

## Annex A Test Notation

### A.1 Introduction

This annex is an integral part of this Recommendation and describes the notation used in the test suite manuals.

The test notation described here is based on the test notation called Tree and Tabular Combined Notation (TTCN) that has been developed jointly by ISO and CCITT.

The notation described in this Recommendation is derived from an early form of TTCN and has been developed specifically for use in the MHS conformance testing specifications.

Each of the MHS test suites is specified in five parts:

- Declaration Part
- Dynamic Part
- Constraints Part
- Test Case Identification
- Cross-References

### A.2 Declaration Part

The Declaration Part declares the environment and objects used in the test suites and is composed of 7 sections:

- Test Configurations
- Test Suite Parameters (TSPs)
- Service Access Points (SAPs)
- Abstract Service Primitives (ASPs)
- Protocol Data Units (PDUs)
- Timers
- Abbreviations

#### A.2.1 Test Configurations

The points of control and observation are declared in this section.

#### A.2.2 Test Suite Parameters

Every MHS Test Suite has a set of parameters whose values are fixed prior to testing and which are used to define a specific testing environment.

TSPs are declared in tabular form as shown in Figure A-1/X.403.

Figure A-1/X.403 Test Suite Parameters.

By convention the name of each Test Suite Parameter in the MHS test suites is of the form:

TSP\_<name>

### A.2.3 Service Access Points (SAPs)

Service Access Points are used as points of control and observation in the MHS Test Suites and are declared in tabular form as shown in Figure A-2/X.403.

Figure A-2/X.403 Service Access Points.

By convention the name of a SAP in the MHS Test Suites is generally one capital letter, such as T, U, V (for tester SAPs) or I, J, K (for IUT SAPs).

### A.2.4 Abstract Service Primitives

Each ASP type and its associated parameters used in a test suite is declared in tabular form as shown in Figure A-3/X.403.

Figure A-3/X.403 Abstract Service Primitives.

The name of the ASP is specified in the "ASP" field and is derived from the corresponding name in the X.400 Series of Recommendations. The SAP at which the ASP occurs is specified in the "SAP" field. The parameters of the ASP are declared in the "NAME" column together with information in "RANGE OF VALUES OR TYPE", "COMMENTS" and "Conditional/Mandatory" columns.

Since there are no IPMS(P2) ASPs defined in the Recommendations, in order to describe conformance tests it has been necessary to construct hypothetical ASPs at the upper boundary of the User Agent Layer. This does not imply, however that manufacturers are required to implement these ASPs within their systems. It serves only to formalize the requirements for observation and invocation of IPMS service elements by the use of these new ASPs. The relation between IPMS service elements and the actual behaviour of the IUT has to be specified in the implementation-dependent PIXIT.

A.2.5 Protocol Data Units

The PDU types used in test suites are declared in the form of tables as shown in Figure A-4/X.403. These PDUs are not defined explicitly in the test suite, but are given a precise reference to the full definition in the X.400 Recommendations, in the type name section of the table.

Figure A-4/X.403 Protocol Data Units.

A.2.6 Timers

This section declares the timers to be used. Timer values are expressions in terms of Test Suite Parameters, and are fixed for the whole test suite. Timer values are declared in tabular form as shown in figure A-5/X.403.

Figure A-5/X.403 Timers.

A.2.7 Abbreviations

Abbreviations used in the Test Suite are defined in the form of a table as shown in Figure A-6/X.403.

Figure A-6/X.403 Abbreviations.

A.3 Dynamic Part

The Dynamic Part defines the test cases of a test suite in terms of trees of behaviour.

Sections A.3.1 and A.3.2 describe generally how trees of behaviour are defined.

Section A.3.3 describes the content and use of Defaults Library.

Section A.3.4 describes the content and use of Test Step Library.

Section A.3.5 describes how each test case in the main body of a test suite is specified.

A.3.1 Proforma table for Test Behaviours

Figure A-7/X.403 Behaviour Description.

