# EUROPEAN TELECOMMUNICATION STANDARD

**ETS 300 658**

**September 1996**

Source: ETSI TC-SPS

Reference: DE/SPS-02030

ICS: 33.080

**Key words:** ISDN, SS7, TC, testing, ASN.1

# Integrated Services Digital Network (ISDN); Signalling System No.7; Transaction Capabilities (TC) version 2; Test responder specification

**ETSI**

European Telecommunications Standards Institute

**ETSI Secretariat**

**Postal address:** F-06921 Sophia Antipolis CEDEX - FRANCE
**Office address:** 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
**X.400:** c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

# Contents

Blank page

## Foreword

This European Telecommunication Standard (ETS) has been produced by the Signalling Protocols and Switching (SPS) Technical Committee of the European Telecommunications Standards Institute (ETSI).

| **Transposition dates** | |
|---|---|
| Date of adoption of this ETS: | 6 September 1996 |
| Date of latest announcement of this ETS (doa): | 31 December 1996 |
| Date of latest publication of new National Standard or endorsement of this ETS (dop/e): | 30 June 1997 |
| Date of withdrawal of any conflicting National Standard (dow): | 30 June 1997 |

Blank page

# 1 Scope

This European Telecommunication Standard (ETS) defines a simple and flexible test responder which enables the use of Abstract Test Suites (ATSs) for Transaction Capabilities (TC) independently from the actual TC-users which reside in a System Under Test (SUT).

No assumption is made about the actual implementation of the interface between this function and TC.

The availability of a standardized TC test responder has the following advantages:

- it makes it possible to write a unique ATS which can be executed against any TC implementation which resides in a system where the test responder is also implemented;

- it allows all the defined TC functionalities to be tested, irrespective of the sub-set of functionalities actually used by the TC-users which are available at the moment when an equipment is under test;

- it helps to isolate faults during testing, since the proper response to a TC message or component will be independent of the proper execution of a real TC-user operation;

- it allows the TC stack to be tested before being delivered to a customer for supporting one ore more particular TC-user applications;

- the test responder can even be used to perform stack-to-stack interoperability testing independently of any particular TC-user application.

# 2 Normative references

This ETS incorporates by dated and undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this ETS only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

[1] ETS 300 134: "Integrated Services Digital Network (ISDN); Signalling System No.7; Transaction Capabilities Application Part (TCAP)".

[2] ETS 300 287: "Integrated Services Digital Network (ISDN); Signalling System No.7; Transaction Capabilities Application Part (TCAP) version 2".

[3] ISO/IEC 8824 (1995): "Information Technology - Abstract Syntax Notation One (ASN.1)" (also published as ITU-T Recommendations X.680 (1994), X.681 (1994), X.682 (1994) and X.683 (1994)).

[4] ISO/IEC 8825-1 (1995): "Information Technology - ASN.1 Encoding Rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)" (also published as ITU-T Recommendation X.690 (1994)).

[5] ISO/IEC 9646-1 (1994): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General concepts".

[6] CCITT Recommendations Q.771-Q.775 (1988): "Specifications of Signalling System No.7, Transaction Capabilities Application Part (TCAP)" (Blue Book).

[7] ITU-T Recommendations Q.771-Q.775 (1993): "Specifications of Signalling System No.7: Transaction Capabilities (TC)" (White Book).

## 3 Abbreviations

For the purposes of this ETS, the following abbreviations apply:

| | |
|---|---|
| AE | Application Entity |
| ASE | Application Service Element |
| ASN.1 | Abstract Syntax Notation One (as specified in ISO/IEC 8824 [3]) |
| ATS | Abstract Test Suite |
| IUT | Implementation Under Test |
| LT | Lower Tester |
| PDU | Protocol Data Unit |
| SCCP | Signalling Connection Control Part (SS7) |
| SUT | System Under Test |
| TC | Transaction Capabilities |
| TMP | Test Management Protocol |
| UT | Upper Tester |

## 4 Architecture

The TC test responder is a particular TC-user which can be implemented together with TC in any system, using several possible configurations.

The communication between the TC test responder which resides in a SUT and a test system relies on the use of a particular Application Service Element (ASE), called "TC Testing User ASE". The TC test responder plays the role of the ASE supplier while the test system plays the role of the ASE consumer.

The TC Testing User ASE can be implemented as the single component of an Application Entity (AE) located at a particular sub-system number, or it can be combined with other application layer elements in which case it is selected by using any application-context-name starting with the following value:

```
{ccitt identified-organization etsi(0) 658 ac(5)}
```

NOTE: ETS 300 009-1, Signalling Connection Control Part (SCCP) will allocate a sub-system number which may be used for addressing an AE which contains the test responder (e.g. when only CCITT Blue Book [6] TC facilities are available).

