INTERNATIONAL TELECOMMUNICATION UNION

ITU-T Q.754

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(03/93)

SPECIFICATIONS OF SIGNALLING SYSTEM No. 7

SIGNALLING SYSTEM No. 7 MANAGEMENT

SIGNALLING SYSTEM NUMBER 7 MANAGEMENT APPLICATION SERVICE ELEMENT (ASE) DEFINITIONS

ITU-T Recommendation Q.754

(Previously "CCITT Recommendation")

FOREWORD

The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the International Telecommunication Union. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, established the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

ITU-T Recommendation Q.754 was prepared by the ITU-T Study Group XI (1988-1993) and was approved by the WTSC (Helsinki, March 1-12, 1993).

NOTES

1 As a consequence of a reform process within the International Telecommunication Union (ITU), the CCITT ceased to exist as of 28 February 1993. In its place, the ITU Telecommunication Standardization Sector (ITU-T) was created as of 1 March 1993. Similarly, in this reform process, the CCIR and the IFRB have been replaced by the Radiocommunication Sector.

In order not to delay publication of this Recommendation, no change has been made in the text to references containing the acronyms "CCITT, CCIR or IFRB" or their associated entities such as Plenary Assembly, Secretariat, etc. Future editions of this Recommendation will contain the proper terminology related to the new ITU structure.

2 In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

CONTENTS

SIGNALLING SYSTEM NUMBER 7 MANAGEMENT APPLICATION SERVICE ELEMENT (ASE) DEFINITIONS

(Helsinki, 1993)

1 Introduction

Note that in the event of a conflict between Recommendations Q.753 and Q.754, Q.753 will take precedence.

This Recommendation defines the OMAP ASE, OMASE. OMASE provides the services invoked using the OM-EVENT-REPORT and OM-CONFIRMED-ACTION primitives across the OMASE-User to OMASE boundary. (See Recommendation Q.753 for a diagram and mapping between the services invoked in the OMASE-User and those of OMASE.)

The OMASE services are derived from those defined in CMIP[1)].

The OMASE primitives are defined in clause 6, the formal syntax defined in Figure 3 uses Transaction Capabilities (TC) OPERATION and ERROR macros. The interworking between OMASE and TC is also given in clause 6.

OMASE provides operations allowing the network administration, via the OMAP Management Process and the OMASE-User, to perform MTP and SCCP Routing Verification Tests (MRVT and SRVT), and circuit validation tests (CVT). This Recommendation contains the ASE definition for MRVT, SRVT and CVT.

The SRVT referred-to here is for the specific test in 3.2.2/Q.753.

The arguments used for primitives across the OMAP Management Process to OMASE-User boundary, and for primitives across the OMASE-User to OMASE boundary, and between OMASE and TC contain the same information if they have the same name. Those arguments are defined in this Recommendation.

2 MTP

2.1 MTP Routing Verification Test (MRVT)[2)]

The MRV Test initiated at the test origin results in an OM-CONFIRMED-ACTION primitive being used from the OMASE-User to OMASE, which includes the testRoute command as a parameter. If a trace of the routes is requested, or a fault exists, the OM-EVENT-REPORT primitive is invoked at the test originator from OMASE, which includes the routeTrace event as a parameter.

testRoute is specified using the CNF-ACTION macro defined in Figure 3, routeTrace is specified using the EVENT macro defined in Figure 3.

For MRVT, the ObjectClass indicates MTP Routing Tables, and the ObjectInstance contains the Point Code of the test destination. The testRoute Action makes use of the BEGIN (MRVT) message with result (MRVA) returning in an END. The routeTrace Event (MRVR) uses a BEGIN message with pre-arranged end.

2.1.1 testRoute Action

The testRoute Action is invoked to initiate an MTP routing verification test. At the initiator node, this invocation is requested by the Administration via the MIS-User or a local interface, through the OMAP Management Process and OMASE-User. At subsequent nodes, the Action is requested implicitly by the receipt of a testRoute Action invocation. A

---

1) CMIP is defined in ISO IS 9596 and in Recommendation X.711.

