5i’

Recommendation Q.775

GUIDELINES FOR USING TRANSACTION CAPABILITIES

(Melbourne, 1988)

1 Introduction

1.1 *General*

The purpose of this Recommendation is to provide guidelines to potential users of Transaction Capabilities (TC-users). The examples given are illustrations only; they indicate how an application may use TCAP, not how TCAP must be used in all cases. The technical basis of this document are Recommendations Q.771 to Q.774; in case of misalignment, these should be considered as the primary reference.

The main purpose of TCAP is to provide support for interactive applications in a distributed environment. TCAP is based on Recommendations X.219 and X.229 (ROSE) enhanced as necessary to provide the services needed by TC-users. Interactions between distributed application entities are modelled by Operations. An operation is invoked by an (originating) entity: the other (destination) entity attempts to execute the operation and possibly returns the outcome of this attempt.

The semantics of an operation (represented by its name and parameters) is not relevant to TCAP; TCAP provides facilities which are independent of any particular operation. The TC-user, when defining an application, must:

1) select operations;

2) select TCAP facilities to support these operations. Such facilities include the handling of individual operations, and the ability to have a number of related operations attached to an association between TC-users, called a dialogue;

3) define the application script.

This Recommendation describes the selection process of defining and using operations. The operations appearing hereafter are fictitious, and are taken for illustration purposes only. Also described are the facilities offered by TCAP for handling one or a sequence of operations in a dialogue. The definition of specific sequences of operations belongs to the application protocol definition and is beyond the scope of this Recommendation; however, Chapter 4 gives a brief indication of what information an application specification should contain.

TCAP services are made accessible to TC-users via primitives; these primitives model the interface between TCAP and its users, but do not constrain any implementation of this interface.

1.2 *Environment*

TCAP defines the end-to-end protocol between TC-users which may be located in a Signalling System No. 7 network, and/or another network supporting TCAP protocols.

Two broad categories of users have been considered (see Recommendation Q.771, § 1.3.2). Only the first category is considered here, i.e. those which are real-time sensitive users, and do not need to exchange large amounts of data. It is considered that for these users, protocols defined for OSI layers 4 to 6 in the X series of Recommendations would result

in excessive overheads and hence are not used. A basic service has been specified, using a connectionless network service approach. Other categories of users might require connection-oriented network and higher layer services.

As a result, TCAP cannot support all kinds of applications, and a number of applications will still require more elaborate services such as specified in the X series of Recommendations. Besides indicating what TCAP can do, this Recommendation indicates what the connectionless approach cannot do, in order to help the application designer choose how to support an application.

2 Operations

2.1 *Definition*

An operation is invoked by an originating TC-user to request a destination TC-user to perform a given action.

A class is attached to an operation. This indicates whether either a successful outcome (result), or an unsuccessful outcome (error), or both, or none have to be reported by the destination. The outcome is reported in a result.

As well as the class, the definition of the operation includes a timer value indicating when the operation should be completed. This value is not indicated to the remote TC-user; it is assumed that the application at both ends has a common understanding of the operations in use.

An **operation** is defined by:

— its operation code and the type of any parameters associated with the operation request;

— its class;

— if the class requires report of success, the possible results corresponding to successful executions are defined by a list of parameters;

— if the class requires report of failure, the possible results corresponding to situations where the operation could not be executed completely by the remote TC-user. Each such situation is identified by a specific error cause; the list of these error causes is part of the operation definition. Diagnostic information can be added to the error cause: if present, it is part of the definition;

— the list of possible linked operations, if replies consisting of linked operations are allowed for this operation. Linked operations have to be described separately;

— a timer value indicating the interval by which the operation has to be completed. This timer value is used to manage the component ID associated with the operation invocation.

2.2 *Examples*

2.2.1 *Simple operations*

Note — The operation invocation should fit into one message, and so should a report of unsuccessful outcome. Reports of success may be segmented using Return Result-Not last and Return Result-Last.

Class 1 (both success and failure reported):

Translate a freephone number into a called subscriber number; return the called number if the translation can be performed, otherwise indicate why it cannot; time allocated: 2 seconds.

No reply being received when the timer expires indicates an abnormal situation (e.g. the operation invocation may have been lost): the local TC-user is informed (operation cancel by TCAP).

Class 2 (only failure reported):

Perform a routine test and send a reply only in case something went wrong; time allocated: 1 minute.

In the case of a class 2 operation, the TC-user is informed if no result has been received when the timer expires. This is interpreted as a successful outcome, even if the invocation was lost. This aspect should be considered when selecting class 2.

Class 3 (only success reported):

Perform a test: this corresponds to a pessimistic view, where failure is considered as the default option, not requiring any reply.

Timer expiry is indicated to the TC-user: this should be interpreted by the TC-user as a failure of the operation (but is considered normal by TC, which considers that the operation has terminated). This aspect should be considered when selecting class 3.

Class 4 (neither success, nor failure reported):

Send a warning, without expecting a reply or acknowledgement of any kind.

In this case, a result never arises from the invocation of the operation. The TC-user relies upon TCAP and the network to deliver the invocation. Notification of the timer expiry is a local matter.

The diagrams in Figure 1/Q.775 illustrate possible sequences of primitives as seen by the TC-user originating an operation.

Figure 1/Q.775, (N), p.

Comparison with ROSE (Recommendation X.219) operation classes :

ROSE provides for five classes of operations: classes 2 to 5, called asynchronous classes, are identical to classes 1 to 4 of TCAP. ROSE's class 1 is a synchronous class; it has no counterpart in TCAP, where full-duplex exchanges of components are considered. However, a TC-user can decide to operate in a synchronous manner (see § 3.2.1).

2.2.2 *More sophisticated operations*

Operations with segmented results

A successfull result may be divided into several segments, each of which is indicated to the originator of the operation by one primitive. This facility, using the TC-RESULT-NL primitive, can be used by TC-users to overcome the absence of segmentation in the underlying layers. The last segment is indicated by the TC-RESULT-L primitive.

The report of an error cannot be segmented.

Apart from abnormal situations, responses are delivered to the remote TC-user in the order in which they have been passed to TCAP by the sending TC-user.

TC cannot identify a specific segment in the case of a segmented result.

Example E1: An operation requests the execution of a test. The result of a correct execution is segmented in three parts P1, P2 and P3 to be returned to the originator.

A possible primitive sequence for example E1 is given in Table 1/Q.775

H.T. [T1.775]

TABLE 1/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(Test, Class = 1)
}
TC-INVOKE ind
(Test)
TC-RESULT-NL req
(P1)
}
	{

TC-RESULT-NL ind (P1)	. TC-RESULT-NL req (P2)
TC-RESULT-NL ind (P2)	. TC-RESULT-L req (P3)
TC-RESULT-L ind (P3)	

Tableau 1/Q.775 [T1.775], p.

The diagram in Figure 2/Q.775 illustrates possible sequences of primitives seen by the originator of an operation (class 1) with segmented results.

Figure 2/Q.775, (N), p.

Linked operations

Another extension to the basic operation scheme is the ability to link an operation invocation to another operation invocation.

Typically, this facility covers situations where the destination of the original (linked-to) operation requires additional information in order to process this operation: this is the case where menu facilities are used (menu facilities allow a user to make a sequence of choices, each being dependent on the previous ones).

Example E2: The operation is the execution of a test with several options; before the test is executed, these options are offered for selection to the test originator (TC-user A). Two operations are nested: operation 1 is the test; operation 2 is the option selection. TC-user A first responds to operation 2 before TC-user B can perform the test with the indicated option(s).

A possible primitive sequence for example E2 is given in Table 2/Q.775.