According to ISO/IEC 9646-1 [5], the set of data units conveyed between two instances of the test responder and a test system can be considered as an "in-band" Test Management Protocol (TMP), which can be used to support co-ordinated test methods between the test responder acting as an Upper Tester (UT) and the Lower Tester (LT) functionality of the test system (see figure 1).

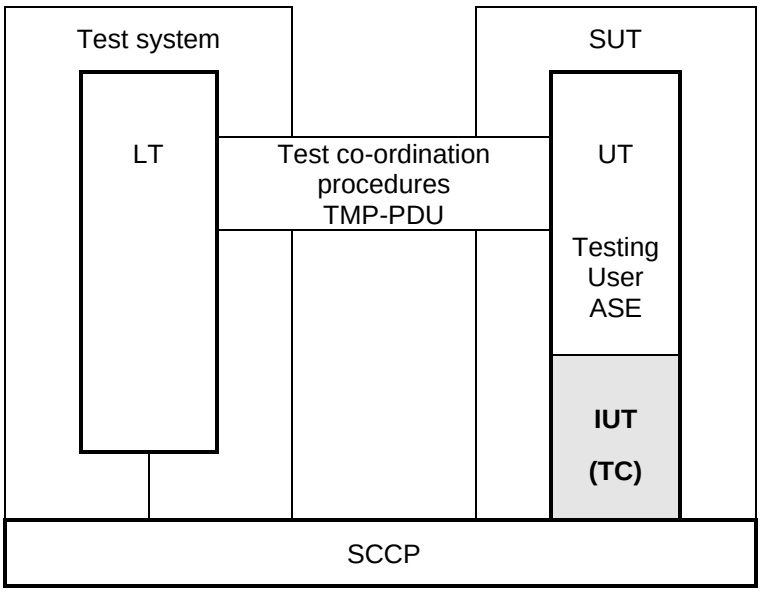


**Figure 1**

The use of this test responder does not require any access to the TC service interface within the SUT, nor does it place any constraints on its implementation. This test responder does not provide means for testing the behaviour of a TC implementation in response to invalid behaviour of the local TC-user.

The "in-band" nature of the TMP implies that, although being under test, the TC service is reliable enough to carry such Protocol Data Units (PDUs) and deliver them to a user. In order to overcome potential difficulties due to a missing or unreliable TC service, the Testing User ASE procedures are defined in such a way that each TMP-PDU can be received in different types of messages or components.

The main purpose of the TMP is to allow series of commands to be sent from the test system to the test responder in order to provoke some behaviour in the TC Implementation Under Test (IUT). Each command is either a TC service primitive to be passed by the test responder to the TC under test or the indication that the test responder should keep waiting for a subsequent event.

# 5 TC testing user Application Service Element (ASE)

## 5.1 General principles

The TC testing user ASE can be built on top of the 1988 version (ETS 300 134 [1], based on the CCITT Blue Book [6]) or the 1993 version (ETS 300 287 [2], based on the ITU-T White Book [7]) of TC[1)]. This ASE can handle simultaneously several dialogues. It embodies the knowledge of seven operations; two of them can be invoked by the ASE consumer (i.e. the test system), five of them can be invoked by the ASE supplier (i.e. SUT). The two operations invoked by the test responder are defined as class 1 operations. However, this has no impact on the behaviour of the test responder[2)].

The TMP-PDUs can be conveyed in the operations arguments, operations result parameters, errors parameters and when available, in the user information parameter of the dialogue portion.

There is no specific relation between a TMP-PDU and the type of message or component which is used to carry it.

NOTE: The upper service interface of the TC Testing User ASE is not standardized. However, such an interface could be made accessible to locally trigger a particular test scenario (e.g., for interoperability testing) or enable the test responder to report to some management function expiry of the timer T-Test.

## 5.2 Operations and Errors

The Abstract Syntax Notation One (ASN.1) module TC-Testing-User defined in annex A contains the specification of the operations which can be invoked by the test system or the test responder during communication.

The local values assigned to the following operations and errors shall be considered as default values. The actual local values used for these operations and errors shall be treated as configuration parameters.

```
class1SupplierOperation
class2SupplierOperation
class3SupplierOperation
class4SupplierOperation
localConsumerError
localSupplierError
```

---

1) It is up to the test suite designer to write the test cases in such a way that the Test Responder does not request 1993 functionalities from a 1988 TC implementation.

2) Whether the test responder reports an outcome for these operations depends on the commands received in the TMP PDUs. This does not violate any rule as far as TC is concerned since the class of an operation is irrelevant to the TC residing at the side where the operation is executed.

# 6 TC test management protocol

## 6.1 Test management protocol PDUs

There are three types of TMP-PDU[3)]:

**testInit**

The testInit PDU requests the test responder to initiate a test session and conveys a set of commands to be executed sequentially by the test responder. This PDU can only be sent by the test system.

**testContinue**

The testContinue PDU conveys a set of additional commands to be executed sequentially by the test responder. This PDU can only be sent by the test system.

