INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

X.291

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(04/95)

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS – OPEN SYSTEMS INTERCONNECTION – CONFORMANCE TESTING

OSI CONFORMANCE TESTING METHODOLOGY AND FRAMEWORK FOR PROTOCOL RECOMMENDATIONS FOR ITU-T APPLICATIONS – ABSTRACT TEST SUITE SPECIFICATION

ITU-T Recommendation X.291

(Previously "CCITT Recommendation")

FOREWORD

The ITU-T (Telecommunication Standardization Sector) is a permanent organ of the International Telecommunication Union (ITU). The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1 (Helsinki, March 1-12, 1993).

ITU-T Recommendation X.291 was revised by ITU-T Study Group 7 (1993-1996) and was approved under the WTSC Resolution No. 1 procedure on the 10th of April 1995.

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

SUMMARY

This Recommendation provides a common approach to the specification of OSI conformance test suites. It describes the process of developing an Abstract Test Suite (ATS), including the design criteria to be used and guidance on its structure and coverage. The text was developed jointly with ISO/IEC JTC1 and the main intent of this revision is to reflect the changes as a result of the work on Protocol Profile Testing Methodology (PPTM) and on Multi-Party Testing Methodology (MPyTM).

INTRODUCTION

This Recommendation provides a common approach to the specification of OSI conformance test suites at a level which is independent of the means of executing those test suites (hereafter called “Abstract Test Suites”). This level of abstraction is suitable for standardization and facilitates the comparison of results produced by different organizations which run the corresponding Executable Test Suites.

Clauses 6 and 7 recall that there are requirements on OSI protocol specifiers which have to be fulfilled before there can be an objective basis for the process of developing an Abstract Test Suite. The need is expressed for consistent conformance clauses and for ICS proformas in relevant base specifications (e.g. ITU-T Recommendations or International Standards which specify OSI protocol standards).

Clauses 8 to 14 describe the process of developing an Abstract Test Suite, including the design criteria to be used and guidance on its structure and coverage. The possible Abstract Test Methods are defined and guidance is given to help the test suite specifier to decide which test method(s) to use in the production of a particular test suite. Requirements and guidance are given on the specification of abstract test cases. These include the subdivision of test cases into test steps and the assignment of test verdicts to test outcomes.

The test suite specifier is also required to provide information to the test realizers (e.g. limitations governing test case selection).

Finally, in clause 15, guidance and requirements are given on test suite maintenance.

This Recommendation is also published as ISO/IEC 9646-2:1994.

Recommendation X.291

OSI CONFORMANCE TESTING METHODOLOGY AND FRAMEWORK FOR PROTOCOL RECOMMENDATIONS FOR ITU-T APPLICATIONS – ABSTRACT TEST SUITE SPECIFICATION[1)]

(Geneva, 1992: revised in 1995)

1 Scope

1.1 This Recommendation specifies the requirements and gives guidance for the production of system-independent conformance test suites for one or more OSI specifications. In particular, it is applicable to the production of all OSI conformance testing specifications including all draft versions of such conformance testing specifications.

1.2 This Recommendation is applicable to the production of abstract test cases which check the conformance of an implementation to the relevant static and/or dynamic conformance requirements by controlling and observing protocol behaviour. The Abstract Test Methods included in this Recommendation are, in fact, capable of being used to specify any test case which can be expressed abstractly in terms of control and observation of Protocol Data Units (PDUs) and Abstract Service Primitives (ASPs). Nevertheless, for some protocols, test cases may be needed which cannot be expressed in these terms. The specification of such test cases is outside the scope of this Recommendation, although the test cases may themselves need to be included in a conformance testing specification.

NOTE – For example, some static conformance requirements related to an Application service may require testing techniques which are specific to that particular Application.

This Recommendation is applicable to the production of test suites for testing implementations of one or more adjacent protocols, whether or not these are embedded under other protocols.

1.3 The following are outside the scope of this Recommendation:

a) the relationship between Abstract Test Suite (ATS) specification and Formal Description Techniques;

b) testing by means of test methods which are specific to particular applications, protocols or systems, including testing by means other than PDU exchange.

NOTE – This Recommendation applies fully to some but not all Physical layer protocols. Nevertheless, many of the concepts apply to all protocols.

2 References

The following Recommendations, and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision: all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of currently valid ITU-T Recommendations is regularly published.

– 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.210 (1993) | ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Conventions for the definition of OSI services*.

---

1) Recommendation X.291 and ISO/IEC 9646-2, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 2: Abstract test suite specification*, are technically aligned.

– CCITT Recommendation X.209 (1988), *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*.

  ISO/IEC 8825:1990, *Information technology – Open Systems Interconnection – Specification of Basic Encoding Rules for 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*.

– CCITT Recommendation X.292 (1992), *OSI conformance testing methodology and framework for protocol Recommendations for CCITT applications – The Tree and Tabular Combined Notation (TTCN)*.

  ISO/IEC 9646-3:1992, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 3: The Tree and Tabular Combined Notation (TTCN)*.

  ISO/IEC 9646-3:1992 Amd 1[2], *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 3: The Tree and Tabular Combined Notation (TTCN) – Amendment 1: TTCN extensions*.

– ITU-T Recommendation X.293 (1995), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – Test realization*.

  ISO/IEC 9646-4:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 4: Test realization*.

– ITU-T Recommendation X.295 (1995), *OSI Conformance testing methodology and framework for protocol Recommendations for ITU-T applications – Protocol profile test specification*.

  ISO/IEC 9646-6:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 6: Protocol profile test specification*.

– ITU-T Recommendation X.296[3], *OSI conformance testing methodology and framework for Protocol Recommendations for ITU-T applications – Implementation conformance statements*.

  ISO/IEC 9646-7:1995, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 7: Implementation conformance statement*.

3 Definitions

For the purposes of this Recommendation, all the definitions given in Recommendation X.290 apply.

4 Abbreviations

For the purposes of this Recommendation, the abbreviations given in Recommendation X.290 apply. The following abbreviations also apply to this Recommendation.

ACSE	Association Control Service Element
ASE	Application Service Element
ASN.1	Abstract Syntax Notation One
ASP	Abstract Service Primitive
ATM	Abstract Test Method

2) To be published.

3) Presently at the stage of draft.

ATS	Abstract Test Suite
ICS	Implementation Conformance Statement
IUT	Implementation under Test
IXIT	Implementation extra Information for Testing
LT	Lower Tester
LTCF	Lower Tester Control Function
MPyT	Multi-Party Testing
OSI	Open Systems Interconnection
PCO	Point of Control and Observation
PCTR	Protocol Conformance Test Report
PDU	Protocol Data Unit
RTS	Remote Transfer Service
SAP	Service Access Point
SPyT	Single Party Testing
SUT	System under Test
TCP	Test Coordination Procedures
TMP	Test Management Protocol
TSS&TP	Test Suite Structure and Test Purposes
TTCN	Tree and Tabular Combined Notation
UT	Upper Tester

5 Compliance

5.1 A base specification which complies with this Recommendation shall satisfy all the requirements stated in clauses 6 and 7.

NOTE – Such compliance is a precondition for the base specification to be an effective basis for conformance testing of implementations.

5.2 An Abstract Test Suite (ATS) specification which complies with this Recommendation,

a) shall be a conformance test suite;

b) shall be specified in a test notation standardized by ITU-T or ISO/IEC;

c) shall satisfy all the requirements stated in clauses 9 to 15 inclusive;

d) shall be within an ITU-T or ISO/IEC published specification or, in the absence of such a specification, shall be within a publicly available specification which is in the process of being standardized within ITU-T or ISO/IEC, which has the highest standardization status available, and which has the status of at least a Committee Draft or equivalent.

NOTE – ATSs outside the standardization process need to be submitted for international standardization before they can fully comply with this Recommendation, in order to ensure that they are subject to public scrutiny, correction and acceptance, internationally.

5.3 It is recommended that the test notation used be the Tree and Tabular Combined Notation (TTCN). If TTCN is used, the ATS shall comply with Recommendation X.292.

6 Conformance requirements in OSI base specifications

6.1 Introduction

The meaning of conformance in OSI is discussed in Recommendation X.290. It is necessary that there be an unambiguous and objective understanding of the conformance requirements of an OSI base specification, as a prerequisite to the production of an ATS for that specification. Clauses 6 and 7 state the requirements on the relevant specifiers to ensure that there is such an understanding of the conformance requirements.

Additional guidance is given in Appendix II.

6.2 General requirements

6.2.1 A clear distinction shall be made between static and dynamic conformance requirements. To avoid ambiguity, they should be stated separately from one another.

6.2.2 It shall be clear what conformance to the specification means, in the sense of what is required to be done (i.e. mandatory), what is permitted but not mandatory (i.e. optional), and what is required not be done (i.e. prohibited), in order to conform to it.

6.2.3 It shall always be decidable whether an instance of communication conforms dynamically or not.

For example, it should be possible to look at a record of Protocol Data Unit (PDU) activity and decide whether or not it is valid with respect to the relevant specification.

6.3 Conformance clauses

6.3.1 Each base specification, which specifies an OSI protocol, abstract syntax, encoding rules, or information object, shall include a conformance clause, which shall be expressed clearly and unambiguously.

6.3.2 Conformance clauses shall distinguish between the following categories of information:

a) references to clauses which state dynamic conformance requirements;

b) static conformance requirements concerning the implementation of the base specification itself;

c) static conformance requirements concerning multi-specification dependencies (see 6.4).

6.3.3 The requirement to produce an Implementation Conformance Statement (ICS), in compliance with the ICS proforma, shall be stated separately from the requirements on the implementation of the specification itself.

6.3.4 The conformance clause of a protocol specification should also include

a) the requirement to be able to accept all correct sequences of PDUs received from peers, and respond with correct PDU sequences;

b) the requirement to be able to respond correctly to all incorrect sequences of PDUs received;

c) in connection oriented protocols, the option to support either the initiation of a connection or the acceptance of a connection, or both;

d) in connectionless protocols, the option to support the transmission of a PDU, the receipt of a PDU, or both.

6.3.5 A checklist for what should be included or referenced in each conformance clause is given in II.5.

6.4 Multi-specification dependencies

Multi-specification dependencies may be specified by each base specification requiring the provision of non-mandatory features in one or more underlying base specifications. If multi-specification dependencies are to be included in an ICS proforma, the ICS proforma shall merely reflect the multi-specification dependencies specified in the conformance clause of the corresponding base specification.

Multi-specification dependencies should usually be specified in terms of what elements of a given underlying service are required in order to support the given protocol or information object. In addition, each underlying protocol specification should specify which units of the protocol are required if a given element of service can be said to be supported. This refers to the functionality implied by the element of service, and does not in any way imply the existence of a service interface.

NOTE – This is not conformance to service, but rather is an expression of the conditional requirements that result from the compliance of a protocol to its service definition.

In cases where it is not possible to express dependencies through the underlying service, they can be specified in terms of the units of the underlying protocol or other specification required to support the higher protocol (the referencing specification).

Multi-specification dependencies should only be specified in a protocol specification if they are needed to preserve the integrity of that protocol. They should be avoided where they are really defining a profile.

Multi-specification dependencies may also be specified in a similar way in information object specifications.

7 Requirements on ICS proformas

7.1 The specific requirements to be met by suppliers in respect of each ICS they are to provide, shall be stated in the relevant base specification. The specification of these requirements shall include an ICS proforma. The ICS proforma shall be in the form of a questionnaire to be completed by the supplier or implementor of an implementation of the relevant base specification.

7.2 The ICS proforma shall cover all major mandatory capabilities, all optional and conditional functions, elements of procedure, parameters, options, PDUs, timers, multi-specification dependencies and other capabilities identified in the base specification.

7.3 There shall be a well-defined mapping (by references) from the ICS proforma to the static conformance requirements. The expression of the static conformance requirements in the ICS proforma shall be consistent with the conformance clause of the base specification.

7.4 Recommendation X.296 provides requirements and guidance on the production of ICS proformas.

8 Abstract Test Suite production process leading to conformance testing specifications

8.1 In order to present the requirements and general guidance for specification of Abstract Test Suites (ATSs), it is useful to assume a normal form of the process of ATS production leading to a conformance testing specification. This clause describes the process in just such a normal form. Test suite specifiers are not required to follow this normal form exactly, however they are recommended to use a similar process involving the same steps, possibly in a different order.