H.T. [T2.775]

TABLE 2/Q.775

TC USER A	TC USER B	

{
TC-INVOKE req
(Test, Class = 1)
}
TC-INVOKE ind
(Test)
TC-INVOKE req
(Option-selection, Class = 1)
}
Operation 1 begin
Operation 2 begin
}
	{
	
	{	

{
TC-INVOKE ind
(Option-selection)
TC-RESULT-L req
(Options)
}
TC-RESULT-L ind
(Options)
TC-RESULT-L req
(Test-result)
}
	{
	
	. Operation 2 end	

{
TC-RESULT-L ind
(Test-result)
}
		Operation 1 end

Tableau 2/Q.775 [T2.775], p.

There is no limit to the number of operation invocations which may be linked to a given operation invocation.

Note that when an operation B is linked to another operation A, they do not have to be nested. The only condition is that the invocation of B should take place before the outcome of A is reported; however, operation B does not have to terminate before operation A.

2.3 *Component-related facilities offered to TC-users*

2.3.1 *Invocation*

So far, operations have been considered from the static point of view. Invocation introduces a dynamic aspect: a specific invocation of an operation has to be differentiated from other possible concurrent invocations of the same or of another operation.

Each particular activation of an operation is identified by a component ID. This component ID must be non ambiguous. It is selected by the TC-user which originates the operation invocation, and passed to the destination TC-user, which will reflect it in its reply(ies): therefore it correlates the replies to an invocation, and the invocation itself.

The TC-user is free to assign any value to the component ID (index, address, . | |).

The component ID associated with an invocation becomes reusable when the last or only segment of a result is received, or when certain abnormal situations are indicated by TCAP; however, the value should not be reallocated immediately for another operation activation, as immediate reallocation would prevent the correct handling of some situations (see below).

The period during which a component ID is released, but cannot be reallocated, is called the freezing period.

As component IDs receive their value dynamically at the time the operation is invoked, their value cannot appear in the specification of the application protocols; rather, a ‘‘logical’’ value, to which a real value is substituted at execution time, should be indicated in order to identify an operation in a single flow.

Taking component IDs into consideration, the sequence of primitives for example E2 above becomes as shown in Table 3/Q.775.

H.T. [T3.775]

TABLE 3/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(1, Test, Class = 1)
}
TC-INVOKE ind
(1, Test)
TC-INVOKE req
(2, 1, Option-selection, Class = 1)
}
	{

{
TC-INVOKE ind
(2, 1, Option-selection)
TC-RESULT-L req
(2, Options)
}
TC-RESULT-L ind
(2, Options)
TC-RESULT-L req
(1, Test-result)
}
	
	{

{
TC-RESULT-L ind
(1, Test-result)
}

Tableau 3/Q.775 [T3.775], p.

where the first parameter of a primitive indicates an invoke ID. When both parameters have to be present, the second one is the linked ID. This is a pure notational convention.

2.3.2 *Cancel (by the TC-user)*

The TC-user requesting invocation of an operation may stop the activity associated with the corresponding component ID, for any reason it finds appropriate. However, cancel should in principle be reserved for abnormal situations: the normal method for terminating an operation is to receive a result or to terminate on timer expiry.

Cancelling has local effect only: it does not prevent the remote TC-user from sending replies to a cancelled operation. When received, these replies will be rejected by TCAP, as illustrated in the following, which represents a sequence of primitives for the example E1 defined above, where TC-user A cancels the test after receiving the first segment of the result.

In Table 4/Q.775, part P2 is not received by TC-user A: TCAP detects a reject situation before delivering it, and any attempt by TC-user B to send more replies is rejected at A’s side.

H.T. [T4.775]
TABLE 4/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(1, Test, Class = 1)
}
TC-INVOKE ind
(1, Test)
TC-RESULT-NL req
(1, P1)
}
	{

{
TC-RESULT-NL ind
(1, P1)
Cancel decision:
TC-CANCEL req
(1)
TC-L-REJECT ind
(1, Problem Code)
}
TC-RESULT-NL req
(1, P2)
}
	
	{

Tableau 4/Q.775 [T4.775], p.

2.3.3 *Reject (by the TC-user)*

A TC-user may decide to reject a component for any reason it finds appropriate, e.g. application protocol error, parameter missing in an operation or a reply, etc.

TCAP covers a number of cases, identified by the list of Problem Codes in Recommendation Q.773. In any of these cases, which correspond to situations where an operation or a reply is not correctly formatted, the TC-user may use the reject facility. Alternately, he may decide to return a failure indication (error component), which allows more detailed error and diagnostic information.

Reject of an operation invocation, or of a result, affect the whole operation: no more replies will be accepted for this invocation. Reject of a linked operation does not affect the linked-to operation.

This is illustrated in Table 5/Q.775 where, in example E2, TC-user A did not expect the option selection process (it may be an optional feature), and rejects the operation with the Problem Code ‘‘Unexpected Linked Operation’’. TC-user B may then decide to execute the test assuming a default option.

H.T. [T5.775]

TABLE 5/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(1, Test, Class = 1)
}
TC-INVOKE ind
(1, Test)
TC-INVOKE req
(2, 1, Option-selection, Class = 1)
}
	{

Loss of an operation invocation

The Table 6/Q.775 sequence illustrates the case, in example E1, where no response to the test is received before the time limit expires.

H.T. [T6.775]

TABLE 6/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(1, Test, Class = 1)
}	

{
Time limit:
TC-L-CANCEL ind
(1)
}	

Tableau 6/Q.775 [T6.775], p.

When a class 1 operation is lost, the TC-user is informed when the timer asociated with the operation expires. When a class 1 operation with a single result is lost, TCAP cannot indicate whether either the operation invocation, or the reply, was lost. If the application needs to discriminate between these two cases, it should do it in the application protocol (e.g. using the time-stamping or acknowledging the operation invocation before replying to it).

For a class 2 operation, loss will be considered as a success (whether the invocation, or the failure report, was lost). This, considering the probability of loss, may be acceptable for non critical operations (e.g. statistical measurements).

For a class 3 operation, loss is treated in the same way as operation failure, whether the invocation, or the success report, has been lost.

For a class 4 operation, loss will not be visible to TCAP.

Loss of a result

— Loss of a non final result is never detected by TCAP.

— Loss of a final result will eventually be indicated to the TC-user when the time limit is reached, but cannot always be unambiguously interpreted as the loss of a reply; of no non final result has been received, it may be that the invocation was lost.

Loss of a linked operation

The loss of a linked operation has the same effect as the loss of a non-linked operation. It has no effect on the linked-to operation.

Loss of a reject component

This case should be extremely infrequent, and no application should try to recover from such a situation. If the lost reject concerns an operation invocation, then when the operation timed out the TC-user which invoked the operation will consider that the invocation (or the reply) was lost, and react accordingly; if it concerns a reply, the originator of the reply will consider that it was correct: it will be up to the originator of the operation to detect the loss.

2.4.2 *Component duplication*

As message duplication is very infrequent in the Signalling System No. 7 network, scripts for No. 7 applications need not define sophisticated scenarios in anticipation of such situations. However, any application in which duplication would be unacceptable

should either define its own duplication detection mechanism or use a connection-oriented service.

Duplicate operation invocation

When an operation invocation is duplicated (by the service provider), the destination TC-user (B) may, or may not, detect the duplication:

— TC-user B detects the duplication: the best it can do in this case is to ignore the duplicate; rejection could be interpreted by the remote TC-user as rejection of the original invocation;

— TC-user B does not detect the duplication: this may happen when there is a master-slave relationship between A and B, and B executes the operation with no knowledge of the context.

Assuming the second case in exaple E1, a possible sequence could be as given in Table 7/Q.775.

