOUR  loopbackTestInfoParamBehaviour;

REGISTERED AS {part14Parameter ??};
```

A.5.13 Protocol integrity controlled results

```
protocolIntegrityControlledResultsParam       PARAMETER
        CONTEXT     EVENT-INFO;
        ATTRIBUTE   TestCategories-ASN1Module.ProtocolIntegrityTestResults;
        BEHAVIOUR  protocolIntegrityControlledResultsParamBehaviour;;
```

A.5.14 Protocol integrity uncontrolled results

```
protocolIntegrityUncontrolledResultsParam    PARAMETER
        CONTEXT      ACTION-REPLY;
        WITH SYNTAX  TestCategories-ASN1Module.ProtocolIntegrityTestResults;
        BEHAVIOUR  protocolIntegrityUncontrolledResultsParamBehaviour;

REGISTERED AS {part14Parameter ??};
```

A.5.15 Protocol integrity test info

```
protocolIntegrityTestInfoParam       PARAMETER
        CONTEXT      ACTION-INFO;
        WITH SYNTAX  TestCategories-ASN1Module.ProtocolIntegrityTestInfo;
        BEHAVIOUR  protocolIntegrityTestInfoParamBehaviour;

REGISTERED AS {part14Parameter ??};
```

A.5.16 Resource boundary controlled results

```
resourceBoundaryControlledResultsParam     PARAMETER
        CONTEXT      EVENT-INFO;
        ATTRIBUTE   TestCategories-ASN1Module.ResourceBoundaryTestResults;
        BEHAVIOUR  resourceBoundaryControlledResultsParamBehaviour          BEHAVIOUR
            DEFINED AS      !The resourceBoundaryControlledResultsParam parameter contains the Resource
                            Boundary intermediate test results.!;;;
```

A.5.17 Resource boundary test info

```
resourceBoundaryTestInfoParam     PARAMETER
        CONTEXT      ACTION-INFO;
        WITH SYNTAX  TestCategories-ASN1Module.ResourceBoundaryTestInfo;
        BEHAVIOUR  resourceBoundaryTestInfoParamBehaviour       BEHAVIOUR
            DEFINED AS      !The resourceBoundaryTestInfoParam parameter contains the information of the
                            Resource Boundary test, that may be filled in for the test. This is the initial content of the
                            ResultReportIndicator and SequenceOfEvents attributes.!;;

REGISTERED AS {part14Parameter ??};
```

A.5.18 Resource self test info

```
resourceSelfTestInfoParam PARAMETER
        CONTEXT      ACTION-INFO;
        WITH SYNTAX  TestCategories-ASN1Module.ResourceSelfTestInfo;
        BEHAVIOUR  resourceSelfTestInfoParamBehaviour;
            DEFINED AS      !This parameter is used to convey additional resource self test information that is related to
                            the function being tested.!;;

REGISTERED AS {part14Parameter ??};
```

A.6 Attribute definitions

A.6.1 Connection test results

```
connectionTestResults   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.ConnectionTestResults;
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.2 Connectivity results

```
connectivityResults   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.ConnectivityTestResults;
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.3 Connectivity type

```
connectivityType ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.Type;
        MATCHES FOR EQUALITY;
        BEHAVIOUR connectivityTypeBehaviour    BEHAVIOUR
            DEFINED AS      !The connectivity attribute provides information to identify whether the test is connection
                            oriented and is tested by establishing a connection or connectionless and is tested by a
                            protocol exchange.!;;

REGISTERED AS {part14Attribute ??};
```

A.6.4 Data integrity results

```
dataIntegrityResults   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.DataIntegrityTestResults;
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.5 Data integrity type

```
dataIntegrityType   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.Type;
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.6 Data units

```
dataUnits   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.DataUnits;
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.7 End connection test results

```
endConnectionTestResults   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.EndConnectionTestResults;
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.8 Error ratio report type

```
errorRatioReportType   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.ErrorRatioReportType;
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.9 Establishment time

```
establishmentTime   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.EstablishmentTime;
        MATCHES FOR EQUALITY, ORDERING;

REGISTERED AS {part14Attribute ??};
```

A.6.10 Loopback results

```
loopbackResults   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.LoopbackTestResults;

REGISTERED AS {part14Attribute ??};
```

A.6.11 Loopback timeout

loopbackTimeout ATTRIBUTE
WITH ATTRIBUTE SYNTAX ???????;
MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};

A.6.12 Loopback type

loopbackType ATTRIBUTE
WITH ATTRIBUTE SYNTAX TestCategories-ASN1Module.LoopbackType;

REGISTERED AS {part14Attribute ??};

A.6.13 PDU sequence

pDUSequence ATTRIBUTE
WITH ATTRIBUTE SYNTAX TestCategories-ASN1Module.PDUSequence;
BEHAVIOUR pDUSequenceBehaviour BEHAVIOUR
DEFINED AS !The pDUSquence specifies a sequence of protocol elements for transmission to the associated object during the protocol integrity test.!;

REGISTERED AS {part14Attribute ??};

A.6.14 PDU reception

pDUReception ATTRIBUTE
WITH ATTRIBUTE SYNTAX TestCategories-ASN1Module.PDUReception;
BEHAVIOUR pDUReceptionBehaviour;

REGISTERED AS {part14Attribute ??};

A.6.15 Protocol integrity results

protocolIntegrityResults ATTRIBUTE
WITH ATTRIBUTE SYNTAX TestCategories-ASN1Module.ProtocolIntegrityTestResults;