**testDataEcho**

The testDataEcho PDU conveys user data echoing the user data received with a particular command. This PDU can only be sent by the test responder.

The testInit and testContinue PDUs can be conveyed in the argument of the operations invoked by the test system, in the result or error parameter returned by the test system in response to an operation invoked by the test responder or in the user information parameter of the dialogue portion of the messages sent by the test system.

The testDataEcho PDU can be conveyed in the argument of the operations invoked by the test responder, in the result or error parameter returned by the System Under Test in response to an operation invoked by the test system, or in the user information parameter of the dialogue portion of the messages sent by the test responder.

## 6.2 Command structure

Each command sent from the test system in a testInit or testContinue PDU indicates to the test responder that it shall keep waiting for an external event from the local TC or that it shall issue a particular request primitive to the local TC.

Commands indicating to the test responder that is has to wait for an external event also indicate whether this event shall occur on a particular dialogue or that this does not matter.

Commands requesting the test responder to issue a primitive comprise up to three items:

1) an indication of the type of request primitive to be passed to the local TC;

2) the reference to the dialogue over which the action should be taken. If this information is not explicitly provided, the test responder assumes that the command refers to the dialogue over which the TMP-PDU has been received. The dialogue reference is always chosen by the test system. Unlike transaction Ids and dialogue Ids, the dialogue reference is a common reference shared by both sides of a dialogue.

   The test responder has to keep track of the one-to-one relation which exists between a dialogue-reference and the local TC dialogue-Id which has been agreed between TC and the Testing User ASE for this dialogue;

3) optionally, a user data parameter to be echoed as user data associated with the service primitive to be issued.

---

**3)** The need for a "TestEnd" PDU is for further study.

### 6.3 Abstract syntax

The ASN.1 module TC-TMP defined in annex B contains the specification of the TMP-PDUs.

When the TMP-PDUs are conveyed in the user information parameter of the dialogue portion, the following abstract-syntax-name is used as a direct-reference to identify the set of data values each of which is a value of type TMP-Protocol.TMP-PDU:

```
{ccitt identified-organization etsi(0) 658 as(4) tmp-pdus(1) version1(1)}
```

The applicable encoding rules are the basic encoding rules defined in ISO/IEC 8825-1 [4].

### 6.4 Procedures

#### 6.4.1 Procedure at the test system side

The Testing User ASE has limited error detection and error recovery functionalities. It is up to the designer of the ATS to ensure that the sequence of commands which are sent to the test responder corresponds to a valid TC-user behaviour and does not conflict with the automatic procedures of this ASE.

It is up to the Test Suite designer to choose how a TMP-PDU is conveyed to the test responder (i.e. which message and which component, if any, etc.). The test system can send a TMP-PDU in any type of message and component with the sole restriction that a TestInit PDU can only be sent in a Begin message or in a Unidirectional message.

It is also the responsibility of the designer of the ATS to ensure that the TMP PDU will be delivered to the Test User ASE (i.e. it should be conveyed in a valid TC message or component).

**Starting a Test Case:** at the beginning of each Test Case, the test system shall send a testInit PDU in order to ensure that no resources associated with a previous test case are still active. This PDU shall include a sequence of commands to be executed by the test responder and optionally the value of a watch-dog timer (T-Test). If no timer value is provided, an implementation value is selected[4)].

**Continuing a Test Case:** if all the commands are not sent in the testInit PDU, the test system can send further commands to the test responder using the testContinue PDU which also contains a sequence of commands to be executed by the test responder. A TC message can convey more than one testContinue PDU.

**Ending a Test Case:** there is no specific command nor TMP-PDU for ending a test case. It is up to the test designer to write the test case postamble in such a way that all active dialogues are closed.

#### 6.4.2 Procedure at the test responder side

##### 6.4.2.1 Generic rules

The test responder refuses any application-context-name whose object identifier value does not start with the following root:

```
{ccitt identified-organization etsi(0) 658 ac(5)}
```

If the test responder accepts a 1993 dialogue (i.e. a dialogue according to ETS 300 287 [2] and the ITU-T White Book [7]), it shall use the received application-context-name in the next dialogue handling primitive it issues.

If the test responder refuses a 1993 dialogue, it shall propose the following application-context-name:

```
{ccitt identified-organization etsi(0) 658 ac(5) testing-ac(1) version1 (1)}
```

---

4) Implementors may choose a very large value if the test responder is intended to be used for running background traffic or load testing.

When requesting TC to send a Begin message or a Unidirectional message, the test responder uses as destination address, the originating address of the message in which the TestInit PDU was received. If an application-context-name has to be provided by the test responder, it shall use the following value:

```
{ccitt identified-organization etsi(0) 658 ac(5) testing-ac(1) version1 (1)}
```

If the test responder detects that it has been reached with a destination address which is not the one it knows from the configuration data, it forces TC to insert the address it knows in the first backward Continue message sent to the test system.