2) See Recommendations X.208 and X.209 for the description of the formal notation.

successful reply indicates successful completion of the test at the point it was invoked and, implicitly, at all subsequent points where the test was invoked. A failure indication is returned to indicate that the test failed in this or a subsequent node.

testRoute CNF-ACTION	*Timer = T1*	*Class = 1*	*Code = 00000001*
ActionArg		*Opt/Man*	*References*
initiatingSP		M	2.1.1.1.1
traceRequested		M	2.1.1.1.2
threshold		M	2.1.1.1.3
pointCodesTraversed		M	2.1.1.1.4
ActionResult			
empty			
Linked Operations			
N/A			
Specific Errors			*References*
failure			2.1.1.3.1
partialSuccess			2.1.1.3.2

```
testRoute CNF-ACTION
ACTIONINFOARG SEQUENCE {
                              initiatingSP            [0] IMPLICIT PointCode,
                              traceRequested          [1] IMPLICIT BOOLEAN,
                              threshold               [2] IMPLICIT INTEGER,
                              pointCodesTraversed     [3] IMPLICIT PointCodeList
}
SPECIFICERRORS                { failure, partialSuccess }
::= 1
```

2.1.1.1 testRoute Action Arguments

2.1.1.1.1 initiatingSP

The initiatingSP identifies the original requestor of the test. It is of type PointCode, defined as an octet string.

Parameter	*Code*
initiatingSP	10000000
Contents	
Bit 0 contains the first bit of the Point Code, Bit 1 contains the second bit of the Point Code, etc.	

```
PointCode ::= OCTET STRING
```

2.1.1.1.2 traceRequested

traceRequested indicates that a trace of all routes used to reach the destination should be reported to the originator (the routeTrace Event is described in 2.1.2). It is of type BOOLEAN.

Parameter	*Code*
traceRequested	10000001
Contents	*Meaning*
TRUE (= 1)	trace was requested, return trace information on success and failure
FALSE (= 0)	trace not requested, return trace information only on failure

2.1.1.1.3 threshold

The originator sets a maximum threshold level of Signalling Points (SP) which are allowed to be crossed in the course of the test (including the initiator if it is an STP). This aids in detecting overly long routes. This threshold is an integral number of SPs, thus it is of type INTEGER.

Parameter	*Code*
threshold	10000010
Contents	
Integer number	

2.1.1.1.4 pointCodesTraversed

As each intermediate SP is crossed, it adds its own Point Code to the list of Point Codes traversed. This aids in detecting loops and is also useful information in case of a failure or if a route trace is requested. It is a list of Point Codes, thus of type PointCodeList. This PointCodeList could be empty.

Parameter	*Code*
pointCodesTraversed	10100011
Contents	
Sequence of Point Codes, tagged as “PointCode” with the contents indicating the exact Point Code	

```
PointCodeList ::= SEQUENCE OF PointCode
```

2.1.1.2 Action Results

There are no contents in a successful return indication.

2.1.1.3 Action Errors

SpecificErrors are possible errors which can occur during this test, and which are unique to this test. These specific errors are in addition to the errors already identified in the om-Action-Confirmed service and appear as parameters to the Processing Failure Error.

2.1.1.3.1 failure

failure indicates a condition of total failure, where no route worked correctly. Most often this will be used as a failure indication from the point which detects the error and does not invoke any further testRoute Actions. The failure SpecificError has with it a parameter to indicate the error condition causing the failure. This parameter, failureType, is represented as a bit string. In addition, the second parameter is to be used when failureType indicates the error UnknownInitiatingSP. traceSent indicates whether or not a routeTrace Event has been invoked to report trace information. It is necessary to indicate this for this error since the node detecting the error cannot send the routeTrace, thus the previous node must. traceSent is a type of BOOLEAN.

Specific Error	*Code*
failure	00000001
Parameters	*References*
failureType	2.1.1.3.1
traceSent	2.1.1.3.1

Parameter	*Code*
failureType	10000000
Bit	*Meaning*
0	detectedLoop
1	excessiveLengthRoute
2	unknownObjectInstance
3	routeInaccessible
4	processingFailure
5	unknownInitiatingSP
6	timerExpired
7	sPNotAnSTP

Parameter	*Code*
traceSent	10000001
Contents	*Meaning*
TRUE	the trace information was sent
FALSE	the trace information was not sent

```
failure         SPECIFIC-ERROR
                PARAMETER SEQUENCE      { failureType       [0] IMPLICIT FailureString,
                                          traceSent         [ l ] IMPLICIT BOOLEAN }
                ::= 1
```

```
FailureString ::= BIT STRING
                        { detectedLoop (0),
                        excessiveLengthRoute (1),
                        unknownObjectInstance (2),
                        routeInaccessible (3),
                        processingFailure (4),
                        unknownInitiatingSP (5),
                        timerExpired (6),
                        sPNotAnSTP (7) }
```

2.1.1.3.2 partialSuccess

This indication is given when at least one testRoute Cnf Action invocation failed and at least one succeeded (at least partially). In this case, each type of error that occurred will be noted and sent in the final reply. The format and contents of partial success are the same as failure.

