5i’

Recommendation Q.773

TRANSACTION CAPABILITIES FORMATS AND ENCODING

1 Introduction

This Recommendation provides the format and encoding of Transaction Capabilities Application Part (TCAP) messages. Formats and Encoding for the Intermediate Service Part (ISP) are for further study. This Recommendation is based on the encoding rules provided in CCITT Recommendation X.209 and is consistent with that Recommendation.

2 Description conventions

This Recommendation does not use Recommendation X.209 formal description language. This Recommendation uses the description method of other Q.700 series Recommendations. Annex A uses the formal decription language to supplement this Recommendation.

3 Standard representation

3.1 *General message structure*

Each information element within TCAP message has the same structure. An information element consists of three fields, which always appear in the following order. The Tag distinguishes one type from another and governs the interpretation of the Contents. The Length specifies the length of the Contents. The Contents is the substance of the element, containing the primary information the element is intended to convey. Figure 1/Q.733 shows an overview of a TCAP message and an information element.

Figure 1/Q.773

Each field is coded using one or more octets. Octets are labelled as shown in Figure 2/Q.773. The first octet is the first transmitted. Bits in an octet are labelled as shown in Figure 3/Q.773, with bit A the least significant and the first transmitted.

Figure 2/Q.773, p.

Figure 3/Q.773, p.

The contents of each element is either one value (Primitive) or one or more information elements (Constructor), as shown in Figure 4/Q.773.

Figure 4/Q.773, p.

3.2 *Tag*

An information element is first interpreted according to its position within the syntax of the message. The Tag distinguishes one information element from another and governs the interpretation of the Contents. It is one or more octets in length. The Tag is composed of ‘‘Class’’, ‘‘Form’’ and ‘‘Tag code’’, as shown in Figure 5/Q.773.

Figure 5/Q.773, p.

3.2.1 *Tag class*

All Tags use the two most significant bits (H and G) to indicate the Tag Class. These bits are coded as shown in Table 1/Q.773.

H.T. [T1.773]

TABLE 1/Q.773

Coding of tag class

Class	Coding (HG)
Universal	**00**
Application-wide	**01**
Context-specific	**10**
Private use	**11**

Table 1/Q.773 [T1.773], p.

The universal class is used for Tags that are exclusively standardized in CCITT Recommendation X.209 and are application independent types. Universal Tags may be used anywhere a universal information element type is used. The universal class applies across all CCITT Recommendations, i.e. across CCITT No. 7 ASEs, X.400 MHS, etc.

The Application-wide class is used for information elements that are standardized across all applications (ASEs) using CCITT No. 7 TC, i.e. TC-Users.

The Context-specific class is used for information elements that are specified within the context of the next higher construction and take into account the sequence of other data elements within the same construction. This class may be used for tags in a construction, and the tags may be re-used in any other construction.

The Private Use class is reserved for information elements specific to a nation, a network or a private user. Such information elements are beyond the scope of the TC Recommendations.

The Tag codes of the Application-wide class not assigned in this Recommendation are reserved for future use.

3.2.2 *Form of the element*

Bit F is used to indicate whether the element is ‘‘Primitive’’ or ‘‘Constructor’’, as is shown in Table 2/Q.773. A primitive element is one whose structure is atomic (i.e. one value only). A constructor element is one whose content is one or more information elements which may themselves be constructor elements.

Both forms of elements are shown in Figure 4/Q.773.

H.T. [T2.773]

TABLE 2/Q.773

Coding element form

Element form	Coding (F)
Primitive	0
Constructor	1

Table [T2.773], p.

3.2.3 *Tag code*

Bits A to E of the first octet of the Tag plus any extension octets represent a Tag code that distinguishes one element type from another of the same class. Tag codes in the range 00000 to 11110 (0 to 30 decimal) are provided in one octet.

The extension mechanism is to code bits A to E of the first octet as 11111. Bit H of the following octet serves as an extension indication. If bit H of the extension octet is set to 0, then no further octets for this tag are used. If bit H is set to 1, the following octet is also used for extension of the Tag code. The resultant Tag consists of bits A to G of each extension octet, with bit G of the first extension octet being most significant and bit A of the last extension octet being least significant. Tag code 31 is encoded as 0011111 in bits G to A of a single extension octet. Higher tag codes continue from this point using the minimum possible number of extension octets.

Figure 6/Q.773 shows the detailed format of the Tag code.

Figure 6/Q.773, p.

3.3 *Length of the Contents*

The Length of the Contents is coded to indicate the number of octets in the Contents. The length does not include the Tag nor the Length of the Contents octets.

The Length of the Contents uses the short, long or indefinite form. If the length is less than 128 octets, the short form is used. In the short form, bit H is coded 0, and the length is encoded as a binary number using bits A to G.