<title> BEHAVIOUR:

Title of the behaviour: DEFAULT for the Default Library;  
DYNAMIC for the Test Step Library and test cases.

IDENTIFIER:

This provides a unique identifier for the behaviour  
description.

DEFAULTS:

This lists the identifiers of default behaviour descriptions  
which are to be used in conjunction with the Dynamic behaviour  
shown in the "BEHAVIOUR DESCRIPTION" part.

BEHAVIOUR DESCRIPTION:

Test behaviour is defined using a tree notation as described  
in A.3.2.

LABEL:

The LABELS column may be used to identify events. Branches  
between events (i.e. "GO TO") are specified by " > Label" in  
the behaviour tree.

CONSTRAINTS REFERENCE:

For each ASP event of a behaviour tree line, this column gives  
the reference to the specific ASP value defined in the  
Constraints Part.

COMMENTS:

This column provides comments which ease understanding of the  
events. Additional comments may be given in the "Extended  
Comments" area. This column can also be used to identify test  
PDUs associated with test events.

## RESULT:

This column indicates which test events generate test verdicts.  
Values of test verdicts are:

- pass: no misbehaviour of the IUT is detected.
- fail: misbehaviour of the IUT is detected.
- inconclusive: the observed behaviour does not allow the assignment of a pass or fail verdict.

### A.3.2 Tree notation for test behaviours.

Trees of behaviour are defined in terms of events which are generally of the form

<SAP>!<event>

or of the form

<SAP>?<event>

The <SAP> is the point of control and observation at which the <event> occurs. The SAPs used are those declared in the Declaration Part.

The "!" symbol indicates that the event is sent from the SAP and "?" indicates that the event is received at the SAP.

The <event> can be

- an ASP event
- a timer event
- a OTHERWISE pseudo-event.

#### A.3.2.1 Single ASP events.

If the <event> is an ASP event then the names for the ASPs are those specified in the Declaration Part (the value is specified as a reference in the CONSTRAINTS REFERENCE column).

Example line for an ASP event:

I?DELind

This means that a Deliver Indication is received at the IUT's SAP I.

#### A.3.2.2 Single Timer events.

If <event> is a timer event then it is of the form

<operation> <parameters>

The "start" operation can take one of two forms:

Start <timer type>

Start (<timer type>,<timer id>)

Where <timer type> is defined in the Declaration Part and has a fixed value associated with it defined in terms of TSPs. The <timer id> allows a name to be attached to an instance of a timer type.

The other operations are:

Cancel: cancels a running or suspended timer

Suspend: suspends a running timer

Resume: resumes a suspended timer

Timeout: expiration of a running timer

These operations take one of two forms:

<operation> <timer type>

<operation> <timer id>

where <operation> denotes the operation. When the timer was started using the form "Start <timer type>", the form "<operation> <timer type>" must be used; when the timer was started using the form "Start <timer id>", the form "<operation> <timer id>" must be used.

Example:

!!Start T/I-timer\_1

means that at the IUT's SAP I the T/I-timer\_1 (e.g. for a transmission time for a UAPDU to be transferred from the tester to the IUT's user) is started.

I?Timeout T/I-timer\_1

means that at the IUT's SAP I the timeout of the above timer is received.

### A.3.2.3 Single OTHERWISE events.

If <event> is the OTHERWISE pseudo-event, this indicates an unspecified event.

Example:

T?OTHERWISE

Means that at the tester's SAP T an unspecified event

is received.

#### A.3.2.4 Trees of behaviour

Trees of behaviour combine events in two ways:

- as sequences of events
- as alternative events

The two combination kinds are distinguished by indented and vertical alignment respectively.

Example of a sequence of events:

```
!!SUBreq
I?SUBcon
T?TRNind
```

This means that first at the SAP I a Submission Request is sent, then at the same SAP a Submission Confirmation is received, after which a Transfer Indication is received at the tester's SAP T.

Example of alternative events:

```
T?DELind
T?Timeout I/T-timer
```

This means that at SAP T either a Deliver Indication is received or alternatively the timeout is received there.