H.T. [T7.775]

TABLE 7/Q.775

TC USER A	TC USER B	

{
TC-INVOKE req
(1, Test, Class = 1)
TC-RESULT-NL ind
(1, P1)
TC-RESULT-NL ind
(1, P1)
A detects an abnormal situation and rejects:
TC-U-REJECT req
(1, Problem Code)
TC detects an abnormal situation and rejects P2:
TC-L-REJECT ind
(1, Problem Code)
}
	{
	TC-INVOKE ind
	(1, Test)
	TC-INVOKE ind
	(1, Test)
	TC-RESULT-NL req
	(1, P1)
	TC-RESULT-NL req
	(1, P1)
	TC-RESULT-NL req
	(1, P2)
	TC-U-REJECT ind
	(1, Problem Code)
	}	

Undetected duplication of invocation }	{ }	
TC-R-REJECT ind (1, Problem Code)	{	

Tableau 7/Q.775 [T7.775], p.

In this sequence, TC-user B considers two independent test invocations, and responds to each of them. The first result P1 is accepted; TC-user A detects that P1 is received a second time, and rejects it; this terminates the operation, and causes result P2 to be rejected when received (reject by TCAP). Therefore, both activities at B's side will terminate on receipt of rejects.

Duplicate non-final result

If a non-final result is duplicated, TCAP cannot detect it, and will deliver it twice to the TC-user. Detection of this situation is left to the application.

Duplicate final result

If a final result is duplicated, TCAP can detect the situation: the second final result is considered as abnormal (the operation has been terminated by the first ''final'' result), and TCAP rejects it.

Table 8/Q.775 shows a sequence for example E1 where the third segment of the result is duplicated (by the network).

H.T. [T8.775]

TABLE 8/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(1, Test, Class = 1)
TC-RESULT-NL ind
(1, P1)
TC-RESULT-NL ind
(1, P2)
TC-RESULT-L ind
(1, P3)
Duplication of P3:
TC-L-REJECT ind
(1, Problem Code)
}
	{
	TC-INVOKE ind
	(1, Test)
	TC-RESULT-NL req
	(1, P1)
	TC-RESULT-NL req
	(1, P2)
	TC-RESULT-L req
	(1, P3)
	}

TC-R-REJECT ind (1, Problem Code) }	{

Tableau 8/Q.775 [T8.775], p.

Comment: Discarding of duplicates in all cases by TCAP would probably appear as a nicer issue. However, it should be noted that:

1) it would require another degree of complexity in TCAP, which contradicts the basic characteristics of TCAP in the connectionless approach;

2) it corresponds to a situation which is extremely infrequent, at least in the No. 7 network.

To cover these situations when required by an application, it would be better to use a connection-oriented network service approach, since duplication could then be detected and handled at the lower layers.

2.4.3 *Component missequencing*

For TCAP, the order of segmented results is not relevant: if the order is important to the TC-user, appropriate mechanisms should be defined in the application protocol (e.g. by introducing a numbering scheme to identify intermediate replies in a parameter of these replies, or by using a connection-oriented service).

Due to missequencing, a non final result may arive after a final result: when this occurs the non final result is rejected by TCAP.

The sequence in Table 9/Q.775 illustrates what happens in example E1 when the last part of the result is received before the second one: both TC-users are informed.

H.T. [T9.775]

TABLE 9/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(1, Test, Class = 1)
}
TC-INVOKE ind
(1, Test)
TC-RESULT-NL req
(1, P1)
}
	{

2) in reaction to this, the TC-user may decide to abort, continue, or end the dialogue. In the last two cases, when the TC-user notifies TCAP of its decision, the peer TC-user is informed of the reject.

Possible cases of reject by TCAP have been encountered in the previous sections. Whenever the component ID is recognised, rejection by TCAP causes the termination of the operation: a possible recovery is a new invocation of the terminated operation. When the rejected component is not identifiable, only the local TC-user is informed, and abort of the dialogue may be the appropriate reaction.

2.4.5 *Operation timer expiry*

When TCAP informs the TC-user of timer expiry (TC-L-CANCEL indication), it indicates that no more information related to the operation invocation (in particular, no reject) can be received. If the peer entity still sends information in relation with this invocation, this information will be discarded when received, provided that the component ID of the cancelled operation has not been reallocated. Premature reallocation of component ID values is normally avoided by correctly setting timer values: in order to

compensate for uncertainties in the amount of time required to send information from TC-user to another without accounting for the absolute worst case (which is also in general the most unlikely), an implementation-dependent mechanism avoiding premature reallocation of component IDs is required.

Timer expiry indication corresponds to an abnormal situation only in the case of a class 1 operation. The TC-user is then aware that either the invocation, or the reply, was lost. If no undesirable side effects arise, another invocation of the same operation can take place after timer expiry. This is illustrated by the sequence in Table 10/Q.775 for example E1.

H.T. [T10.775]

TABLE 10/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(1, Test, Class = 1)
}
	. TC-INVOKE ind (1, Test)

{
Timer expiry:
TC-L-CANCEL ind
(1)
TC-INVOKE req
(2, Test, Class = 1)
}	

Tableau 10/Q.775 [T10.775], p.

Timer expiry for a class 2 operation indicates that no failure was received nor will be accepted for this invocation: it is a definite indication of success (for class 2). A parallel situation applied to class 3 in case of failure. The indication of timer expiry for a class 4 operation is a local decision.

3 Dialogues

Whenever one of the operation handling primitives considered in § 2 is issued, a request is passed to TCAP, but nothing is sent to the remote TC-user until a primitive requesting transmission is issued. These primitives, and their relation with operation handling primitives, are considered now.

3.1 *Grouping of components in a message*

The effect of TC-user issuing a component handling primitive (unless this primitive has local effect only), is to build a **component** to be included in a **message** . The message is not transmitted until the TC-user requests it.

Note that a component may also be generated as a result of a TCAP reject: in this case this component is put in the next message for the dialogue unless it is aborted.

Provided that the maximum size of a message is not exceeded, several components can be grouped and sent to the remote end as a single message, thereby saving transmission overhead. This is done under control of the TC-user, which explicitly specifies when it wants (a) component(s) to be sent.

Example E3, as given in Table 11/Q.775, shows the beginning of a dialogue with a network service centre where a switch requests instructions (operation 1) and receives a request to connect the call to a given destination address, and a request to send information (e.g. announcement or message to be displayed) to the calling party. Both components are contained in a single message.

H.T. [T11.775]
TABLE 11/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(1, Provide-Instructions, Class = 1)
TC-BEGIN req
(control parameters)
}	

TC-BEGIN ind (control parameters) TC-INVOKE ind (1, Provide-instructions) TC-INVOKE req (2, 1, Connect-Call) TC-RESULT-L req (1, Send-Info) TC-CONTINUE req (control parameters) }	{

{
TC-CONTINUE ind
(control parameters)
TC-INVOKE ind
(2, 1, Connect-Call)
TC-RESULT-L ind
(1, Send-Info)
}	

Tableau 11/Q.775 [T11.775], p.

TC-BEGIN and TC-CONTINUE are transmission primitives described in § 3.2 below.

There may be one transmission primitive for each component, but the separation of primitives allows the grouping of components within a message. In addition, the information contained in the parameters of the transmission primitives (e.g. addressing information) applies to all the components included in the message.

At the originating side, the primitive requesting transmission appears after a component handling primitive; this indicates that transmission of the preceeding components has to take place immediately; it avoids indicating specific components to be transmitted with a given transmission primitive, and allows transmission primitives without any associated component.

At the destination side, the primitive requesting transmission appears first: it contains control information which is necessary for TCAP to deliver each of the components (if any) in the message; the last component of the message is indicated to the TC-user by the ‘‘Last Component’’ parameter. The components are delivered to the destination TC-user in the same order as they were passed to TCAP by the originating TC-user.