8.2 For the purposes of this Recommendation, the ATS production process is assumed to be as follows:

a) study the relevant specifications and ICS proformas to determine what the conformance requirements (including options) are which need to be tested (see clause 9);

b) decide which test groups will be needed to achieve the appropriate coverage of the conformance requirements (see 10.2);

c) optionally develop test group objectives: the common testing objectives of the elements of each test group (see 10.3);

d) develop test purposes which reflect the test group objectives (if any) of the test groups in which they are contained, and which provide adequate coverage of the conformance requirements to be tested (see 10.3 and 10.4);

e) choose the abstract testing context and the Abstract Test Method(s) for which the complete abstract test cases need to be specified, and decide what restrictions need to be placed on the capabilities of the Lower Tester(s) and, if appropriate to the chosen Abstract Test Methods(s), the Upper Tester(s) and Test Coordination Procedures (see clause 11);

f) apply a standardized test notation to specify the set of abstract test cases, including the test step structure to be used (see clause 12);

g) specify the interrelationships:

1) among the test cases;

2) between the test cases and the ICS(s); and

3) as far as possible, between the test cases and the Implementation extra Information for Testing (IXIT) statement(s);

in order to determine the restrictions on the selection and parameterization of test cases for execution, and the restrictions, if any, on the orders in which they can be executed (see clause 14);

h) consider the procedures for maintaining the ATS (see clause 15).

8.3 It is also assumed that during the ATS production process an overall structure for the conformance testing specification(s) will be developed, with appropriate parts for:

a) the Test Suite Structure and Test Purposes (TSS&TP) (see clause 10);

b) one or more ATSs for one or more Abstract Test Methods (see clause 11);

c) the specification of a Test Management Protocol (TMP), if applicable (see clause 13).

8.4 Clauses 9 to 15 provide requirements and guidance which relate to each step in the above process.

8.5 Testing for conformance to a profile is based on the use of appropriate test cases from base specification ATSs for the base specification(s) referenced in the profile. These base specification ATSs may be either single-protocol or multi-protocol ATSs. In some cases, it may be appropriate to standardize a base specification ATS for that subset of the base specification used by one or more specific profile(s), in which case subsequent amendments to the ATS should be made to extend the coverage to meet the needs of other profiles or the complete base specifications as and when necessary.

Additional profile specific test cases may be needed to address conformance requirements that are relevant to the profile but are outside the scope of the TSS&TP for any base specifications. These additional profile specific test cases are standardized in the Profile Specific Test Specification. The requirements and guidance on the production of Profile Test Specifications for protocol profiles are given in Recommendation X.295.

9 Conformance requirements and ICS proforma

9.1 Before an ATS can be specified, the test suite specifier shall first determine what the conformance requirements are for the relevant base specification(s) and what is stated in the ICS proforma(s) concerning the implementation of those specification(s).

9.2 Clauses 6 and 7 specify the requirements to be met by specifiers of the base specifications as a prerequisite to the production of an ATS for a particular base specification or combination of base specifications.

9.3 If the static conformance requirements are not properly specified, the test suite specifier should contribute to the production of an amendment to or revision of the relevant specification to clarify the conformance requirements.

10 Test Suite Structure and Test Purposes (TSS&TP)

10.1 Basic requirements

10.1.1 The test suite structure and set of test purposes applicable to all ATSs to be specified for the same base specification or combination of base specifications shall be specified in the relevant conformance testing specification, preferably in a separate part.

10.1.2 Each ATS shall comprise a number of test cases, each of which is designed to achieve one of the specified test purposes. The test cases may be grouped into test groups, if necessary nested. The structure shall be hierarchical; thus, an item at a lower level shall be completely contained within a higher level item. Similar test groups may occur in more than one higher level test group.

10.1.3 The test suite specifier shall ensure that a subset of the test purposes of each ATS is concerned with capability testing, and another subset is concerned with behaviour testing. This need not lead to distinct test cases for behaviour and capability testing because it may be possible to use a combined behaviour and capability test purpose. The test suite specifier shall provide an explanation of how the test purposes are derived from or relate to the base specification. The test suite specifier shall also provide a summary of the coverage achieved by the ATS.

10.2 Specification of the test suite structure

10.2.1 In order to ensure that the resulting ATS provides adequate coverage of the relevant conformance requirements, the test suite specifier is advised to design the test suite structure in terms of nested test groups in a top down manner. There are many ways of structuring the same test suite into test groups; no one way is necessarily right and the best approach for one test suite may not be appropriate for another test suite. Nevertheless, the test suite specifier shall ensure that the test suite includes test cases for whichever of the following categories are relevant:

a) capability tests (for static conformance requirements);

b) behaviour tests of valid behaviour;

c) behaviour tests that investigate the reaction of the Implementation under Test (IUT) to invalid test events; these may be subdivided into those concerned with syntactically invalid test events, semantically invalid test events, and inopportune test events, as relevant to the protocol concerned;

d) tests focusing on the different roles of the IUT;

e) tests focusing on PDUs sent to the IUT;

f) tests focusing on PDUs received from the IUT;

g) tests focusing on interactions between PDUs sent and PDUs received;

h) tests related to each mandatory capability;

i) tests related to each optional capability;

j) tests related to each protocol phase;

k) variations in the test event occurring in a particular state;

l) timing and timer variations;

m) PDU encoding variations;

n) variations in values of individual parameters;

o) variations in combinations of parameter values;

p) combinations of related requirements from more than one base specification;

q) tests specific for multi-party behaviour.

This list is not exhaustive; additional categories might be needed to ensure adequate coverage of the relevant conformance requirements for a specific test suite. Furthermore, these categories overlap one another and it is the task of the test suite specifier to arrange them into an appropriate hierarchical structure.

10.2.2 The following structure is an example of a single-layer protocol test group for a particular role in the Single Party Testing context, provided for guidance:

A. Capability tests

A.1 Mandatory capabilities

A.2 Optional capabilities

B. Behaviour tests: response to valid behaviour by peer implementation

B.1 Connection establishment phase (if relevant)

B.1.1 Focus on what is sent to the IUT

B.1.1.1 Test event variation in each state

B.1.1.2 Timing/timer variation

B.1.1.3 Encoding variation

B.1.1.4 Individual parameter value variation

B.1.1.5 Combination of parameter values

B.1.2 Focus on what the IUT is requested to send

– substructured as B.1.1

B.1.3 Focus on interactions

– substructured as B.1.1

B.2 Data transfer phase

– substructured as B.1

B.3 Connection release phase (if relevant)

– substructured as B.1

C. Behaviour tests: response to syntactically or semantically invalid behaviour by peer implementation

C.1 Connection establishment phase (if relevant)

C.1.1 Focus on what is sent to the IUT

C.1.1.1 Test event variation in each state

C.1.1.2 Encoding variation of the invalid event

C.1.1.3 Individual invalid parameter value variation

C.1.1.4 Invalid parameter value combination variation

C.1.2 Focus on what the IUT is requested to send

C.1.2.1 Individual invalid parameter values

C.1.2.2 Invalid combinations of parameter values

C.2 Data transfer phase

– substructured as C.1

C.3 Connection release phase (if relevant)

– substructured as C.1

D. Behaviour tests: response to inopportune events by peer implementation

D.1 Connection establishment phase (if relevant)

D.1.1 Focus on what is sent to the IUT

D.1.1.1 Test event variation in each state

D.1.1.2 Timing/timer variation

D.1.1.3 Special encoding variations

D.1.1.4 Major individual parameter value variations

D.1.1.5 Variation in major combination of parameter values

D.1.2 Focus on what the IUT is requested to send

– substructured as D.1.1

D.2 Data transfer phase

– substructured as D.1

D.3 Connection release phase (if relevant)

– substructured as D.1

10.2.3 This test group structure does not cover basic interconnection tests. These may be provided as a list of selected capability and/or behaviour tests, but shall not involve any additional test purposes.

10.3 Specification of the test purposes

10.3.1 The test suite specifier shall create a set of test purposes, with each test purpose focused on a single conformance requirement or set of related conformance requirements (e.g. in the case of multi-protocol testing) of the relevant specification(s).

It is suggested that test groups of related test purposes are identified first (as described in 10.2) and that text defining the test group objective be produced for each test group. Within each test group, several more specific test objectives should be defined, to become either nested test group objectives or individual test purposes. By successive refinement of the test group objectives in this way, a structured set of test purposes may be produced.

The test purposes may be produced directly from studying those clauses in the relevant specification(s) which are appropriate to the test group concerned. For some test groups, the test purposes may be derivable directly from the protocol state table; for others, they may be derivable from the PDU encoding definitions or the descriptions of particular parameters, or from text which specifies the relevant conformance requirements.

This orderly construction technique helps to ensure the adequate coverage of the conformance requirements to be tested. It also avoids unnecessary duplication of text in the test purposes, because the full description of each test purpose does not have to be written explicitly, but can be assembled by tracing a path down through the nested structure of test group objectives.

NOTE – If the test suite specifier employs a formal description of the base specification(s) concerned, test purposes may be derived from that by means of some automated method. If an automated method is used, the same requirements apply. However, methods based on Formal Description Techniques (FDTs) are outside the scope of this Recommendation. Nevertheless, if an FDT is to be used for this purpose, it is preferred that it be a standardized one.

10.3.2 In order to increase the efficiency of testing individual parameters on a single PDU, test purposes covering a combination of parameters may be specified for a single abstract test case. However, the testing of invalid parameter values shall not be combined with the testing of other valid or invalid values in a single test purpose.

10.3.3 As part of the process of designing the TSS&TP, it is suggested that test purposes be identified initially for each conformance requirement (e.g. specific parameter) that is to be tested. As a second stage, test purposes for combinations of related conformance requirements may be specified. If this is done,

a) a new test purpose covering a combination of related conformance requirements shall be written referencing those test purposes which cover the individual conformance requirements;

b) an indication shall be given that one abstract test case is to be produced for that new test purpose, rather than a distinct test case for each of the referenced test purposes that have been superseded;

c) each superseded test purpose shall remain in the set of test purposes, but shall identify the new test purpose(s) which supersede it.

10.3.4 The result of identifying and then combining particular conformance requirements to form test purpose(s) is a specification of a test suite structure and a list of names of the test cases that shall apply to both the test purposes and to any ATS produced for those test purposes.

10.3.5 Whatever method is used to derive the test purposes, the test suite specifier should ensure, as far as possible, that they provide an adequate coverage of the conformance requirements of the relevant specification(s). There shall be at least one test purpose related to each distinct conformance requirement or set of related conformance requirements.

10.3.6 Test purposes should be specified not only for clearly testable conformance requirements, but also for conformance requirements that may be untestable using the test methods defined in this Recommendation.

NOTE – Test purposes for untestable requirements serve to inform the specifiers of the base specifications which conformance requirements are untestable, by indicating gaps in the ATSs.

10.4 Coverage

Ideally, the TSS&TP should provide coverage of all of the conformance requirements of the relevant base specification(s). However, if the related ATS(s) are to be developed only to meet the needs of testing particular profile(s), then the TSS&TP may initially be developed just to provide coverage for those conformance requirements of the base specification(s) relevant to the particular profile(s). Such a TSS&TP should be expanded to provide full coverage of the base specification(s) when resources permit.

It is possible to give guidance on the meaning of "adequate" coverage with reference to the test suite structure example in 10.2. In order to express this, a shorthand notation will be used: the letter "x" will represent all appropriate values for the first digit in the test group identifier, and similarly "y" for the second digit, so that B.x.y.1 stands for B.1.1.1, B.1.2.1, B.1.3.1, B.2.1.1, B.2.2.1, B.2.3.1, B.3.1.1, B.3.2.1 and B.3.3.1.

With this notation, a minimum "adequate" coverage for the example given in 10.2 is considered to be as follows:

a) For capability test groups (A.1, A.2):

   1) at least one test purpose per relevant capability;

   2) at least one test purpose per relevant PDU type and each major variation of each such type, using "normal" or default values for each parameter.

b) For test groups concerned with test event variation in each state (B.x.y.1, C.x.1.1, D.x.y.1), at least one test purpose per relevant state/event combination.

c) For test groups concerned with timers and timing (B.x.y.2, D.x.y.2), at least one test purpose concerned with the expiry of each defined timer.

d) For test groups concerned with encoding variations (B.x.y.3, C.x.1.2, D.x.y.3), at least one test purpose for each relevant kind of encoding variation per relevant PDU type.

e) For test groups concerned with valid individual parameter values (B.x.y.4, D.x.y.4):

   1) for each relevant integer parameter, test purposes concerned with the boundary values and one randomly selected mid-range value;

   2) for each relevant bitwise parameter, test purposes for as many values as practical, but not less than all the "normal" or common values;

   3) for other relevant parameters, at least one test purpose concerned with a value different from what is considered "normal" or default in other test groups.

      NOTE 1 – Tests for valid parameter values should focus on the relevant claims made in the ICS.

f) For test groups concerned with syntactically or semantically invalid individual parameter values (C.x.1.3, C.x.2.1):

   1) for each relevant integer parameter, test purposes concerned with invalid values adjacent to the allowed boundary values defined in the base specification, plus one other randomly selected invalid value;

   2) for each relevant bitwise parameter, test purposes for as many invalid values as practical;

3) for all other relevant types of parameter, at least one test purpose per parameter.

NOTE 2 – Tests for invalid parameter values should focus on values outside the range defined in the relevant base specification, rather than valid values outside the range claimed in the ICS.

g) For test groups concerned with combinations of parameter values (B.x.y.5, C.x.1.4, C.x.2.2, D.x.y.5):

1) at least one test purpose for each important combination of specific values (e.g. boundary values);

2) at least one test purpose per set of interrelated parameters to test an arbitrary combination of relevant values.