Specific Error	*Code*
partialSuccess	00000010
Parameters	*References*
failureType	2.1.1.3.1
traceSent	2.1.1.3.1

```
partialSuccess  SPECIFIC-ERROR
                PARAMETER SEQUENCE      { failureType       [0] IMPLICIT FailureString,
                                          traceSent         [1] IMPLICIT BOOLEAN }
                ::= 2
```

2.1.2 routeTrace Event

The routeTrace Event reports trace information. Trace information consists of zero, one or more Point Codes, such as the Point Code detecting an error or the entire list of Point Codes traversed along a route. This event is invoked either at the explicit request of the originating node (indicated by traceRequested, see 2.1.1.1.2) or by failure at any point along the route. This event is not confirmed, therefore no replies to this invocation are expected (no error or success indications are expected).

routeTrace EVENT	*Timer* = 0	*Class* = 4	*Code* = 00000010
EventInfo		*Opt/Man* (Note)	*References*
success		O	2.1.2.1.1
detectedLoop		O	2.1.2.1.2
excessiveLengthRoute		O	2.1.2.1.3
unknownObjectInstance		O	2.1.2.1.4
routeInaccessible		O	2.1.2.1.5
processingFailure		O	2.1.2.1.6
unknownInitiatingSP		O	2.1.2.1.7
timerExpired		O	2.1.2.1.8
sPNotAnSTP		O	2.1.2.1.9
NOTE – One and only one of these must be present.			

```
routeTrace      EVENT
                EVENTINFO CHOICE {
                                success                 [0] IMPLICIT PointCodeList,
                                detectedLoop            [1] IMPLICIT PointCodeList,
                                excessiveLengthRoute    [2] IMPLICIT PointCodeList,
                                unknownObjectInstance   [3] IMPLICIT NULL,
                                routeInaccessible       [4] IMPLICIT PointCode,
                                processingFailure       [5] IMPLICIT NULL,
                                unknownInitiatingSP     [6] IMPLICIT PointCode,
                                timerExpired            [7] IMPLICIT PointCodeList,
                                sPNotAnSTP              [8] IMPLICIT PointCodeList,
                }
                ::= 2
```

2.1.2.1 Event Information

2.1.2.1.1 success

On successful completion, the trace of the Point Codes (one or more) of the crossed SPs are included.

Parameter	*Code*
success	10100000
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	2.1.1.1.4

2.1.2.1.2 detectedLoop

When a loop is detected, the Point Codes (three or more) contained in the loop are included.

Parameter	*Code*
detectedLoop	10100001
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	2.1.1.1.4

2.1.2.1.3 excessiveLengthRoute

When an excessively long route is found (threshold exceeded), the entire route is included.

Parameter	*Code*
excessiveLengthRoute	10100010
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	2.1.1.1.4

2.1.2.1.4 unknownObjectInstance

If the object instance is unknown, no additional information is required.

Parameter	*Code*
unknownObjectInstance	10000011
Contents	*Reference*
empty	–

2.1.2.1.5 routeInaccessible

The Point Code of the node where the route was inaccessible is included.

Parameter	*Code*
routeInaccessible	10000100
Contents	*Reference*
Bit 0 contains the first bit of the Point Code, Bit 1 contains the second bit of the Point Code, etc.	2.1.1.1.1

2.1.2.1.6 processingFailure

If a processing failure occurs, no additional information is required.

Parameter	*Code*
processingFailure	10000101
Contents	*Reference*
empty	–

2.1.2.1.7 unknownInitiatingSP

The Point Code of the node detecting the unknown Initiating SP is included.

Parameter	*Code*
unknownInitiatingSP	10000110
Contents	*Reference*
Bit 0 contains the first bit of the Point Code, Bit 1 contains the second bit of the Point Code, etc.	2.1.1.1.1

2.1.2.1.8 timerExpired

The Point Code(s) of the node(s) from where no result for the testRoute Action was received is included.

Parameter	*Code*
timerExpired	10100111
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	2.1.1.1.4

2.1.2.1.9 sPNotAnSTP

If the intermediate SP receiving an MRVT message does not have the MTP transfer function, the list of crossed SPs to reach this SP is included.

Parameter	*Code*
sPNotAnSTP	10101000
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	2.1.1.1.4

3 SCCP

3.1 SCCP Routing Verification Test (SRVT) ASE

This specific SRVT's functions are defined in 3.2.2/Q.753. The SRV Test initiated at the test origin results in an OM-CONFIRMED-ACTION primitive being used from the OMASE-User to OMASE, which includes the testRoute command as a parameter. If a trace of the routes is requested, or a fault exists, the OM-EVENT-REPORT primitive is invoked at the test originator from OMASE, which includes the routeTrace event as a parameter.

testRoute is specified using the CNF-ACTION macro defined in Figure 3, routeTrace is specified using the EVENT macro defined in Figure 3.