If the Length of the contents is greater than 127 octets, then the long form of the Length of the Contents is used. The long form Length is from 2 to 127 octets long. Bit H of the first octet is coded 1, and bits A to G of the first octet encode a number one less than the size of the Length in octets as an unsigned binary number whose MSB and LSB are bits G and A, respectively. The length itself is encoded as an unsigned binary number whose MSB and LSB are bit H of the second octet and bit A of the last octet, respectively. This binary number should be encoded in the fewest possible octets, with no leading octets having the value 0.

The indefinite form is one octet long and may (but need not) be used in place of the short or long form, whenever the element is a constructor. It has the value 10000000. When this form is employed, a special end-of-contents (EOC) indicator terminates the Contents.

There is no notation for the end-of-contents indicator. Although considered part of the Contents syntactically, the end-of-contents indicator has no semantic significance.

The representation for the end-of-contents indicator is an element whose class is universal, whose form is primitive, whose ID Code has the value 0, and whose Contents is unused and absent:

EOC	Length	Contents
00(hex)	00(hex)	Absent

Figure 7/Q.773 shows the formats of the Length field described above. The maximum value that may be encoded is constrained by the network message size limitations in the connectionless case. Limitations in the connection-oriented case are for further study.

Figure 7/Q.773 [T3.773], p. (traiter comme tableau MEP)

3.4 *Contents*

The contents is the substance of the element and contains the information the element is intended to convey. Its length is variable, but always an integral number of octets. The contents is interpreted in a type-dependent manner, i.e. according to the tag value.

4 TCAP message structure

A TCAP message is structured as a single constructor information element. It consists of a Transaction Portion which contains information elements used by the Transaction sub-layer, and a Component Portion which contains information elements used by the Component sub-layer. One of the Transaction Portion elements is called the Component Portion, and it contains the Component sub-layer information elements. Each Component is a constructor information element.

Figure 8/Q.773 shows the detailed TCAP message structure described above.

Figure 8/Q.773 [T4.773], p. (traiter comme tableau MEP)

5 Transaction Portion

Transaction Portion information elements use the Application Wide class as defined in § 3.2.1.

5.1 *Structure of the Transaction Portion*

The Transaction Portion fields for various message types are shown in Tables 3/Q.773 to 8/Q.773.

H.T. [T5.773]

TABLE 3/Q.773

Transaction Portion fields

Unidirectional message type

Element Form	Fields of Transaction Portion	Mandatory Indication
Constructor Message Type tag Total message length | ua) }	{ Mandatory	
Constructor Component Portion tag Component Portion length }	{ Mandatory	
Constructor One or more Components (Not a part of Transaction Portion) (Described in § 6) }	{ Mandatory	

a) See Note a) to Figure 8/Q.773.

Table 3/Q.773 [T5.773], p.

H.T. [T6.773]

TABLE 4/Q.773

Transaction portion fields

Begin message type

Element Form	**Fields of Transaction Portion**	**Mandatory Indication**
Constructor **Message type tag** **Total message length | ua)** **}**	**{** **Mandatory**	
Primitive **Originating Transaction ID tag** **Transaction ID length** **Transaction ID** **}**	**{** **Mandatory**	
Constructor **Component Portion tag** **Component Portion length** **}**	**{** **Mandatory | ub)**	
Constructor **One or more Components** **(Not a part of Transaction Portion) (Described in § 6)** **}**	**{** **Optional**	

a) See Note a) to Figure 8/Q.773.

b) The Component Portion tag is not required if there are no Components being sent in the message.

Table 4/Q.773 [T6.773], p.

H.T. [T7.773]

TABLE 5/Q.773

Transaction Portion fields

End message type

Element Form	Fields of Transaction Portion	Mandatory Indication
Constructor **Message type tag** **Total message length | ua)** **}**	**{** **Mandatory**	
Primitive **Destination Transaction ID tag** **Transaction ID length** **Transaction ID** **}**	**{** **Mandatory**	
Constructor **Component Portion tag** **Component Portion length** **}**	**{** **Mandatory | ub)**	
Constructor **One or more Components** **(Not a part of Transaction Portion) (Described in § 6)** **}**	**{** **Optional**	

a) See Note a) to Figure 8/Q.773.

b) See Note b) to Table 4/Q.773.

Tableau 5/Q.773, [T7.773], p.

