The drawings contained in this Recommendation have been done in Autocad.
Recommendation Q.774

TRANSACTION CAPABILITIES PROCEDURES

1 Introduction

Transaction capabilities (TC) allows TC users to exchange components via transaction capabilities application part (TCAP) messages. Procedures described in this section specify the rules governing the information content and the exchange of TCAP messages between TC users.

1.1 *Basic guideline*

To maximize flexibility in service architecture and implementation style, TCAP procedures restrict themselves to supporting the exchange of components between TC users. Application specific (TC user) procedures are not part of TCAP.

When the selection of a parameter value associated with a primitive that is required by a lower layer (sub–layer) is not relevant to that layer (sub–layer), the value is simply passed down through the primitive interface. The same assumption applies to the parameters received from a lower layer through the primitive interface which are not required for TCAP functions.

1.2 *Overview*

Section 2 describes addressing rules for TC messages. Section 3 describes transaction capabilities based on a connectionless network service. Section 4 describes transaction capabilities based on a connection oriented network service.

2 Addressing

In a Signalling System No. 7 environment using a connectionless network service, TC messages will use any of the addressing options afforded by the signalling connection control part (SCCP). Assignment and use of global titles may be network and/or application specific.

Addressing options available for the intermediate service part (ISP) are for further study. Addressing options whene other network providers are used are for further study.

3 Transaction capabilities based on a connectionless network service

3.1 *Sub–layering in TCAP*

TCAP procedure is divided into component sub–layer procedure and transaction sub–layer procedure. The component sub–layer procedure provides a TC user with the capability of invoking remote operations and receiving replies. The component sub–layer also receives dialogue control information from a TC user, and, in turn, uses transaction sub–layer capabilities for transaction control.

The component sub–layer provides two kinds of procedures:

– dialogue handling;

– component handling.

3.2 *Component sub–layer procedures*

3.2.1 *Normal procedure*

3.2.1.1 *Component handling procedure*

3.2.1.1.1 *Mapping of TC component handling service primitives to component types*

Recommendation Q.771 describes the services provided by the component sub–layer by defining the service interface between the TC user and the component sub–layer and the interface between the component sub–layer and the transaction sub–layer. Component handling procedures map component handling service primitives onto components, which constitute the protocol data units (PDUs) of the component sub–layer. A mapping of these primitives to component sub–layer PDUs is indicated in Table 1/Q.774.

3.2.1.1.2 *Management of component IDs*

Component IDs are assigned by the invoking end at operation invocation time. A TC–user need not wait for one operation to complete before invoking another. At any point in time, a TC–user may have any number of operations in progress at a remote end (although the latter may reject an invoke component for lack of resources).

Each component ID value is associated with an operation invocation and its corresponding component state machine. Management of this component ID state machine takes place only at the end which invokes the operation. The other end reflects this component ID in its relies to the operation invocation, and does not manage a state machine for this connection ID. Note that both ends may invoke operations in a full–duplex manner: each end manages state machines for the operations it has invoked, and is free to allocate component IDs independently of the other.

A component ID value may be reallocated when the corresponding state machine returns to idle. However, immediate reallocation could result in difficulties when certain abnormal situations arise. A released ID value (when the state machine returns of idle) should therefore not be real–located immediately; the way this is done is implementation–dependent, and thus is not described in this Recommendation.

Component states and state transitions are described in § 3.2.1.1.3.

TABLE 1/Q.774

Mapping of TC component handling service primitives to components

Service Primitive	Abbreviation	Component Type

TC–INVOKE

INV

INVOKE (Note 1)

TC–RESULT

RR–L

Return Result (Last) (Note 1)

TC–U–ERROR

RE

Return Error (Note 1)

TC–U–REJECT

RJ

Reject (Note 1)

TC–R–REJECT

RJ

Reject (Note 1)

TC–L–REJECT

(Note 2)

TC–RESULT–NL

RR–NL

Return Result (Not Last)

TC–L–CANCEL

(Note 3)

TC–U–CANCEL

(Note 3)

local.

3.2.1.1.3 *Operation classes*

TABLE 2/Q.774

Operation Classes

Operation Class	Description
1	Reporting success or failure
2	Reporting failure only
3	Reporting success only
4	Outcome not reported

A different type of state machine is defined for each class of operation, the state transitions of which are represented by Figures 1/Q.774 to 4/Q.774. These state machines are described here from a protocol point of view (sent/received components), whereas they are described in Recommendation Q.771 from a service (primitives) point of view.