The test suite specifier shall not assume that the test realizer or test laboratory will perform any checking of test events against the values specified in the ICS other than that checking which is specified in the abstract test cases. Therefore, the test purposes and abstract test cases shall make explicit use of values given in the ICS whenever checking of valid parameter values is specified. The test suite shall include test cases to check for the support of parameter values that are allowed by the base specification(s) and are within the ranges stated in the ICS. Such test cases shall make use of test suite parameters that contain the relevant ICS values. The test suite shall also include test cases to check for valid reactions to parameter values that are invalid with respect to the base specification(s). Parameter values that are valid with respect to the base specification(s) but outside the ranges stated in the ICS shall not be tested.

NOTE 3 – The progression of the work on formal methods in conformance testing may provide analytical approaches to assess the appropriate coverage of an ATS, especially for the state/event variations, as in b) above. This Recommendation, however, does not recommend any particular analytical approach.

10.5 TSS&TP compliance clause

The TSS&TP part shall include a compliance clause concerning the development of test suites for that TSS&TP. That clause shall require, as a minimum, that an ATS complying with the TSS&TP part:

a) consists of a set of test cases corresponding to the set or to a subset of the test purposes specified in the TSS&TP part;

b) uses a test suite structure which is an appropriate subset of the whole of the test suite structure specified in the TSS&TP part;

NOTE – The only subsetting of the test suite structure that should take place is as follows:

– in the absence of a complete ATS for a base specification, the subsetting of the TSS&TP to give complete coverage of that base specification for one or more profiles;

– the omission of test purposes that are untestable in the chosen Abstract Test Method; in particular, for embedded method variants, this will be necessary due to the limitations imposed by the use of the base specification(s) above the one that is the focus of the test purposes.

c) uses the same naming conventions for the test groups and test cases;

d) maintains the relationship, if any, specified in the TSS&TP between the test purposes and the entries in the ICS proforma(s) and partial IXIT proforma(s) to be used for test case deselection;

e) complies with this Recommendation.

11 Abstract testing methodology

11.1 Introduction

11.1.1 In the abstract testing methodology, there are two contexts in which test cases may be specified: the Single-Party Testing (SPyT) context, and the Multi-Party Testing (MPyT) context.

11.1.2 The SPyT context is needed when an IUT is required, by the test purpose, to communicate with only one real other open system.

11.1.3 The MPyT context is needed when an IUT is required, by the test purpose, to communicate with multiple other real open systems concurrently. In the MPyT context, the IUT may communicate with all of the real open systems by means of the same service provider, or it may communicate with individual real open systems by means of different service providers.

11.1.4 The abstract testing methodology makes use of four abstract testing functions, named the Lower Tester (LT), the Lower Tester Control Function (LTCF), the Upper Tester (UT), and the Test Coordination Procedures (TCP).

11.1.5 In the SPyT context the functions required are as follows:

a) an LT which behaves as the peer real open system to the IUT, and assigns the verdict for the test case;

b) a UT behaving as a user of the IUT;

c) TCP between the LT and the UT.

11.1.6 In the MPyT context the functions required are as follows:

a) a set of LTs which execute in parallel, each LT behaving as a peer real open system to the IUT;

b) an LTCF, that coordinates the activity of the LTs, and UTs if any, and assigns a verdict for the test case;

c) optionally, a set of UTs which execute in parallel, each UT behaving as a user of the IUT;

d) TCP between each associated LT and UT, among the LTs, between LTs and the LTCF, among the UTs, and between the UTs and the LTCF.

11.1.7 An Abstract Test Method (ATM) describes an abstract testing architecture consisting of a configuration of abstract testing functions (LTs, UTs, LTCF and TCP) as applicable to either the SPyT or MPyT context, and the relationships of these functions to the test system and the System under Test (SUT). Each ATM determines the Points of Control and Observation (PCOs) and test events (i.e. Abstract Service Primitives (ASPs) and PDUs) which shall be used in an abstract test case for that ATM.

11.1.8 In the SPyT context, there are just four main ATMs defined, varying in the extent of control and observation of the IUT that they provide. These are called the Local, Distributed, Coordinated and Remote test methods, which are defined in 11.3. There are several variants of these ATMs that may be used in an ATS (see 11.4).

11.1.9 In the MPyT context, any configuration of LTCF, one or more LTs, zero or more UTs and TCP may be used as an MPyT ATM. Such ATMs may include any combination of SPyT ATMs for LT/UT pairs, but may also include the use of LTs without any corresponding UTs.

11.1.10 Each ATS shall be specified in accordance with one or more ATMs. Each ATS shall identify which ATM is used for each test case or test group.

11.2 General specification of the Single Party Testing context

11.2.1 Introduction

The SPyT context applies to end system SUTs.

11.2.2 Requirements on the Lower Tester

11.2.2.1 An LT is the representation of the means of providing, during test execution, indirect control and observation of the lower service boundary of the IUT via the underlying service-provider.

This is performed by specifying events at the LT PCO.

11.2.2.2 In this Recommendation, a notation is used to refer to protocols both within the SUT and within the LT.

The IUT itself is defined in terms of services provided at its upper and lower boundaries. The IUT may be an implementation of a single protocol within a single OSI layer. Alternatively, it may include the implementation of multiple adjacent protocols in one or more OSI layers. The protocols in the IUT are designated P_1 to P_n.

The highest protocol under test is numbered P_n and the lowest protocol is numbered P_1. For single-protocol IUTs, n is equal to 1.

The SUT may implement protocols lower than P_1 but these are not of interest in the abstract testing methodology. Nevertheless, the SUT shall include the Physical layer.

The underlying service-provider, beneath protocol P_1, is designated the X-service.

The X-service may use the physical medium only, or one or more OSI layers. There is no requirement that the underlying service is provided at a LAYER boundary beneath the layer of P_1. In some instances, notably Application layer testing, the underlying service provider may be in the same layer as P_1.

The same notation applies to the LT, therefore the test events specified at the LT PCO are specified in terms of X-ASPs and/or (P_1 to P_n)-PDUs.

NOTE – This notation is purposely made independent of the numbered layers of the OSI Basic Reference Model (see Recommendation X.200), to enable the description of any IUT including those whose boundaries do not match the OSI layer boundaries, e.g. Commitment, Concurrency and Recovery protocol. But when an IUT is an implementation of a protocol spanning a whole OSI layer, then X-ASP is equivalent to (N – 1) – ASP and Y – ASP is equivalent to (N)-ASP, consistent with the use of the N and N – 1 notation in Recommendation X.200.

11.2.2.3 If the communication between the LT and the IUT needs to be split over parallel connections (e.g. to test a splitting and recombining function) then the LT may use multiple PCOs in the SPyT context. In other cases in which multiple connections are required the MPyT context should be used.

11.2.3 Requirements on the Upper Tester

The general requirements on the UT vary with the ATM.

The main difference between the ATMs is in the nature of the UT and its coordination with the LT.

In some test methods, a PCO is to be employed for the UT, in addition to the PCO for the LT. In these test methods, the definition of the test events at the PCO for the UT shall be specified in accordance with the appropriate OSI service definition and base specification. The service at the PCO for the UT is called the Y-service.The activity at the UT PCO shall not require that the SUT or IUT support ASP parameters, PDUs or capabilities that are not part of a relevant base specification.

If the PCO is at a humanly accessible interface, the SUT's user interface shall serve as the PCO.

11.2.4 Test Coordination Procedures

For effective and reliable execution of conformance tests, some set of rules is required for the coordination of the test process between the LT and the UT. The general objective of these rules is to enable the LT to control the operation of the UT, in ways necessary to run the test suite chosen for the IUT.

These rules lead to the development of TCP to achieve the synchronization between the LT and the UT and the management of information exchanged during the testing process. The details of this synchronization and how the required effects are achieved are closely related to the characteristics of the SUT as well as to the test methods.

The requirements on the TCP shall be specified for each ATS. The TCP shall include provision for passing, to the LT, events which are controlled (and if applicable, observed) at the UT, and which need to be logged.

11.3 Abstract Test Methods for Single Party Testing methods

11.3.1 Introduction

For IUTs within end-system SUTs, there are four categories of ATMs:

- Local;
- Distributed;
- Coordinated; and
- Remote.

11.3.2 The Local test method

In this test method:

a) the test events at the LT PCO are specified only in terms of X-ASPs and/or (P_1 to P_n)-PDUs;

b) the test events at the UT PCO are specified in terms of Y-ASPs;

c) the upper service boundary of the IUT shall be a standardized hardware interface which can be used for testing purposes; the test suites shall not place any requirements on the realization of the interface in the SUT, additional to those in the standardized hardware interface specification;

d) the specification of the hardware upper interface of the IUT shall define the mapping between the relevant ASPs and/or PDUs and their realization at the interface;

e) the UT is located within the test system;

f) the requirements for the TCP shall be specified in the ATS but are realized locally within the test system.

This test method is illustrated in Figure 1.

11.3.3 The Distributed test method

In this test method:

a) the test events at the LT PCO are specified only in terms of X-ASPs and/or (P_1 to P_n)-PDUs;

b) the test events at the UT PCO are specified in terms of Y-ASPs;

c) the upper service boundary of the IUT shall be either a human user interface or a standardized programming language interface which can be used for testing purposes; the test suites shall not place any requirements on the realization of the interface in the SUT, additional to those in the standardized programming language interface specification, if applicable;

d) there shall be a mapping between the relevant ASPs and their realization at the upper interface of the IUT;

e) the UT is located within the SUT;

f) the requirements for the TCP shall be specified in the ATSs, although the procedures themselves shall not be;

g) if the upper interface of the IUT is a human user interface, then the human operator of the SUT fulfils the requirements of the TCP;

h) if the upper interface is a standardized programming language interface, then the UT is realized in software and the UT and LT together fulfil the requirements of the TCP.

This test method is illustrated in Figure 2.

ATSs for the Distributed test method shall not themselves specify a UT interface.

In order to avoid placing requirements on the internal design of SUTs, the ATSs shall not require that a programming language interface is standardized for the sole purpose of testing.

NOTE – In the Application layer, until application programming interfaces are standardized to provide a common means of access to OSI Application services, the use of the Distributed test method is in practice limited to using human user interfaces to OSI applications (e.g. File Transfer Access and Management initiators).

11.3.4 The Coordinated test method

In this test method:

a) the test events at the LT PCO are specified in terms of X-ASPs, and/or (P_1 to P_n)-PDUs plus Test Management PDUs (TM-PDUs);

b) Y-ASPs are not used in the specification of the ATS; no assumption is made about the existence of an upper service boundary of the IUT;

c) the UT is located within the SUT;

d) the requirements for the TCP shall be specified in the ATS by means of a standardized TMP, referenced by the ATS;

e) the UT shall be required by the test suite specifier to implement the TMP and achieve the appropriate effects on the IUT;

f) test cases shall be added to the ATS for the purpose of testing that the UT conforms to the requirements of the TMP specification; such test cases do not contribute to the conformance assessment of the IUT.

A standardized TMP is applicable to a particular ATS specification for the Coordinated test method and may not be applicable to other ATSs for the Coordinated test method.

Concerning the TMP:

a) the TMP shall be required to implemented within the SUT directly above the abstract service boundary at the top of the IUT;

b) the IUT shall not be required to interpret TM-PDUs, only pass them to and from the UT;

c) a TMP is defined only for testing a particular protocol and so does not need to be independent of the underlying protocol;

d) verdicts on test cases shall not be based on the ability of the SUT to exhibit any ASP or parameter of an ASP at the upper service boundary of the IUT, since this would contradict the definition of the Coordinated test method: the upper service boundary of the IUT is not a PCO in this test method. However, it is recommended that the TMP be defined separately from the ATS(s) in order to facilitate the task of the implementor of a UT. The specification of the TMP (as with the specification of any OSI protocol) may refer to the ASPs of its underlying service (i.e. the ASPs at the upper service boundary of the IUT).

This test method is illustrated in Figure 3.

11.3.5 The Remote test method

In this test method, provision is made for the case where it is not possible to observe and control the upper service boundary of the IUT. Also in this test method:

a) the test events at the LT PCO are specified only in terms of X-ASPs, and/or (P_1 to P_n)-PDUs;

b) Y-ASPs are not used in the specification of the ATS; no assumption is made about the existence of an upper service boundary of the IUT;

c) some requirements for TCP may be implied or informally expressed in the ATS but no assumption shall be made regarding their feasibility or realization;

d) abstractly the SUT needs to carry out some UT functions to achieve whatever effects of the TCP and whatever control and/or observation of the IUT are implied or informally expressed in the ATS for the given base specification(s); these functions are not specified nor are any assumptions made regarding their feasibility or realization;

e) the LT should attempt to achieve the implied or informally expressed TCP in accordance with the relevant information in the IXIT(s).