H.T. [T8.773]

TABLE 6/Q.773

Transaction Portion fields

Continue message type

Element Form	Fields of Transaction Portion	Mandatory Indication
Constructor Message type tag Total message length | ua) }	{ Mandatory	
Primitive Originating Transaction ID tag Transaction ID length Transaction ID }	{ Mandatory	
Primitive Destination Transaction ID tag Transaction ID length Transaction ID }	{ Mandatory	
Constructor Component Portion tag Component Portion length }	{ Mandatory | ub)	
Constructor One or more Components (Not a part of Transaction Portion) (Described in § 6) }	{ Optional	

a) See Note a) to Figure 8/Q.773.

b) See Note b) to Table 4/Q.773.

Tableau 6/Q.773, [T8.773] p.

H.T. [T9.773]

TABLE 7/Q.773

Transaction Portion fields

Abort message type (P-Abort)

Element Form	Fields of Transaction Portion	Mandatory Indication
Constructor **Message type tag** **Total message length | ua)** **}**	**{** **Mandatory**	
Primitive **Destination Transaction ID tag** **Transaction ID length** **Transaction ID** **}**	**{** **Mandatory**	
Primitive **P-Abort Cause tag** **P-Abort Cause length** **P-Abort Cause** **}**	**{** **Mandatory | ub)**	

a) See Note a) to Figure 8/Q.773.

b) P-Abort Cause is only present when the Abort is generated by the Transaction sub-layer.

Tableau 7/Q.773, [T9.773] p.

H.T. [T10.773]

TABLE 8/Q.773

Transaction Portion fields

Abort message type (U-Abort)

Element Form	Fields of Transaction Portion	Mandatory Indication
Constructor Message type tag Total message length | ua) }	{ Mandatory	
Primitive Destination Transaction ID tag Transaction ID length Transaction ID }	{ Mandatory	
Constructor User Abort Information tag User Abort Information length User Abort Information }	{ Optional | ub)	

a) See Note a) to Figure 8/Q.773.

b) The User Abort Information is optional, and may only be present when the Abort is generated by the TC-User.

Tableau 8/Q.773, [T10.773] p.

5.2 *Message Type Tag*

This field consists of one octet and is mandatory for all TCAP messages. Message Type tags are coded as shown in Table 9/Q.773.

H.T. [T11.773]
TABLE 9/Q.773
Coding of message type tag

Message type	H	G	F	E	D	C	B	A
Unidirectional	0	1	1	0	0	0	0	1
Begin	0	1	1	0	0	0	1	0
(reserved)	0	1	1	0	0	0	1	1
End	0	1	1	0	0	1	0	0
Continue	0	1	1	0	0	1	0	1
(reserved)	0	1	1	0	0	1	1	0
Abort	0	1	1	0	0	1	1	1

Table 9/Q.773 [T11.773], p.

5.3 *Transaction ID tags*

Two types of Transaction IDs, i.e. Originating Transaction ID and Destination Transaction ID, may be used. Zero, one or two ID information elements are required depending upon the Message type used. Table 10/Q.773 depicts this relationship.

H.T. [T12.773]
TABLE 10/Q.773
Transaction ID(s) in each message type

Message type	Originating ID	Destination ID
Begin	Yes	No
End	No	Yes
Continue	Yes	Yes
Abort	No	Yes

Table 10/Q.773 [T12.773], p.

The Originating and Destinastion Transaction ID Tags are coded as shown in Table 11/Q.773.

H.T. [T13.773]

TABLE 11/Q.773

Coding of Transaction ID tags

	H	G	F	E	D	C	B	A

{
Originating Transaction ID Tag
}
	0	1	0	0	1	0	0	0

{
Destination Transaction ID Tag
}
	0	1	0	0	1	0	0	1

Table 11/Q.773 [T13.773], p.

The length of a Transaction ID is 1 to 4 octets.

5.4 *P-Abort Cause tag*

The P-Abort Cause tag is coded as shown in Table 12/Q.773.

H.T. [T14.773]

TABLE 12/Q.773

Coding of P-Abort Cause tag

	H	**G**	**F**	**E**	**D**	**C**	**B**	**A**
P-Abort Cause Tag	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**0**

Table 12/Q.773 [T14.773], p.

The P-Abort cause values are coded as shown in Table 13/Q.773.

H.T. [T15.773]

TABLE 13/Q.773

Coding of P-Abort Cause values

P-Abort Cause	H	G	F	E	D	C	B	A
Unrecognized Message Type	0	0	0	0	0	0	0	0
Unrecognized Transaction ID	0	0	0	0	0	0	0	1

{
Badly Formatted Transaction Portion
}
	0	0	0	0	0	0	1	0

Incorrect Transaction Portion	0	0	0	0	0	0	1	1
Resource Limitation	0	0	0	0	0	1	0	0

Table 13/Q.773 [T15.773], p.

5.5 *User Abort Information tag*

The User Abort Information element Tag is coded as shown in Table 14/Q.773.