REGISTERED AS {part14Attribute ??};

A.6.16 Resource Boundary Test Results

resourceBoundaryTestResults ATTRIBUTE
WITH ATTRIBUTE SYNTAX TestCategories-ASN1Module.ResourceBoundaryTestResults;
BEHAVIOUR resourceBoundaryTestResultsBehaviour BEHAVIOUR
DEFINED AS !The resource boundary test results attribute contains the intermediate results of the test. The information in the attribute depends on the event that triggers the reporting. There are four cases that trigger the reporting of intermediate test results:

- **Case 1 – Completion of last signal in sequence of test events (if indicated by the Result Report Indicator).**
- **Case 2 – Receipt of a signal at a PCO while no event sequence list is active and therefore no receive signal is to be matched.**
- **Case 3 – Receipt of a signal at a PCO while another receive signal is to be matched according to the active event sequence list.**
- **Case 4 – Wait duration timer expired in the case of a receive signal.**

For these cases, the intermediate test results are as follows:

In case 1:

- **Intermediate Resource Boundary Test outcome (passed)**

In case 2:

- **Intermediate Resource Boundary Test outcome (unexpected)**
- **Signal Type of the received signal**
- **Signal values of the signal type**
- **Identities of the MORTs from which the signal was received**
- **Identities of the AOs where the signal was received**

In case 3:

- **Intermediate Resource Boundary Test outcome (wrongSignalReceived)**
- **Event Id of failed test event**
- **Signal Type of the received signal**
- **Signal values of the signal type**
- **Identities of the AOs where the signal was received**

In case 4:

- **Intermediate Resource Boundary Test outcome (waitDurationTimerExpired).**

This attribute contains either the details of a failed test and/or the information on the received signal at a PCO. It contains information on a received signal if a signal was received that was not specified as expected in a sequence of events. It contains the details of a failure in the case of a failed test. Two kind of failures may be reported:

- **A wrong signal was received: another signal was received than the signal as specified in the sequence of events.**
- **The wait duration timer expired: an expected signal was specified, but no signal was received.**

In both cases, also the event identifier of the failed test event is reported.!;

REGISTERED AS {part14Attribute ??};

A.6.17 Resource self test results

```
resourceSelfTestResults   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX       TestCategories-ASN1Module.ResourceSelfTestResults;
```

REGISTERED AS {part14Attribute ??};

A.6.18 Result interval

```
resultInterval   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX       TestCategories-ASN1Module.ResultInterval;
        MATCHES FOR EQUALITY, ORDERING;
```

REGISTERED AS {part14Attribute ??};

A.6.19 Result Report Indicator

```
resultReportIndicator   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX       TestCategories-ASN1Module.ResultReportIndicator;
        BEHAVIOUR resultReportIndicatorBehaviour        BEHAVIOUR
            DEFINED AS      !The resultReportIndicator attribute indicates whether a result report is required when a
                            sequence of test events was passed.!;;
```

REGISTERED AS {part14Attribute ??};

A.6.20 Sequence of Events

```
sequenceOfEvents   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX       TestCategories-ASN1Module.SequenceOfEvents;
        BEHAVIOUR sequenceOfEventsBehaviour   BEHAVIOUR
            DEFINED AS      !The sequenceOfEvents attribute specifies a sequence of signals that have to be inserted or
                            received at the specified MORT and AO. In the case of a send signal, it is inserted in the
                            MORT via the specified AO. In the case of a receive signal, it is received at the indicated
                            AO from the indicated MORT. Each event is numbered by an event identifier.

                            The sequence of test events may contain a single signal that has to be inserted or received,
                            but also a sequence of events may be specified that consists of send and receive signals. The
                            test performer shall then handle the events sequentially. In the case of a receive event, the
                            test performer shall wait for the receipt of the indicated signal before continuing with the
                            next event. It waits until the wait duration timer has expired. If expired, a failure occurred.
                            In the case of a send signal, the wait duration timer specifies the time that is waited upon
                            before the signal is inserted after the former event was completed. In the case of a failure
                            the sequence of events is not completed, but is aborted.!;
```

REGISTERED AS {part14Attribute ??};

A.6.21 State transition time interval

```
statetTransitionTimeInterval   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.StatetTransitionTimeInterval;
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.22 Test conditions

```
testConditions   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.TestConditions
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.23 Test pattern

```
testPattern   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.TestPattern;

REGISTERED AS {part14Attribute ??};
```

A.6.24 Test patterns

```
testPatterns   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.TestPatternSequence;

REGISTERED AS {part14Attribute ??};
```

A.6.25 Test threshold

```
testThreshold   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.TestThreshold;
        MATCHES FOR EQUALITY;

REGISTERED AS {part14Attribute ??};
```

A.6.26 Waiting interval

```
waitingInterval   ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      TestCategories-ASN1Module.WaitingInterval;
        MATCHES FOR EQUALITY, ORDERING;

REGISTERED AS {part14Attribute ??};
```

A.7 Supporting productions

This subclause specifies the required ASN.1 value notation for the value reference used in the MANAGED OBJECT CLASS template.