To build up a complex behaviour tree, the two kinds of combination can be mixed.

Example:

```
!!SUBreq
I?SUBcon
T?TRNind
T?DELind
T?Timeout I/T-timer
```

This means that after sending a Submission Request at I, either a Submission Confirmation is received at I, followed by the receipt of a Transfer Indication at T, or a Disconnect Indication is received at T.

#### A.3.3 Defaults Library

General default behaviours which are used by several test cases are defined in the Defaults Library using the format shown in figure A-8/X.403. The name of the default is of the form:



LIB\_<name>

or

LIB\_<name> [X]

where X is a place holder which is replaced by an actual SAP when applying the default element in a particular Test Case.

Note:

Where particular default behaviour applies to a single test case only the behaviour table is associated with that test case and the identifier is not prefixed with "LIB\_".

Figure A-8/X.403 Default Behaviour.

#### A.3.4 Test Step Library

Where a sequence of test steps is of use in several test cases they can be included in the Test Step Library and given a name of the form:

LIB\_<name>

Note:

Where a test step applies to a single test case the behaviour table is associated with that test case and the identifier is not prefixed with "LIB\_".

Figure A-9/X.403 Test Step Behaviour.

#### A.3.5 Test Case

Each test case in the main body of the test suite is described in terms of three headings (a) - (c), and a behaviour tree (d)

##### (a) Test Reference and Test Identifier

These elements give a unique reference and identifier for each test case and are described fully in section A.5.

##### (b) Summary

A brief overview of the purpose of the test is provided.

##### (c) Test Description (optional)

This provides an informal description of the actions and events that should take place during the test and an informal verdict criteria.

##### (d) Behaviour Tree

Dynamic behaviour is described using the tree notation defined in section A.3.2.

Figure A-10/X.403 Test Case Behaviour.

Note 1: In this field all Default Library Identifiers used are inserted. Where necessary, the SAP at which they are applied is also identified. If for example the field contains the entry:

LIB\_unexpected [T]

it means that the subtree associated with this Default Behaviour is considered to be associated with the SAP T.

Note 2: Test Step Library behaviour is included in the behaviour tree using the following notation:

+ <Test Step Library Identifier>

Note 3: The behaviour tree of every Test Case provides the verdicts pass, fail, and where appropriate inconclusive.

Note 4: When using Default Library elements it is possible that some of the verdict alternatives are "hidden" in the Default Library element.

#### A.4 Constraints Part

The Constraints Part of a Test Suite specifies the values and their encoding of all instances of ASPs, Test PDUs, Base PDUs and Library Components. The Constraints Parts is divided into the following sections:

- Introduction to Constraints Part
- ASP Constraints
- Test PDU Constraints
- Base PDU Constraints
- Components Library

Figure A-11/X.403 The structure of the Constraints Part.

#### A.4.1 ASP Constraints

Values of ASPs are defined as specific instances of a generic ASP.

##### A.4.1.1 Specification of a "Generic" ASP

A generic ASP is defined using the format shown in Figure A-12/X.403.

Figure A-12/X.403 Generic ASP specification.

The "FIELDS" column is used to list all the parameters of the ASP.

The "VALUE or REFERENCE" column is used to specify a value for each parameter and this can be done in four ways:

- (a) As a reference which can be a TSP name or a library component name.
- (b) As an explicit value.
- (c) As "-" to indicate that this parameter may be omitted in specific instances of this ASP.

(d) As "?" to indicate that for "request" ASP's this parameter must have a value defined in a specific instance if it is a component of interest.

A.4.1.2 Specification of ASP Instances

Specific values of ASPs are defined using the tabular format shown in Figure A-13/X.403.

Figure A-13/X.403 Specific ASP value.

The "INSTANCE NAME" column is used to identify specific instances of the ASP used in the test suite.

The "MODIFIED PARAMETER" column identifies, for "request" ASP's those parameters whose values are to be modified from the generic ASP specification, and for "notification" ASP's those parameters whose values are to be checked.