During one dialogue, the following rules apply:

- the first invoke Id used by the test responder has the value 0, then this value is incremented each time the test responder invokes an operation;

- if the invocation received by the test responder gives reason for a user reject to be generated, the test responder automatically requests TC to return a reject component (e.g. it rejects any operation which is not defined in the TC-Testing-User module) with the appropriate reject problem;

- the test responder always requests the return option when issuing a TC-BEGIN request primitive. In all other cases the setting of this parameter is an implementation option;

- when requesting the sending of partial results, the test responder always requests the sequencing option. In all other cases the setting of this parameter is an implementation option;

- the timer value provided by the test responder when invoking an operation is a configuration parameter.

#### 6.4.2.2 Handling of TMP-PDUs

If an event is received without any TMP-PDU, the test responder does nothing[5)].

If an event is received with several TMP-PDUs, they are processed sequentially, starting with the ones included in the user information field of the dialogue portion (if any).

On receipt of a testInit PDU, the test responder releases all the active resources (e.g. it closes any active dialogue, releases the dialogue references, etc.)[6)] and starts the T-Test timer. Then the commands are executed sequentially and the test responder keeps waiting for the next event or expiry of timer T-Test.

On Receipt of a testContinue PDU, the test responder executes sequentially the specified commands and keeps waiting for the next event or expiry of timer T-Test.

On receipt of a testDataEcho PDU, the test responder does nothing.

According to the generic rules, receipt of any other data value will lead to:

- a TC-user reject procedure if the invalid PDU value is received in an operation argument, a result parameter or an error parameter;

- a TC-user abort procedure if it is received in the user information field of the dialogue portion.

When the T-Test timer, expires all the resources associated with the test are automatically released.

The test responder interprets and executes sequentially each command received from the test system, as described in subclauses 6.4.2.3 and 6.4.2.4.

---

5) This includes receipt of a TC-NOTICE indication primitive which may can contain a returned TMP-PDU.

6) How the resources are released is an implementation matter. However, this should not lead to the sending of any external message (e.g., END or ABORT message).

#### 6.4.2.3 Execution of the wait command

The test responder keeps waiting for an incoming event from the IUT. When the incoming event has been received it executes the next command (if any) or keeps waiting for the next TMP-PDU[7)].

Conceptually, an incoming event corresponds to one or more related TC indication primitives. The latter case corresponds to a situation where the test responder receives a dialogue handling primitive indicating that component are present. In such a case, the test responder consumes all the component handling primitives until the "last component" parameter takes the value "TRUE". However this does not place any constraint on the actual implementation of the interface between the Test responder and TC.

If the command is accompanied by an explicit dialogue reference, primitives whose dialogue identifier does not correspond to this reference, are ignored (the test responder keeps waiting for a next event). Otherwise the dialogue identifier value is not checked.

#### 6.4.2.4 Execution of other commands

The test responder interprets the service type as follows:

- v1988uniReq: the test responder requests the IUT to send a 1988 Unidirectional message;
- v1993uniReq: the test responder requests the IUT to send a 1993 Unidirectional message;
- v1988beginReq: the test responder requests the IUT to send a 1988 Begin message;
- v1993beginReq: the test responder requests the IUT to send a 1993 Begin message including a dialogue portion;
- continueReq: the test responder requests the IUT to send a Continue message for the dialogue which corresponds to the dialogue-reference value included in the command;
- basicEndReq: the test responder requests the IUT to send an End message for the dialogue which corresponds to the dialogue-reference value included in the command;
- localEndReq: the test responder requests the IUT to locally terminate the dialogue which corresponds to the dialogue-reference value included in the command;
- uAbortReq: the test responder requests the IUT to abort the dialogue which corresponds to the dialogue-reference value included in the command. If the dialogue has been established using the 1993 procedure, the abort-reason is set to "user-specific";
- class1invokeReq: the test responder requests the IUT to invoke the class 1 operation defined in the module TC-Testing-User (see annex A);
- class2invokeReq: the test responder requests the IUT to invoke the class 2 operation defined in the module TC-Testing-User (see annex A);
- class3invokeReq: the test responder requests the IUT to invoke the class 3 operation defined in the module TC-Testing-User (see annex A);
- class4invokeReq: the test responder requests the IUT to invoke the class 4 operation defined in the module TC-Testing-User (see annex A);
- linkedInvokedReq: the test responder request the IUT to invoke the class 1 operation defined in the module TC-Testing-User (see annex A) as a linked operation to the oldest pending operation;
- resultNlReq: the test responder requests the IUT to send a result not last component in response to the oldest pending operation;

---

7) The Test Responder is always waiting for events. The wait command is only required if there is a need to explicitly wait for an event before executing a command which has been sent in advance.