The states of each component state machine are defined as follows:

– Idle:The component ID value is not assigned to any pending operation.

– Operation Sent: The component ID value is assigned to an operation which has not been completed or rejected.

– Wait for Reject: When a component indicating the completion of an operation is received, the receiving TC–user may reject this result. The Wait for Reject State is introduced so that the component ID is retained for some time, thereby making the rejection possible.

State transitions are triggered by:

– a primitive received from the TC–user, causing a component to be built, and eventually sent;

– receipt of a component from the peer entity;

– a number of situations indicated on Figures 1/Q.774 to 4/Q.774, corresponding to the following situations:

Cancel: A timer is associated with an operation invocation. This invocation timer is started when the invoke component is passed to the transaction sub–layer. The TC–INVOKE request primitive indicates a timer value. A cancel situation occurs when the invoking TC–user decides to cancel the operation (TC–U–CANCEL request primitive) before either the final result (if any) is received, or a timeout situation occurs. On receipt of a TC–U–CANCEL request, the component sub–layer stops the timer; any further replies will not be delivered to the TC–user, and TCAP will react according to abnormal situations as described in § 3.2.2.2.

End situation: When an End or Abort message is received, or when prearranged end is used, TCAP returns any pending operations to Idle.

Invocation timeout: A timeout situation occurs when the timer associated with an operation invocation expires: the state machine returns to idle, with notification to the TC–user by means of a TC–L–CANCEL indication (in the case of a class 1, 2 or 3 operation). This notification indicates an abnormal situation for a class 1 operation, or gives the definite outcome of a class 2 or 3 operation for which no result has been received (normal situation).

Reject timeout: A Reject timeout situation occurs when the timer associated with the Wait for Reject state expires. If this occurs, the component sub–layer assumes that the TC–user has accepted the component.

In the diagrams that follow, components contain either single ID values, or ordered pairs of IDs (i, y), where i is the invoke ID and y is the linked ID. The state diagrams are modeled for a single operation invocation with ID i. The value of y is not relevant to the ID i. A linked invoke operation can only be accepted if the linked to state machine is in the Operation Sent state.

Components can be received “well–formed” or “malformed”. The diagrams show where this is significant. If it does matter whether the component is received “well–formed” or “malformed” then the diagram indicates “receive” only.

Class 1 operations report failure or success. A rejection in the case of a protocol error may also occur. Upon invoking a class 1 operation, the invoking end will keep the ID i active until a “last” reply is received and can no longer be rejected. An ID may be released locally, at the option of the TC–user. This is indicated in Figure 1/Q.774.

Fig. 1/Q.774 /T1113720-88 = 15 cm

Class 2 operations report failure only. A rejection in the case of a protocol error may also occur. Upon invoking a class 2 operation, the invoking end will keep the ID i active until a reply has been received and can no longer be rejected or until a timeout[1]) cancel or end situation occurs. This is indicated in Figure 2/Q.774.

1) A timeout for a class 2 operation is a “normal” situation.

Fig. 2/Q.774 /T1113731-88 = 15 cm

Class 3 operations report success only. A rejection in the case of a protocol error may also occur. Upon invoking a class 3 operation, the invoking end will keep the ID i active until a reply has been received and can no longer be rejected or until a timeout[2]) cancel or end situation occurs. This is indicated in Figure 3/Q.774.

2)
A timeout for a class 3 operation is a “normal” situation.

Fig. 3/Q.774 /T1113730-88 = 15 cm

Class 4 operations do not report their outcome. A rejection in the case of a protocol error may also occur. Upon invoking a class 4 operation, the invoking end will keep the ID i active until a reject has been received or until a timeout[3]) cancel or end situation occurs. This is indicated in Figure 4/Q.774.

3) A timeout for a class 4 operation is a “normal” situation.

Fig. 4/Q.774 /T1113751-88 = 15 cm

3.2.1.2 *Sample component flows*

Some sample component flows that are compatible with Recommendation X.229 (Remote operations) are indicated in Figure 5/Q.774. The flows show cases of valid component sequences correlated to an invoked operation.