The "VALUE or REFERENCE" column can contain either specific values or references to library components ASPs or test PDUs.

A.4.2 Specifying PDU values

The MHS test suite contains a large number of test PDU values. Each PDU is defined in terms of modifications to one of the small number of "base" PDUs.

For convenience commonly used PDU components are defined in a library and are referenced by test PDUs and base PDUs.

A.4.3 Base PDUs

A.4.3.1 Base PDU specification

Base PDUs are not themselves used as test PDUs but they serve as a basis from which to derive the test PDUs. Usually only a few Base PDUs need to be specified.

The name of a Base PDU is of the form:

BASE\_<PDUtypename>\_<number>

An example of a Base PDU is shown below.

Example:

The value or value reference of each element of the structure is specified within square brackets ("[" and "]") under the VALUE or REFERENCE heading.

When specifying the encoding of a PDU for encoding/decoding tests, two additional columns are used to specify the ID Code [ID] and Length Indicator [LI] of each element of the PDU. The format for doing this is shown in the example below.

The values of ID and LI can be specified explicitly to allow invalid and various forms of valid codings to be defined. The mnemonic "LI" is used to indicate that any valid encoding of length is allowed.

#### A.4.3.2 Identifying the components to be modified

A component which is to be replaced in a PDU is identified by a path through the declaration of the PDU. The path is written as a list of elements, each separated from the next by a ".". The elements in the list can be labels which appear in a BASE PDU, components which appear in the left-hand side of a labeled declaration, or components which appear in the left-hand side of the expansion of a library reference in the right-hand side of a declaration.

For example, consider the following definitions:

In order to reference "a", the path would be instance\_1.a.  
In order to reference "e", the path would be instance\_1.b.d.e.

#### A.4.4 Test PDUs

Test PDUs are defined in terms of operations on Base PDU's. These operations refer to Library Components, TSPs or specific values.

There are two kinds of test PDU:

- PDUs sent by the tester (IUT as recipient)

By convention the names of these PDUs are of the form

<PDU name>\_x\_<number>

where x is the number of the base PDU from which the test PDU is derived.

- PDUs received by the tester (IUT as originator)

By convention the names of these PDUs are of the form

<PDU name>\_0\_<number>

where "0" indicates that these test PDUs are not derived from a base PDU.

#### A.4.4.1 Test PDUs sent by the tester

A test PDU sent by the tester to the IUT is normally constructed from a Base PDU by means of the REPLACE operation.

The specification has the form:

For the conventions of value assignments see section A.4.6.

Example:



To construct invalid components in test PDUs to be sent by the tester, the abstract REDEFINE operation is sometimes used. It is used together with the REPLACE operation in the following form:

The scope of the newly defined type is restricted to the PDU definition containing the REDEFINE operation.

Note that if the <value> is a reference to an element defined elsewhere (i.e. a TSP or a Library Component), then the new type definition does not affect the referenced element itself but only its usage in the actual PDU.

Example:

The error to be constructed here is the wrong tag of the ORName type (the correct tag would be [APPLICATION 0]). The scope of the erroneous type-definition constructed by "REDEFINE" is restricted

to all occurrences of ORName in the definition of IM\_UAPDU\_1\_3. This means that L\_ORDescriptor\_1 is used here with the modified ORName type, whereas the usage of this library component in other PDUs or components remains unaltered.

#### A.4.4.2 Test PDUs received by the tester

For received PDUs normally only a portion of the PDU relates to the purpose of the test.

A component of interest is identified and its value assigned using the techniques described in A.4.3.

The specification scheme has the following form:

Example:

#### A.4.5 Component Library

Components of PDUs are defined in the library and are referenced in Base PDU specifications, Test PDU specifications and by other library components.

The name of a Library Component is of the form:

L\_<ASN.1 type name>\_<number>

and is specified using the techniques described in A.4.3.

Example:

#### A.4.6 Value Conventions

The following conventions are used when defining values or value references for PDU components.