The ObjectClass indicates SCCP Global Title Translation Tables and the ObjectInstance contains the Global Title Indicator and Tested Global Title. The GTI is coded as defined in the SCCP Address Indicator. The testRoute Action (SRVT) makes use of the BEGIN message with the result (SRVA) returning in an END. The routeTrace Event (SRVR) uses a BEGIN message with prearranged end.

3.1.1 testRouteAction

The testRoute Action is invoked to initiate an SCCP Routing Verification Test. At the initiator node, this invocation is requested by the Administration via the MIS-User or a local interface, through the OMAP Management Process and OMASE-User. At subsequent nodes, the Action is requested implicitly by the receipt of a testRoute Action invocation. A successful reply indicates successful completion of the test at the point it was invoked and, implicitly, at all subsequent points where the test was invoked. A failure indication is returned to indicate that the test failed in this or a subsequent node.

testRoute CNF-ACTION	*Timer* = T2	*Class* = 1	*Code* = 00000001
ActionArg		*Opt/Man*	*References*
initiatingSP		M	3.1.1.1.1
traceRequested		M	3.1.1.1.2
threshold		M	3.1.1.1.3
pointCodesTraversed		M	3.1.1.1.4
formIndicator		M	3.1.1.1.5
mtpBackwardRoutingRequested		M	3.1.1.1.6
testInitiatorGT		O	3.1.1.1.7
destinationPC		O	3.1.1.1.8
destinationSSN		O	3.1.1.1.9
backupDPC		O	3.1.1.1.10
backupSSN		O	3.1.1.1.11
originalGT		O	3.1.1.1.12
ActionResult			
empty			
Linked Operations			
N/A			
Specific Errors			References
failure			3.1.1.3.1
partialSuccess			3.1.1.3.2

```
testRoute CNF-ACTION
ACTIONINFOARG SEQUENCE {
        initiatingSP                    [0] IMPLICIT PointCode,
        traceRequested                  [1] IMPLICIT BOOLEAN,
        threshold                       [2] IMPLICIT INTEGER,
        pointCodesTraversed             [3] IMPLICIT PointCodeList,
        formIndicator                   [4] IMPLICIT FormIndicator,
        mtpBackwardRoutingRequested     [5] IMPLICIT BOOLEAN,
        testInitiatorGT                 [6] IMPLICIT GlobalTitle OPTIONAL,
        destinationPC                   [7] IMPLICIT PointCode OPTIONAL,
        destinationSSN                  [8] IMPLICIT SubsystemNumber OPTIONAL,
        backupDPC                       [9] IMPLICIT PointCode OPTIONAL,
        backupSSN                       [10] IMPLICIT SubsystemNumber OPTIONAL,
        originalGT                      [11] IMPLICIT GlobalTitle OPTIONAL
}
SPECIFICERRORS                          { failure, partialSuccess }
::= 1
```

3.1.1.1 testRoute Action Arguments

3.1.1.1.1 initiatingSP

The initiatingSP identifies the test initiator. It is of type PointCode.

Parameter	*Code*
initiatingSP	10000000
Contents	
Bit 0 contains the first bit of the Point Code, Bit 1 contains the second bit of the Point Code, etc.	

```
PointCode ::= OCTET STRING
```

3.1.1.1.2 traceRequested

traceRequested indicates that a trace of all routes used to reach the destination should be reported to the originator (the routeTrace Event is described in 3.1.2). It is of type BOOLEAN.

Parameter	*Code*
traceRequested	10000001
Contents	*Meaning*
TRUE (= 1)	trace was requested, return trace information on success and failure
FALSE (= 0)	trace not requested, return trace information only on failure

3.1.1.1.3 threshold

The originator sets a maximum threshold level of Translation Signalling Points (TSP) which are allowed to be crossed in the course of the test (including the initiator if it is an SCCP Relay Node). This aids in detecting overly long routes. This threshold is an integral number of SP's, thus it is of type INTEGER.

Parameter	*Code*
threshold	10000010
Contents	
Integer number	

3.1.1.1.4 pointCodesTraversed

As each Translation SP is crossed, it adds its own Point Code to the list of Point Codes traversed. This aids in detecting loops and is also useful information in case of a failure or if a route trace is requested. It is a list of Point Codes, thus of type PointCodeList. This PointCodeList could be empty.