In addition, in order to overcome the lack of specification of behaviour above the IUT, where necessary, the required behaviour of the SUT shall be specified in terms of the X-ASPs or (P_1 to P_n)-PDUs which need to be observed by the LT. This form of implicit specification shall be taken to mean “do whatever is necessary within the SUT in order to provoke the required behaviour”.

However, it is possible that some of the test cases in the ATS cannot be executed (e.g. transmission of consecutive unacknowledged Data PDUs, etc.).

With such implicit specification of control of the IUT, in this test method, it is possible to specify control but not observation above the IUT. This is a major difference between this and the other test methods.

This test method is illustrated in Figure 4.

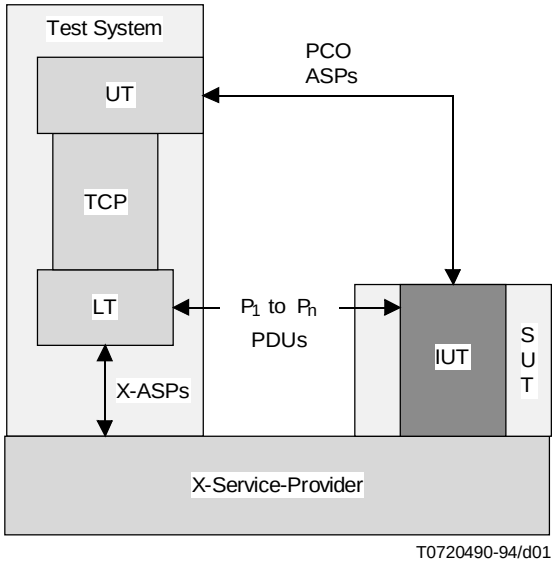

FIGURE 1/X.291

The Local test methods

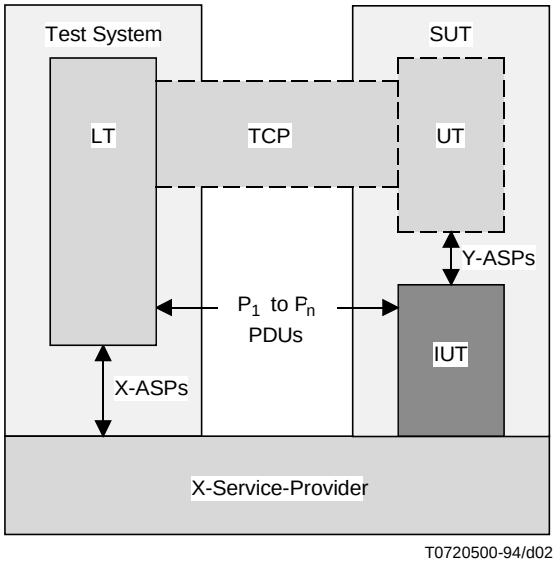

FIGURE 2/X.291

The Distributed test methods

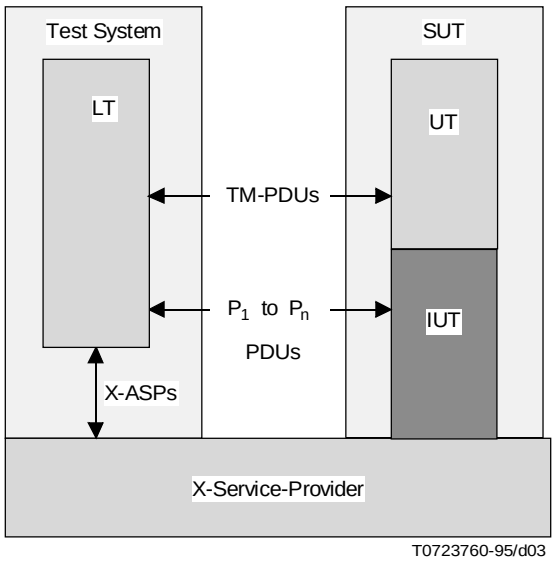

FIGURE 3/X.291

The Coordinated test methods

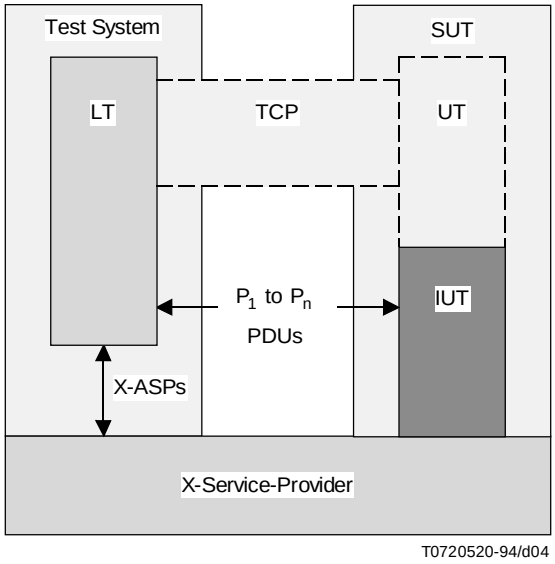

FIGURE 4/X.291

The Remote test methods

11.4 Test method variants

11.4.1 Embedded and Non-Embedded variants of test methods in the Single Party Testing context

In the SPyT context, if control and observation are applied to a means of access to the upper service boundary of the IUT, the test methods are said to be non-embedded. If, however, control and observation are applied through the implementation of one or more base specifications above those to be tested, the test methods are called embedded.

Each of the test methods has a non-embedded variant which can be applied to either single or multi-protocol IUTs. If a non-embedded variant is applied to a multi-protocol IUT, then the IUT is tested as a whole, with test purposes that combine conformance requirements from each of the protocols.

Each of the test methods also has an embedded variant to test protocols within a multi-protocol IUT one at a time.

When testing an IUT, consisting of protocols P_1 to P_n, the embedded variant has test purposes which only focus on protocol(s) P_1 to P_e ($e < n$). In this variant protocol(s) P_1 to P_e are being tested embedded under protocol(s) P_{e+1} to P_n. The non-embedded variant has test purposes which focus on all protocol(s) P_1 to P_n.

NOTE – This description of the embedded variants assumes that the protocols of the IUT are ordered in a continuous adjacent user/provider relationship.

The embedded variants are defined to test a part of a multi-protocol IUT. This does not mean that there cannot be accessible service boundaries within the multi-protocol IUT: it means that no such boundaries are used by the test methods. Thus, all protocols between the protocol(s) under test and the highest protocol for which PDUs are expressed as test events in the ATS shall be regarded as being part of the multi-protocol IUT.

With embedded testing it is possible to test an IUT consisting of more than one protocol, incrementally. First protocol P_1 can be tested embedded under protocol(s) P_2 to P_n. If protocol P_1 is tested, this protocol can be considered part of a new service provider, and as a result, the IUT then consists of protocols P_2 to P_n. This process can be repeated for the rest of the protocols. The last protocol (P_n) is tested using a non-embedded variant.

11.4.2 Multi-user variants

Some protocols require only one communication path between the LT and the IUT, but have more than one user of the IUT. This situation may require several UTs to be specified to fulfil the requirements of the test purpose. (This is illustrated in Figure 5.)

If more than one UT is used, TCP may be specified between the UTs to provide coordination of their activities without going via the LT.

11.5 General specification of the Multi-Party Testing context

11.5.1 Introduction

The MPyT context is a context in which the IUT is required to communicate with multiple other real open systems. Only one IUT is tested but multiple LTs are used to test it.

Open relay systems are tested in the MPyT context.

In the MPyT context, each LT represents one of the real open systems with which the IUT needs to communicate. Each LT communicates with the appropriate part of the IUT, by observing and controlling ASPs and PDUs as in the SPyT context.

In the MPyT context, the LTs shall provide preliminary results, but shall not assign a verdict.

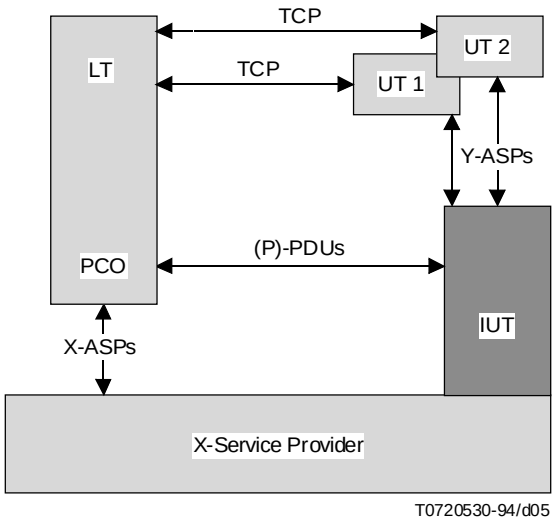

FIGURE 5/X.291

Multi-user Single-Party Testing

11.5.2 Lower Tester Control Function

The LTCF coordinates the activity of the LTs and assigns the verdict to the test case. The LTCF is architecturally distinct from the LTs, although its behaviour may be specified in combination with the behaviour of one of the LTs. The LTCF does not have direct responsibility for controlling and observing behaviour at a PCO.

An LTCF shall be specified in each test case which is in the MPyT context.

11.5.3 Upper Testers

In the MPyT context, UTs may be used as in the SPyT context or may be absent, depending on the requirements of the ATMs used for a particular test suite.

NOTE – For practical reasons, to avoid unnecessarily complex requirements to be met by the SUT, only a small number of different configurations of UTs should be used within a single MPyT ATS.

11.5.4 Test Coordination Procedures

The requirements for TCP shall be specified in each test case.

The TCP between an LT and a UT may be specified in accordance with one of the four SPyT ATMs. There is no requirement to use the same SPyT ATM for all LT/UT pairs.

TCP shall be specified between the LTCF and each LT, in each test case. In addition, TCP may be specified between different LTs without going via the LTCF.

NOTE – For example TCP between the LTCF and LTs may be used to:

a) start and stop LTs;

b) suspend and resume the execution of LTs;

c) exchange information such as preliminary results.

TCP may also be specified between the LTCF and one or more UTs, and may be specified between different UTs.

11.5.5 Illustration of Abstract Test Methods for Multi-Party Testing

Figure 6 shows the general model for MPyT with multiple UTs.

Figure 7 shows the model if a single UT is used.

Figure 8 illustrates the use of MPyT with no UT.

Figure 9 illustrates the use of MPyT with no UT, as appropriate for testing a relay system between two subnetworks, in which there are two LTs, one on each subnetwork at Service Access Points (SAPs) external from the relay. This enables an open relay system to be tested in its normal mode of operation, with its behaviour on each subnetwork being observed. TCP between the two LTs require a connection between the LTs, either via the relay system or by any other means.

11.6 Choice of Abstract Test Method

11.6.1 Introduction

Before an ATS can be defined, it is necessary to study all the environments in which the base specification(s) is/are likely to be tested and establish accordingly an abstract testing context and the ATM(s) to be used for the production of one or more ATSs.

ATMs vary in the extent of control and observation of an IUT that they provide. The choice of test method, therefore, influences the expressibility of the behaviour in test case descriptions.

11.6.2 Comprehensive testing service

Test suite specifiers shall place a requirement in the conformance testing specification defining which ATM(s) shall be supported as a minimum by any organization claiming to provide a comprehensive testing service for the base specification(s) in question. If an organization supports this minimum set of ATMs, then it may claim to provide a comprehensive testing service even when other ATMs may be applicable to the base specification(s) in question.

A comprehensive testing service shall offer at least one ATS which places no additional requirements upon the SUT other than those contained in the base specification(s) to which the SUT claims to conform.

To meet this requirement in the SPyT context, an ATS for the Remote test method shall be included in a comprehensive testing service, unless one of the other test methods also meets this requirement. For some protocols in, or embedded under, the Application layer, it may be possible to meet this requirement by including a test suite for the embedded variant of the Distributed test method. For IUTs with hardware upper interfaces it may also be possible to meet the requirement by including a test suite for the Local test method.

To meet this requirement in the MPyT context the relevant ATS shall only use ATMs (i.e. in TTCN, test component configurations) which do not require interfaces in the SUT other than those required by the relevant base specification(s) to which the SUT is claimed to conform.

If a standardized ATS specification is produced which does not meet the above requirement for provision of a comprehensive testing service, then it shall contain the following statement in the Scope clause:

“This Abstract Test Suite is insufficient on its own for the provision of a comprehensive testing service as defined in Recommendation X.291 for the <name of base specification>.”

A comprehensive testing service requirement statement shall be located, as a separate clause in the part of the conformance testing specification containing the test purposes for a particular base specification(s).