TestCategories-ASN1Module { joint-iso-ccitt ms(9) function(2) part14(14) asn1Module(2) 1}

```
DEFINITION IMPLICIT TAGS

::= BEGIN
--      EXPORTS everything
        IMPORTS     AttributeId
        FROM CMIP-1 { joint-iso-ccitt ms(9) cmip(1) modules0(0) protocol(3) }
        ManagementExtension, StartTime, StopTime
        FROM        Attribute-ASN1Module { joint-iso-ccitt ms(9) smi(3) part2(2) asn1Module(2) 1 }
        TimeoutPeriod, TestOutcome, Timespec, MORTs, AssociatedObjects
        FROM Test-ASN1Module { joint-iso-ccitt ms(9) function(2) part12(12) asn1Module(2) 0 };

part14ObjectClass       OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part14(14) managedObjectClass(3) }
part14Package           OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part14(14) package(4) }
part14Parameter         OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part14(14) parameter(5) }
part14NameBinding       OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part14(14) nameBinding(6) }
part14Attribute         OBJECT IDENTIFIER ::= { joint-iso-ccitt ms(9) function(2) part14(14) attribute(7) }
```

```
ConnectionTestInfo ::= SEQUENCE {
        testPattern [0] TestPattern OPTIONAL,  -- test pattern sent
        testDirection [1] TestDirection OPTIONAL,
        testDuration [2] TestDuration OPTIONAL,
        reportingInterval [3] TimeSpec OPTIONAL }

ConnectionTestResults ::= SEQUENCE {
        receivedTestPattern [0] TestPattern OPTIONAL,
        errorRatio [1] Real OPTIONAL,
        testDirection [2] TestDirection OPTIONAL,
        testDurationMade [3] Timespec OPTIONAL }

ConnectivityTestInfo ::= SEQUENCE {
        timeoutPeriod  [0] CHOICE {
            timeUnits Timespec,
            attributeId AttributeId },
        effectiveTime   [1] Timespec }

ConnectivityTestResults ::= CHOICE {
        establishmentTime          [0] TimeSpec,
        timeoutPeriod     [1] TimeoutPeriod,
        testThreshold     [2] TestThreshold,
        specificError     [3] OBJECT IDENTIFIER }

ConnectivityThreshold  ::= CHOICE {
        time [0]    SEQUENCE {
        unitsType       UnitsType,
        unitsTotal      UnitsTotal },
        rawData [1]     RawData }

ContCond  ::= CHOICE {
        intervalTime      Timespec,
        pDUReception      PDUReception }

DataCategory ::= INTEGER {
        bits        (0),
        octets      (1),
        blocks      (2),
        packets     (3)
        -- ... }

DataIntegrityTestInfo ::= SEQUENCE  OF {
        dataUnits DataUnits }

DataIntegrityTestResults ::= CHOICE {
        testResult          [0] SEQUENCE {
                establishmentTime       [1] EstablishmentTime OPTIONAL,
                originalData                    [2] DataUnits,
                corruptedData           [3] DataUnits OPTIONAL },
        timeoutPeriod     [4] TimeoutPeriod,
        testThreshold     [5] TestThreshold
        }

DataRate ::= CHOICE { REAL, OBJECT IDENTIFIER }

DataSize ::= INTEGER

DataType ::= CHOICE {
        integerDataType                 IntegerDataType,
        objectIdentifierDataType        OBJECT IDENTIFER }

DataUnits ::= SEQUENCE {
        dataType        [0] DataType,
        dataCategory    [1] DataCategory    OPTIONAL,
        dataSize        [2] DataSize        OPTIONAL,
        dataNumber      [3] UnitsTotal      OPTIONAL,
        dataRate        [4] DataRate        OPTIONAL }
```

```
ErrorRatioReportType ::= ENUMERATED {
        errorBitNumber (0),
        percentErrorSecond (1) }

ErrorUnitThreshold ::= SEQUENCE {
        unitType        [0] UnitType,
        unitsTotal          [1] UnitsTotal }

EstablishmentTime ::= Timespec

FailedCase ::= ENUMERATED {
        lineDisconnected(0),
        counterOverflow(1) }                -- error counter overflow

IntegerDataType ::= INTEGER {
        allBitOn(0),
        allBitOff(1),
        incrementNumber(2),
        pn11(3),   -- CCITT 0.152
        pn15(4),   -- CCITT 0.151
        pn20(5)
        -- ...}

IntermediateResourceBoundaryTestOutcome ::= ENUMERATED {
        passed                          (0),
        unexpected                      (1),
        wrongSignalReceived                 (2),
        waitDurationTimerExpired        (3) }

IntermediateResponse ::= ENUMERATED {
        inProgress                                                      (0),
        delayedMeasurement                                                  (1),
        interruptedMeasurement                                          (2),
        repeatLater                                                     (3),
        noAcknowledgement (4) } -- abnormal condition, a correct response cannot be sent

InternalResourceTestResults ::= SEQUENCE {
        functionTested                      OBJECT IDENTIFIER,
        testResult                      ANY DEFINED BY functionTested }

LoopbackError ::= CHOICE { errorBitNo     INTEGER, percent          REAL }

LoopbackTestInfo ::= SEQUENCE OF {
        loopbackData  TestSignalSequence     OPTIONAL,
        testStartTime  GeneralizedTime       OPTIONAL,
        testIntervalTime                     TimeSpec       OPTIONAL,
        reportingIntervalTime                TimeSpec       OPTIONAL,
        loopbackType  OBJECT IDENTIFIER OPTIONAL,