3.2 *Dialogue handling facilities*

When two TC-users co-operate in an application, more than one operation invocation is generally required. The resulting flow of components has to be identified so that:

1) components of the same flow can be related

2) flows corresponding to several instances of the same application can be identified and allowed to run in parallel.

Each such flow is identified, for the TC-user, by a dialogue and a corresponding Dialogue ID parameter. The dialogue handling facility provided for this purpose is the structured dialogue.

When only a single message is required to complete a distributed application, the Unidirectional message of the unstructured dialogue may be used. The originator does not expect a report of the outcome of the operation (i.e. may only invoke class 4 operations), but may receive a report of a protocol error if one occurs.

3.2.1 *Structured dialogue*

3.2.1.1 *General*

The use of dialogues allows several flows of components to co-exist between two TC-users. The Dialogue ID parameter is used in both operation handling and transmission (dialogue) handling primitives to determine which component(s) pertain(s) to which dialogue.

The Dialogue ID parameter is represented (by convention) by the first parameter in these primitives, starting with letter D. Each TC-user has its own reference for a given dialogue. Local references (those used on the interface) are represented here; mapping of these local references onto protocol references included in messages is done by TCAP.

Three primitives have been defined for handling dialogues under normal circumstances; they indicate dialogue begin (TC-BEGIN), continuation (TC-CONTINUE) or end (TC-END). Each of these primitives may be used to request transmission of 0, 1 or several components; these components may contain information relating to one or several operations.

Table 12/Q.775 illustrates a possible sequence for example E2, where the test request starts the dialogue, which ends when the test result has been sent.

H.T. [T12.775]

TABLE 12/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(D1, 1, Test, Class = 1)
TC-BEGIN req
(D1, Address)
}	

	{
	TC-BEGIN ind
	(D2, Address)
	TC-INVOKE ind
	(D2, 1, Test)
	TC-INVOKE req
	(D2, 2, 1, Option-selection, Class = 1)
	TC-CONTINUE req
	(D2)
	}

{
TC-CONTINUE ind
(D1)
TC-INVOKE ind
(D1, 2, 1, Option-selection)
TC-RESULT-L req
(D1, 2, Options)
TC-CONTINUE-req
(D1)
TC-END ind
(D1, normal)
TC-RESULT-L ind
(D1, 1, Test-result)
}
	{
	TC-CONTINUE ind
	(D2)
	TC-RESULT-L ind
	(D2, 2, Options)
	TC-RESULT-L req
	(D2, 1, Test-result)
	TC-END req
	(D2)
	}

Tableau 12/Q.775 [T12.775], p.

Any grouping of components is allowed in the messages of a dialogue: TCAP does not check, for instance, that a message terminating a dialogue does not include operation invocations of class 1. Full-duplex exchange of components is assumed: if a TC-user wants to introduce some restrictions, e.g. working in a synchronous mode as defined in ROSE, it would have to introduce the necessary procedures itself.

3.2.1.2 *Exchange of messages*

Transmission of messages is accomplished with the quality of service of the underlying layer services: no flow control or error recovery mechanisms are provided by TCAP.

— The first dialogue handling primitive of a dialogue must indicate dialogue begin (TC-BEGIN). Further messages must not be sent from the side originating the dialogue until a message is received in the backward direction, indicating dialogue continuation.

— If a TC-user tries to send a large number of messages in a short amount of time, no flow control mechanism in TCAP will prevent it.

— SCCP class 1 in-sequence delivery can be requested as an option, indicated by the Quality of Service parameter. Note that this option may not be available end to end when interworking with a network which does not provide it.

3.2.1.3 *Dialogue end*

TCAP places no restriction on the ability for a TC-user to request dialogue end. It follows that messages may be lost if no precautions are taken in the application on when the dialogue may end. In particular, if the application protocol allows both TC-users to issue TC-END primitives at about the same time, and if these primitives trigger transmission of components, it is likely that some (if not all) of these components will not be delivered to their respective destination TC-users.

It is up to the application to define, if necessary, its own rules concerning the right to end a dialogue: TCAP will not check them. Any message received for a terminated dialogue is discarded if it requests dialogue end, and otherwise causes the dialogue to be aborted at the remote entity.

The differences between the three ways of ending a dialogue are as follows.

Prearranged end

A typical application is the access to a distributed database, where the requesting user (TC-user A) does not know where the information it seeks is located. TC-user A broadcasts a request to each location which might have the information required, and will eventually receive a response from the TC-user which holds this information. Prearranged end avoids messages from the other destinations saying: ‘‘I do not have this information’’. Only the responding destination may continue the dialogue (if so wished); all other destination will, by convention, end the dialogue locally; the originator of the requests will also end the dialogues with the non-responding destinations locally, when it receives the response to its request. Note that the convention is between applications: TCAP does not check that it is respected, nor is it indicated in the TCAP protocol.

Example E4 in Table 13/Q.775 illustrates this situation, with two destinations B1 and B2; two dialogues (D1, D2) and (D3, D4) are started; B1 happens to own the requested information, and decides to continue the dialogue.

H.T. [T13.775]
TABLE 13/Q.775

TC USER A	TC USER B1	TC USER B2
{		
TC-INVOKE req		
(D1, 1, Question)		
TC-BEGIN req		
(D1, Address)		
TC-INVOKE req		
(D3, 1, Question)		
TC-BEGIN req		
(D3, Address)		
TC-CONTINUE ind		
(D1)		
TC-RESULT-L ind		
(D1, 1, Response)		
D1 goes on		
D3 ends locally		
TC-END req		
(D3, local)		
}	{	
TC-BEGIN ind		
(D2, Address)		
TC-INVOKE ind		
(D2, 1, Question)		
TC-RESULT-L req		
(D2, 1, Response)		
TC-CONTINUE req		
(D2)		
}	{	
TC-BEGIN ind		
(D4, Address)		
TC-INVOKE ind		
(D4, 1, Question)		
B2 does not have the information:		
TC-END req		
(D4, local)		
}		

Tableau 13/Q.775 [T13.775], p.

Prearranged end may also be used when a TC-user wants to send information, and does not expect a reply of any kind afterwards.

Basic end

When a TC-user issues the TC-END request primitive, it causes transmission of any pending components to the remote end. TCAP does not check that all operation invocations have received a response when dialogue end is requested: no notification is given to the TC-user that any pending operation invocations have not received a final result.

At the receiving end, the dialogue is considered terminated when all the components received within the message indicating the end have been delivered to the TC-user.

Example: the dialogue ends when the test in example E1, Table 14/Q.775, receives a response.

H.T. [T14.775]

TABLE 14/Q.775

TC USER A	TC USER B
{	
TC-END ind (D1)	
TC-RESULT-NL ind (D1, 1, P1)	
TC-RESULT-NL ind (D1, 1, P2)	
TC-RESULT-L ind (D1, 1, P3)	
End of dialogue for A	
}	{
TC-RESULT-NL req (D2, 1, P1)	
TC-RESULT-NL req (D2, 1, P2)	
TC-RESULT-L req (D2, 1, P3)	
TC-END req (D2, normal)	
End of dialogue for B	
}	

Tableau 14/Q.775 [T14.775], p.

Abort by the TC-user

The abort facility allows the TC-user to stop the dialogue at any time. A typical case is when the user abandons the service. The main differences between this and normal ending are:

— any components for which transmission is pending are not sent to the peer entity;

— peer-to-peer information can be indicated at the time the abort is issued, and this is delivered to the remote TC-user.

The sequence given in Table 15/Q.775 shows a user abandonment in example E2.

3.2.1.4 *Message-related abnormal situations*

These are considered independently from the effects of such events in the Component sub-layer.