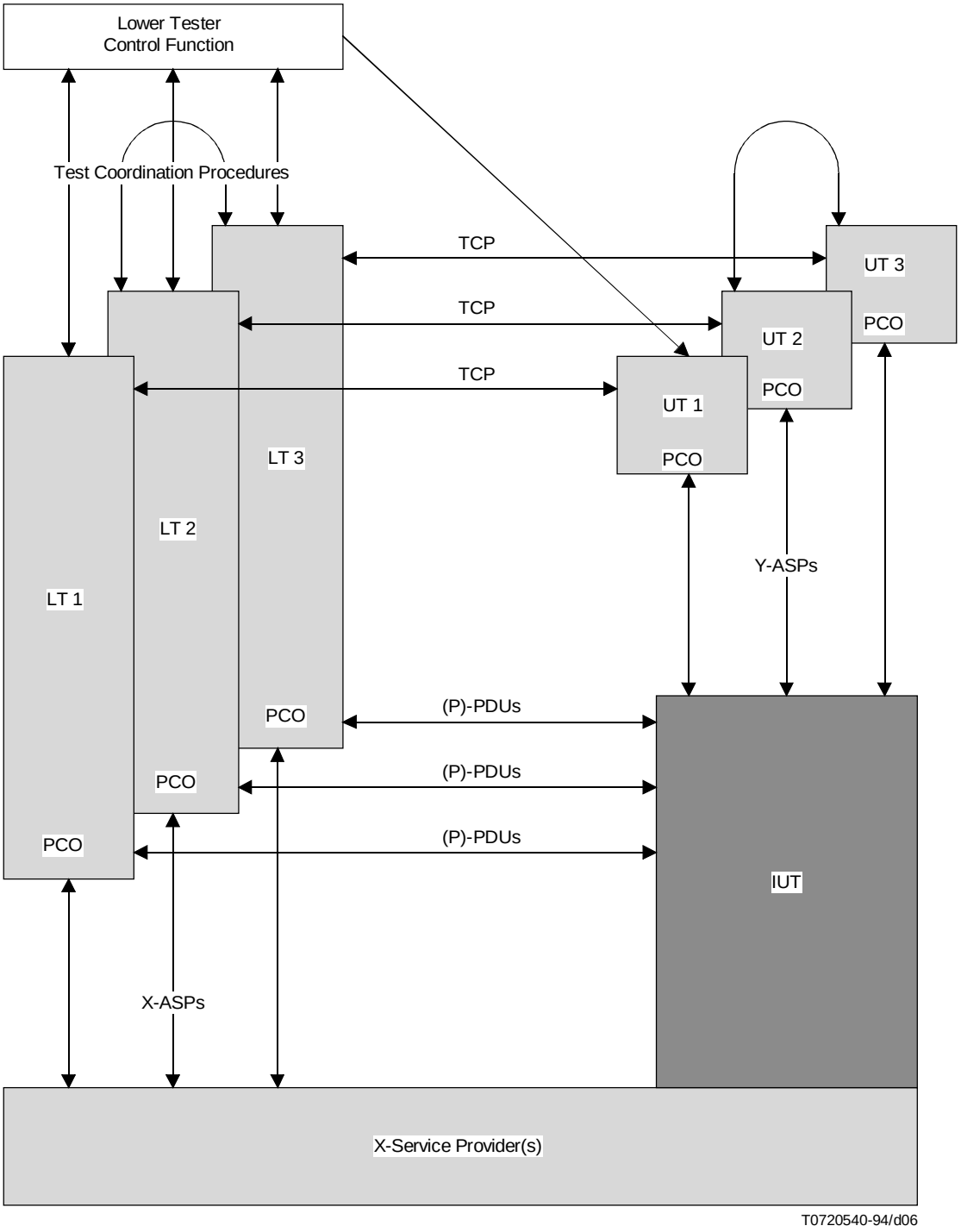

FIGURE 6/X.291

General model for Multi-Party Testing

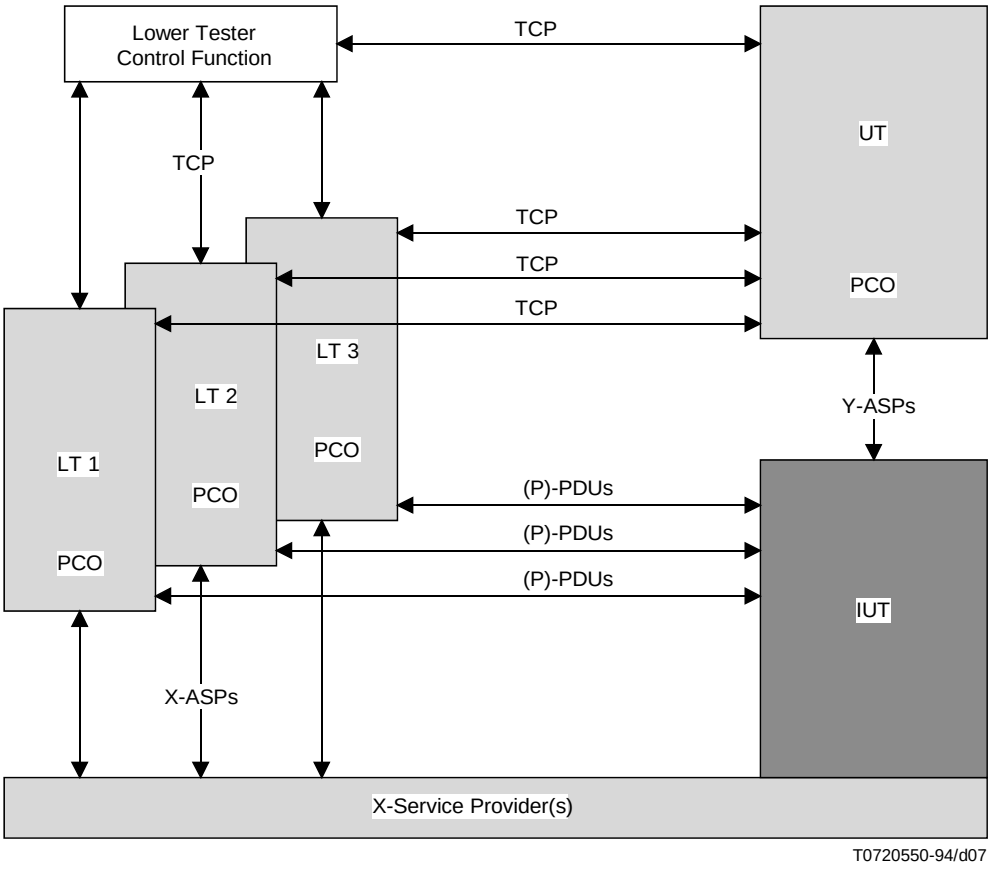

FIGURE 7/X.291

Multi-Party Testing with a single UT

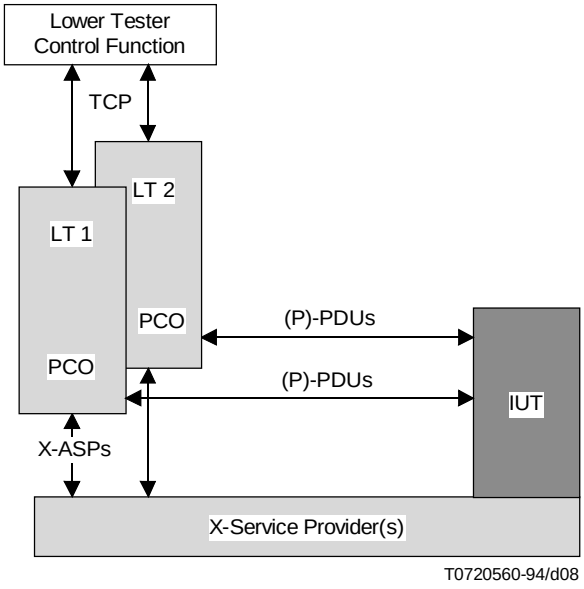

FIGURE 8/X.291

Multi-Party Testing with no UT

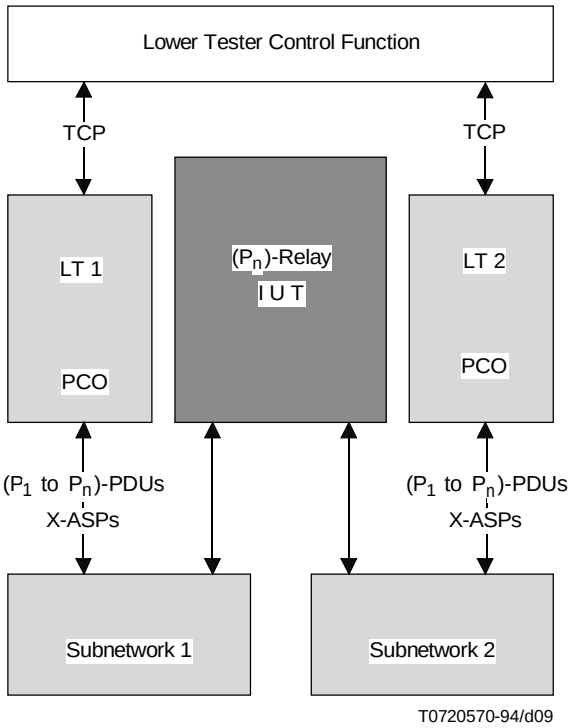

FIGURE 9/X.291

Possible realization with no UT

11.6.3 Types of Implementation under Test

There is a relationship between the test methods and the configurations of the real open systems to be tested.

Subclause 7.2/X.290 gives a full account of the classification of SUT and IUTs.

When choosing a test method, the test suite specifier shall identify, if they have not already done so, whether the test suites they plan to produce are for IUTs which

a) are single-protocol, multi-protocol or include a combination of protocol(s) and information object(s);

b) belong to end or relay systems;

c) belong to complete or partial systems;

d) belong to fully open or mixed systems;

e) have service boundaries accessible or not;

f) are special purpose (i.e. to be used by a single application) or general purpose (i.e. to be used by several applications);

g) require communication with only one real open system at a time or concurrently with more than one real open system.

11.6.4 Applicability of the Abstract Test Methods

Some considerations concerning the applicability of the test methods to different protocols are discussed in Appendix I.

One or more appropriate ATMs shall be selected for the base specification(s) being considered.

For each base specification or combination of base specifications for which ATSs are to be produced, priorities should be assigned to the standardization of different ATSs for different applicable test methods, giving highest priority to those that are most likely to be applicable to the majority of real systems.

12 Specification of Abstract Test Suites

12.1 General

An ATS comprises a set of test cases and optionally test steps for a particular test method or set of test methods.

An ATS for an embedded test method may be specified in two parts: one which is independent of the base specification(s) under which the embedding is to be done, and one which is specific to the embedding base specification(s). The former is called the common part and the later the specific part.

Preceding the test cases themselves the ATS specification shall include the following information:

a) ATS name, date of origin and version number;

b) references to (and version numbers of) the base specification(s) [for protocol(s), abstract syntax, encoding rules and information object(s) as appropriate] which are used in the specification of the test cases;

c) references to (and version numbers of) the OSI service definitions, the ASPs of which are used in the test cases;

d) reference to (and version number of) the specification defining the test notation;

e) description of the coverage of the test suite; for example, the functional subsets of the base specification(s) that are tested;

f) description of the structure of the test suite, in terms of test groups and their relationship to the base specification(s);

g) whether SPyT and/or MPyT contexts are required by the ATS;

h) identification of the SPyT or MPyT ATM(s) used;

i) description of the TCP or a reference to the specification of the TMP (if applicable in the test method);

j) optionally, a list of which capability and behaviour test cases may be used as basic interconnection tests;

k) information to assist the test realizer and test laboratory in their use of the ATS (see clause 14);

l) an identification of the Technical Corrigenda (or ITU-T equivalent) which are related to the base specification(s), and which have been taken into account in the ATS.

12.2 Use of Tree and Tabular Combined Notation (TTCN)

12.2.1 The test suite specifier shall apply a standardized test notation in which to specify the abstract test cases. TTCN, defined in Recommendation X.292, is recommended for this purpose.

12.2.2 If an ATS uses facilities additional to those in TTCN as defined in Recommendation X.292, then such additions shall be documented in the ATS and submitted for inclusion in Recommendation X.292 by means of defect reports or an amendment, as appropriate.

Test suites and test cases in the MPyT context shall be specified by using "Concurrent TTCN" as specified in ISO/IEC 9646-3:1994/Amd 1.

LTs may be specified as test components, the main test component playing the role of the LTCF. UTs may similarly be specified as test components.

For embedded test methods, an ATS in TTCN may be subdivided into common and specific parts.

12.3 Specification of abstract test cases

12.3.1 Once the test notation and test method have been chosen, the abstract test cases can be specified.

Each abstract test case shall:

a) reflect only a single test purpose (which may cover a set of related conformance requirements) as defined by the test purpose specifier;

b) specify all sequences of test events that comprise the test body;

c) specify all sequences of test events that comprise the test preamble(s), if any, necessary to ensure that the test case is capable of being started in the idle testing state and, optionally, in one or more other stable testing states (see 12.3.3);

d) specify all sequences of test events that comprise the test postamble(s), if any, necessary to ensure that the test case is capable of ending in the idle testing state and, optionally, in one or more stable testing states;

e) be specified using the chosen test notation and the appropriate one of the chosen SPyT or MPyT ATM(s);

f) specify the test verdict to be associated with each possible sequence of test events comprising a complete path through the test case.