H.T. [T16.773]

TABLE 14/Q.773

Coding of User Abor Information tag

	H	G	F	E	D	C	B	A
User Abort Information tag	0	1	1	0	1	0	1	1

Table 14/Q.773 [T16.773], p.

The TC-User may provide any information element desired as the contents of the User Abort Information element.

5.6 *Component Portion tag*

The Component Portion Tag is coded as shown in Table 15/Q.773.

H.T. [T17.773]

TABLE 15/Q.773

Coding of Component Portion tag

	H	G	F	E	D	C	B	A
Component Portion Tag	0	1	1	0	1	1	0	0

Table 15/Q.773 [T17.773], p.

6 Component Portion

The Component Portion, when present, consists of one or more Components. The Components are based on, and extended from, the Remote Operations Service Element (ROSE) Application Protocol Data Units (APDUs) of Recommendation X.229 as indicated in Section 3/Q.772.

6.1 *Component type tag*

Each Component is a sequence of information elements. The Component types, as defined for TCAP, have the structure indicated in the following tables.

The information elements for the various Components shown in Tables 16/Q.773 to 19/Q.773 are all mandatory except the Linked ID and the parameters. The parameter may be one of the following:

— A Sequence of parameters

— A Set of parameters

— A specific parameter with its own tag (i.e. not part of a sequence or set)

— Nothing at all (i.e. absent)

Section 6.4 and Table 24/Q.773 define the Sequence and Set tags.

H.T. [T18.773]
TABLE 16/Q.773
Invoke component

Invoke component	Mandatory Indication
{ **Component type tag** **Component length** **}**	**M**
{ **Invoke ID tag** **Invoke ID length** **Invoke ID** **}**	**M**
{ **Linked ID tag** **Linked ID length** **Linked ID** **}**	**O**
{ **Operation Code tag** **Operation Code length** **Operation Code** **}**	**M**
Parameters	**O**

Tableau 16/Q.773, [T18.773] p.

H.T. [T19.773]
TABLE 17/Q.773
Return Result (Last) and Return Result (Not Last) components

{
Return Result (Last) and Return Result (Not Last) components
}
	Mandatory Indication

{
Component type tag
Component length
}
	M

{
Invoke ID tag
Invoke ID length
Invoke ID
}
	M

Sequence tag Sequence length	O | ub)

{
Operation Code tag
Operation Code length
Operation Code
}
	O | ub)

Parameters	O | ub)

a) ROSE has only one APDU called Return Result. See § 3.1.2/Q.772.

b) Omitted when no information elements are included in the parameters.

Tableau 17/Q.773, [T19.773] p.

H.T. [T20.773]

TABLE 18/Q.773

Return Error Component

Return Error component	Mandatory Indication

{
Component type tag
Component length
}
	M

{
Invoke ID tag
Invoke ID length
Invoke ID
}
	M

{
Error Code tag
Error Code length
Error Code
}
	M

Parameters	O

Tableau 18/Q.773, [T20.773] p.

H.T. [T21.773]

TABLE 19/Q.773

Reject component

Reject component	Mandatory Indication

{
Component type tag
Component length
}
	M

{
Invoke ID tag | ua)
Invoke ID length
Invoke ID
}
	M

{
Problem Code tag
Problem Code length
Problem Code
}
	M

Parameters	O

a) If the Invoke ID is not available, Universal Null (Table 22/Q.773) with length = 0 should be used.

Tableau 19/Q.773, [T21.773] p.

The Component Type Tag is coded context-specific, constructor as indicated in Table 20/Q.773.

H.T. [T22.773]

TABLE 20/Q.773

Component type tag

Component type tag	H	G	F	E	D	C	B	A
Invoke	1	0	1	0	0	0	0	1
Return Result (Last)	1	0	1	0	0	0	1	0
Return Error	1	0	1	0	0	0	1	1
Reject	1	0	1	0	0	1	0	0
(reserved)	1	0	1	0	0	1	0	1
(reserved)	1	0	1	0	0	1	1	0
Return Result (Not Last)	1	0	1	0	0	1	1	1

Table 20/Q.773 [T22.773], p.

The format of a Return Result (Not Last) is identical to that of a Return Result (Last).

6.2 *Component ID tag*

The term Component ID refers to the Invoke ID or the Lined ID. The Component ID tag is coded as shown in Table 21/Q.773.

H.T. [T23.773]

TABLE 21/Q.773

Coding of Component ID Tag

	H	G	F	E	D	C	B	A
Invoke ID	0	0	0	0	0	0	1	0
Linked ID [a]	1	0	0	0	0	0	0	0

a) This tag differs from the Invoke ID, which is coded as a universal INTEGER, in order to distinguish it from the following tag (Operation Code) which is also coded as a universal INTEGER.