- resultLReq: the test responder requests the IUT to send a result last component in response to the oldest pending operation;

- uErrorReq: the test responder requests the IUT to send an error component in response to the oldest pending operation. The error code is the single error code allowed by the operation definition;

- uCancelReq: the test responder requests the IUT to cancel the oldest pending operation;

- uRejectReq: the test responder requests the IUT to send a reject component indicating "invoke-problem: resource limitation" for the oldest pending operation.

NOTE: In this context, a "1988 message" is a message defined according to ETS 300 134 [1] and the CCITT Blue Book [6], while a "1993 message" is a message defined according to ETS 300 287 [2] and the ITU-T White Book [7].

#### 6.4.2.5 Data echo

When requesting a dialogue handling service from the 1993 TC (i.e. as defined in ETS 300 287 [2] and the ITU-T White Book [7]) and if a "to-be-echoed" element was present in the associated command received from the test system, the test responder shall include one or more testDataEcho TMP-PDUs as value of the user information parameter. Each "testDataEcho" value is identical to the received "to-be-echoed" value. During the dialogue establishment phase or when the unstructured dialogue mode is used, the number of "testDataEcho" values provided by the test responder is a configuration parameter. Once the dialogue has been established, only one value can be sent.

When requesting a component handling service from TC, the test responder includes a testDataEcho TMP-PDU as user parameter (i.e. operation argument, result parameter, error parameter, etc.) if a "to-be-echoed" element was present in the associated command received from the test system. If included, the "testDataEcho" value is identical to the received "to-be-echoed" value.

If the test responder receives some unknown information in the user information parameter of a dialogue handling primitives, it provides it unchanged in the user information parameter of the next dialogue handling primitive it issues.

## Annex A (normative): TC test responder Operations and Errors

The following module defines the TC Testing User ASE in terms of which operations can be invoked by the consumer (the test system) and the supplier (the SUT).

```
TC-Testing-User {ccitt identified-organization etsi(0) 658 modules(0) testing-user(1)
version1(1)}

DEFINITIONS ::=

BEGIN

IMPORTS

OPERATION, ERROR
    FROM TCAPMessages {ccitt recommendation q 773 modules(2) messages(1) version2(2)}

APPLICATION-SERVICE-ELEMENT
    FROM Notation-Extensions {joint-iso-ccitt remote-operations (4) notation-extension(2)}

TMP-PDU
    FROM TC-TMP {ccitt identified-organization etsi(0) 658 modules(0) tmp(2) version1(1)}
;

-- application-context-names

ac-id           OBJECT IDENTIFIER   ::= {ccitt identified-organization etsi(0) 658 ac(5)}
testing-ac-id   OBJECT IDENTIFIER   ::= {ac-id testing-ac(1) version1(1)}


-- ase

Testing-User-ASE APPLICATION-SERVICE-ELEMENT

CONSUMER INVOKES    {
                    localConsumerOperation,
                    globalConsumerOperation
                    }
-- consumer is the test system

SUPPLIER INVOKES    {
                    class1SupplierOperation,
                    class2SupplierOperation,
                    class3SupplierOperation,
                    class4SupplierOperation,
                    globalSupplierOperation
                    }
-- supplier is the test responder

::= {ccitt identified-organization etsi(0) 658 ase(3) testing-user (1) version1(1)}


LocalConsumerOperation  ::= OPERATION
ARGUMENT    TMP-PDU
RESULT      TMP-PDU
ERRORS      {localSupplierError}

GlobalConsumerOperation ::= OPERATION
ARGUMENT    TMP-PDU
RESULT      TMP-PDU
ERRORS      {globalSupplierError}

Class1SupplierOperation ::= OPERATION
ARGUMENT    TMP-PDU
RESULT      TMP-PDU
ERRORS      {localConsumerError}
LINKED      {localConsumerOperation}

Class2SupplierOperation ::= OPERATION
ARGUMENT    TMP-PDU
ERRORS      {localConsumerError}
LINKED      {localConsumerOperation}

Class3SupplierOperation ::= OPERATION
ARGUMENT    TMP-PDU
RESULT      TMP-PDU
```

```
Class4SupplierOperation ::= OPERATION
ARGUMENT    TMP-PDU

GlobalSupplierOperation ::= OPERATION
ARGUMENT    TMP-PDU
RESULT      TMP-PDU
ERRORS      {globalConsumerError}
LINKED      {globalConsumerOperation}


ConsumerError   ::= ERROR
PARAMETER   TMP-PDU

SupplierError   ::= ERROR
PARAMETER   TMP-PDU


localConsumerOperation  LocalConsumerOperation  ::= localValue : 0
globalConsumerOperation GlobalConsumerOperation ::=
            globalValue : {ccitt identified-organization etsi(0) 658 operations(1) consumer(1)}

class1SupplierOperation Class1SupplierOperation ::= localValue : 1
class2SupplierOperation Class2SupplierOperation ::= localValue : 2
class3SupplierOperation Class3SupplierOperation ::= localValue : 3
class4SupplierOperation Class4SupplierOperation ::= localValue : 4
globalSupplierOperation Class1SupplierOperation ::=
            globalValue : {ccitt identified-organization etsi(0) 658 operations(1) supplier(2)}

localConsumerError      ConsumerError           ::= localValue : 1
globalConsumerError     ConsumerError           ::=
            globalValue : {ccitt identified-organization etsi(0) 658 errors(2) consumer(1)}

localSupplierError      SupplierError           ::= localValue : 2
globalSupplierError     SupplierError           ::=
            globalValue : {ccitt identified-organization etsi(0) 658 errors(2) supplier(2)}


END
```