Message loss

TCAP provides no protection against message loss. Three cases are identified:

1) the message begins a new dialogue: the dialogue will exist at the originating side only, and no message will be allowed in either direction. Eventually, an implementation-dependent mechanism of TCAP ends the dialogue at the originating end;

2) the message continues an existing dialogue: loss is not detected. TCAP will react (or not) to the loss of included components as indicated in § 2.4.1 above;

3) the message ends a dialogue: TCAP will eventually react if this message contained a response to a class 1 operation: otherwise an implementation-dependent mechanism may end the dialogue at the destination end.

H.T. [T15.775]

TABLE 15/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(D1, 1, Test, Class = 1)
TC-BEGIN req
(D1, Address)
}	

TC-BEGIN ind (D2, Address) TC-INVOKE ind (D2, 1, Test) TC-INVOKE req (D2, 2, 1, Option-selection, Class = 1) TC-CONTINUE req (D2) }	{

{
TC-CONTINUE ind
(D1)
TC-INVOKE ind
(D1, 2, 1, Option-selection)
User abandon:
TC-U-ABORT req
(D1, Cause)
}	

TC-U-ABORT ind (D2, Cause) }	{

Tableau 15/Q.775 [T15.775], p.

Message duplication

Duplication of a BEGIN message causes two transactions to be opened, as indicated below: each of these transactions has its own local ID, and the same destination ID. The TC-user eventually detects that something is wrong, and both dialogues are aborted.

The sequence given in Table 16/Q.775 illustrates a duplication of the BEGIN message in Example E2.

H.T. [T16.775]

TABLE 16/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(D1, 1, Test, Class = 1)
TC-BEGIN req
(D1, Address)
}	

{
TC-CONTINUE ind
(D1)
TC-INVOKE ind
(D1, 2, 1, Option-select)
}
	{
	TC-BEGIN ind
	(D2, Address)
	TC-INVOKE ind
	(D2, 1, Test)
	Duplicated BEGIN:
	TC-BEGIN ind
	(D3, Address)
	TC-INVOKE ind
	(D3, 1, Test)
	Response to the first Begin
	TC-INVOKE req
	(D2, 2, 1, Option-select, Class = 1)
	TC-CONTINUE req
	(D2)
	Response to the second Begin
	TC-INVOKE ind
	(D3, 2, 1, Option-select, Class = 1)
	TC-CONTINUE req
	(D3)
	}

{
TC-CONTINUE ind
(D1)
TC-INVOKE ind
(D1, 2, 1, Option-select)
TC-user considers that this invocation is abnormal, and may reject it, or abort one of the dialogues:
TC-U-ABORT req
(D1, Cause)
}	

	{
	TC-U-ABORT ind
	(D3, Cause)
	}

Tableau 16/Q.775 [T16.775], p.

At that moment, there is still one dialogue (with local ID D2) at TC-user B's side, but no dialogue at A's side. TC-user B will receive an indication from TCAP when operation 2 of dialogue D2 timeouts with no reply (TC-L-CANCEL ind), and may then decide to abort D2. Note that the situation would be more difficult to detect, had TC-user B not invoked a class 1 operation.

Duplication of a CONTINUE message is not detected by TCAP.

When an END message is duplicated, the second message is received with an ID which does not correspond to an active dialogue: TCAP reacts by discarding the duplicate message.

Missequencing of messages

When the missequenced messages involve neither the beginning, nor the end of a dialogue, missequencing is not detected by TCAP, and may result in component missequencing, to which TCAP would react as indicated in § 2.5.3 above.

When a message indicating dialogue continuation arrives after a message indicating the end of the same dialogue, it is not delivered, and causes TCAP to abort the dialogue; the TC-user will probably detect the loss when receiving a premature dialogue end indication. If the application needs to recover from this case, a new dialogue should be started.

Message corruption

When receiving a corrupted message, TCAP reacts as indicated in Recommendation Q.774.

Table 17/Q.775 shows the sequence of primitives when TCAP decides to abort the dialogue after receiving a corrupted message in example E2.

H.T. [T17.775]

TABLE 17/Q.775

TC USER A	TC USER B

{
TC-INVOKE req
(D1, 1, Test, Class = 1)
TC-BEGIN req
(D1, Address)
}	

 TC-BEGIN ind (D2, Address) TC-INVOKE ind (D2, 1, Test) TC-INVOKE req (D2, 2, 1, Option-select, Class = 1) TC-CONTINUE req (D2) }	{

{
Corrupted message:
TC-ABORT ind
(D1, Cause)
}
	
	. TC-ABORT ind (D2, Cause)

Tableau 17/Q.775 [T17.775], p.

3.2.1.5 *Relations between dialogue handling and operation handling*

Depending on the moment when the dialogue end is requested, the TCAP facilities associated with an operation will be available until the end of the dialogue, or not. The following gives some guidelines on when dialogue end can be requested; if these are not respected, TCAP will not refuse the request for dialogue end.

The problems that may result from the collision of messages requesting dialogue end have been considered above.

Normal end should not be requested when:

— there are operation invocations pending for the dialogue;

— the application protocol anticipates that replies being transmitted with the termination request could be rejected.

In addition, a request for dialogue end must not trigger transmission of operation invocations, since no reply could be received for these operations.

Many applications might not define recovery scenarios in response to a rejected reply. This legitimises the transmission of replies or of class 4 operations in a message indicating dialogue end. The other applications should either use the connection-oriented network service approach, or end the dialogue with a message containing no component, that would be sent only when a reject indication can no longer be received.

3.2.2 *Unstructured dialogue*

A Unidirectional message will contain either only class 4 operation invocations or reports of protocol errors in such invocations. Multiple components can be transmitted in a Unidirectional message provided that the maximum size of a message is not exceeded.

4 Application service elements and application entities

4.1 *Introduction*

This material supplements preceding material providing guidelines on the usage of TC by describing what needs to be included in an Application Entity (AE) specification. This material is based on CCITT Recommendations X.219 and X.229 and requires further study.

CCITT Recommendation Q.700, § 3.2.3.6, describes how Application Service Elements (ASEs) and Application Entities (AEs) are structured and how an AE is addressed in Signalling System No. 7.

This section illustrates that architecture, considering the functional decomposition of an application, and describes how AEs, ASEs, operations and errors should be defined.

4.2 *Decomposition of functionality*

Application process functions communicate through one or more Application Entities (AEs). The combination of two peer AEs plus their interaction is called the Application Context. An AE consists of communications for one or more functions of an application. Each communications function forms an ASE which is an integrated set of actions and may be used in more than an AE. TCAP is itself an ASE which is used by other ASEs as well as being common to AEs (see § 3.2.3.6/Q.700). An ASE identifies one or more operations and specifies how those operations are used; that is, which peer entity may invoke which operations, and in what order. Operations may be selected from one or more libraries.

An ASE provides a service to the user of the ASE. An ASE is used by two complementary AEs: the consumer of the service and the supplier of the service. The consumer of the service is the end that initiates the AE to AE communication. An ASE user is thus generally asymmetric.

Within an ASE, the mechanism for providing the ASE service is the invocation of operations by the service requestor on the service provider. Each operation provides a part of the service in an inherently asymmetric manner since it is invoked by one AE and executed by the peer AE. An ASE generally includes more than one operation. An ASE user is, in general, not limited to either invoking or performing operations, but may both invoke or perform the same or different operations. Also, an ASE user may exist at a pair of nodes such that either node may request the same service from the other node. That is, the AEs at the nodes may be symmetric, both invoking and executing the same operations.

Note — Primitives which provide a standard service interface for the access of ASEs within AEs are for further study.

Figure 3/Q.775 illustrates the decomposition of this functionality and provides examples.

Figure 3/Q.775, (N), p.