Table 21/Q.773 [T23.773], p.

The length of a Component ID is 1 octet.

An Invoke Component has one or two Component IDs: an Invoke ID, and if it is desired to associate the Invoke with a previous Invoke, then the Linked ID is provided in addition to the Invoke ID.

Return Result and Return Error Components have one Component ID, called an Invoke ID which is the reflection of the Invoke ID of the Invoke Component to which they are responding.

The Reject Component uses as its Invoke ID, the Invoke ID in the Component being rejected. If this ID is unavailable (e.g. due to mutilation of the message undetected by lower layers), then the Invoke ID tag is replaced with a universal NULL tag (which always has length = 0) as shown in Table 22/Q.773.

H.T. [T24.773]

TABLE 22/Q.773

Coding of NULL tag

	H	G	F	E	D	C	B	A
NULL tag	0	0	0	0	0	1	0	1

Table 22/Q.773 [T24.773], p.

If an Invoke containing both Invoke and Linked IDs is being rejected, only the Invoke ID is used in the Reject Component.

6.3 *Operation Code tag*

Each operation is assigned a value to identify it. Operations can be classified as local or global operations. A local operation code follows an Operation Code Tag and Operation Code length. The Operation Code Tag is coded as shown in Table 23/Q.773.

H.T. [T25.773]

TABLE 23/Q.773

Coding of Operation Code tag

	H	G	F	E	D	C	B	A
Local Operation Code tag	0	0	0	0	0	0	1	0
Global Operation Code tag	0	0	0	0	0	1	1	0

Table 23/Q.773 [T25.773], p.

The Global Operation Code is coded as described in Recommendation X.209.

6.4 *Sequence and Set tags*

When there is more than one parameter in a Component (applicable to all Component types), they follow the Sequence or Set Tag, which are coded universal, constructor, as shown in Table 24/Q.773. The choice of Sequence or Set is at the discretion of the Application Service Element using TCAP.

H.T. [T26.773]

TABLE 24/Q.773

Coding of Sequence and Set tags

	H	G	F	E	D	C	B	A
Sequence Tag	0	0	1	1	0	0	0	0
Set Tag	0	0	1	1	0	0	0	1

Table 24/Q.773 [T26.773], p.

6.5 *Error Code tag*

Each error is assigned a value to identify it. Errors can be classified as local or global errors. A local error code follows the Error Code Tag and Error Code Length. The Error Code Tag is coded as shown in Table 25/Q.773.

H.T. [T27.773]

TABLE 25/Q.773

Coding of Error Code tag

	H	G	F	E	D	C	B	A
Local Error Code Tag	0	0	0	0	0	0	1	0
Global Error Code Tag	0	0	0	0	0	1	1	0

Table 25/Q.773 [T27.773], p.

The Global Error Code is coded as described in Recommendation X.209.

6.6 *Problem Code*

The Problem Code consists of one of the four elements General Problem, Invoke Problem, Return Result Problem or Return Error Problem. The tags for these elements are coded as shown in Table 26/Q.773. Their values are shown in Tables 27/Q.773 to 30/Q.773.

H.T. [T28.773]

TABLE 26/Q.773

Coding of Problem Type tags

Problem Type	H	G	F	E	D	C	B	A
General Problem	1	0	0	0	0	0	0	0
Invoke	1	0	0	0	0	0	0	1
Return Result	1	0	0	0	0	0	1	0
Return Error	1	0	0	0	0	0	1	1

Table 26/Q.773 [T28.773], p.

H.T. [T29.773]

TABLE 27/Q.773

Coding of General Problem

	H	**G**	**F**	**E**	**D**	**C**	**B**	**A**
{ Unrecognized Component | ua) }	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
{ Mistyped Component | ua) }	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
{ Badly Structured Component | ua) }	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**

a) TCAP Components are equivalent to ROSE APDUs.

Table 27/Q.773 [T29.773], p.

H.T. [T30.773]

TABLE 28/Q.773

Coding of Invoke Problem

	H	G	F	E	D	C	B	A
Duplicate Invoke ID	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Unrecognized Operation	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
{								
Mistyped Parameter | ua)								
}	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
Resource Limitation	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
{								
Initiating Release | ub)								
}	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
Unrecognized Linked ID	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
Linked Response Unexpected	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
{								
Unexpected Linked | uc) Operation								
}	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**

a) TCAP Invoke parameter is equivalent to ROSE Invoke argument.

b) ROSE uses ‘‘Initiator releasing’’ as only the initiator of the underlying association may release it. In TCAP, either entity may release the association.

c) ROSE refers to a linked operation as a child operation.

Table 28/Q.773 [T30.773], p.