## Annex B (normative): Test management protocol PDUs

The following module defines the TMP-PDUs.

```
TC-TMP {ccitt identified-organization etsi(0) 658 modules(0) tmp(2) version1(1)}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

TMP-PDU ::= CHOICE
    {
    testInit            [0] TestInit,
    testContinue        [1] CommandSequence,
    testDataEcho        [2] UserData
    }

CommandSequence ::= SEQUENCE SIZE(0..maxNbOfCommands) OF TestCommand

maxNbOfCommands INTEGER ::= 30

TestInit ::= SEQUENCE
    {
    timeout             INTEGER (1..127)    OPTIONAL,           -- T-Test (unit is 30 sec)
    commands            CommandSequence,
    ...
    }

UserData ::= CHOICE
    {
    simple              OCTET STRING (SIZE(0..maxUserDataLength)),
    complex             [0] ABSTRACT-SYNTAX.&Type
    }

maxUserDataLength INTEGER ::= 2048

TestCommand ::= CHOICE
    {
    wait                [0] DialogueReference,
    action              [1] ActionInfo
    }

DialogueReference ::= CHOICE
    {
    unspecified         NULL,
    dialogue            INTEGER (0..255)
    }

ActionInfo ::= SEQUENCE
    {
    service             ServiceType,
    dialogueReference   DialogueReference   DEFAULT unspecified : NULL,
    to-be-echoed        UserData            OPTIONAL,
    ...
    }

ServiceType ::= ENUMERATED
    {
    v1988uniReq (10),
    v1993uniReq (11),
    v1988beginReq (12),
    v1993beginReq (13),
    continueReq (14),
    basicEndReq (15),
    localEndReq (16),
    uAbortReq (17),
    class1invokeReq (21),
    class2invokeReq (22),
    class3invokeReq (23),
    class4invokeReq (24),
    linkedInvokeReq (25),
    resultNlReq (26),
    resultLReq (27),
    uErrorReq (28),
    uCancelReq (29),
    uRejectReq (30),
    ...
    }
```

```
-- abstract syntax name for TMP-PDUs

tmp-pdus-as OBJECT IDENTIFIER ::= {ccitt identified-organization etsi(0) 658 as(4) tmp-pdus(1)
version1(1)}


END
```

## Annex C (informative): Example of use of the TC test responder

This annex illustrates how some of the test purposes defined in ITU-T Recommendation Q.787 can be implemented using the TC test responder.

### a) Test 2.1.6 Valid functions, User cancel

| Test responder/TC interface | Message flow | Test system |
|---|---|---|
| | BEGIN [Invoke (localConsumerOperation, 1)] <---------- | |
| TC-Begin-Ind<br>TC-Invoke-Ind (1) | | |
| TC-Invoke-Req (0)<br>TC-ContinueReq | | |
| | CONTINUE[Invoke (class1SupplierOperation, 0)] ----------> | |
| TC-U-Cancel-Req (0) | | |
| | CONTINUE[Return-Result-L (0)] <---------- | |
| TC-L-Reject-Ind (0)<br>TC-End-Req | | |
| | END[Reject (0)] | |

**Figure C.1**

The localConsumerOperation argument is defined as follows:

```
testInit : {
    timeOut 30,
    commands
        {
        action : {service class1InvokeReq},
        action : {service continueReq},
        action : {service uCancelReq},
        wait   : {unspecified : NULL},
        action : {service basicEndReq}
        }
    }
```