Value references identify components defined either within the Component Library or within the Test Suite Parameters section. CharacterString Values can specified within double-quotes (eg "abc"); Bit String Values are specified within single-quotes (eg '0A'H or '0001'B; hexadecimal or binary notation); Integer Values are specified as numeric characters (eg 2); sets and sequences of values are specified within curly brackets separated by comma (eg {"abc",'0A'H}).

For PDU's sent by the tester :

[ ? ] indicates that the value has no influence on the test and may be anything that is legal according to the relevant service or protocol standard.

[ - ] indicates that the parameter shall be absent.

[ \* ] indicates that the value is to be inserted by the tester before test execution.

For PDU's received by the tester :

[ ? ] indicates that the tester need make no verification of the value of the parameter .

[ - ] indicates that the tester shall check that the parameter is absent.

Note that the "?" and "-" symbols in value assignments of PDU components have got other meanings than "?" and "-" in generic ASP schedules.

## A.5 Test Case Identification

Test cases are completely identified using four components:

- a test group identifier.
- a subgroup identifier.
- a validity identifier.
- a test number.

These four components are specified in two equivalent ways:

- As a Test Reference where the four components are textual and descriptive.

Example:

OriginalEncodedInfoTypeIndication/Recipient/Valid/2

- As a Test Identifier where the four components are numeric and concise.

Example:

307.2.1.2

### A.5.1 IPMS(P2) and MTS(P1) identification

#### A.5.1.1 Test Groups

Number ranges have been allocated for the test groups as shown below:

Initial Tests	001 - 099
X.409 Tests	100 - 199
Protocol Elements Tests (for frequently occurring Elements)	200 - 299
X.400 Service Elements Tests	300 - 399
Additional Tests	400 - 499

#### A.5.1.2 Subgroups

Numeric identifiers have been allocated to the test subgroups as shown below:

Originator	1
Recipient	2
encode	1
decode	2
Relay	3
Relaying-Recipient	4
Relaying-Originator	5

#### A.5.1.3 Validity identifiers

Test cases which exercise valid behaviour are distinguished from those which exercise the IUT's reaction to invalid behaviour using the numeric identifiers shown below:

Valid	1
Invalid	2

#### A.5.1.4 Test case numbers

Test cases for a particular group/subgroup/validity are numbered sequentially.

### A.5.2 RTS Identification

#### A.5.2.1 Test Groups

Number ranges have been allocated for the test groups as shown below:

Association Establishment	1
Association Release	2
Data Transfer	3
Association Recovery	4
X.409 Tests	5

#### A.5.2.2 Subgroups

Numeric identifiers have been allocated to the RTS subgroups as shown below:

Initiator	1
Responder	2
Sender	1
Receiver	2

### A.5.2.3 Validity identifiers

Test cases which exercise valid behaviour are distinguished from those which exercise the IUT's reaction to invalid and inopportune behaviour using the numeric identifiers shown below:

Valid	1
Invalid	2
Inopportune	3

## A.6 Cross Referencing

### A.6.1 Cross Reference Numbering

The MTS(P1) and IPMS(P2) test suites contain a cross referencing system for the ASPs, test PDUs and library components. The cross referencing appears in the left and right margins of the test suite as shown in Figure A-14/X.403.

Figure A-14/X.403 Cross referencing.

Numbers in the left hand margin of the test suite are in sequential order and are "place identifiers". They occur whenever an ASP, test PDU or library component occurs in the test suite.

Whenever an ASP, test PDU or library component occurs, numbers are also placed in the right hand margin. These numbers are forward and backward references to the place identifiers of the other occurrences of the ASP, test PDU or library component.

Where a forward or backward reference can not be found then a dot (".") is printed in the right hand margin. This should not occur in fully defined test suites.

Where a line in the test suite contains more than one ASP, test PDU or library component, the cross references for each item in the line are separated by vertical bars (" ") in the right hand margin as shown in Figure A-15/X.403.

Figure A-15/X.403 Multiple cross references.