H.T. [T31.773]

TABLE 29/Q.773

Coding of Return Result Problem

	H	G	F	E	D	C	B	A
Unrecognized Invoke ID	0	0	0	0	0	0	0	0
Return Result Unexpected	0	0	0	0	0	0	0	1
{								
Mistyped Parameter | ua)								
}	0	0	0	0	0	0	1	0

a) TCAP Return Result parameter is equivalent to ROSE Return Result result.

Table 29/Q.773 [T31.773], p.

H.T. [T32.773]

TABLE 30/Q.773

Coding of Return Error Problem

	H	G	F	E	D	C	B	A
Unrecognized Invoke ID	0	0	0	0	0	0	0	0
Return Error Unexpected	0	0	0	0	0	0	0	1
Unrecognized Error	0	0	0	0	0	0	1	0
Unexpected Error	0	0	0	0	0	0	1	1
Mistyped Parameter	0	0	0	0	0	1	0	0

Table 30/Q.773 [T32.773], p.

ANNEX A
(to Recommendation Q.773)

Specification of Transaction Capabilities in ASN

TCAPMessage { ccittRecommendationQ.773ModuleA } EFINITIONs ::=

BEGIN EXPORTS OPERATION, ERROR;

-- *Transaction Sub-Layer fields*

MessageType ::= CHOIC { Unidirectional [APPLICATION 1] IMPLICIT Unidirectional, begin [APPLICATION 2] IMPLICIT Begin, end [APPLICATION 4] IMPLICIT End, continue [APPLICATION 5] IMPLICIT Continue, abort [APPLICATION 7] IMPLICIT Abor }

Unidirectional ::= ComponentPortion Begin ::= SEQUENC { | rigTransactionID, ComponentPortion OPTIONAL } End ::= SEQUENC { | estTransactionID, ComponentPortion OPTIONAL } Continue ::= SEQUENC { | rigTransactionID, DestTransactionID, SEQUENCE] | ComponentPortion OPTIONAL } Abort ::= SEQUENC { | est-TransactionID, SEQUENC { | CHOIC { P-AbortCause, SEQUENC { | HOIC { UserAbortInformation OPTIONAL }

OrigTransactionID ::= [APPLICATION 8] IMPLICIT OCTET STRING DestTransactionID ::= [APPLICATION 9] IMPLICIT OCTET STRING

P-AbortCause ::= { APPLICATION 10] IMPLICIT INTEGE { unrecognizedMessageType (0), unrecognizedTransactionID (1), badlyFormattedTransactionPortion (2), incorrectTransactionPortion (3), resourceLimitation (4) }

UserAbortInformation ::= [APPLICATION 11] ANY OPTIONAL

-- *COMPONENT PORTION. The last field in the transaction portion of the TCAP message is the -- ComponentPortion. The Component Portion may be empty.*

ComponentPortion ::= [APPLICATION 12] IMPLICIT SEQUENCE OF Component

-- *Component Sub-Layer fields.*

-- *COMPONENT TYPE. Recommendation X.229 defines four Application Protocol Data Units (APDUs). -- TCAP adds returnResult-NotLast to allow for the segmentation of a result. Note: in X.229 EXPLICIT -- rather than IMPLICIT tagging is used*

Component ::= CHOICE { invoke [1] IMPLICIT Invoke, returnResultLast [2] IMPLICIT ReturnResult, returnError [3] IMPLICIT ReturnError, reject [4] IMPLICIT Reject, returnResultNot-Last [7] IMPLICIT ReturnResult }

-- *The Components are sequences of data elements.* .bp

Invoke ::= SEQUENC { invokeID INTEGER, linked-ID[0] IMPLICIT INTEGER, OPTIONAL, operation code OPERATION, parameter ANY DEFINED BY operation code OPTIONAL } -- *ANY is filled by the single ASN.1 data type -- following the key word ARGUMENT in the type -- definition of a particular operation.*

ReturnResult ::= SEQUENC { invokeID INTEGER, SEQUENC { operation code OPERATION, SEQUENC { parameters ANY DEFINED BY operation code -- *ANY is filled by the single ASN.1 data -- type following the key word RESULT in -- the type definition of a particular operation* } PTIONAL }

ReturnError ::= SEQUENC { invokeID INTEGER error code ERROR, parameter ANY DEFINED BY error code OPTIONAL } -- *ANY is filled by the single ASN.1 data type -- following the key word PARAMETER in the type -- definition of a particular error.*

Reject ::= SEQUENC { invokeID CHOICE { INTEGER NULL } problem CHOIC { [0] IMPLICIT GeneralProblem, [1] IMPLICIT InvokeProblem [2] IMPLICIT ReturnResultProblem, [3] IMPLICIT ReturnErrorProblem }

-- *OPERATIONS.*

-- Operations are specified with the OPERATION MACRO. When an operation -- is specified, the valid parameter set, results, and errors for that -- operation are indicated. Default values and optional parameters are -- permitted.