12.3.2 When specifying test cases for an embedded test method, the test suite specifier may find that parameterizing the ATS allows it to be used, unchanged, under several higher base specifications. In this situation the ATS specification will be a multi-part document. One part specifies the common part, containing the test cases parameterized with "formal parameters". Other parts each specify a specific part for a particular higher base specification, or combination of base specifications, and this specific part contains appropriate "actual parameters". To have a complete ATS it is necessary to have both the common part and one of the specific parts.

12.3.3 If a test purpose can be achieved only by means of system-dependent actions in the SUT, it is not possible to specify an abstract test case for that test purpose in an ATS. This limitation shall be documented in the ATS specification.

NOTE – The possibility of writing ad hoc conformance resolution tests to achieve the test purpose on a case-by-case basis should be indicated, but such tests are outside the scope of standardization.

If a test purpose cannot be achieved due to the specific nature of the chosen ATM, that limitation shall also be documented in the ATS specification.

Thus, for each specified test purpose, the ATS shall either specify an abstract test case to achieve that test purpose or shall document why such a test case is not included.

12.3.4 A choice of more than one test preamble may be specified in a given abstract test case, one for each of the stable testing states in which the test case can start. Each test preamble takes the test case from a particular stable testing state to the initial testing state of the test body. Thus, a small set of stable testing states, in which test cases may start and end, shall be defined for the ATS; this set shall include the appropriate idle testing state.

NOTE 1 – It is likely that not more than two or three test preambles will need to be used per test case.

In each abstract test case in which the initial testing state of the test body is not the idle testing state, the test suite specifier shall define a test preamble to take the test case from the idle testing state to the initial testing state of the test body. In addition, in each abstract test case in which the test body does not necessarily end in the idle testing state, the test suite specifier shall define one or more test postambles to enable the abstract test case to end in the idle testing state.

NOTE 2 – The ability to start and end an abstract test case in an idle testing state is necessary in order to be able to run each abstract test case individually, in isolation from the other abstract test cases.

If more than one test preamble or postamble is defined for an abstract test case, then the test suite specifier shall specify the conditions under which each test preamble or postamble is to be used. The choice of test preamble shall depend upon the stable testing state in which the test case starts. The choice of test postamble shall depend upon the testing state in which the test body ends and the stable testing state in which the test case is to end.

The omission of a test preamble from an abstract test case shall be permitted only if the initial testing state of the test body is the desired starting stable testing state of the test case. Similarly, the omission of a test postamble from an abstract test case shall be permitted only if the final testing state of each path of the test body is one of the desired ending stable testing states of the test case. Each test postamble takes the test case from the end of the test body to a stable testing state in which the test case can end.

If it is the intention to be able to make use of test preambles that start in some stable testing state other than the idle testing state, the test suite specifier shall specify that the identity of the ending stable testing state of each abstract test case is stored for access by the next test case. The next test case can then compare the identity of that state with the possible stable testing states, in order to determine which test preamble to use. In this way, the use of test preambles is made conditional on the starting stable testing state, not unconditionally optional.

If the initial testing state of the test body is a transient testing state, then the test body shall not be run without first running a test preamble.

12.3.5 Each test preamble, test body and test postamble may be explicitly identified as test steps, but they need not be.

In designing the test step structure within abstract test cases, the test suite specifier can benefit from using the same test steps in several abstract test cases.

12.4 Assignment of Verdicts

12.4.1 The test suite specifier shall ensure that each abstract test case explicitly defines:

a) each sequence of test events to be associated with a “pass” verdict;

b) each sequence of test events to be associated with an “inconclusive” verdict;

NOTE – This verdict would be associated with sequences of test events representing behaviour from the IUT which although valid prevents the test purpose(s) from being accomplished.

c) all remaining sequences of test events to be associated with a “fail” verdict, either defined individually or categorized by using an unidentified test event.

12.4.2 The test verdict for a test case in the MPyT context shall only be assigned by the LTCF. The individual LTs are only concerned with the PCO(s) with which they are associated. Therefore, they are only capable of assigning a

preliminary result. When an LT exits in error or otherwise, it shall assign a preliminary result. When all LTs have completed their assigned behaviour the LTCF shall determine the verdict by evaluating all preliminary results.

12.4.3 The checking that is to be performed in a test case, for the validity of test events received from the IUT with respect to the relevant base specification(s), shall be specified explicitly within the abstract test case. The test suite specifier shall not assume that the test realizer or test laboratory will perform any checking of the test events against the base specification(s) other than that which is specified in the abstract test cases.

12.5 Abstract Test Suite specification conformance clause

The ATS specification shall include a conformance clause.

The conformance clause shall contain the following statement:

"The test realizer shall comply with the requirements of Recommendation X.293. In particular, these concern the realization of an Executable Test Suite based on each Abstract Test Suite.

Test laboratories running conformance test services using this Abstract Test Suite shall comply with Recommendation X.294."

12.6 Consistency with base specification

An ATS shall represent accurately the base specification(s) to which it tests conformance. If errors or ambiguities are discovered in a base specification during development of the ATS, the test suite specifier shall forward, to the proper ITU-T or ISO/IEC group, defect reports which identify the problems. If differences are discovered between an ATS and a base specification after the ATS is standardized, then the base specification shall have precedence in problem resolution.

NOTE – FDTs may facilitate validation of a test suite against a base specification.

12.7 Copyright

ATSs need to be processed by appropriate software tools and need to be exchanged between test suite specifiers and test realizers in a machine processable form. Nevertheless, once an ATS specification has been published, for example as an ITU-T Implementation Manual, the version of the ATS used by test realizers shall be semantically identical to the published version. This raises a copyright issue.

The following statement shall appear in the ATS specification, as a footnote on the first page of the ATS itself, referenced from the title of the ATS [e.g. Abstract Test Suite[1)]]:

"[1)] **Copyright release for this Abstract Test Suite**

Users of this <specification> may freely produce, reproduce and process machine-processable versions of this Abstract Test Suite so that it can be used for its intended purpose and may print out the Abstract Test Suite in a format given in this <specification> for the purpose of checking the correctness of the machine-processable version and making it human-readable."

The term <specification> may be modified suitably to reflect the exact form of publication, e.g. ITU-T Implementation Manual, International Standard, or Technical Report.

Also the words "Unless otherwise specified" shall be added before "no part of this publication may be reproduced...." in the copyright statement on the appropriate contents page.

13 Specification of a Test Management Protocol (TMP)

In the case of the SPyT Coordinated test method, the TCP are realized by the standardization of a TMP, as a separate part of the conformance testing specification. Similarly, a TMP may be used within an MPyT test method.

The TMP needs to be able to convey requests to the IUT to achieve the effect of an ASP and to convey back to the LT the record of observations of effects equivalent to the occurrence of ASPs. The UT should be an implementation of the TMP. Test cases shall be specified in the ATS specification for the purpose of testing that the UT conforms to the requirements of the TMP specification. Such test cases do not, however, contribute to the conformance assessment of the IUT.

If a TMP part of the conformance testing specification is produced, then a proforma shall be provided for a TMP implementation statement which shall include an entry for each of the TM-PDUs.

14 Information in an ATS specification concerning use of the ATS

14.1 An ATS specification contains a standardized ATS along with other information necessary to facilitate the production of an Executable Test Suite and run a test campaign.

The test suite specifier shall provide information in the ATS specification to assist the test realizer and test laboratory in their use of the ATS. This information shall include, but is not limited to, the following:

a) a mapping of the abstract test cases to the ICS proforma entries to determine whether or not each abstract test case is to be selected for the particular IUT; the mapping should be specified in a notation suitable for Boolean expressions;

b) the specification of a partial IXIT proforma for each ATS: this proforma shall contain a list of all parameters for which the test suite requires values; if any of the required parameter values are to be found in the ICS, the IXIT proforma entry for each such parameter shall reference the corresponding entry in the ICS proforma;

NOTE – Other aspects of the IXIT proforma are discussed in Recommendations X.290, X.293, X.294 and X.296.

c) a mapping of the abstract test cases to the partial IXIT proforma, to parameterize the test suite for the particular IUT; the mapping shall identify requirements for testing which may prevent test cases from being run against a particular IUT; the mapping should be specified in a notation suitable for Boolean expressions;

d) the order of listing the abstract test cases for the presentation of results in the Protocol Conformance Test Report (PCTR) (see 14.2);

e) any restrictions that there may be on the order in which the test cases may be executed;

f) identification of test cases or test groups which shall be realized in a Means of Testing claimed to conform to the standardized ATS specification;

g) the requirements on the TCP or a reference to the specification of the TMP [if applicable in the chosen test method(s)];

h) test cases for testing that the UT conforms to the requirements of the TMP specification [if applicable in the chosen test method(s)];

i) any necessary timing information.

14.2 The order in which the abstract test cases are to be listed in the PCTR, for the purposes of presentation of results shall be specified either explicitly in the ATS specification as a list, or implicitly (by default) as the order in which the abstract test cases are specified in the ATS. In addition, the ATS specification may provide, or reference, information on the status of each test case, in which case this information is to be preserved in the PCTR.

If any listed Basic Interconnection Tests are run as a preliminary stage in the conformance assessment process, the test verdicts associated with them shall be listed in the PCTR in the positions indicated for the corresponding capability or behaviour test cases (i.e. as if they were run as capability or behaviour tests).

14.3 The order in which the abstract test cases are listed in the ATS does not imply a precise order of execution. However, restrictions may be specified on the possible orders of execution (i.e. defining a partial ordering, e.g. it may be desirable to run a simple abstract test case before running more complex and detailed variants of that test case).

NOTE – Optimization of the order of execution of test cases in order to minimize execution time is considered to be a performance matter. This area is outside the scope of standardization.

15 Maintenance of Abstract Test Suite specifications

Once an ATS specification has been produced and is in use, it can be expected that errors and omissions in it will be detected by those who are using the ATS. The test suite specifier shall in such circumstances progress the updating of the ATS specification via the relevant defect reporting procedures.

In addition, from time to time, changes will be made to the base specification(s) to which the test suite relates. The test suite specifier shall ensure that the ATS specification is updated as soon as possible after changes to the relevant base specification have been ratified.

Appendix I

Applicability of the test methods to OSI protocols

(This appendix does not form an integral part of this Recommendation)

I.1 The Physical layer

In the Physical layer, test events include the act of measuring some characteristic of, or generating, a physical signal (e.g. an electrical or optical signal). Nevertheless, Recommendations X.290 to X.296 do not fully address the requirements of the Physical layer (e.g. no standardized test notation is provided for the Physical layer).

For the Physical layer functions of physical components, such as modems and transceivers, the Local test method is directly applicable.

The Remote and Coordinated embedded test methods are likely to be the most practical for local area networks.

In some cases for local area networks, sufficient control and observation above the IUT can be provided by the normal activity of a Data Link protocol. In such cases, the Data Link protocol implementation in the SUT provides the UT functionality and employs the Data Link protocol for test coordination. This is an example of the embedded variant of the Remote test method. If, however, there are no protocols used above the Data Link protocol, then this can be considered to be an example of the Coordinated test method.

I.2 Data Link and Media Access Control protocols

For testing Data Link protocols, the following points should be considered:

a) the Local test method is applicable only if the IUT has a standardized hardware upper interface;

b) the test methods are applicable only if an LT can be realized with control over Physical service primitives (or perhaps more realistically Physical and Data Link PDUs). This may be difficult for some types of subnetwork.

For Media Access Control protocol testing:

c) sufficient control and observation above the IUT may be provided by the normal activity of the Logical Link Control protocol. In such cases, the Logical Link Control protocol implementation in the SUT provides the UT functionality and employs the Logical Link Control protocol for test coordination. This is an example of the embedded variant of the Remote test method. If, however, there are no protocols used above the Logical Link Control protocol, then this can be considered to be an example of the Coordinated test method.

If single-protocol testing of a Data Link protocol is not possible, embedded test methods should be considered.

I.3 Network protocols

For Network protocols, the test methods to be used are dependent upon whether the IUT is an end-system or an open relay system.

It should be recognized that with some subnetwork technologies there are more than three protocols required to provide the Network Service. Each of these protocols may be tested separately or in any combination of adjacent protocols.

Considering the layer as a whole, both Network and Data Link ASPs are controllable and observable. Thus, for end-systems, all four SPyT ATMs (non-embedded variants) are applicable, but since the Data Link Service is not end-to-end, the LT has to be connected to the SUT over a single link.

For Network layer relay systems, MPyT test methods using two or more LTs are applicable. The number of LTs will vary based on the number of IUT components required to accomplish the test purpose, in conjunction with the number of IUT components supported by the SUT. For example, a simple test purpose might require two LTs, each communicating by means of a different service provider, exchange PDUs via the Network layer relay IUT. This example is illustrated in Figure I.1. As the relay system test purposes become more complex, LTs are added to communicate with the additional relay components.