4.3 *How to specify an AE*

CCITT Recommendation Q.700, § 3.2.3.6, describes how two Signalling System No. 7 Application Processes communicate via Application Entities, and also the structure of an AE.

The application designer should provide a definition for each type of AE. It should contain:

— A general description of the services supported by the combination of the two peer AEs and communicating by a dialogue. (In Recommendation X.229 terminology, this corresponds to the ‘‘Application Context’’).

— A definition of the complete application protcol between the peer AEs by:

— identifying each ASE constituting the AE, and

— indicating which of the peer AEs initiates the service.

— Any special constraints to ensure that peer AEs with different versions are compatible.

A formal specification of the application context using the Recommendation X.229 APPLICATION-CONTEXT macro is for further study.

Since each AE constitutes a single coding domain for operation and error code values (addressed by SCCP subsystem number in a connectionless network service environment), each operation or error code value must be unique within the AE (see § 4.5).

4.4 *How to specify an ASE*

The definition of an ASE is part of the stage 3 of the service description methodology, as defined by Recommendation I.220.

The ASE description should provide:

— A general description of the ASE and its procedures.

— The information flows between the entities which are communicating to support the service, based on stage 2, with additions and enhancements that are needed as part of the protocol design.

— A detailed description of the ASE protocol. This includes the sequence in which operations may be invoked, and the reaction to abnormal situations. The definition should include how protocol version interwork. Dialogue begin, continuation and end should be specified. This section should describe the interaction between the ASE and the TCAP component sub-layer expressed in terms of the primitive interface.

— SDL diagrams.

Recommendation X.229 (ROSE) defines an APPLICATION-SERVICE-ELEMENT macro which may be used to specify an ASE formally. It identifies which operations are contained in the AE and how they are invoked. The use of this macro in Signalling System No. 7 is for further study.

4.5 *How to specify operations and errors*

4.5.1 *Information needed to specify operations and errors*

To specify an operation, the following items must be defined:

— The operation name.

— The operation code. This may be local or global. See § 4.5.2.

— The operation class. A value in the range 1 to 4 as defined in § 2.2.1.

— The parameters accompanying the operation invocation (input parameters). Further essential information to supplement that provided in the parameters with the original invocation may be requested using linked operations.

— The parameters that may be returned as the result of a successful outcome (Return Result), whenever the operation reports success (possitive output parameters). The way these parameters are actually passed (in a single component or several) is no part of the operation description.

— The error codes and associated parameters that may be returned as the result of an unsuccessful outcome (Return Error) of the operation execution, whenever this operation reports failure (negative output parameters). An error code must be present when reporting failure, and all the possible values be defined as part of the operation description.

— The allowed linked operations (see § 2.2.2).

— The timer value for completion of the operation.

The operation description consists of a Table indicating the eight items above, together with a short prose description of what the operation does. A formal definition using Annex A/Q.773 OPERATION and ERROR macros should also be included to unambiguously indicate which parameters are mandatory, which are optional with default values as applicable, and which individual, sets or sequences of parameters are legal as input, positive output, and negative output. The OPERATION and ERROR type (macro)

definitions are exported from the TCAP definitions (Annex A/Q.773) and need to be imported into the ASE being defined in order to define operations and errors.

The syntax of the OPERATION MACRO (reproduced from Annex A/Q.773) is as follows:

OPERATION MACRO ::=

BEGIN

TYPE NOTATION ::= Parameter Result Errors Linked Operations

VALUE NOTATION ::= valu { ALUE CHOIC { value(VALUE CHOICE localValue INTEGER, value(VALUE CHOICE globalValue OBJECT IDENTIFIER }

Parameter ::= ''PARAMETER'' Named Type | empty

Result ::= ''RESULT'' ResultType | empty

ResultType ::= NamedType | empty

Errors ::= ''ERRORS'' { *UErrorNames } *U | empty

LinkedOperations ::= ''LINKED'' { *ULinkedOperationNames } *U | empty

ErrorNames ::= ErrorList | empty

ErrorList ::= Error | ErrorList '','' Error

Error ::= value (ERROR) —— *shall reference an error value* | type —— *shall reference an error type if no error value* | type —— *is specified*

LinkedOperationNames ::= OperationList | empty

OperationList ::= Operation | OperationList '','' Operation

Operation ::= value (OPERATION) —— *shall reference an operation value* | type —— *shall reference an operation type if no error value* | type —— *is specified*

NamedType ::= identifier type | type

END

ERROR MACRO ::=

BEGIN

TYPE NOTATION ::= Parameter

VALUE NOTATION ::= value (VALUE CHOIC { valu { ALUE CHOICE localValue INTEGER, valu { ALUE CHOICE globalValue OBJECT IDENTIFIER }

Parameter ::= ''PARAMETER'' NamedType | empty

NamedType ::= identifier type | type

END

The use of local and global values is explained in § 4.5.2.

As an example, the CUGCheck2 operation, which is used to check whether an incoming call is compatible with the CUG characteristics of the called party, is described here in both (abbreviated) formal notation, and in the form of a table.

4.5.2 *Example of operation description*

(*Note* — Arbitrary section numbers are used in this example.)

3.4.3.1 *Description of operations*

3.4.3.1.1 *CUG check 1*

This operation is used between the originating exchange of a call and a dedicated point for CUG validation check of the calling user.

3.4.3.1.2 *CUG check 2*

This operation is used between the terminating exchange of a call and a dedicated point for CUG validation check of the called user.

3.4.3.2 *Parameters of operations and outcomes*

3.4.3.2.1 *CUG Check 1*

H.T. [T18.775]

CUG Check 1	Timer = x sec	Class = 1	Code = 00000001
Parameters with Invoke	Opt/Man	Reference	

{
CallingUserIndex
CUGCallIndicator
CallingPartyNumber
}
	O M M	3.4.3.3.1 3.4.3.3.2 3.4.3.3.3	

Parameters with Return Result			

{
CUGInterlockCode
CUGCallIndicator
}
	O M	3.4.3.3.5 3.4.3.3.2	

Linked Operations			
Not applicable			
Errors			
UnsuccessfulCheck		3.4.3.3.7	

Tableau du § 3.4.3.2.1, [T18.775], p.

cUGCheck1 OPERATION PARAMETER SEQUENC { | allingUserIndex OPTIONAL, cUGCallIndicator, SEQUENC { | callingPartyNumber } RESULT SEQUENC { | UGInterlockCode OPTIONAL, cUGCallIndicator } ERRORS SEQUENCE { | nsuccessfulCheck } ::= 1

3.4.3.2.2 *CUG check 2*

H.T. [T19.775]

CUG Check 2	Timer = x sec	Class = 1	Code = 00000010
Parameters with Invoke	Opt/Man	Reference	

{
CUGInterlockCode
CUGCallIndicator
CalledPartyNumber
}
	M M M	3.4.3.3.5 3.4.3.3.2 3.4.3.3.4	

Parameters with Return Result			

{
CalledUserIndex
CUGCallIndicator
}
	O M	3.4.3.3.6 3.4.3.3.2	

Linked Operations			
Not applicable			
Errors			
UnsuccessfulCheck		3.4.3.3.7	

Tableau du § 3.4.3.2.2, [T19.775], p.

cUGCheck 2 OPERATION PARAMETER SEQUENC { | UGInterlockCode, cUGCallIndicator, SEQUENC { | calledPartyNumber } RESULT SEQUENC { | alledUserIndex OPTIONAL, cUGCallIndicator } ERRORS SEQUENCE { | nsuccessfulCheck } ::= 2

3.4.3.3 *Parameter coding*

3.4.3.3.1 The CallingUserIndex is the local index at the calling user to identify a particular CUG he belongs to.

H.T. [T20.775]

CallingUserIndex	Code = 10000001
Contents	Meaning
IA5 Character String One IA5 character represents one digit of the CUG index value }	{

Tableau du § 3.4.3.3, [T20.775], p.

callingUserIndex ::= [1] IMPLICIT LocalIndex LocalIndex ::= IA5 STRING —— *The maximum number of digits is four*. .bp

3.4.3.3.2 The CUGCallIndicator indicates whether the call is requested or designated as a CUG call and whether outgoing access is requested or allowed.