## A.6.2 Cross Reference Listing

At the end of MTS(P1) and IPMS(P2) test suites there is a separate cross reference listing of all the ASPs, test PDUs and library components together with the place identifiers of all their occurrences in the test suite.

Example:

```
      :  
      :  
IM_UAPDU_1_14  586 1467  
IM_UAPDU_1_15  587 1470  
      :  
      :
```

The numbers on the right side indicate the places where the item occurs in the test suite.

## Annex B IPMS(P2) PICS Proformas

### B.1 General

As a prerequisite to conformance testing the supplier of an IPMS(P2) implementation must provide a Protocol Implementation Conformance Statement (PICS).

The proforma IPMS(P2) PICS contained in this Annex specifies the information to be supplied.

This information is needed for test case selection. Suppliers should note that tests will be performed to check that services shown as not supported are in fact not present rather than improperly implemented.

The IPMS(P2) PICS is in two parts:

- a part requesting information concerning the support of service elements,
- a part requesting information concerning the support of protocol elements.

Information on service element support is requested in tabular form where, for each service element,

- the status of the service element is indicated as mandatory (M), optional (O) or not applicable (-) in columns labelled "STD",
- the actual support of the service element by the implementation on origination and reception is indicated by the supplier in columns labelled "IMP".

Information on protocol element support is requested in tabular form where, for each protocol element,

- the status of the protocol element on origination and reception is indicated as mandatory (M) or optional (O) in columns labelled "STD",
- any implementation constraints are indicated in the column labelled "CONST STD" where constraints are interpreted as a minimum for reception and a maximum for origination,
- the actual support of the protocol element by the implementation on origination and on reception is indicated by the supplier in the column labelled "STATUS IMP",
- the actual constraints of the implementation on origination and on reception is indicated by the supplier in the columns labelled "CONST IMP".

Constraints may be expressed as a length or size (octets, bits,...), a value (32k-1) or a number of occurrences (4) depending on the element being constrained.

B.2 IPMS(P2) PICS Service Elements Proforma

The requirements of the X.400 Recommendations are shown in the STD columns of the proforma using the following keys:

- M : Mandatory element (X.401 Basic or Essential Optional)
- O : Optional element (X.401 Additional Optional)
- : Not applicable service element

Suppliers of an implementation should use the IMP columns in the proforma to specify information concerning the support of service elements. For convenience it is suggested that suppliers need only indicate with an "X" those service elements that are not supported.

Table: B-1/X.403 IPMS(P2) Service Elements Proforma (Part 1 of 2)



Table: B-1/X.403 IPMS(P2) Service Elements Proforma (Part 2 of 2)

### B.3 IPMS(P2) PICS Protocol Elements Proformas

The requirements of the X.400 Recommendations are shown in the STATUS STD columns of the proformas in Tables B-2/X.403 to B-4/X.403 using the following keys:

- M : Mandatory element (X.401 Basic or Essential Optional)
- O : Optional element (X.401 Additional Optional)

Protocol elements which correspond directly to service elements are indicated as mandatory if their corresponding service elements are shown in X.401 (1984) as Basic or Essential Optional, and as optional if their corresponding service elements are shown in X.401 (1984) as Additional Optional. Other protocol elements are indicated as mandatory or optional according to their designation in the UAPDU definitions in X.420 (1984).

The pragmatic constraints of the X.400 Implementor's Guide are

shown in the CONST STD columns of the proformas in Tables B-2/X.403 to B-4/X.403.

Suppliers of an implementation should use:

- the STATUS IMP columns in each proforma to specify information concerning the support of protocol elements. For convenience it is suggested that suppliers need only indicate with an "X" those protocol elements that are not supported.
- the CONST IMP columns in each proforma to specify the actual constraints of the implementation.

Table: B-2/X.403 ORDescriptor proforma.

Table: B-3/X.403 IM-UAPDU proforma (part 1 of 3)

Table: B-3/X.403 IM-UAPDU proforma (part 2 of 3)

Table: B-3/X.403 IM-UAPDU proforma (part 3 of 3)

Table: B-4/X.403 SR-UAPDU proforma