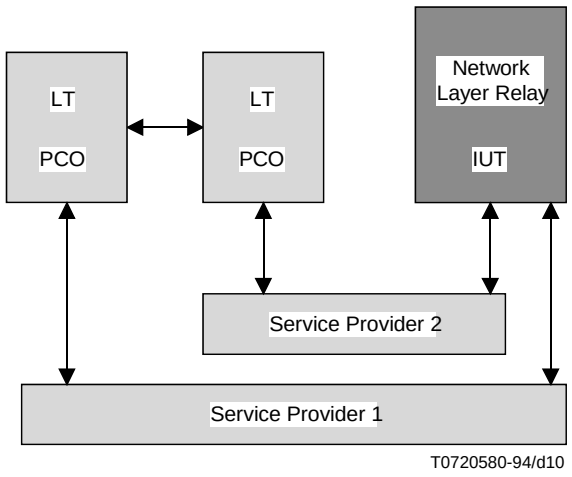

FIGURE I.1/X.291

Use of a simple MPyT test method for a Network Relay

I.4 Transport protocol

The Coordinated and Remote test methods, and the embedded variant of the Distributed test method are applicable to Transport protocol conformance testing.

I.4.1 Requirement for multi-user Single-Party Testing

In the testing of the multiplexing function of the Transport protocol, one LT may be used to communicate with more than one UT (i.e. multiple users). In this example, multiple Transport connections are multiplexed over one Network connection, and there is communication and coordination between the LT and each UT; each UT handles one Transport connection. This example is illustrated in Figure I.2. As the multiplexing becomes more complex more UTs are added to handle the new Transport connections.

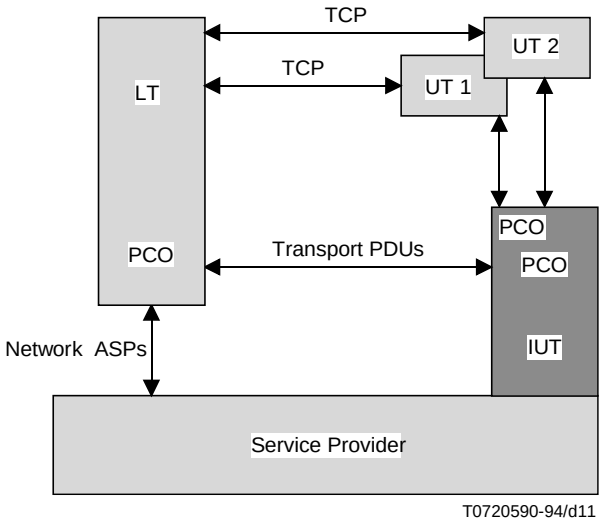

FIGURE I.2/X.291

Use of the Distributed test method in multi-user testing

I.5 Session protocol

The Coordinated and Remote test methods, and the embedded variant of the Distributed test method are applicable to Session protocol conformance testing.

For a large group of systems it will be appropriate to test the Session protocol in combination with the Presentation protocol and appropriate Application Service Elements (ASEs). Testing of the Session protocol should, therefore, be done in one of the two following ways:

a) as a single-protocol implementation, in order to test the provision of a general purpose Session service capable of supporting several different ASEs – the Coordinated test method is likely to be appropriate;

b) in combination with the Presentation protocol and ASEs, in order to test it in a specific Application context – the Remote test method and embedded variant of the Distributed test method are likely to be appropriate.

I.6 Presentation and Application protocols

I.6.1 General comments

The Presentation protocol and protocols for ASEs in a specific Application context are interrelated to a large extent. Invalid Application PDUs will (e.g. in the case of syntax errors) have to be detected by the Presentation protocol implementation, and, in the case of semantic errors, by the relevant ASE implementation. Real systems may choose to combine these functions.

It is therefore, in general, not feasible to test Presentation and Application protocols separate from each other.

I.6.2 Presentation

The ASPs are potentially observable and controllable to the same extent as for lower layers. Thus, all four SPyT ATMs (non-embedded variants) are theoretically applicable. However, the testing of Presentation protocol in isolation from an ASE is of limited value, because it could only test the protocol machine, leaving untested the more interesting aspect of the Presentation layer, namely the mapping between abstract and transfer syntaxes. Therefore, the testing of Presentation protocol embedded under Association Control Service Element (ACSE) and other ASEs in a specific Application context is preferred. Thus, the relevant applicable test methods are the Remote test method and the embedded variant of the Distributed test method.

I.6.3 Transfer syntaxes

Transfer syntaxes (e.g. ASN.1 encoded with the Basic Encoding Rules) are rather different from OSI protocol specifications with respect to conformance. In general, there would not be conformance testing of the encoding rules of a transfer syntax independent of the ASE using those rules. In any case, the transfer syntax encoding rules will be tested with the Presentation protocol, and the test methods appropriate to that protocol will be used.

I.6.4 Application

Conformance tests can be specified abstractly in terms of ASPs, whether or not there is any notion of a SAP associated with them. Thus, provided that there is some mapping between ASE ASps and effects which can be observed and/or controlled, tests can be specified in terms of those ASE ASPs. The observation and control of the ASPs may be indirect because of the nature of the mapping onto corresponding effects, but as long as that mapping is possible then tests specified in these terms can be run.

It is accepted that, in some circumstances, specifications for Applications, defining Application contexts, may specify non-protocol conformance requirements which have to be achieved as a result of protocol exchanges. However, these requirements should be kept quite distinct from the normal protocol conformance requirements, possibly even in separate specifications. Testing of non-protocol conformance requirements will in general require application-specific test methods, and so fall outside the scope of Recommendations X.290 to X.296.

When testing specific ASEs in an Application context that includes the ACSE, the PCO below the LT will be characterized by the set of possible ASPs that can occur at it. These will include both ACSE and Presentation ASPs.

I.6.5 MHS requirements for multi-user Single-Party Testing

In general, the Remote and Distributed test methods are applicable to MHS, depending on the role of the SUT (i.e. when the SUT is initiator then the Distributed test method is appropriate).

One aspect of the MHS protocols deserves special mention. This involves the receipt of a single message that should be delivered to more than one recipient, and generation of the appropriate separate responses to the originating source. Test cases for this situation need to use a multi-user variant of the Distributed test method. (See Figure I.3.)

I.6.6 Transaction Processing

Single association aspects of Transaction Processing can be tested in the SPyT context.

Multiple association aspects of Transaction Processing need to be tested in the MPyT context. For example, to test an IUT in the role of an intermediate system communicating with a superior and two subordinates in a dialogue tree, three LTs would be needed and it may not be necessary to use a UT. This is illustrated in Figure I.4. If however, a UT is required by the test purpose then it may need TCP with one of the LTs and this may be realized by a TMP. This is illustrated in Figure I.5.

I.7 Connectionless protocols

Since each test method described in Recommendations X.290 to X.296 is defined in terms of observation and control of ASPs and PDUs, and not in terms of connections, they are all applicable to the testing of connectionless protocols, taking into account the restrictions applying to each layer.

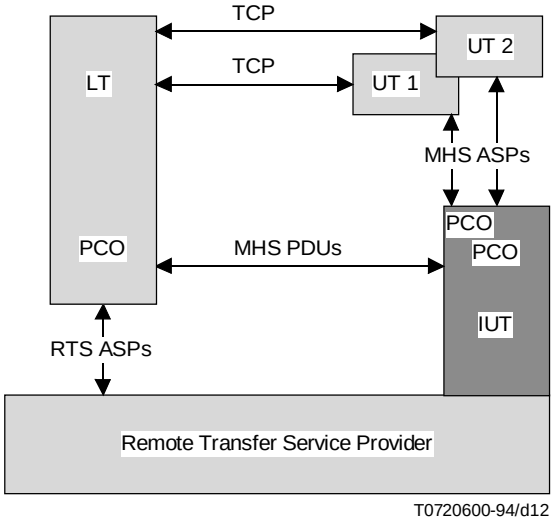

FIGURE I.3/X.291

Example of MHS requirement for multi-user testing

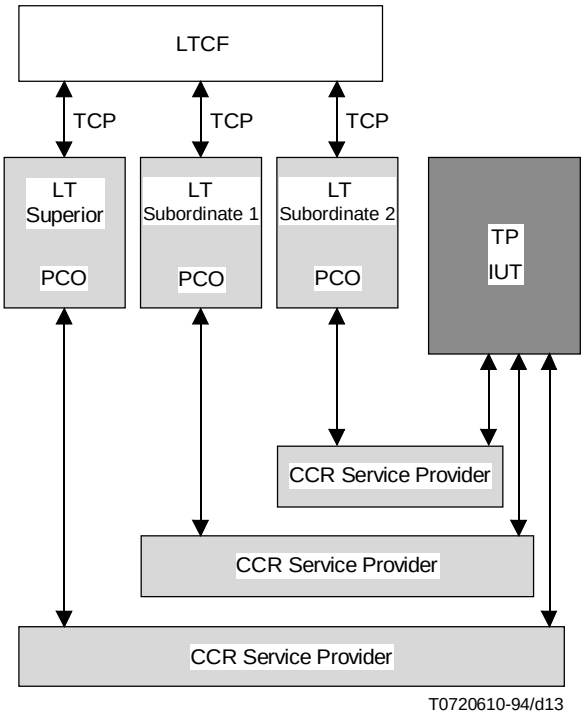

FIGURE I.4/X.291

Use of the MPyT test method for Transaction Processing

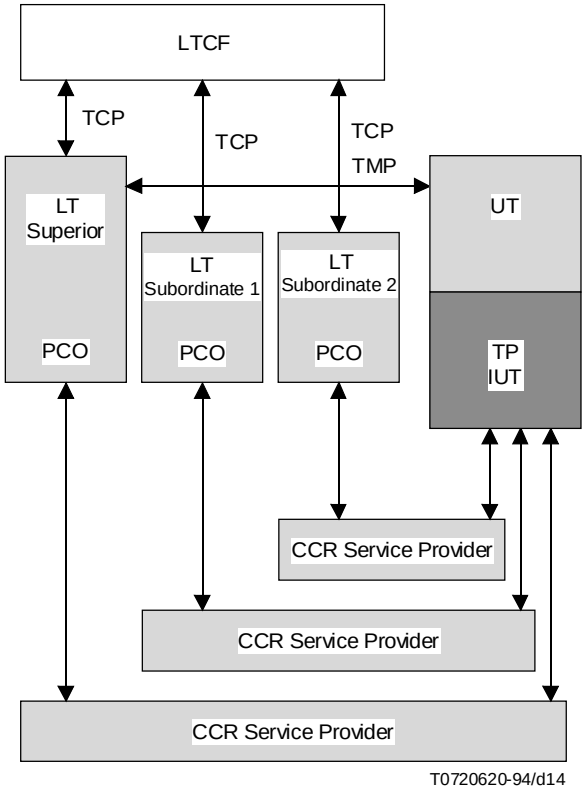

FIGURE I.5/X.291

Use of a TMP in an MPyT test method for Transaction Processing

Appendix II

Guidance for protocol specifiers to facilitate conformance testing

(This appendix does not form an integral part of this Recommendation)

II.1 Introduction

This appendix gives guidance, primarily for the specifiers of new specifications which specify protocols, to facilitate conformance testing by ensuring a very clear understanding of the conformance requirements.

The guidance in this appendix on implementation requirements and options should be read in conjunction with the requirements and guidance on ICS proformas in Recommendation X.296.

II.2 Guidance on scope

II.2.1 Precision in the scope clause sets the tone for precision in the rest of the specification. The requirements stated in the specification should be consistent with the scope and field of application and vice versa.

II.2.2 The scope should distinguish clearly between the following three types of information included in the protocol specification:

a) the definition of the procedures for communication to be followed at the time of communication;

b) requirements to be met by suppliers of implementations of the procedures;

c) guidance on how to implement the procedures.

Guidance on how to implement the procedures does not constitute additional requirements nor does it have any bearing on conformance. If such guidance is included, the scope should make these points and indicate how guidance can be distinguished from the requirements of the specification. This distinction is much easier to make if guidance is separated from requirements.

II.2.3 It should be clear to whom the specification applies.

II.2.4 It should be clear under what conditions the specification applies.

Protocol procedures apply between pairs of communicating parties at the time of communication. If there might be any ambiguity over which communicating parties are involved, this should be resolved in the scope.

It is best if protocol specifications are written in such a way that the requirements are to be met by a single communicating party (the "first" communicating party for this purpose) for the benefit of one or more other communicating parties (the "second" communicating parties). Then when two (or more) communicating parties are all expected to communicate in conformance with the specification, the specification is first applied to one party, treating it as the "first" one, and then to the other(s) in turn. This ensures that if the procedures are violated, it is clear which party is at fault.

II.2.5 If any guidance is given about factors which are not standardized definitively, the scope should make it clear that any such guidance can be ignored without affecting conformance.

II.2.6 The aspects which are excluded from the scope should be identified clearly.

Not all factors relevant to the procedures or to products which implement them need to be standardized; indeed it is often desirable to leave some implementor freedom. For instance, it may be desirable to omit in a protocol specification any requirements for explicit values of timeouts, but to give guidance instead.