H.T. [T21.775]

CUGCallIndicator	Code = 10000010
Contents	Meaning

{
00000000
00000001
00000010
00000011
}
Non-CUG call
Non-CUG call
CUG call with outgoing access
CUG call without outgoing access
}
	{

Tableau du § 3.4.3.3.2, [T21.775], p.

cUGCallIndicator ::= [2] IMPLICIT CallIndicator

CallIndicator ::= INTEGE { nonCUGCall (0), nonCUGCall (1), outgoingAccessAllowedCUGCall (2), outgoingAccessNotAllowedCUGCall (3) }

3.4.3.3.3 The CallingPartyNumber is the network (e.g. E.164) number of the calling party. It is expressed in the same manner as the ISUP Calling party number in § 3.7 of Recommendation Q.763. The code of this parameter is ‘‘10000011’’.

H.T. [T22.775]

CallingPartyNumber	Code = 10000011
Contents	Meaning

{
— | (em encoded per § 3.7/Q.763
}	

Tableau du § 3.4.3.3.3, [T22.775], p.

callingPartyNumber ::= [3] IMPLICIT OCTET STRING —— *contents encoded per § 3.7/Q.793*

3.4.3.3.4 The CalledPartyNumber is the network (e.g. E.164) number of the called party. It is expressed in the same manner as the ISUP Called party number in § 3.6 of Recommendation Q.763. The code of this parameter is ‘‘10000100’’.

H.T. [T23.775]

CalledPartyNumber	Code = 10000100
Contents	Meaning

{
— | (em encoded per § 3.6/Q.763
}	

Tableau du § 3.4.3.3.4, [T23.775], p.

calledPartyNumber ::= [4] IMPLICIT OCTET STRING —— *contents encoded per § 3.6/Q.793* .bp

3.4.3.3.5 The CUGInterlockCode is the code to uniquely identify a CUG inside the network. It is expressed in the same manner as the ISUP CUG interlock code in § 3.13 of Recommendation Q.763. The code of this parameter is ‘‘10000101’’.

H.T. [T24.775]

CUGInterlockCode	Code = 10000101
Contents	Meaning

{
— | (em encoded per § 3.13/Q.763
}	

Tableau du § 3.4.3.3.5, [T24.775], p.

CUGInterlockCode ::= [5] IMPLICIT OCTET STRING —— *contents encoded per § 3.13/Q.793*

3.4.3.3.6 The CalledUserIndex is the local index at the called user to identify a particular CUG he belongs to. Refer to § 3.4.3.3.1. The code of this parameter is ‘‘10000110’’.

H.T. [T25.775]

CalledUserIndex	Code = 10000110
Contents	Meaning
IA5 Character String One IA5 character represents one digit of the CUG Index value }	{

Tableau du § 3.4.3.3.6, [T25.775], p.

CalledUserIndex ::= [6] IMPLICIT LocalIndex

3.4.3.3.7 *Errors*

H.T. [T26.775]

UnsuccessfulCheck	Code = 00000001
Parameters	
Cause	3.4.3.3.8

Tableau du § 3.4.3.3.7, [T26.775], p.

unsuccessfulCheck ERROR PARAMETER { | ause } ::= 1

3.4.3.3.8 The Cause indicates the reason why the CUG check is unsuccessful.

H.T. [T27.775]

Cause	Code = 10000111
Contents binary (decimal)	Meaning
00110010 (50) Requested facility not subscribed }	{
00110101 (53) Outgoing calls barred within CUG }	{
00110111 (55) Incoming calls barred within CUG }	{
00111110 (62) InconsistencyInDesignatedOutgoingAccessInformationAndSubscriberClass }	{
01010110 (90)	Non-existent CUG
01010111 (87)	Called user not member of CUG
01011000 (88)	Incompatible destination
10000000 (110)	Inconsistency in data

Tableau du § 3.4.3.3.8, [T27.775], p.

cause ::= [7] IMPLICIT CauseCode

CauseCode ::= INTEGE { requestedFacilityNotSubscribed (50), outgoingCallsBarredWithinCUG(53), incomingCallsBarredWithinCUG(55), inconsistencyInDesignatedOutgoingAccessInformationAndsubscriberClass(62), nonExistentCUG(90), calledUserNotMemberOfCUG(87), incompatibleDestination(88), inconsistencyInData(110) }

4.5.3 *Allocation and management of operation and error codes*

The simple approach is to provide one module containing the definition of the operations and errors it uses as a self-contained local domain.

Before defining a new operation, the application designer should check all modules to see whether a similar operation already exists. To avoid redefining the operation in a number of modules, methods are required which allow a module to import the definition of the operations it uses from other modules. If the opertion does not exist, the designer should specify it locally.

Example: | Operation code 00000010 has one meaning for ASE1, and probably a completely different meaning for ASE2; two domains are involved.

Note that many domains may be used by one ASE; however, for simplicity, it is assumed in the following that an ASE uses only one domain.

In addition to its local operation, an ASE may need to make use of operations which are already defined in another domain. There are two methods for doing so:

— import operation and error types from other modules;

— import operation and error values from other modules.

4.5.3.1 *Import of types*

The definition of an operation type includes the notational aspects (see the OPERATION MACRO above), without allocating the code values.

It may be desirable to import the type of an already existing operation, however the importing module may want to allocate its own local codepoint to the imported operation or error. The imported operation or error becomes a member of the local domain of that module.

If two different modules import a given operation by type, its codepoint in each of the importing local domains is generally different.

Importing by type allows a common description of operations. A module importing by types only uses a single domain (its local domain), as represented in Figure 4/Q.775.

Figure 4/Q.775, (N), p.

4.5.3.2 *Import of values*

When operation values are imported, the type and the coding are the same in the exporting and importing ASEs.

A module importing operations or errors by value makes use of:

— a local domain for its local operations and

— the exporting domains for its imported operations.

A global value is required in the second case to avoid ambiguity between local codepoints and imported codepoints, as represented in Figure 5/Q.77.

Figure 5/Q.775, (N), p.

4.6 *Applying the concept to service protocols*

The first step, before assigning operation codes, is to examine the service ASEs (each an integrated set of actions) and assign them to AEs. The extremes are, on one hand, that all service ASEs are assigned to one AE and, on the other hand, that each AE is composed of only one service ASE. The likely case is several groupings of service ASEs.

Each AE should be identified by a SSN, but not necessarily a fixed SSN specified in Recommendation Q.713. Within an AE, an operation code assignment scheme is used, so that no two operations can have the same operation code.

MONTAGE: PAGE 134 = PAGE BLANCHE

SECTION 2

TEST SPECIFICATION

Recommendation Q.780

SIGNALLING SYSTEM NO. 7 TEST SPECIFICATION

GENERAL DESCRIPTION

1 General

This Recommendation is an introductory Recommendation to the test specifications of Signalling System No. 7. The test specifications are contained in Recommendations Q.781-Q.783. This Recommendation defines the scope and purpose of the test specification and identifies guidelines that are either specific to the particular protocol under test, or are more general. In addition it identifies functional requirements imposed by the test specification.