Fig. 5/Q.774 /T1113760-88 = 12 cm

Figure 6/Q.774 depicts that, as an extension to Recommendations X.219 and X.229, TCAP permits multiple return results to respond to the same Invoke operation for the purpose of segmenting a result over a connectionless network service.
Fig. 6/Q.774 /T1113770-88 = 6 cm

3.2.1.3 *Dialogue control via TC primitives*

The TC–UNI, TC–BEGIN, TC–CONTINUE and TC–END request primitives are used by a TC–user to control the transfer of components. Components in a message are delivered to the remote TC–user in the same order in which they are received by the originating component sub–layer from the local TC–user. The corresponding indication primitives are employed by the component sub–layer to inform the TC–user at the receiving end of the state of the dialogue.

A TC–user employs a dialogue control request primitive to trigger transmission of all previously passed components with the same dialogue identifier. A component sub–layer dialogue control primitive in turn triggers a corresponding service request to the transaction sub–layer, the sub–layer where the transaction control service is provided. A mapping of TC to TR transaction control primitives is provided in Table 3/Q.774.

TABLE 3/Q.774

Mapping of TC Dialogue Handling Service Primitives to TR Primitives

TC Primitive	TR Primitive
TC–UNI	TR–UNI
TC–BEGIN	

TR–BEGIN

TC–CONTINUE

TR–CONTINUE

TC–END

TR–END

TC–U–ABORT

TR–U–ABORT

TC–P–ABORT

TR–P–ABORT

Dialogue begin

A TC–BEGIN request primitive results in a TR–BEGIN request primitive, which begins a transaction, and transmits any (0 or more) components passed on the interface with the same dialogue ID.

At the destination end, a TR–BEGIN indication primitive is received by the component sub–layer. It causes a TC–BEGIN indication primitive starting a dialogue to be delivered to the TC–user, followed by component handling primitives associated with each of the components received (if any).

Dialogue continuation

A TC–CONTINUE request primitive results in a TR–CONTINUE request primitive which transmits any components passed on the interface with the same dialogue ID. If reject components (see § 3.2.2.2) have been built by the component sub–layer for this dialogue, they are also transmitted.

At the destination end, a TR–CONTINUE indication received by the component sub–layer causes a TC–CONTINUE to be delivered to the TC–user, followed by component handling primitives associated with each of the components received.

Dialogue end

In the case of basic end of a dialogue, any components passed on the interface plus any reject components built by the component sub–layer for this dialogue are passed for transmission to the transaction sub–layer in a TR–END request primitive, then the dialogue is ended.

At the destination end, a dialogue ends when each component (if any) accompanying the TR–END indication primitive have been delivered to the TC–user by an appropriate component handling primitive following the TC–END indication.

The component sub–layer does not check, when a TC–user requests the end of a dialogue, that all the component state machines associated with this dialogue have returned to Idle. Similarly, no check is made by the component sub–layer that all the state machines associated with a dialogue have returned to Idle when it has delivered the components accompanying a TR–END indication primitive. In an end situation, any non–idle–state machine is returned to Idle when the TR–END request primitive is passed to the transaction sub–layer (at the originating side), or when all accompanying components have been delivered to the TC–user at the destination side; any components pending transmission are discarded.

Prearranged end and TC–user abort of a dialogue do not trigger transmission of pending components. All state machines associated with the dialogue are returned to idle, and the components are discarded.

3.2.2 *Abnormal procedures*

3.2.2.1 *Dialogue control*

Any abnormal situation detected by the component sub–layer results in the rejection of a component, and in notification to the local TC–user. The component sub–layer never decides to abort a dialogue. Abort of a dialogue is always the reflection of a decision by:

- the transaction sub–layer to abort the underlying transaction. The component sub–layer idles the operation state machines of the dialogue, discards any pending component, and passes an abort indication to the TC–users (TC–P–ABORT indication primitive);
- the TC–user to abort the dialogue. At the originating side, a TC–U–ABORT request is received from the TC–user: active component state machines for this dialogue are idled, and a TR–U–ABORT request is passed to the transaction sub–layer. At the destination side, a corresponding TR–U–ABORT indication is received from the transaction sub–layer, any active component state machines for the dialogue are idled, and a TC–U–ABORT indication is passed to the TC–user;

In both cases, accompanying information (P–Abort cause, or user–provided information) passes transparently through the component sub–layer.

Handling of the notification of abnormal situations which cannot be related to a particular dialogue is for further study.