The scope should make it clear which aspects are standardized definitively, which are covered by guidance but not by any requirements, and which are excluded from consideration by the specification. Any aspects which one might think would be covered, on the basis that they are closely related to aspects which are standardized, need explicit mention.

II.2.7 All options should, if possible, be identified clearly in the scope.

Options are one of the most troublesome, but unfortunately necessary, parts of protocol specifications. They fall somewhere between what is standardized and what is not. They will be covered in greater depth below. What is important is that major options are not buried deep inside the specification but are declared clearly at the beginning. If the number and detailed nature of the options makes this impractical, one should seriously ask whether such complexity is really necessary. Can detailed options be grouped together in some way (e.g. classes) to simplify the specification?

II.2.8 The scope clause should be reviewed after considering the rest of the specification.

It is often not possible to satisfy some of the above suggestions until the rest of the specification has been considered. Therefore, it is generally necessary to return to the scope, to check that it really does agree with the contents of the specification. It is common to find that clauses quite outside the scope have been included.

II.3 Guidance on normative references

II.3.1 Specifications which specify OSI protocols should refer to the OSI reference model, the relevant specifications which define OSI services and to any relevant specifications for appropriate conventions, guidelines, or FDTs.

II.3.2 It should be made clear whether conformance to the specification which specifies the protocol requires conformance to any part of any other specification.

II.3.3 It should be made clear whether a reference is to a particular version or edition of the referenced specification or to each successive version or edition.

Normally, the latest version and edition is required, but this can cause problems as changes to another specification might affect conformance to the one in hand.

II.4 Guidance on requirements and options

II.4.1 The status of every requirement should be unambiguous.

Optional and conditional conformance requirements are common but there is the possibility of interpreting too much as being optional. As an example, it may be optional to implement the sending of a given PDU. However, if that PDU is part of a confirmed service then there is usually a requirement to accept the response. Such a conformance requirement on the response is conditional and should be explicitly stated as such. It would be a mistake to say that if the sending of the PDU is optional, then the reception of the response is also optional.

II.4.2 It should be possible for an instance of communication to conform with all the mandatory dynamic conformance requirements.

II.4.3 The conditions under which conditional requirements apply should be spelt out clearly.

II.4.4 It should not be impossible for the implementor or supplier to know what these conditions are.

II.4.5 There should be no possibility of confusion between what is optional dynamically and what is optional statically.

There may be mandatory static conformance requirements for the support of features whose use at the time of communication is optional. Conversely, a message whose use is mandatory in a given context at the time of communication may be part of a protocol mechanism whose support is optional statically.

II.4.6 If the specification contains a “shopping list” of options, and there are restrictions on the allowed combinations of such options, then the restrictions should be specified clearly. These should include identification of any mutual exclusions and any minimum and maximum limits to the allowed range of options.

II.4.7 If the specification does not give any rules for selection of options, it should be made clear in the scope that only the total range and individual options are standardized, but not the selection.

II.4.8 “Legitimizing options” should be avoided. These are options which allow alternative and incompatible versions of the same thing to claim conformance to the same specification. Although they do not of themselves prevent an objective understanding of conformance, they may frustrate the aims of OSI.

II.4.9 There should be no options which give the implementor permission to ignore important requirements of the specification. Such options devalue the specification and the meaning of conformance to it.

II.4.10 If there are prohibitions in the specification, they should be sufficiently precise to be meaningful.

Many specifications have clauses which say in effect "do all of this and nothing else". Such prohibitions may be meaningless, because every protocol conveys some information which is not standardized, the so-called "user data", and every standardized product has attributes which are not standardized, e.g. weight. It may be difficult to draw a clear objective distinction between things the specification cannot forbid and those which the writers of the specification want to forbid, unless the prohibitions are stated explicitly.

II.5 Checklist for conformance clauses

The following is a checklist for what should be included or referenced, if relevant, in each conformance clause to specify the static conformance requirements:

a) which conformance classes are required to be supported;

b) which role(s) are required to be supported;

c) which functional units and PDUs (or operations and notifications) within those functional units are required unconditionally because they are essential to the integrity of the protocol or information object;

d) which functional units and PDUs (or operations and notifications) within those functional units become mandatory as the result of evaluation of a condition based on a given conformance class (i.e. if the conformance class is supported then they become mandatory);

e) which functional units and PDUs (or operations and notifications) within those functional units become mandatory as the result of evaluation of a condition based on a given role (i.e. if the role is supported then they become mandatory);

f) which PDUs (or operations and notifications) become mandatory as the result of evaluation of a condition based on a given optional functional unit being supported (i.e. if the functional unit is supported then they become mandatory);

g) in the case of a protocol being used to provide a service, which PDUs become mandatory as the result of the evaluation of a condition based on a given element of that service being supported (i.e. if that element of the service is supported then they become mandatory);

h) which PDUs (or operations and notifications) become mandatory as the result of the evaluation of a condition based on some combination of conformance class, role, functional unit and element of service (i.e. if the combination is supported then they become mandatory);

i) which elements of underlying services are required for each conformance class, role, functional unit, or PDU.

Similar considerations apply to the support of parameters and parameter values, but these are usually best specified in tabular form in the ICS proforma, rather than in words in the conformance clause. Indeed, some of the more complex conditions are easier to specify in the ICS proforma, using a conditional expression and logical operators, rather than in words in the conformance clause. If, however, the ICS proforma does amplify the conformance requirements stated in the conformance clause, rather than just repeat them in a different form, then the conformance clause should recognize this and refer to more detail being provided in the status column of the ICS proforma.

Given such a specification, all other functional units, PDUs, operations and notifications are optional.

II.6 Guidance on PDUs

II.6.1 The permitted set of PDU types and parameter encodings should be stated clearly.

II.6.2 The permitted range of values should be stated clearly for each parameter.

II.6.3 All values which fall outside the stated permitted range should be stated explicitly to be invalid.

If not, some people will argue that such values are undefined but allowed, whilst others will argue that they are invalid.

II.6.4 It should be clear whether or not undefined PDU types are allowed.

It is safer to declare all undefined PDU types to be invalid.

II.6.5 Critical undefined values should be mentioned explicitly in the scope as being undefined.

II.6.6 There should be a defined procedure to be followed by the first communicating party in each case of it receiving an invalid or undefined PDU type or parameter.

II.6.7 It should be possible to detect whether the defined procedure has been followed in such cases. If it is not clear, then this should be because it does not matter whether or not the procedure has been followed.

Sometimes the procedure to be followed upon receipt of an invalid PDU is intentionally the same as when some valid PDUs are received in the same circumstances. For example, the procedure might be to do nothing until one specific type of PDU is received, everything else being ignored. In such situations, it probably does not matter that the error has apparently gone undetected. In other situations, it may be the intention that error cases should be given special treatment, but the procedure has been poorly chosen. In these latter situations, it may be that the action cannot be distinguished from that in the non-error cases.

II.6.8 If, in the encoding of PDUs, there are any fields declared to be "reserved", then there should be a clear statement of what values, if any, are allowed or disallowed in these fields.

II.6.9 If interrelated parameters can be carried on separate PDUs, then the set of permitted relationships between the values of these parameters should be precisely and clearly defined.

II.6.10 If the parameter encoding allows for parameters to be specified in any order, and the PDU format places restrictions on the permitted orders, then these restrictions should be clearly stated. It should be recognized that if many different orders are permitted, then a large representative sample of different orders ought to be tested. The added complexity of testing conformance should, therefore, be adequately compensated by some advantage in allowing this freedom.

II.6.11 The order in which the bits, octets, etc. should be carried in the underlying protocol should be stated clearly.

For example, should a two octet integer travel most or least significant octet first? It is surprising how often such simple causes of ambiguity are overlooked.

II.6.12 The relationship between service data units and PDUs should be defined clearly.

II.7 Guidance on states

II.7.1 Protocol procedures are often defined using a finite state approach, whether formalized or not. The specification of these states is often incomplete.

II.7.2 Each state should be defined clearly.

II.7.3 If there are events which can occur only in a subset of the possible states, then possible occurrence of an event should be distinguished from valid occurrence.

II.7.4 The required actions and state transitions should be defined for each possible state/event pair. In particular, they should be defined for possible but invalid state/event pairs.

II.8 Guidance on FDTs

II.8.1 The following guidance applies only to those specifications which include a formal description. Precise, unambiguous specifications can be written without the aid of an FDT but, in complex specifications such as protocols, formal descriptions are recommended. It should, however, be realized that they can create problems themselves in relation to conformance.

II.8.2 It should be clear whether the formal description forms a normative part of the specification or is provided only for guidance.

It is very important to have a clear understanding of the status of the formal description. Ideally there should be no discrepancies between the text and the formal description, but because this is very difficult to achieve in practice it is important that the reader knows which takes precedence. If the formal description is provided only for guidance, it cannot define conformance requirements.

II.8.3 The FDT should be a standardized one and should be properly referenced.

II.8.4 If the formal description defines requirements, but not all the requirements of the specification, then it should be stated clearly that the text includes requirements which are not covered by the formal description and these additional requirements should be identified clearly.

II.8.5 If the formal description defines requirements, and it also defines an allowed way of implementing some aspects of the protocol, but there is intended to be freedom for the implementor to implement those aspects in some other way, then this constitutes over-definition. This is all too common in formal descriptions, and creates difficulties in relation to conformance. If the formal description is an essential part of the specification, then text should be provided to qualify it, indicating where such over-definition exists and what the real requirements are.

The problem usually arises because the formal description describes the internal behaviour of an idealized implementation, rather than the observable external behaviour that is required. It is only the observable external behaviour which can be tested, and therefore it is only this which should constitute requirements for conformance purposes. It may well be that a different FDT should be used for defining the requirements from that used to provide guidance to implementors.

II.9 Miscellaneous guidance

Information which may appear obvious should nevertheless be stated.

If something is omitted because it is “obvious”, some readers will assume it is required because it is “obvious”, but others will assume that it is omitted to provide freedom for implementors. For example, does the existence of a checksum imply that it has to be checked?

Appendix III

Relationship between Recommendations X.290 to X.296 and Recommendation X.200 service notation

(This appendix does not form an integral part of this Recommendation)

III.1 Figure III.1 illustrates an IUT which spans a whole OSI layer, showing the equivalence of X-ASPs to the (N–1)-Service and the Y-ASPs to the (N)-Service consistent with the use of (N – 1) and (N) in Recommendation X.200 related to an (N)-Protocol entity.

III.2 Figure III.2 illustrates an IUT for a protocol which does not span the whole of an OSI layer. An example of the IUT (3) would be an implementation of Commitment, Concurrency and Recovery (CCR), in which case (1) represents Presentation, (2) represents ACSE, and (4) could represent Transaction Processing. In this case the lower layer boundary of the IUT partly coincides with an OSI layer boundary (i.e. Presentation Service) and partly with a non-layer boundary (i.e. Association Control Service). The ASPs at this combined boundary to the IUT are represented by the X-ASPs in the abstract testing methodology, whereas only the Presentation service can be represented by the (N – 1)-Service notation of the OSI basic reference model. Furthermore, the Y-ASPs in this case are at the CCR service boundary, which is not an OSI layer service boundary, and therefore cannot be represented by the (N)-Service notation.

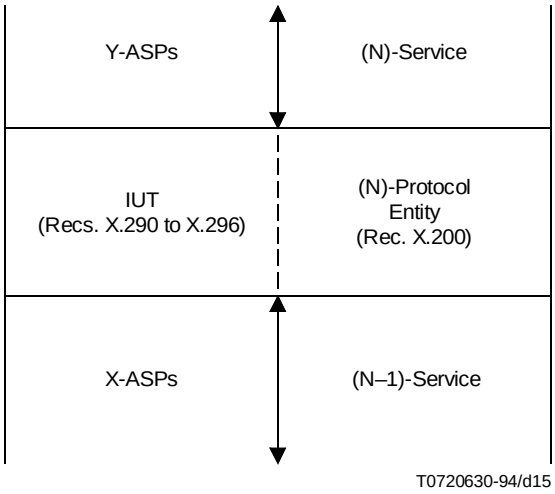

FIGURE III.1/X.291

Service notations for an (N) entity

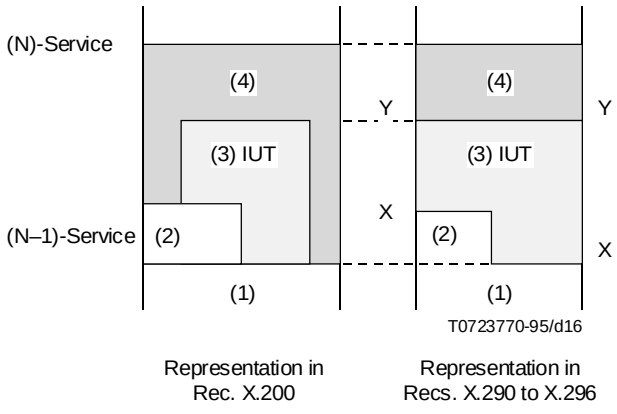

FIGURE III.2/X.291

Two representations of an IUT within an OSI layer