OPERATION MACRO ::=

BEGIN

TYPE NOTATION ::= Parameter Result Errors Linked Operations

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

Parameter ::= ‘‘PARAMETER’’ NamedTyped | 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 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 operation value is* | type -- *specified*

NamedType ::= identifier type | type

END

—— *ERRORS*

—— *Errors are specified with the ERROR MACRO. When an error is —— specified, the valid parameters for that error are indicated. —— Default values and optional parameters are permitted.*

ERROR MACRO ::=

BEGIN

TYPE NOTATION ::= Parameter

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

Parameter ::= ‘‘PARAMETER’’ NamedType | empty

NamedType ::= identifier type | type

END

—— *PROBLEMS.*

GeneralProblem ::= INTEGE { unrecognizedComponent (0), mistypedComponent (1), badlyStructuredComponent (2 }

InvokeProblem ::= INTEGE { duplicateInvokeID (0), unrecognizedOperation (1), mistypedParameter (2), resourceLimitation (3), initiatingRelease (4), unrecognizedLinkedID (5), linkedResponseUnexpected (6), unexpectedLinkedOperation (7) }

ReturnResultProblem ::= INTEGE { unrecognizedInvokeID (0), returnResultUnexpected (1), mistypedParameter (2) }

ReturnErrorProblem ::= INTEGE { unrecognizedInvokeID (0), returnErrorUnexpected (1), unrecognizedError (2), unexpectedError (3), mistypedParameter (4) }

END

APPENDIX I
(to Recommendation Q.773)

Formats and encoding for the Unidirectional message

I.1 *Introduction*

This Appendix provides the formats and encoding for the additional message type: Unidirectional.

I.2 *Structure of the Transaction Portion*

Table I-1/Q.773 relates to § 5.1. It shows the Transaction Portion fields for this message type.

H.T. [T33.773]
TABLE I-1/Q.773
Transaction Portion fields — Unidirectional message type

Element Form	Fields of Transaction Portion	Mandatory Indication
Constructor Message Type tag Total message length | ua) }	{ Mandatory	
Constructor Component Portion tag Component Portion length }	{ Mandatory | ub)	
Constructor One or more Components (Not a part of Transaction Portion) (Described in § 6) }	{ Optional	

a) See Note a) to Figure 8/Q.773.

b) The Component Portion Tag is not required if there are no Component being sent in the message.

Table I-1/Q.773 [T33.773], p.

I.3 *Message type tag*

Table I-2/Q.773 relates to § 5.2. It shows the coding of the Message Type tag. Note that the tag value included here is marked reserved in Table 8/Q.773.

H.T. [T34.773]
TABLE I-2/Q.773
Coding of Message type tag

Message Type	H	G	F	E	D	C	B	A
Unidirectional	0	1	1	0	0	0	0	1

Table I-2/Q.773 [T34.773], p.

I.4 *Transaction IDs*

Table I-3/Q.773 shows the usage of Transaction IDs in the Unidirectional message type. No Transaction IDs are present.

H.T. [T35.773]
TABLE I-3/Q.773
Transaction ID(s) in each message type

Message Type	Originating ID	Destination ID
Unidirectional	No	No

Table I-3/Q.773 [T35.773], p.

I.5 *Component Portion*

The Component Portion in Unidirectional messages is as specified in § 6.

I.6 *Specification of the Unidirectional message in ASN*

—— *The ASN specification of the Unidirectional message (in —— conjunction with Annex A) is provided here. The following —— line should be added to the CHOICE of Message Type:* unidirectional ::= [APPLICATION 1] IMPLICIT Uni

—— *The structure of the Unidirectional Message Type is:* Uni ::= ComponentPortion

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 when 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.

H.T. [T1.774]

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)**	

Note 1 — X.219 and X.229 Compatible.

Note 2 — Treatment of this primitive is described in § 3.2.2.2.

Note 3 — There is no component type associated with this primitive since the effect is purely local.

Tableau 1/Q.774 [T1.774], p.

3.2.1.1.3 *Operation classes*

H.T. [T2.774]

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**

Table 2/Q.774 [T2.774], p.

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.

Figure 1/Q.774, p.

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 cancel or end situation occurs. This is indicated in Figure 2/Q.774.

Figure 2/Q.774, p.

A timeout for a class 2 operation is a ‘‘normal’’ situation.

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 cancel or end situation occurs. This is indicated in Figure 3/Q.774.

Figure 3/Q.774, p.

A timeout for a class 3 operation is a ‘‘normal’’ situation.