3.2.2.2 *Abnormal procedures relating to operations*

The following abnormal situations are considered:

- no reaction to class 1 operation invocation (see § 3.2.1.1.3);
- receipt of a malformed component: the component type and/or the Invoke ID cannot be recognized (i.e. the state machine cannot be identified);
- receipt of a well–formed component in violation of authorized state transitions.

The actions taken by the component sub–layer to report component portion errors are shown in Table 4/Q.774. The following considerations have guided the choices indicated in this Table:

- When a protocol error has been detected by the local TC–user, this TC–user is not subsequently advised via the TC–Reject (as indicated in Table 4/Q.774) since it is already aware of the protocol error.
- In other cases (reject by component sub–layer), the local TC–user is always advised so that it can issue a dialogue control primitive (see the reject mechanism described below).
- When a component is rejected, the associated state machine returns to Idle.
- The reject mechanism applies whenever possible: even if the Invoke ID is not assigned or not recognized (i.e. no state machine can be identified), the reject mechanism should be initiated. The only case where rejection is purely local is when the component to be rejected is itself a reject component.

Protocol errors in the component portion of a TCAP message are reported using the Reject component. The Reject component is sent in response to an incorrect component other than Reject.

When an invoke ID is available in a component to be Rejected, this ID is reflected in the Reject component.

TABLE 4/Q.774

Action Taken on Protocol Errors in Component Portion

Component Type received	Type of error	Local action	Component State Machine	Local user advised	Component state machine	Remote user advised
Syntax error	Init. Reject	Inv: NA Link: No change	Yes a)			

Return
to Idle

Yes

INVOKE

Linked ID unassigned

Init. Reject

Inv: NA
Link: NA

Yes a)

Inv:
Return to
Idle

Yes

RETURN–RESULT (L/NL)
or

Syntax error

Init. Reject

Return
to Idle

Yes a)

NA

Yes

RETURN–ERROR

Invoke ID unassigned

Init. Reject

NA

Yes a)

NA

Yes

RETURN–RESULT (L/NL)

Operation
Class 2/4

Init. Reject

Return
to Idle

Yes a)

NA

Yes

RETURN–ERROR

Operation
Class 3/4

Init. Reject

Return to Idle

Yes a)

NA

Yes

REJECT

Syntax Error

Local Reject

Return to NA b)

Yes

NA

No

UNKNOWN

Invoke ID derivable

Init. Reject

No Change
(NA)

Yes a)

Return to
Idle

Yes

Invoke ID
non derivable

Init. Reject

(NA)

Yes a)

NA

Yes

NA: Not applicable.
a) This is to alert the TC User so it can issue a dialogue control primitive to send the Reject component formulated by the
Component Sub–Layer.
b) If Invoke ID present, and Invoke Problem, return Component State machine to idle.

Component type abbreviations are identified in Table 1/Q.774.

In the case of multiple components within a message, when a malformed component is detected by the component sub–layer, subsequent components in the message are discarded.

Rejection of any portion of a segmented result shall be equivalent to rejecting the entire result.

The associated state machine is returned to idle. Subsequent portions of the same segmented result shall also be rejected on the basis of no active state machine.

The reject mechanism: when the component sub–layer detects a situation where (non–local) reject should be initiated (as per Table 4/Q.774), it builds a reject component, stores it, and informs the local TC–user by means of TC–L–REJECT indication primitive. The TC–user may decide:

a) to continue the dialogue, or

b) to end the dialogue using the basic scenario, or

c) to abort the dialogue.

In cases a) and b), the first dialogue handling primitive (TC–CONTINUE request or TC–END request respectively) issued by the TC–user triggers transmission of the stored reject component(s) built for this dialogue by the component sub–layer. The remote component sub–layer receives the reject component(s) built for this dialogue, idles the corresponding component state machine(s) if possible (as per Table 4/Q.774) and informs the TC–user of the (remote) rejection via TC–R–REJECT information primitive(s).

If the component sub–layer generated reject combined with accumulated components from the TC–user exceeds the message length limitations, then the TC–user, being aware of the reject component, must initiate two dialogue handling primitives. The component sub–layer, also being aware of the length problem, will send all the components, except the reject, with the first primitive. The reject will be sent with the next dialogue handling primitive together with any further components provided by the TC–user.

3.3 *Transaction sub–layer procedures*

3.3.1 *General*

The transaction sub–layer provides for an association between its users (TR–users). This association is called a transaction.