**b) Test 2.1.2.1.1 Valid functions, Linked operations, Class 1 original operation, IUT as sender**

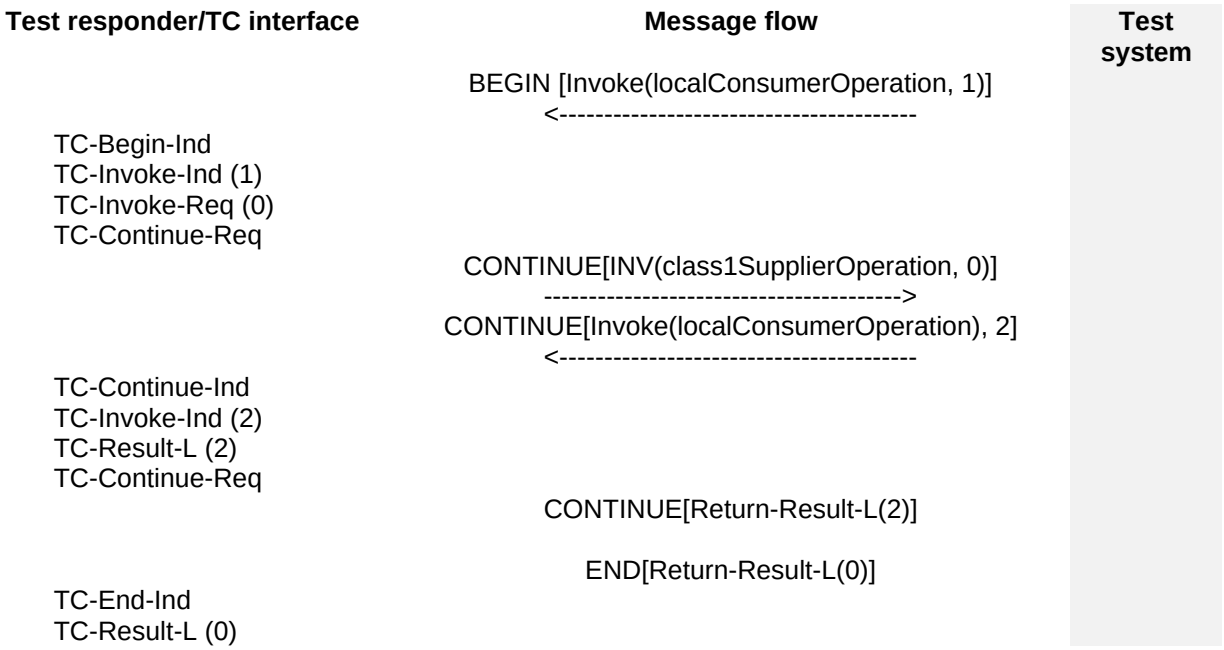


**Figure C.2**

The first localConsumerOperation invocation argument is defined as follows:

```
testInit : {
    timeOut 30,
    commands {
        action : {service class1InvokeReq},
        action : {service continueReq},
        wait   : {unspecified : NULL}
        }
    }
```

The second localConsumerOperation invocation argument is defined as follows:

```
testContinue : {
        action : {service resultLReq},
        action : {service continueReq},
        wait   : {unspecified : NULL}
        }
    }
```

**c) Test 1.1.2.2.1.1-3: Valid functions, Clearing after Continue message, IUT Abort by TC-User**

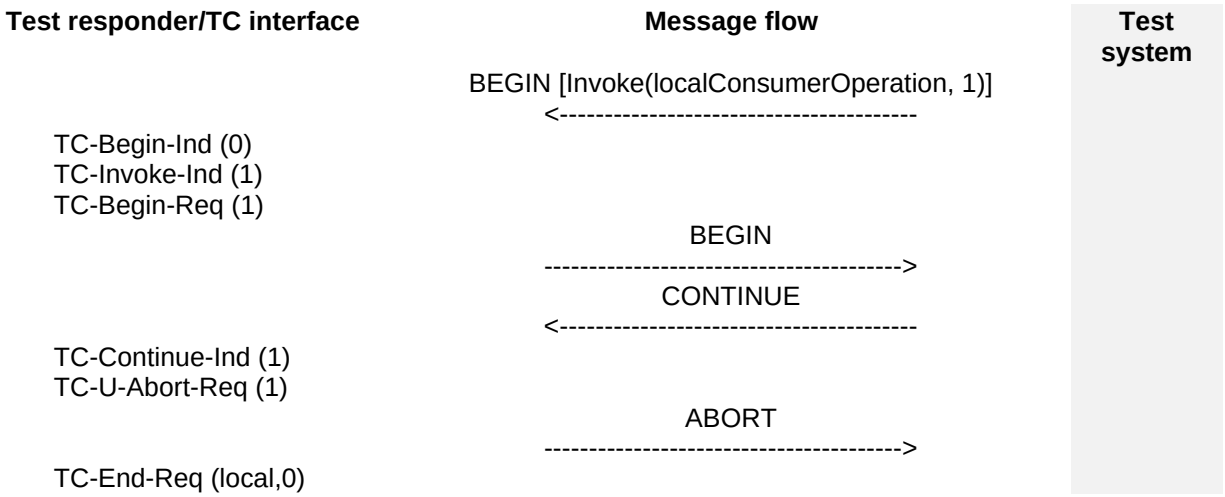


**Figure C.3**

The localConsumerOperation invocation argument is defined as follows:

```
testInit : {
    timeOut 30,
    commands {
        action : {service v1988beginReq, dialogueReference value : 1},
        wait   : {dialogueReference value : 1},
        action : {service uAbortReq, dialogueReference value : 1},
        action : {service localEndReq, dialogueReference value : 0}
        }
    }
```

## Annex D (informative): Implementing loops using the TC test responder

This annex illustrates the use of the TC test responder for implementing loops. The test case is such that both the test system and the test responder keep continuously opening dialogues which are immediately ended by the other entity.