        loopbackTimeout                      TimeSpec       OPTIONAL,
        loopbackErrorThreshold   LoopbackError    OPTIONAL }

LoopbackTestResult ::= CHOICE {
        passed     SEQUENCE
                   { loopbackDataReceived  TestSignalSequence,
                     loopbackErrorReceived  LoopbackError },
        fail   INTEGER (errorRatioThresholdExceeded (0), loopbackTimeoutExceeded (1)),
        timeout NULL, -- test timeout period exceeded --
        prematureTermination NULL,
        inconclusive     NULL }

LoopbackType ::= OBJECT IDENTIFIER

Parameter ::= SEQUENCE {
        attributeType  AttributeId,
        value              ANY DEFINED BY attributeType }

Pattern Type ::= DataType
```

```
PDUReception ::= SEQUENCE {
        pDUType         PDUType,
        parameter       Parameter,
        responseTimeout ResponseTimeout  OPTIONAL }

PDUSequence ::= SEQUENCE {
        pDUType      PDUType,
        parameter    Parameter,
        contCond     ContCond,
        waitDuration WaitDuration        OPTIONAL}

PDUType ::= OBJECT IDENTIFIER

ProtocolIntegrityTestInfo ::= SEQUENCE {    -- This ProtocolIntegrityTestInfo is optional.
        pDUSequence      [0] PDUSequence,
        pDUReception     [1] PDUReception OPTIONAL,    -- one, and only one, of the pDUReception
        waitingInterval  [2] WaitingInterval OPTIONAL, -- and the waitingInterval shall be present
        startTime        [3] StartTime OPTIONAL,
        stopTime         [4] StopTime OPTIONAL }

ProtocolIntegrityTestResults ::= INTEGER {
        wrongPDUResponse                (0),
        pDUResponseNotReceived (1) }

ResourceBoundaryTestInfo ::=  SEQUENCE {
        resultReportIndicator           [0] IMPLICIT ResultReportIndicator OPTIONAL,
        sequenceOfEvents                [1] IMPLICIT SequenceOfEvents OPTIONAL }

ResourceBoundaryTestResults ::= SEQUENCE {
        signalReceived                                  [0] IMPLICIT SignalReceived OPTIONAL,
        intermediateResourceBoundaryTestOutcome         [1] IMPLICIT IntermediateResourceBoundaryTestOutcome
                                                            OPTIONAL,
        eventId                                         [2] IMPLICIT INTEGER OPTIONAL }

ResourceSelfTestInfo ::= SEQUENCE {
        diagnosticType [0] INTEGER OPTIONAL, --type of diagnostic to be run
        phases SET OF INTEGER, -- phases to be run
        iteration [1] INTEGER  -- number of iterations for each type
        timeoutPeriod [2] TimeoutPeriod }

ResourceSelfTestResults ::= CHOICE {

intermediateResponse        IntermediateResponse,
        SEQUENCE {
                phases SET OF INTEGER,
                iteration  [0] INTEGER,
                timeoutPeriod [1] TimeoutPeriod,
                finalResponse  PrintableString OPTIONAL } }

ResultInterval ::= INTEGER

ResultReportIndicator ::= ENUMERATED {
        resultReportForPassedSequences          (0),
        noResultReportForPassedSequences        (1) }

SequenceOfEvents ::= SET OF SEQUENCE {
        eventId             INTEGER,
        signalType          OBJECT IDENTIFIER,
        signalValue         ANY DEFINED BY signalType,
        signalDirection     SignalDirection,
        mORTs               MORTs,
        associatedObjects           AssociatedObjects,
        waitDuration        WaitDuration }

SignalDirection ::= ENUMERATED {
        send        (0),
        receive     (1) }
```

```
SignalReceived  ::= SET OF SEQUENCE {
        signalType          OBJECT IDENTIFIER,
        signalValue         ANY DEFINED BY signalType,
        mORTs               MORTs,
        associatedObjects           AssociatedObjects }

StateTransitionTimeInterval ::= Timespec

TerminationReason ::= OBJECT IDENTIFIER

TestConditions ::= SEQUENCE {
        INTEGER {
                testIfBusy          (0),
                rejectIfBusy        (1),
                waitIfBusy          (2) },
        INTEGER {
                customerOverrideTest        (0),
                noCustomerOverrideTest      (1) }   OPTIONAL }

TestDirection ::= INTEGER { atoZ (0), ztoA (1), bothDirections (2), transmitFromNearEnd (3), receiveFromNearEnd (4) }

TestDuration ::= CHOICE {
                        signalDuration      Timespec,
                        signalLength        SEQUENCE {
                                                size        INTEGER
                                                dataRate    dataRate OPTIONAL

TestPatternSequence ::= SEQUENCE OF { TestSignal }-- There should be some size limitation here
                                                   -- if test signal pattern is to fit in one pdu!!!!!

TestPattern ::= CHOICE {
        rawData  OCTET STRING,
        standardType        DataType,
        undefinedType       ManagementExtension }

TestThreshold  ::= CHOICE {
        bitErrorThreshold                       [0]     INTEGER,
        percentageErrorThreshold        [1]     REAL,
        errorUnitThreshold                      [2]     UnitErrorThreshold }

Type ::= INTEGER {
        connectionOriented          (0),
        connectionless      (1) }

UnitsTotal ::= INTEGER

UnitType ::= OBJECT IDENTIFIER

WaitDuration ::= Timespec

WaitingInterval ::= Timespec

END
```