The transaction sub–layer procedure associates each TCAP message and, therefore, all the contained components with a particular transaction.

The transaction sub–layer processes the transaction portion (message type and transaction ID) of a TCAP message. Transaction IDs identify a transaction. Each end assigns a local transaction identification; local transaction IDs are exchanged in the transaction portion of messages as indicated in Q.773.

The component portion of a TCAP message is passed between the component sub–layer and the transaction sub–layer as user data in the transaction sub–layer primitives.

3.3.2 *Mapping of TR service primitives to message types*

Recommendation Q.771 describes the services performed by the transaction sub–layer by defining the service interface between the TR user and the transaction sub–layer and the transaction sub–layer and the SCCP. Similarly, state transition diagrams appear in Recommendation Q.771 based on service primitives. In this section, a message based description of the protocol is provided. A mapping of TR–primitives to transaction sub–layer protocol data units is indicated in Table 5/Q.774.

TABLE 5/Q.774

Mapping of TR Service Primitives to Messages

Service Primitive	Message Type
TR–UNI	Unidirectional
TR–P–ABORT	Abort

TR–BEGIN

Begin

TR–CONTINUE

Continue

TR–U–ABORT

Abort

TR–END

End

3.3.3 *Normal procedures*

3.3.3.1 *Message transfer without establishing a transaction*

3.3.3.1.1 *Actions of the sending end*

The TR–UNI request primitive is used when a TR–user sends a message to another TR–user but does not need to enter into a transaction. A unidirectional message, which does not have a transaction ID, is used in this case.

3.3.3.1.2 *Actions of the receiving end*

The receipt of a unidirectional message causes a TR–UNI indication primitive to be passed to the TR–user. No further action is taken by the transaction sub–layer.

3.3.3.2 *Message transfer within a transaction*

3.3.3.2.1 *Transaction begin*

In the following discussion, the sending node of the first TCAP message is labelled node "A", and the receiving node is labelled node "B".

3.3.3.2.1.1 *Actions of the initiating end*

The TR–user at node "A" initiates a transaction by using a TR–BEGIN request primitive, which causes a begin message to be sent from node "A" to node "B".

The begin message contains an originating transaction ID. This transaction ID value, when included in any future message from node "A" as the originating transaction ID or in a message to node "A" as the destination transaction ID, identifies the transaction to node "A".

Once the transaction sub–layer at node "A" has sent a begin message it cannot send another message to the transaction sub–layer at node "B" for the same transaction until it receives a continue message from node "B" for this transaction.

3.3.3.2.1.2 *Actions of the receiving end*

The receipt of a Begin message causes a TR–BEGIN indication primitive to be passed to the TR–user at node "B". In response to a TR–BEGIN indication primitive, the TR–user at node "B" decides whether or not to establish a transaction. If the TR–user does want to establish a transaction, it passes a TR–CONTINUE request primitive to the transaction sub–layer; otherwise, it terminates the transaction (see § 3.3.3.2.3). These conditions are defined by the TR–user.

The Begin message contains only an originating transaction ID. If, after receiving a Begin message with a given originating transaction ID, the transaction sub–layer receives another Begin message with the same originating transaction ID, the transaction sub–layer does not consider this as an abnormal situation: a second transaction is initiated at node "B".

3.3.3.2.2 *Transaction continuation*

A Continue message is sent from one node to another when a TR–CONTINUE request primitive is passed from the TR–user to the transaction sub–layer at the sending node.

A Continue message includes the destination transaction ID which is identical to the originating transaction ID received in messages from the peer node. Each node assigns its own originating transaction ID at transaction initiation time. The transaction IDs remain constant for the life of the transaction.

A Continue message includes both an originating transaction ID and a destination transaction ID. The originating transaction ID, in successive continue messages is not examined.

Receipt of a Continue message causes a TR–CONTINUE indication primitive to be passed to the destination TR–user.

Once the user at node "B" has responded with a TR–CONTINUE request primitive to establish a transaction, all subsequent interactions at either end between the TR–user and the transaction sub–layer are via TR–CONTINUE primitives until the transaction is to be terminated. In message terms, once a Continue message is sent from node "B", all subsequent messages shall be Continue messages until the transaction is to be terminated.

3.3.3.2.3 *Transaction termination*

The basic method: A TR–user at either end may terminate a transaction by passing a TR–END request primitive (indicating basic end) to the transaction sub–layer. An end message is sent to the peer entity which, in turn, passes a TR–END indication promitive to its TR–user. The end message contains a destination transaction ID.