Parameter	*Code*
pointCodesTraversed	10100011
Contents	
Sequence of Point Codes, tagged as "PointCode" with the contents indicating the exact Point Code	

```
PointCodeList ::= SEQUENCE OF PointCode
```

3.1.1.1.5 formIndicator

The formIndicator identifies the form of the SRVT message, i.e. either Request, Verify or Compare. lt is of type INTEGER, with the values defined as below.

Parameter	*Code*
formIndicator	10000100
Contents	
Value 0 = Compare Value 1 = No Compare	

```
FormIndicator ::=        INTEGER
                             { compare (0),
                             noCompare (l) }
```

3.1.1.1.6 mtpBackwardRoutingRequested

The mtpBackwardRoutingRequested identifies whether MTP backward routing towards the OPC is required for test success. It is of type BOOLEAN.

Parameter	*Code*
mtpBackwardRoutingRequested	10000101
Contents	
TRUE (= 1) Routing Requested FALSE (= 0) Routing Not Requested	

3.1.1.1.7 testInitiatorGT

The testInitiatorGT identifies the Global Title Indicator and the initiator's Global Title. It is of type OCTET STRING.

Parameter	*Code*
testInitiatorGT	10000110
Contents	
Octet 1 = GlobalTitle Indicator Octet 2, 3, ... = Initiator Global Title	

GlobalTitle ::= OCTET STRING

3.1.1.1.8 destinationPC

The dcstinationPC identifies the Destination Point Code (PPC or TPC). It is of type PointCode.

Parameter	*Code*
destinationPC	10000111
Contents	
Bit 0 contains the first bit of the Point Code, Bit 1 contains the second bit of the Point Code, etc.	

3.1.1.1.9 destinationSSN

The destinationSSN identifies the destination Subsystem Number. It is of type OCTET STRING.

Parameter	*Code*
destinationSSN	10001000
Contents	
Bit 0 contains the first bit of the Subsystem Number, Bit 1 contains the second bit of the Subsystem Number, etc.	

SubsystemNumber ::= OCTET STRING

3.1.1.1.10 backupDPC

The backupDPC identifies the backup Destination Point Code (SPC). It is of type PointCode.

Parameter	*Code*
backupDPC	10001001
Contents	
Bit 0 contains the first bit of the Point Code, Bit 1 contains the second bit of the Point Code, etc.	

3.1.1.1.11 backupSSN

The backupSSN identifies the backup destination Subsystem Number. It is of type OCTET STRING.

Parameter	*Code*
backupSSN	10001010
Contents	
Bit 0 contains the first bit of the Subsystem Number, Bit 1 contains the second bit of the Subsystem Number, etc.	

3.1.1.1.12 originalGT

The originalGT field is only present in an SRVT message if translation of a GT in the called party address yields or has already yielded a replacement GT.

In this case, the field in an SRVT message to be sent out is as follows:

i) if the SRVT message to be sent out is not the compare form, and the test is initiated by receipt of an SRVT message containing an originalGT, then the field is copied across; or

ii) in all other cases, the originalGT sent out is the GT of the called party address for the SRVT message before translation.

The field is used as the GT of the calling party address in any SRVR message sent and, for the compare form of SRVT message, by the mate TSP receiving it to check that its translation yields the GT in the called party address field of the compare SRVT message received.

The type of originalGT is GlobalTitle.

Parameter	*Code*
originalGT	10001011
Contents	
Octet 1 = Global Title Indicator Octet 2, 3,... = Original Global Title	

3.1.1.2 Action Results

There are no contents in a successful return indication.

3.1.1.3 Action Errors

SpecificErrors are possible errors which can occur during this test, and which are unique to this test. These specific errors are in addition to the errors already identified in the om-Action-Confirmed service and appear as parameters to the Processing Failure Error.

3.1.1.3.1 failure

failure indicates a condition of failure, where a Translation could not be successfully done, or was incorrect. Most often this will be used as a failure indication from the point which detects the error and does not invoke any further testRoute Actions. The failure SpecificError has with it a parameter to indicate the error condition causing the failure. This parameter, failureType, is represented as a bit string. In addition, the second parameter is to be used when failureType indicates the error Unknown Initiating SP. traceSent indicates whether or not a routeTrace Event has been invoked to report trace information. It is necessary to indicate this for this error since the node detecting the error cannot send the routeTrace, thus the previous node must. traceSent is a type of BOOLEAN and is optional.