Annex B

MCS proforma

(This proforma will be published as an amendment to this Recommendation | International Standard)

Annex C

MICS proforma

(This proforma will be published as an amendment to this Recommendation | International Standard)

Annex D

MOCS proforma

(This proforma will be published as an amendment to this Recommendation | International Standard)

Annex E

MRCS proforma

(This proforma will be published as an amendment to this Recommendation | International Standard)

Annex F

Summary of test categories

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

This annex provides, in the form of a matrix, a summary of all the test categories, identifying for each test category the TO attributes used, the use made of Test Info parameters and the possible Test Results.

Test category	Test category description	TO attributes	Test info	Test results
Connection	Ability of a comm path (real or virtual – MORT) to support a desired service with level of functionality, test = send test pattern over service in one direction	MORTs, AOs, testPatterns, connTestResults (Null for pass or testPattern and specError for fail), adminSt, timeoutPeriod, (test patterns twice, once as attribute, once in results)	–	Null for pass or testPattern and specError for fail (doesn't indicate if test pattern sent or received)
Connectivity	Verifies that connectivity may be established between two entities (MORT and AO) within specified time: for CO – establish conn, CL – exchange data units	MORTs, AOs, connectivityType, connectivityResults, establishment time (2X), timeout period	timeoutPeriod (timeUnits or attId), effectiveTime	CHOICE: estabTime, timeout Period, testThreshold, specificError
Data integrity	Whether two entities (MORT and AO) can exchange data without any corruption and measure time taken for exchange (test = sendPDU, get it back)	MORTs, AOs, actualTestTimePkg, dataIntegrityType, test patterns, timeoutPeriod, dataIntegrityResults, test threshold	DataUnits	CHOICE: testResult (estabTime, origData, corruptData), timeout Period, testThreshold (similar to test results for CY)
Loopback	Verifies that data can be sent and received, over a comm path within specific interval of time with acceptable rate (MORT is comm path – includes end pts) (test = send test pattern)	MORTs, AOs, loopbackType, testPatterns, dataUnit(s), errorRatioReportType, timeoutPeriod, testConditions, dataUnits, resultInterval, loopbackResults, loopbackThreshold	loopbackData, testStartTime, testTimeDuration, errorThresh, loopbackReportInterval, loopbackTestResultType (errorRatioReportType)	testTimeDuration, dataUnits, errorRatio, errorCause
Protocol integrity	Determines if MORT can conduct proper protocol interactions with specified AO	actualTesttime (actualStartTime, actualStopTime), PDU sequence, timeoutPeriod, PIResults, waitingInterval, pdu reception, test conditions	pduSeq, pduRec, waitingInterval, startTime, stopTime	wrongPDU response, pudResponse Not Received
Resource boundary	Multi-resource test using disconnected info streams; Verifies correctness of resources (MORT) internal to system that have points of control and observation (AOs)	AOs, resultReportIndicator, seqOfEvents (includes signal type (ANY)), RBResults, timeoutPeriod, testConditions	resultReportIndicator, seqOfEvents (signal type)	signalReceived, intermRBTestOutcome, eventId
Resource self	Ability of a resource (MORT, physical or logical function) to perform its allotted function	MORTs, timeoutPeriod, testConditions	–	–
Test infra-structure	Testing the test system; (same as testing TARR functionality)	–	–	–

Annex G

Example TARR Package (for Connection Test)

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

This annex provides an example of a TARR package for the connection test. A managed object containing such a package would support receiving a connection test request in the form of either a testRequestControlledAction action or a testRequestUncontrolledAction action with the appropriate parameters defined for connection test. A managed object which supports such a package is said to have “TARR functionality”.

connectionTestTARRPackage PACKAGE

BEHAVIOUR "A managed object containing this package supports receiving a connection test request in the form of either a testRequestControlledAction action or a testRequestUncontrolledAction action with the appropriate parameters defined for connection test. It will return the results of the test in a testResultNofication with the connectionControlledResultsParam for a controlled test and in the action reply for the testRequestUncontrolledAction with the connectionUncontrolledResultsParam.";

ACTIONS

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testRequestControlledAction
connectionTestInfoParam

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":associatedObjectNotAvailable

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":independentTestInvocationError

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994"mistypedTestCategoryId

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994"mistypedTestRequestInformation

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":mORTNotAvailable

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":noSuchAssociatedObject

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":noSuchMORT

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":relatedTOError,

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testRequestUncontrolledAction
connectionTestInfoParam

connectionUncontrolledResultsParam

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":associatedObjectNotAvailable

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994"mistypedTestCategoryId

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994"mistypedTestRequestInformation

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":mORTNotAvailable

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":noSuchAssociatedObject

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":noSuchMORT,

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testSuspendResumeAction

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":noSuchTestInvocationId

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":noSuchTestSessionId

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":invalidTestOperation

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testSuspendResumeError,

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testTerminateAction

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":noSuchTestInvocationId

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":noSuchTestSessionId

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":invalidTestOperation

"Rec. X.745 (1993) | ISO/IEC 10164-12:1994":testTerminateError;

REGISTERED AS {...};

Annex H

Example of use of resource boundary test category

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

H.1 The resource boundary test provides the user with a means to test parts of a system separately as if they were not encapsulated in the system. This type of test is very suitable for resources that have a complex behaviour that cannot (completely) be tested by observing external communication interfaces.

This test category is suitable for several applications. Two typical applications, each described in more detail below, might be the verifcation of protocol implementations and the boundary scan of hardware.

