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

Fig. 1/Q.773 /T1120490-88 = 4.5 cm

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.

Fig. 2/Q.773 /T1120500-88 = 4.5 cm

Fig. 3/Q.773 /T1120510-88 = 4.5 cm

The contents of each eleent is either one value (Primitive) or one or more information elements (Constructor), as shown in Figure 4/Q.773.
Fig. 4/Q.773 /T1120520-88 = 5 cm

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.
Fig. 5/Q.773 /T1120530-88 = 3 cm

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.

TABLE 1/Q.773

Coding of tag class

Class	Coding (HG)
Universal	00
Application–wide	01

Context–specific	10
Private use	11

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.

TABLE 2/Q.773

Coding element form

Element form

	Coding (F)
Primitive	0
Constructor	1

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 /T1106430-87 = 6.5 cm

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	Contents
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.

0

Length of Contents

MSB

LSB

a) Short form

1

(Length of Field Size -1

MSB

LSB

MSB

Length of Contents

LSB

b) Long form

Constructor Element Tag

L = 10000000

Tag
Length (Note)
Contents

Tag
Length (Note)
Contents

EOC Tag = 00000000

EOC Length = 00000000

c) Indefinite form

Note – The Length may take any of three forms: short, long, and indefinite.

FIGURE 7/Q.773

Format of length field

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.

Message Type Tag

Total Message Length a)

Transaction Portion Information Element

:
:

Component Portion Tag

Component Portion Length

Component Type Tag

Component Length

Component Portion Information Element

:
:

:
:

Component

a) The user should be aware of total message length limitations when using TCAP in the SS No. 7 connectionless environment.

FIGURE 8/Q.773

Detailed TCAP message structure

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.

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 a)	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 4/Q.773

Transaction portion fields
Begin message type

Element Form

Fields of Transaction Portion

Mandatory Indication

Constructor

Message type tag
Total message length a)

Mandatory

Primitive

Originating Transaction ID tag
Transaction ID length
Transaction ID

Mandatory

Constructor

Component Portion tag
Component Portion length

Mandatory b)

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 5/Q.773

Transaction Portion fields
End message type

Element Form

Fields of Transaction Portion

Mandatory Indication

Constructor

Message type tag
Total message length a)

Mandatory

Primitive

Destination Transaction ID tag
Transaction ID length
Transaction ID

Mandatory

Constructor

Component Portion tag
Component Portion length

Mandatory b)

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.

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 a)

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 b)

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.

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 a)

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 b)

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.

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 a)

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 b)

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.

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.

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

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

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.

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

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

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.

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

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

5.5 *User Abort Information tag*

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

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.

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

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.

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

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 b)

Operation Code tag
Operation Code length
Operation Code

O b)

Parameters

O b)

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.

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

TABLE 19/Q.773

Reject component

Reject component

Mandatory Indication

Component type tag
Component length

M

Invoke ID tag a) 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.

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

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.

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.

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.

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

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.

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

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.

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

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.

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

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.

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 27/Q.773

Coding of General Problem

H

G

F

E

D

C

B

A

Unrecognized Component a)

0

0

0

0

0

0

0

0

Mistyped Component a)

0

0

0

0

0

0

0

1

Badly Structured Component a)

0

0

0

0

0

0

1

0

a) TCAP Components are equivalent to ROSE APDUs.

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 a)

0

0

0

0

0

0

1

0

Resource Limitation

0

0

0

0

0

0

1

1

Initiating Release b)

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 c) 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 29/Q.773

Coding of Return Result Problem

H

G

F

E

B

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 a)

0

0

0

0

0

0

1

0

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

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

ANNEX A

(to Recommendation Q.773)

Specification of Transaction Capabilities in ASN

```
TCAPMessages{ ccittRecommendationQ.773ModuleA }DEFINITIONs ::=

BEGIN

EXPORTS  OPERATION, ERROR;

— Transaction Sub–Layer fields

MessageType ::=     CHOICE{            Unidirectional       [APPLICATION                         1]
           IMPLICIT Unidirectional,
                             begin            [APPLICATION                                       2]
           IMPLICIT Begin,
                             end              [APPLICATION                                       4]
           IMPLICIT End,
                             continue         [APPLICATION                                       5]
           IMPLICIT Continue,
                             abort            [APPLICATION                                       7]
           IMPLICIT Abort}
Unidirectional ::=  ComponentPortion
Begin ::=           SEQUENCE{ OrigTransactionID, ComponentPortion OPTIONAL }
End ::=        SEQUENCE{ DestTransactionID, ComponentPortion OPTIONAL }
Continue ::=        SEQUENCE{ OrigTransactionID, DestTransactionID,
                              ComponentPortion OPTIONAL }
Abort ::=           SEQUENCE{ DestTransactionID,
                             CHOICE{ P–AbortCause,

                UserAbortInformation OPTIONAL }}
OrigTransactionID ::= [APPLICATION 8] IMPLICIT OCTET STRING
DestTransactionID ::= [APPLICATION 9] IMPLICIT OCTET STRING
P–AbortCause ::= {[APPLICATION 10] IMPLICIT INTEGER{
                            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 returnResultNotLast 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,
                        returnResultNotLast
        [7] IMPLICIT ReturnResult }
— The Components are sequences of data elements.
Invoke ::=       SEQUENCE{

                      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 ::=      SEQUENCE{

                      invokeID INTEGER,
                                 SEQUENCE{ operation code OPERATION,

                      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
                         }OPTIONAL }
ReturnError ::=       SEQUENCE{
```

```
                    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 ::=      SEQUENCE{

                    invokeID CHOICE { INTEGER NULL }

                    problem CHOICE{

                      [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 CHOICE{




                    localValue INTEGER,

                 globalValue OBJECT IDENTIFIER })
```

```
Parameter ::=
          “PARAMETER” NamedTyped | empty

Result ::=                    “RESULT”ResultType | empty

ResultType ::=                NamedType | empty

Errors::=
          “ERRORS” “{”ErrorNames“}” | empty

LinkedOperations ::=
          “LINKED” “{”LinkedOperationNames“}” | 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 CHOICE{




                        localValue INTEGER,

                    globalValue OBJECT IDENTIFIER })
Parameter ::=
        “PARAMETER” NamedType | empty
NamedType ::=           identifier type | type
END
```

```
— PROBLEMS.
GeneralProblem ::=       INTEGER{
         unrecognizedComponent (0),



                              mistypedComponent (1),

                         badlyStructuredComponent (2)}
InvokeProblem ::=
         INTEGER{ duplicateInvokeID (0),



                              unrecognizedOperation (1),

                         mistypedParameter (2),

                         resourceLimitation (3),

                         initiatingRelease (4),

                         unrecognizedLinkedID (5),

                         linkedResponseUnexpected (6),

                         unexpectedLinkedOperation (7) }
ReturnResultProblem                                                                                  ::=
         INTEGER{ unrecognizedInvokeID (0),



                              returnResultUnexpected (1),

                         mistypedParameter (2) }
ReturnErrorProblem ::=
         INTEGER{ 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 theTransaction Portion*

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

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 a)

Mandatory

Constructor

Component Portion tag
Component Portion length

Mandatory b)

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.

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.

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

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.

TABLE I–3/Q.773

Transaction ID(s) in each message type

Message Type	Originating ID	Destination ID
Unidirectional	No	No

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