Specific Error	*Code*
failure	00000001
Parameters	*References*
failureType traceSent	3.1.1.3.1 3.1.1.3.1

Parameter	*Code*
failureType	10000000
Bit	*Meaning*
0	detectedLoop
1	excessiveLengthRoute
2	unknownObjectInstance
3	routeInaccessible
4	processingFailure
5	unknownInitiatingSP
6	timerExpired
7	wrongSP
8	incorrectTranslation-Primary
9	incorrectTranslation-Secondary
10	incorrectTranslation-Intermediate
11	notPrimaryDestination
12	notSecondaryDestination
13	notRecognizedPrimary
14	notRecognizedSecondary
15	routingProblem

Parameter	*Code*
traceSent	10000001
Contents	*Meaning*
TRUE	the trace information was sent
FALSE	the trace information was not sent

```
failure         SPECIFIC-ERROR
                PARAMETER SEQUENCE      { failureType       [0] IMPLICIT FailureString,
                                          traceSent         [l ] IMPLICIT BOOLEAN }
                ::= 1
```

```
FailureString ::= BIT STRING
                        { detectedLoop (0),
                        excessiveLengthRoute (1),
                        unknownObject (2),
                        routeInaccessible (3),
                        processingFailure (4),
                        unknownInitiatingSP (5),
                        timerExpired (6),
                        wrongSP (7),
                        incorrectTranslation-Primary (8),
                        incorrectTranslation-Secondary (9),
                        incorrectTranslation-Intermediate (10),
                        notPrimaryDestination (11),
                        notSecondaryDestination (12),
                        notRecognizedPrimary (13),
                        notRecognizedSecondary (14),
                        routingProblem (15) }
```

3.1.1.3.2 partialSuccess

This indication is given when at least one testRoute Cnf Action invocation failed and at least one succeeded (at least partially). In this case, each type of error that occurred will be noted and sent in the final reply. The format and contents of partial success are the same as failure.

Specific Error	*Code*
partialSuccess	00000010
Parameters	*References*
failureType	3.1.1.3.1
traceSent	3.1.1.3.1

```
partialSuccess      SPECIFIC-ERROR
                    PARAMETER SEQUENCE      { failureType     [0] IMPLICIT FailureString,
                                              traceSent       [1] IMPLICIT BOOLEAN }
                    ::= 2
```

3.1.2 routeTrace Event

The routeTrace Event reports trace information. Trace information consists of one or more Point Codes, such as the entire list of Translation Point Codes traversed along a route. This event is invoked either at the explicit request of the originating node (indicated by traceRequested, see 3.1.1.1.2) or by failure at any point along the route. This event is not confirmed, therefore no replies to this invocation are expected (no error or success indications are expected).

routeTrace EVENT	*Timer* = 0	*Class* = 4	*Code* = 00000010
EventInfo		*Opt/Man* (Note)	*References*
success		O	3.1.2.1.1
detectedLoop		O	3.1.2.1.2
excessiveLengthRoute		O	3.1.2.1.3
unknownObjectInstance		O	3.1.2.1.4
routeInaccessible		O	3.1.2.1.5
processingFailure		O	3.1.2.1.6
unknownInitiatingSP		O	3.1.2.1.7
timerExpired		O	3.1.2.1.8
wrongSP		O	3.1.2.1.9
incorrectTranslation-Primary		O	3.1.2.1.10
incorrectTranslation-Secondary		O	3.1.2.1.11
incorrectTranslation-Intermediate		O	3.1.2.1.12
notPrimaryDestination		O	3.1.2.1.13
notSecondaryDestination		O	3.1.2.1.14
notRecognizedPrimary		O	3.1.2.1.15
notRecognizedSecondary		O	3.1.2.1.16
routingProblem		O	3.1.2.1.17
Note – One and only one of these indications must be present.			

```
routeTrace      EVENT
                EVENTINFO CHOICE {
                success                          [0] IMPLICIT PointCodeList,
                detectedLoop                     [1] IMPLICIT PointCodeList,
                excessiveLengthRoute             [2] IMPLICIT PointCodeList,
                unknownObjectInstance            [3] IMPLICIT NULL,
                routeInaccessible                [4] IMPLICIT PointCode,
                processingFailure                [5] IMPLICIT NULL,
                unknownInitiatingSP              [6] IMPLICIT PointCode,
                timerExpired                     [7] IMPLICIT PointCodeList,
                wrongSP                          [8] IMPLICIT PointCodeList,
                incorrectTranslation-Primary     [9] IMPLICIT PointCodeList,
                incorrectTranslation-Secondary   [10] IMPLICIT PointCodeList,
                incorrectTranslation-Intermediate [11] IMPLICIT PointCodeList,
                notPrimaryDestination            [12] IMPLICIT PointCodeList,
                notSecondaryDestination          [13] IMPLICIT PointCodeList,
                notRecognizedPrimary             [14] IMPLICIT PointCodeList,
                notRecognizedSecondary           [15] IMPLICIT PointCodeList,
                routingProblem                   [16] IMPLICIT PointCodeList }
                ::= 2
```

3.1.2.1 Event Information

3.1.2.1.1 success

On successful completion, the trace of the Point Codes (one or more) of the crossed SCCP Relay Nodes are included.