The pre–arranged method: This method implies that the peer entities know *a priori* – at a given point in the application script – that the transaction will be released. In this case, the TR–user passes a TR–END request primitive (indicating pre–arranged end) to its transaction sub–layer, and no End message is sent.

3.3.3.2.4 *Abort by the TR–user*

When a TR–user wants to abort a transaction, it passes a TR–U–ABORT request primitive to the transaction sub–layer, which sends an abort message with user–provided (cause and diagnostic) information.

At the receiving side, the transaction sub–layer receiving an Abort message containing user–provided information passes this information without analyzing it to the TR–user in a TR–U–ABORT indication primitive.

3.3.3.2.5 *Example of message exchange*

Figure 7/Q.774 depicts an example of exchanges of TCAP messages between two TR–users.

Fig. 7/Q.774 /T1106500-87 = 8 cm

3.3.3.2.6 *Transaction state transition diagrams*

A state machine is associated with a transaction at each end of this transaction. Four transaction states are introduced:

– Idle: no state machine exists;

– Init Sent (IS): a Begin message has been sent; an indication from the peer entity whether the transaction has been established or not is awaited;

– Init Received (IR): a Begin message has been received; a request from the TR–user either to continue the transaction, or to terminate it, is awaited;

– Active: the transaction is established: continue messages can be exchanged in both directions simultaneously.

Figure 8/Q.774 shows the transaction state transition diagram.

3.3.4 *Abnormal procedures relating to transaction control*

The following abnormal situations are covered by the transaction sub–layer:

1) no reaction to transaction initiation;
2) receipt of an indication of abnormal situation from the underlying layer;
3) receipt of a message with an unassigned or non–derivable destination transaction ID (non–derivable means that the information is not found or not recognized): the message cannot be associated with a transaction;
4) receipt of a message with a recognized destination transaction ID: the message can be associated with a transaction, but the message type is not compatible with the transaction state.

Fig. 8/Q.774 /T1113780-88 = 12 cm

Case 1 is covered by a local, implementation–dependent, mechanism which results in aborting the transaction locally, as described below.

Case 2 is for further study.

When a transaction portion error is found (cases 3 and 4 above), the transaction sub–layer should take the following actions.

The status of the originating transaction ID should be checked. Actions are the following:

1) If the originating transaction ID is not derivable, the local end (which received the message) discards the message and does not take any other action; e.g. it does not send an abort message or terminate the transaction; or,

2) If the originating transaction ID is derivable, the following actions are taken:

i)
P–Abort cause and transmit it to the originating end. The originating end will then take the appropriate action to terminate the transaction if the originating transaction ID is assigned.

ii)
the transaction sub–layer takes no action to terminate the transaction at its end.

iii)

a)
idle;

b)
the transaction via the transaction sub–layer abort; and

c)

release all component IDs associated with this transaction,

discard any pending components for that transaction,

inform the TC–user of the transaction abort.

Finally, regardless of the disposition of the transaction IDs, the entire erroneous TCAP message should be discarded.

TABLE 6/Q.774

Actions when an Abnormal Transaction Portion is Received

Local End (detects protocol error)

Remote End

	Message Type Received	Origin. Tr. Id.	Destin. Tr. Id.	Action	Transaction State Mach.	Local User Advised	Transaction State Mach.	User Advised
UNIDIREC-TIONAL								
	–	–	Discard	– c)	No	– c)	No	
	not der.	–	Discard	NA	No	NA	No	
BEGIN								
	der.	–	Abort	NA	No	Ret to Idle a)	Yes a)	

not der.

–

Discard

NA

No

NA

No

CONTINUE

der.

not der unass.

Abort

NA

No

Ret to Idle a)

Yes a)

der.

ass.

Abort

Ret to Idle

Yes

Ret to Idle a)

Yes a)

END/ABORT

–

not der unass.

Discard

NA

No

NA

No

–

ass.

Discard

Ret to Idle

Yes

NA

No

not der

–

Discard

NA

No

NA

No

UNKNOWN

der.

not der unass.