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 cancel or end situation occurs. This is indicated in Figure 4/Q.774.

Figure 4/Q.774, p.

A timeout for a class 4 operation is a ‘‘normal’’ situation.

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.

Figure 5/Q.774, p.

Figure 6/Q.774 depics 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.

Figure 6/Q.774, p.

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.

H.T. [T3.774]

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

Table 3/Q.744 [T3.774], p.

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.

H.T. [T4.774]

TABLE 4/Q.774

Action Taken on Protocol Errors in Component Portion

Local	Remote

Component Type received	Type of error	Local action	Component State Machine	Local user advised	Component state machi
INVOKE	Syntax error Linked ID unassigned	Init. Reject Init. Reject	Inv: NA Link: No change Inv: NA Link: NA	Yes | ua) Yes | ua)	Return to Idle Inv: Return to Idle
{	Syntax error Invoke ID unassigned	Init. Reject Init. Reject	Return to Idle NA	Yes | ua) Yes | ua)	NA NA
RETURN _RESULT (L/NL)	Operation Class 2/4	Init. Reject	Return to Idle	Yes | ua)	NA
RETURN _ERROR	Operation Class 3/4	Init. Reject	Return to Idle	Yes | ua)	NA
REJECT	Syntax Error	Local Reject	Return to NA | ub)	Yes	NA
UNKNOWN	Invoke ID derivable Invoke ID non derivable	Init. Reject Init. Reject	No Change (NA) (NA)	Yes | ua) Yes | ua)	Return to Idle NA

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.

Table 4/Q.774 [T4.774], p.

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.

H.T. [T5.774]

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

Table 5/Q.774 [T5.774], p.

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.

Figure 7/Q.774, p.

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.

Figure 8/Q.774, p.14

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) The transaction sub-layer should form an abort message with an appropriate 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) If the destination transaction ID is not derivable or derivable but not assigned, the transaction sub-layer takes no action to terminate the transaction at its end.

iii) If the destination transaction ID is derivable and assigned:

a) the transaction sub-layer terminates the transaction at its end, i.e. return to idle;

b) the transaction sub-layer informs the component sub-layer of the abort of the transaction via the transaction sub-layer abort; and

c) the component sub-layer should:

— 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.

H.T. [T6.774]

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.
UNIDIRECTIONAL	—	—	Discard	— | uc)	No	— | uc)
BEGIN	not der. der.	— —	Discard Abort	NA NA	No No	NA Ret to Idle | ua)
CONTINUE	not der. der.	— not der unass.	Discard Abort	NA NA	No No	NA Ret to Idle | ua)
END/ABORT	— —	not der unass. ass.	Discard Discard	NA Ret to Idle	No Yes	NA NA
UNKNOWN	not der der.	— not der unass.	Discard Abort	NA NA	No No	NA Ret to Idle | ub)

NA: Transition to the Idle state is Not Applicable \| ub)

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.

Table 6/Q.774 [T6.774], p.

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 Not last component

SCCP Signalling connection control part

TC Transaction capabilities

TCAP Transaction capabilities application part

TCU TC-user

TSL Transaction sub-layer

ISP Intermediate service part

IS Initiation sent state

IR Initiation received state

DHA Dialogue handling

CHA Component handling

RJ Reject

RE Return error

RR Return result

INV Invoke

ISM Invocation state machine

CCO Component coordinator

UNI Unidirectional

A.3 *Drafting conventions*

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

Figure, p.

Figure A-1/Q.774, p.16

Figure A-2a/Q.774, p.17

Figure A-2b/Q.774, p.18

Figure A-3/Q.774 (page 1 sur 6), p.19

Figure A-3/Q.774 (page 2 sur 6), p.20

Figure A-3/Q.774 (page 3 sur 6), p.21

Figure A-3/Q.774 (page 4 sur 6), p.22

Figure A-3/Q.774 (page 5 sur 6), p.23

Figure A-3/Q.774 (page 6 sur 6), p.24

Figure A-4/Q.774 (page 1 sur 2), p.25

Figure A-4/Q.774 (page 2 sur 2), p.26

Figure A-5/Q.774 (page 1 sur 4), p.27

Figure A-5/Q.774 (page 2 sur 4), p.28

Figure A-5/Q.774 (page 3 sur 4), p.29

Figure A-5/Q.774 (page 4 sur 4), p.30

Figure A-6/Q.774 (page 1 sur 6), p.31

Figure A-6/Q.774 (page 2 sur 6), p.32

Figure A-6/Q.774 (page 3 sur 6), p.33

Figure A-6/Q.774 (page 4 sur 6), p.34

Figure A-6/Q.774 (page 5 sur 6), p.35

Figure A-6/Q.774 (page 6 sur 6), p.36