Parameter	*Code*
success	10100000
Contents	*Reference*
Sequence of one or more Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.2 detectedLoop

When a loop is detected, the Point Codes (three or more) contained in the loop are included.

Parameter	*Code*
detectedLoop	10100001
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.3 excessiveLengthRoute

When an excessively long route is found (threshold exceeded), the entire route is included.

Parameter	*Code*
excessiveLengthRoute	10100010
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.4 unknownObjectInstance

If the object is unknown, no additional information is required. For the SRVT, this refers to the case when no translation data exists for the GTI + GT.

Parameter	*Code*
unknownObjectInstance	10000011
Contents	*Reference*
empty	–

3.1.2.1.5 routeInaccessible

The Point Code of the node where the route was inaccessible is included.

Parameter	*Code*
routeInaccessible	10000100
Contents	*Reference*
Bit 0 contains the first bit of the Point Code, Bit 1 contains the second bit of the Point Code, etc.	3.1.1.1.1

3.1.2.1.6 processingFailure

If a processing failure occurs, no additional information is required.

Parameter	*Code*
processingFailure	10000101
Contents	*Reference*
empty	–

3.1.2.1.7 unknownInitiatingSP

The Point Code of the node detecting the unknown initiating Signalling Point is included.

Parameter	*Code*
unknownInitiatingSP	10000110
Contents	*Reference*
Bit 0 contains the first bit of the Point Code, Bit 1 contains the second bit of the Point Code, etc.	3.1.1.1.1

3.1.2.1.8 timerExpired

The Point Code(s) of the node(s) from where no result for the testRoute Action was received is included.

Parameter	*Code*
timerExpired	10100111
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.9 wrongSP

The complete list of Translation SPs traversed in the route to the invalid SP is included.

Parameter	*Code*
wrongSP	10101000
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.10 incorrectTranslation-Primary

The complete list of Translation SPs traversed in the route to the incorrect primary destination is included.

Parameter	*Code*
incorrectTranslation-Primary	10101001
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.11 incorrectTranslation-Secondary

The complete list of Translation SPs traversed in the route to the incorrect secondary destination is included.

Parameter	*Code*
incorrectTranslation-Secondary	10101010
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.12 incorrectTranslation-Intermediate

The complete list of Translation SPs traversed in the route to the incorrect intermediate point is included.

Parameter	*Code*
incorrectTranslation-Intermediate	10101011
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.13 notPrimaryDestination

The complete list of Translation SPs traversed in the route to the invalid primary destination is included.

Parameter	*Code*
notPrimaryDestination	10101100
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.14 notSecondaryDestination

The complete list of Translation SPs traversed in the route to the invalid secondary destination is included.

Parameter	*Code*
notSecondaryDestination	10101101
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.15 notRecognizedPrimary

The complete list of Translation SPs traversed in the route to the secondary destination is included.

Parameter	Code
notSecondaryDestination	10101110
Contents	References
Sequence of Point Codes, Tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.16 notRecognizedSecondary

The complete list of Translation SPs traversed in route to the primary destination is included.

Parameter	*Code*
notRecognizedSecondary	10101111
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

3.1.2.1.17 routingProblem

The complete list of Translation SPs traversed in the route to the possible routing problem. This occurs when the Point Code from the translation is not recognized.

Parameter	*Code*
routingProblem	10110000
Contents	*Reference*
Sequence of Point Codes, tagged as "PointCode", with contents indicating the exact Point Code	3.1.1.1.4

4 Circuit management

4.1 Circuit Validation Test (CVT) ASE

The Circuit Validation Test ASE provides the services accessed via om-Action-Confirmed as described in Figure 3. lt uses an instance based on a Circuit Management Object Class defined in Recommendation Q.751. The BaseManagedObjectClass indicates cvt-Cic-Tables-1992, and the BaseManagedObjectInstance identifies the CktGrpInfo (circuit group information identifying the pre-defined identifier for the circuit and its group agreed between the exchanges at the circuit group's ends) and its CIC known in the sending SP.