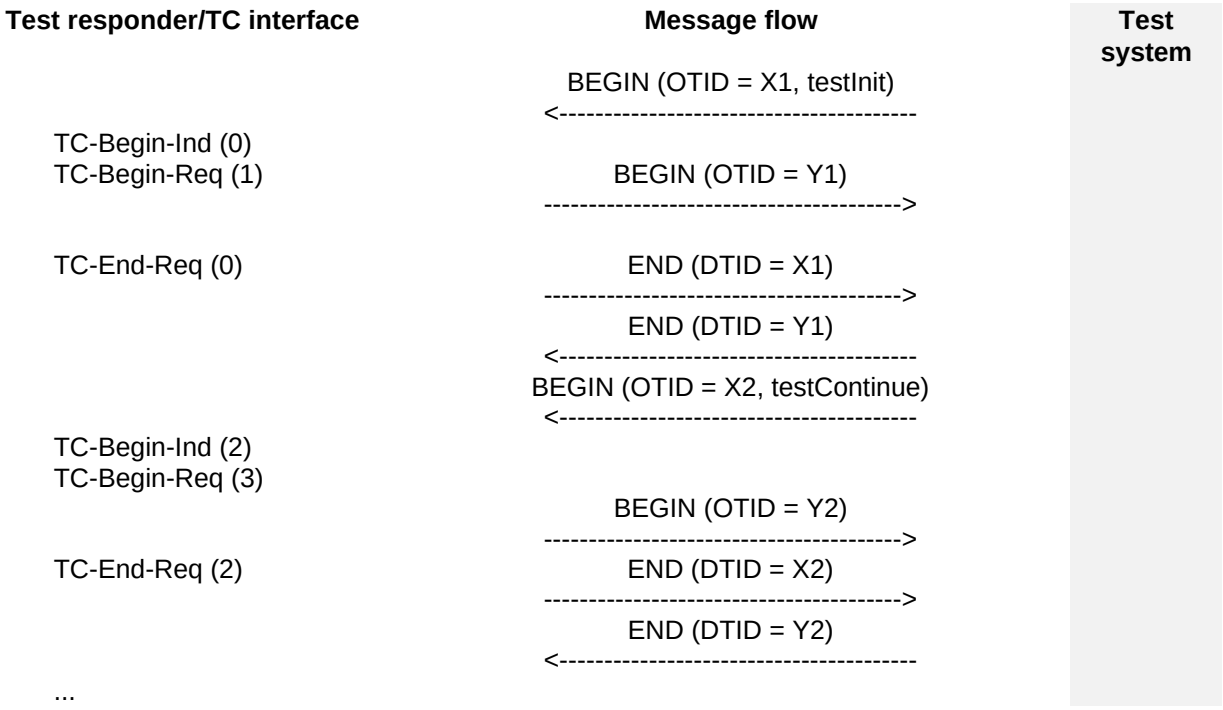


**Figure D.1**

The first BEGIN message sent from the Tester to the test responder includes a testInit PDU defined as follows:

```
testInit : {
    timeOut 30,
    commands {
        action : {service v1988beginReq, dialogueReference value : 1},
        action : {service basicEndReq, dialogueReference value : 0},
        wait   : {dialogueReference value : 1}
        }
    }
```

The subsequent BEGIN messages include a testContinue PDU defined as follows:

```
testContinue : {
        action : {service v1988beginReq, dialogueReference value : i+1},
        action : {service basicEndReq, dialogueReference value : i},
        wait   : {dialogueReference value : i+1}
        }
    }
```

where i is incremented from 1 to the number of desired loops (N).

When i = N, the test case can be terminated by sending a testContinue PDU without the v1988beginReq command. In this example, there exist at most two dialogues at a time. However, for load testing purposes, the test case could be rewritten in such a way that the Tester does not send any END message and the TMP-PDUs (except the one which terminates the test case) do not contain any "basicEndRequest" in the list of commands.

## Annex E (informative): Bibliography

- ETS 300 009-1: "Integrated Services Digital Network (ISDN); Signalling System No.7; Signalling Connection Control Part (SCCP) (connectionless and connection-oriented class 2) to support international interconnection; Part 1: Protocol specification [ITU-T Recommendations Q.711 to Q.714 and Q.716 (1993), modified]".

- ITU-T Recommendation Q.787 (1993): "Transaction Capabilities (TC) test specification".

## History

| Document history | | | |
|---|---|---|---|
| December 1995 | Public Enquiry | PE 97: | 1995-12-04 to 1996-04-12 |
| June 1996 | Vote | V 106: | 1996-06-24 to 1996-08-30 |
| September 1996 | First Edition | | |
| | | | |
| | | | |

ISBN 2-7437-0969-3
Dépôt légal : Septembre 1996