H.1.1 Verification of protocol implementations

Efficient testing of protocols highly depends on the accessibility of the protocol. When using the resource boundary test, the protocol layers can be tested independently from each other. This is defined as the conceptual testing architecture in ISO 9646-1. Figure H.1 shows the applied configuration.

The resources of the test category represent protocol layers, the PCOs represent the Service Access Points (SAPs) between the layers, and the signals represent the Protocol Data Units (PDUs) and Service Primitives (SP) that are exchanged between protocol layers.

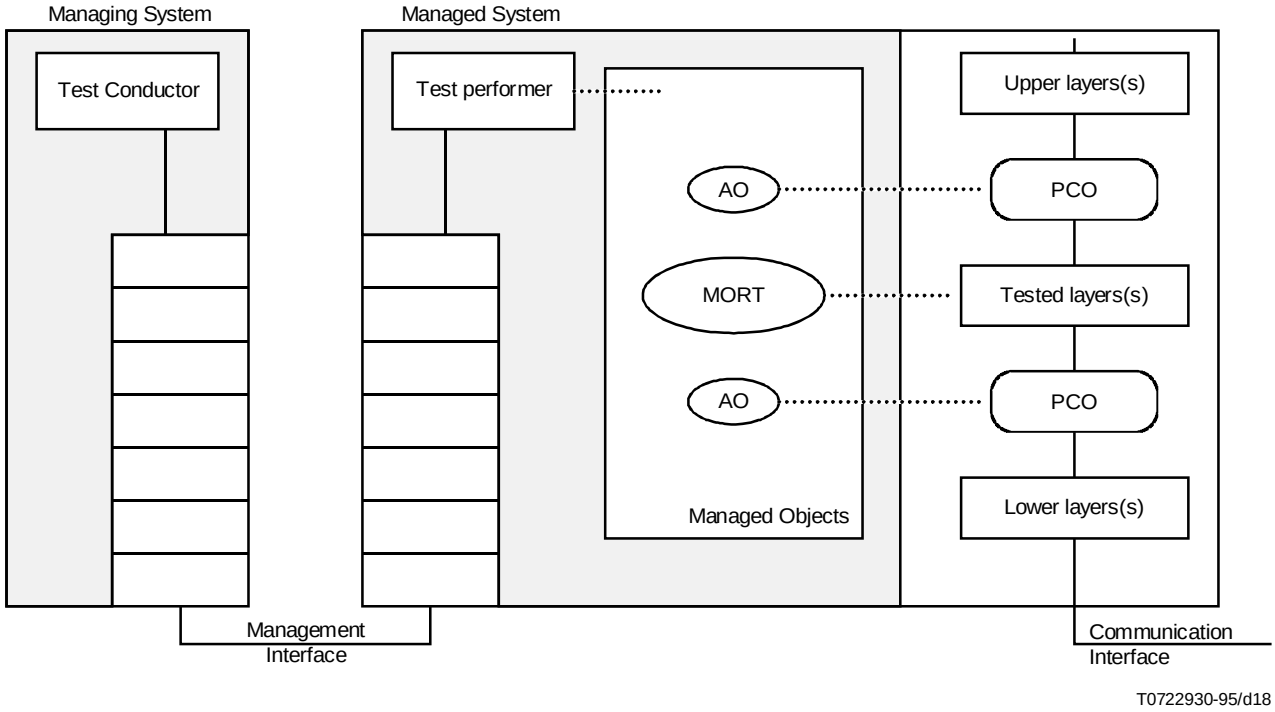

Figure H.1 – Testing protocol implementations

H.1.2 Controlling boundary scan tests in hardware

Digital equipment that contains facilities for boundary scan testing of integrated circuits can be controlled from a management system as defined in IEEE 1149.1. Figure H.2 shows the applied configuration.

The resources of the test category represent integrated circuits, the PCOs represent the boundary scan cells, and the signals represent the test vectors.