2 Geneal principles of test specifications

The test specification aims at testing protocol conformance in a given implementation. This is independent of a given implementation and does not generally imply any modification of the signalling point under test. However, it is recognized that certain tests require capabilities of the system that are not explicitly defined in the relevant Recommendation, and these capabilities may not be present in all implementations. As a consequence, certain tests may not be possible in all implementations.

3 Scope of the test specification

The test specification is intended to cover all aspects of Signalling System No. 7. However the initial Recommendations cover the message transfer part Q.701-Q.707, and the telephone user part Q.721-Q.724. The test specification is not a definition of the protocol, this is contained in Recommendations Q.701-Q.707 and Q.721-Q.724 as appropriate.

4 Field of application

The test specification applies in the international network, and if appropriate in the national network. In the international network, the actual tests to be performed will be the subject of appropriate bilateral agreements beween the two or more Administrations/RPOAs concerned.

5 Method of application

The test specification fulfils the requirements for both validation testing and compatibility testing. See §§ 5.1 and 5.2 for an explanation of these terms.

All tests in the test specification are validation tests (VAT), and in addition those marked with an asterisk are also compatibility tests (CPT).

5.1 *Validation testing*

The function of validation testing is to check that a given implementation conforms to the relevant CCITT Recommendations of the Signalling System. These validation tests could apply both in the national and international networks. The validation test is a pre-requisite of compatibility testing (see § 5.2) and is performed under the responsibility of each Administration/RPOA. These tests will generally be performed without the cooperation of another Administration/RPOA, although this is not precluded should this arrangement prove convenient. Validation testing will be performed on a signalling point that is not in service.

The validation test is performed on one signalling point.

It is suggested that the validation test, or subset, is repeated when the implementation is upgraded or modified in any functional way.

Validation testing may require the use of a simulator to check the operation of the signalling point under test. The specification of this simulator is not explicitly covered by these Recommendations although the general requirements are implicit in the test specification.

In validation testing, the signalling point under test is called SP‘‘A’’.

5.2 *Compatibility testing*

The objective of compatibility testing is to check for the correct interworking of two implementations. To perform compatibility testing the two nodes involved are interconnected. The specification is written for the interconnection of two given implementations for the first time. For subsequent interconnections of the same two implementations a subset of tests may prove sufficient. These tests will not only be performed on a new signalling point, but also on a signalling point already in service.

Each Recommendation identifies a list of tests that may be suitable for compatibility testing, but the actual tests to be performed will be bilaterally agreed between the Administrations/RPOAs concerned.

Certain of the tests identified in the test list as compatibility test may disturb the operation of the exchange, whereas others may not. Any tests which may cause disturbance to the exchange should be carefully selected to meet the operational criteria of the two Administrations/RPOAs.

The satisfactory completion of compatibility testing should be bilaterally agreed.

When a change to the signalling network is made, tests selected from those identified as compatibility tests may be appropriate. In general the tests performed under these circumstances will be the minimum number to ensure that compatibility between points in the network is still maintained.

In compatibility testing, each signalling point may in turn consider itself to be SP‘‘A’’, i.e. tests are performed on both signalling points involved.

5.3 *Test configuration*

For both validation and compatibility testing the point under test is connected to the test environment and becomes part of the ‘‘test configuration’’. The test configuration satisfies all of the following three criteria:

— The point under test will be connected by one or more signalling linksets (real or simulated), which may or may not be interconnected.

— The capability of generation and reception of test traffic, where applicable.

— The ability to perform the described test, notably the facility to store and analyze messages to the appropriate degree.

6 Functional requirements imposed by the test specification

The functional description that follows is intended to identify the functional requirements imposed by the test specification. It does not imply any physical partitioning of equipment in real systems. See also Recommendation Q.701, § 2.2.1.

6.1 *Level 1*

The test specification assumes the availability of a suitable signalling data link with the parameters identified in the relevant Q Recommendations, e.g. Q.702 (referring to Recommendation G.821).

In validation testing the signalling data link may be a pseudo-signalling data link, in which case it should preferably have similar/identical characteristics to the signalling data links likely to be encountered in service. Simulation of deterioration of the transmission link may not be necessary if the emulator includes the capability to simulate abnormal conditions on the signalling data link.

In compatibility testing the signalling data link is the actual signalling data link that will be used in service.

6.2 *Level 2*

The level 2 test environment consists of four items (see Figure 1/Q.780):

— the level 3 simulator;

— the test simulator;

— the signalling link monitor (see § 7);

— the signalling data link.

6.2.1 *Level 3 simulator*

During the level 2 tests it is necessary to inject signalling messages and indications to and from the level 2 under test. It is desirable that the level 3 function used is the actual level 3 of the MTP with some additional functions for test purposes.

6.2.2 *Test simulator*

During level 2 testing it is necessary to inject some abnormal signal units (as well as normal signal units) to fully test the level 2 under test, the test simulator should have this function. In addition the simulator should have the capability to receive and check signal units from the level 2 under test. The generation of certain abnormal sequences of signal units should also be a capability of the test simulator.

6.3 *Level 3*

The level 3 test specification assumes that the level 2 has already been tested satisfactorily. However, certain tests will in addition explicitly test the level 2/3 interface.

The level 3 test environment consists of 3 items (see Figure 2/Q.780):

— the simulator of upper levels;

— simulated network including test simulator and signalling data links;

— the signalling link monitor(s) (see § 7).

6.3.1 *Simulator of upper levels*

During level 3 testing it is necessary to inject signalling messages into level 3 for testing, e.g. message loss during changeover. It is desirable that the simulator used should be as close as possible to the actual upper level to be used. In addition an MML interface is assumed. The level 3 under test must use an already tested level 2.

6.3.2 *Simulated network including test simulator*

During level 3 testing it is necessary to inject some abnormal messages (as well as normal messages) to check the level 3 under test, the simulated network including test simulator should have this function. In addition the test simulator should have the capabilities to receive and check messages from the level 3 under test. The generation of certain abnormal sequences of messages should also be a capability of the test simulator. The test simulator must include an already tested level 2.

6.4 *TUP*

The TUP test specification assumes a tested MTP for compatibility tests but no assumption is made about message transfer between the TUP under test and the TUP tester for validation tests.

The TUP test environment consists of three items (see Figure 3/Q.780):

— the TUP tester;

— a stable signalling relation and telephone circuits;

— a monitor of TUP messages and telephone circuits.

6.4.1 *TUP tester*

The TUP tester is required to simulate TUP protocol operations and some exchange call control operations.

6.4.2 *Monitor*

The monitor is required to monitor and record TUP message sequences and to monitor the result of call control operations on the controlled telephone circuits. This includes checking that tones are correctly received and that speech/information transfer is possible.

7 Signalling link monitor(s)

The test specification assumes the availability of a signalling link monitor and a suitable access point for connection of the monitor as specified in Recommendation Q.702, § 4.

The test specification does not attempt to specify what a signalling link monitor should be, but instead the functional requirements are identified in general terms. A signalling link monitor will be used for decoding of signal unit sequences during testing and to give the operator confidence that the signalling protocol has been correctly observed.

The requirements imposed on a signalling link monitor will be different for the two types of testing. For validation testing detailed decoding down to a field level will be required, but for compatibility testing decoding down to a message level may be adequate.

In addition it should be noted that compatibility testing will be a function performed numerous times on a signalling point, whereas validation testing will be performed once only, except under certain circumstances of upgrading of the signalling point.

Note — It should be oserved that implementations may include a signalling link monitor as an intrinsic part of the signalling point, however, for validation testing this cannot necessarily be relied upon. In addition, the test specification does not attempt to perform the function of testing the accuracy of any signalling link monitor implemented in the signalling point, however, certain conclusions will inevitably be made from the performance of validation testing.

Figure 1/Q.780, (N), p.

Figure 2/Q.780, (N), p.

Figure 3/Q.780, (N), p.