Abort

NA

 Fascicle VI.9 – Q.774

No
Ret to Idle b)
Yes a)

der.
ass.
Abort
Ret to Idle
Yes
Ret to Idle a)
Yes a)

NA: Transition to the Idle state is Not Applicable b)
not der.: not derivable.
der.: derivable.
ass.: derivable and assigned.
unass.: derivable but unassigned.
a) If the Transaction ID is assigned at this end, otherwise the state transition is not applicable, and the user is not informed.
b) The expression NA is used in those cases where the normal procedure of Return to Idle at both ends following the
appearance of an abnormal situation is Not Applicable because it is impossible to identify the Transaction ID(s) and
therefore to relate the damaged message to a specific transaction at either ends (Local and/or Remote end).
c) The Unidirectional message does not refer to an explicit transaction and therefore it does not affect the Transaction State
Machine.

When receiving an Abort message, the destination transaction sub–layer does the following:

- if the Abort message contains user–abort information (or no information), inform the TR–user by means of the TR–U–ABORT indication primitive;
- if the Abort message contains a P–Abort cause information, inform the TR–user by means of the TR–P–ABORT indication primitive. Notification to the management is for further study;
- in both cases, discard any pending messages for that transaction and return the transaction state machine to Idle.

4 Transaction capabilities based on a connection oriented network service

For further study.

ANNEX A

(to Recommendation Q.774)

Transaction capabilities SDLs

A.1 *General*

This Annex contains the description of the transaction capability procedures described in Recommendation Q.774 by means of SDLs according to the CCITT specification and description language. In order to facilitate the functional description as well as the understanding of the behaviour of the signalling system, the transaction capabilities application part (TCAP) is divided into the component sub–layer and the transaction sub–layer (Figure A–1/Q.774). The component sub–layer again is divided into a component handling block (CHA) and a dialogue handling block (DHA) (Figure A–2/Q.774).

The SDL is provided according to this functional partitioning which is used only to facilitate understanding and is not intended to be adopted in a practical implementation of the TCAP. The functional blocks and their associated service primitives are shown in Figure A–2/Q.774.

A.2 *Abbreviations used in the SDL diagrams*

CSL

Component sub–layer

L

Last component

NL
SCCP
TC
TCAP
TCU
TSL

Transaction sub–layer

ISP
IS Initiation sent state
IR Initiation received state
DHA
CHA
RJ
RE
RR
INV

Invoke

ISM

Invocation state machine

CCO

UNI
Unidirectional

A.3 *Drafting conventions*

To indicate the direction of each interaction the symbols are used as shown below:

Fig, /T1120540-88. = 8 cm

Fig. A–1/Q.774 /T1120550-88 = 13 cm

Fig. A–2a/Q.774 /T1120560-88 = 18 cm

Fig. A–2b/Q.774 /T1120570-88 = 13 .5cm

Fig. A–3/Q.774 /T1120580-88 = 19 cm

Fig. A–3/Q.774 (sheet 2 of 6) /T1120590-88 = 16.5 cm

Fig. A–3/Q.774 (sheet 3 of 6) /T1120600-88 = 20 cm

Fig. A–3/Q.774 (sheet 4 of 6) /T1120610-88 = 17 cm

Fig. A–3/Q.774 (sheet 5 of 6) /T1120620-88 = 13 cm

Fig. A–3/Q.774 (sheet 6 of 6) /T1120630-88 = 23 cm

Fig. A–4/Q.774 (sheet 1 of 2) /T1120640-88 = 2.5 cm

Fig. A–4/Q.774 (sheet 2 of 2) /T1120650-88 = 17 cm

Fig. A–5/Q.774 (sheet 1 of 4) /T1120660-88 = 25 cm

Fig. A–5/Q.774 (sheet 2 of 4) /T1120670-88 = 25.5 cm

Fig. A–5/Q.774 (sheet 3 of 4) /T1120680-88 = 21.5 cm

Fig. A–5/Q.774 (sheet 4 of 4) /T1120690-88 =19 cm

Fig. A–6/Q.774 (sheet 1 of 6) /T1120670-88 = 13 cm

Fig. A–6/Q.774 (sheet 2 of 6) /T1120710-88 = 18 cm

Fig. A–6/Q.774 (sheet 3 of 6) /T1120720-88 = 11.5 cm

Fig. A–6/Q.774 (sheet 4 of 6) /T1120730-88 = 17.5 cm

Fig. A–6/Q.774 (sheet 5 of 6) /T1120740-88 = 16.5 cm

Fig. A–6/Q.774 (sheet 6 of 6) /T1120750-88 = 15 cm