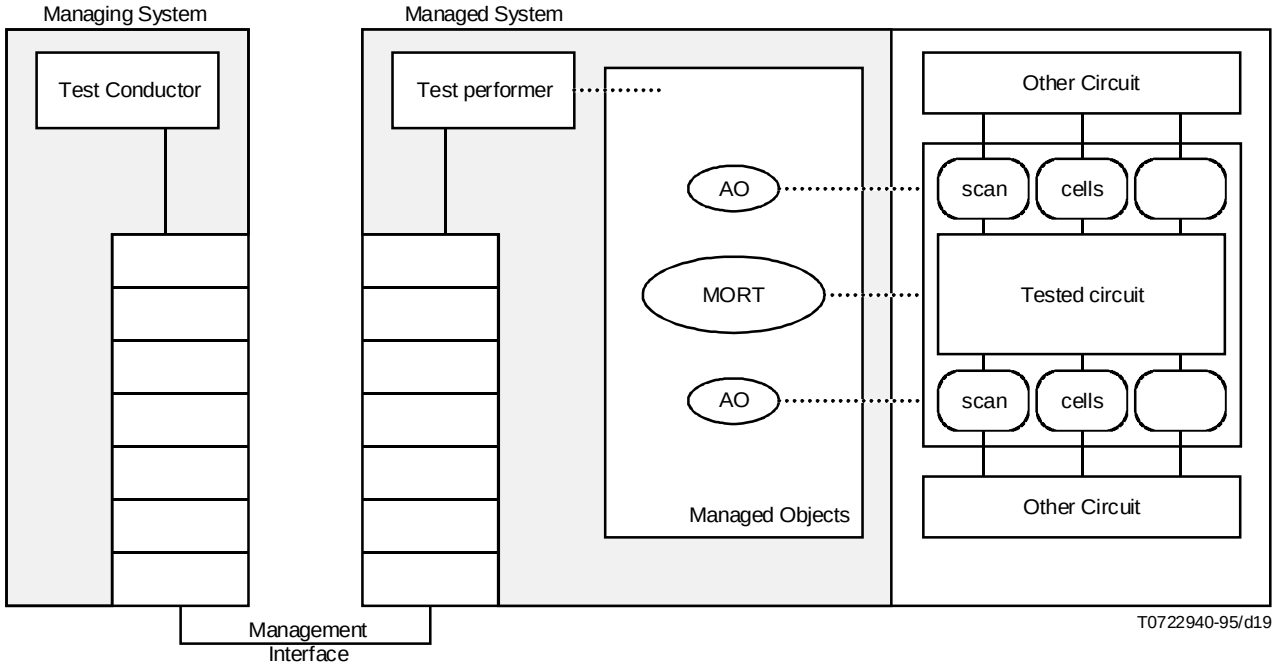

Figure H.2 – Hardware boundary scan tests

Annex I

Example test process for managed communications network

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

This annex illustrates how the test categories defined in this Recommendation | International Standard might be applied to the problem of testing a network of managed communication systems.

Figure I.1 shows an example network. This network contains:

- A test conductor (a part of the managing open system) that requests tests and receives result reports according to ITU-T Rec. X.745 | ISO/IEC 10164-12. Note that result reports may be generated by any open system participating in the test.
- A number of managed open systems, each containing a test performer that receives test requests and initiates the internal operations required to respond.
- Connections that carry information between managed open systems.
- Connections between the open systems that contain test performers and open or other systems not containing test performers.

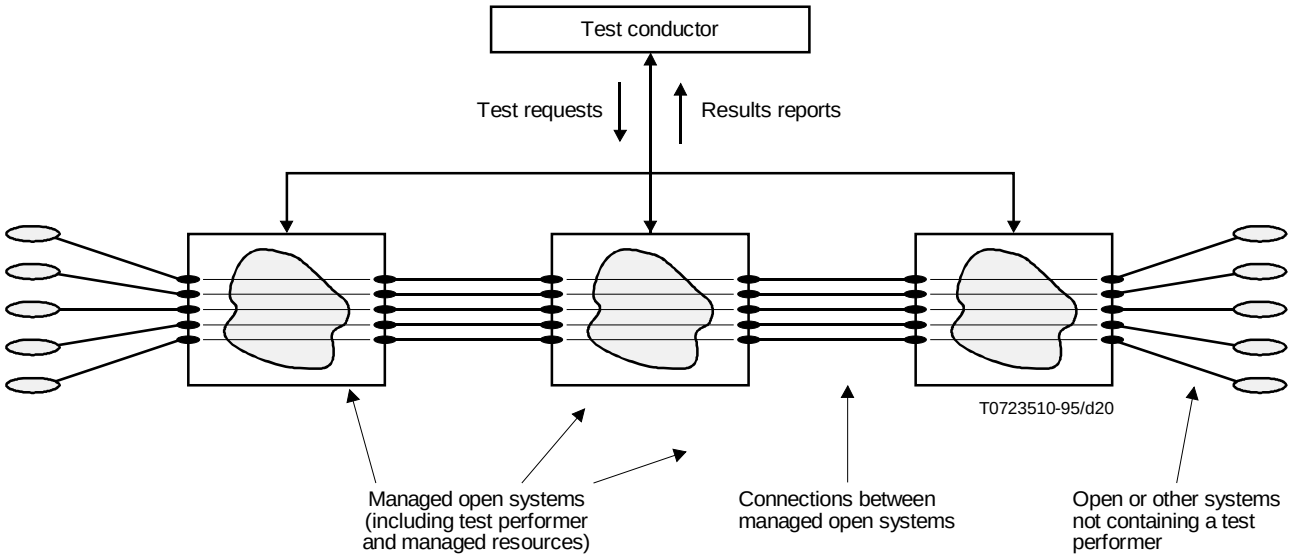

Figure I.1 – Network of managed communication systems

With reference to this network model, a generic test process can be defined that has five basic parts. However, an actual test process may not have all the five parts (e.g. it may apply only the internal resource test).

1) *Testing the Interface to Resources in the Network*

   First, the test conductor must prove both the integrity of the paths along which test requests and result reports will pass and the ability of the test performers in the managed open systems to receive and respond to test requests and (where appropriate) controls. This may be achieved using the **Test Infrastructure Test** defined in this Recommendation | International Standard.

2) *Testing the Resources in the Network*

   Having verified the test infrastructure, this can now be used to initiate and observe tests. **Resource Self-Tests** and **Resource Boundary Tests** permit the operability of resources (functional or physical components within a managed open system) to be investigated. The results of these tests will provide an indication of the health of the resource and, where appropriate, diagnostic information.

Note that use of the **Resource Self-Test** requires only limited user participation, because the details of the tests to be performed are built into the resource undergoing test (the MORT). Use of the **Resource Boundary Test** requires the user to specify the test patterns or exercises to be applied and, optionally, the results that are expected. It is possible, therefore, that **Resource Boundary Tests** may be used following detection of a fault by a **Resource Self-Test,** to obtain further diagnostic information.

3) *Testing Connections Between Resources*

Presuming that the various managed resources are fault-free (or have been proven to be fault-free through execution of internal resource tests), they may now be used to apply tests to the various system-to-system connections within the network. This may be achieved by use of the following tests defined in this Recommendation | International Standard:

- Connection Test;
- Connectivity Test;
- Data Integrity Test;
- Loopback Test;
- Protocol Integrity Test.

The **Connection Test** is comparable to the **Resource Self-Test** in that the nature of the test performed is determined by the type of the MORT and the test patterns or exercises are "built into" the AOs involved in the test. Therefore, only limited user participation is required. To ensure that connected managed resources can cooperate in the performance of such tests, agreement is required on the detail of the tests to be performed. For example, the precise sequence of instructions or data transmitted from the sending resource must be known in advance by the receiving resource, such that a pass or fail result may be generated. The detail of point-to-point connection tests will be specific to the type of connection concerned.

4) *Testing Connections that Enter or Leave the Network*

Tests may also be performed of the extremities of the managed communications network – for example, of connections to or from terminal equipment such as telephones and modems.

The **Connection Test** is an example of a test that might be used to meet this objective. This tests is performed by a single managed resource (the AO) and will require, at most, only limited participation of the "far end" terminal equipment. For example, it may be necessary to request the user to disconnect the terminal equipment or to manually place a loopback condition before testing can start. Alternatively, the managed resource that is performing the test may control the establishment of a loopback condition by sending a defined tone or instruction.

Alternatively, the **Loopback Test** may be used.

5) *Testing Overall Network Integrity*

So far, only the individual parts of the network have been tested – that is, the component resources and their interconnections. It is possible that, while the parts may work when tested separately, the whole network may not perform as required (e.g. due to performance mismatches, timing problems, etc.). The overall capability of the network to provide the required service must therefore be confirmed.

This can be achieved by performing a "service-integrity" test, which can be considered to be a **Resource Self-Test** applied to the complete managed network (i.e. the network is considered at a higher level of abstraction to be a single managed open system). The nature of the tests performed will be specific to the network design and must therefore be developed as a part of the task of designing the network.

The overall test process is illustrated in Figure I.2.

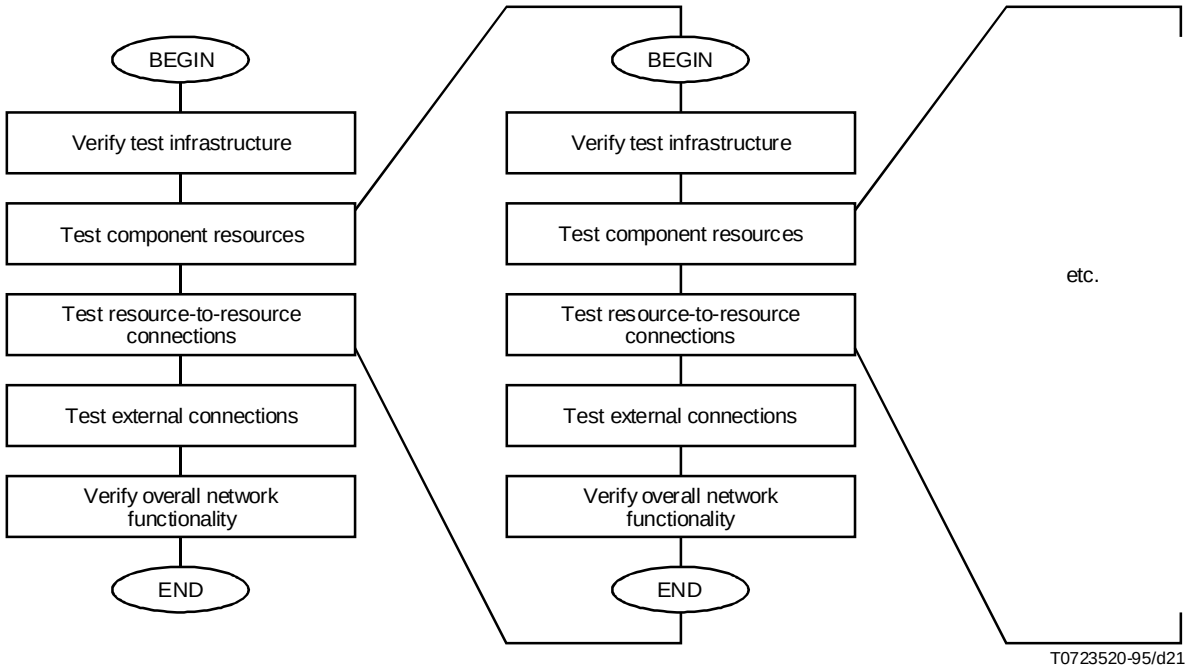

Figure I.2 – Generic test process

Note that this process is hierarchic. A managed open system may itself be a network of subordinate managed open systems, in which case it can respond to a request to perform an internal resource or connection test by initiating a "child" test process of the form described above.

Note also that, while this process is independent of the functions and implementation of the managed open systems, it is able to prove the network's operability or to identify the location of a fault to a particular managed open system or to a particular connection. Therefore, the process is sufficient to perform initial diagnosis of faults in a multi-vendor network.