4.1.1 cktValidTest CnfAction

The Circuit Validation Test Request and subsequent return of the Circuit Validation Response is mapped into a confirmed action. The action is the request for the far-end test.

cktValidTest CNF-ACTION	*Timer* = T_c	*Class* = 1	*Code* = 00000001
ActionArg		*Opt/Man*	*References*
requestingSP		M	4.1.1
timer		O	4.1.1
ActionResult			*Reference*
success			4.1.1
Linked Operations			
N/A			
Specific Errors			
failure			

See 4.2/Q.753 for the possible values of T_c and timer.

```
cktValidTest                CNF-ACTION
ACTIONARG SEQUENCE {
                            requestingSP        RequestingSP,
                            timer               Timer OPTIONAL }
                            ACTIONRESULT                        Success
                            SPECIFICERRORS                      { failure }
                            ::= 3
```

4.1.2 Action Arguments

The requesting SP is the Point Code of the signalling point initiating the test procedure. It is of type Octet String as stated below.

```
RequestingSP ::= OCTET STRING
```

4.1.3 Action Results

The Action Results are returned in a return result component upon success. The contents of the two parameters are to be defined based on the CVT procedure.

```
Success         ::= SEQUENCE {
                cktGrpInfo      [0] IMPLICIT CktGrpInfo,
                cICName         [1] IMPLICIT OCTET STRING OPTIONAL
                }
```

Note that CktGrpInfo is defined as OCTET STRING.

4.1.4 Specific Error

The specific error indicates the failure and reason for failure. The contents of the two parameters are to be defined based on the CVT procedure.

```
failure         SPECIFIC-ERROR
                PARAMETER SEQUENCE {
                        cktGrpInfo      [0] IMPLICIT CktGrpInfo,
                        cICName         [1] IMPLICIT OCTET STRING OPTIONAL
                }
```

Note that CktGrpInfo is defined as OCTET STRING.

The CVT failure reasons are:

a) CIC not assigned at near end;

b) wrong near end data for circuit;

c) valid tone not received at near end;

d) overall test timer T_c expired before CVR received;

e) CVR message received before synchronization achieved in bit pattern test;

f) T_c expired before synchronization is achieved in bit pattern test;

g) bit pattern still being received when T_c expires;

h) bit pattern still being received when CVR message is received;

i) tone being received when T_c expires;

j) tone being received when CVR message is received;

k) near end CIC and far end CIC do not match (near end check on receipt of CVR message);

l) CVR message received indicating failure:

- CIC not assigned at far end;
- wrong far end data for circuit;
- group characteristics unavailable at far end;

m) failure – unspecified.

5 Transaction capabilities

For further study.

6 General definitions

6.1 Objects and operations

OMAP runs tests on objects such as the MTP and SCCP routing tables. These objects are described here as "Object Classes" and are identified by an object identifier which specifies this Recommendation and the type of object. This structure is shown below for the OMAP object identifiers mtp-Routing-Tables, sccp-Routing-Tables, and cvt-Cic-Tables.

oMAP	OBJECT IDENTIFIER ::= { ccitt recommendation q 754 }
mtp-Routing-Tables-1992	OBJECT IDENTIFIER ::= { oMAP 0 }
sccp-Routing-Tables-1992	OBJECT IDENTIFIER ::= { oMAP 1 }
cvt-Cic-Tables-1992	OBJECT IDENTIFIER ::= { oMAP 5 }

The Object Class of MTP Routing Tables is 0011857200 (hexadecimal), for SCCP Routing Tables is 0011857201 (hexadecimal), and for CVT CIC Tables is 0011857205 (hexadecimal). See Annex C/X.208 and 22/X.209.

Tables 1 and 2 show the OM-primitives, Figure 1 shows the OMAP operations derived from CMIP (IS 9596), and Figure 3 contains a formal syntax of OMASE.

The operations derived from CMIP are listed below.

Currently Defined Operations	
0	eventReport
7	confirmedAction

6.2 Primitives and procedures of the OMASE protocol

6.2.1 General

The OMASE protocol uses the TC-service as defined in Recommendation Q.771. The Invoke ID and the Dialogue ID correspond with those defined for the TC-service.

OMASE is modelled using a Protocol Machine (referred-to as OMPM below). The abbreviation APDU stands for Application Protocol Data Unit in what follows, it refers to the contents of the primitive(s) passed between OMASE and TC.

Figure A.1 shows the model including TC and SCCP, the OMPM resides in OMASE. Figure A.2 shows an example of particular instances of the primitives in an MRV Test (but without OM-EVENT-REPORT).

6.2.2 OM-EVENT-REPORT

6.2.2.1 Service primitive

The OM-EVENT-REPORT primitive used between the OMASE-User and OMASE is defined in Table 1.

The specific event that occurred is interpreted in the context of the object class specified.

TABLE 1/Q.754

OM-EVENT-REPORT Parameters

Parameter Name	Req/Ind
CallingPartyAddress	M
CalledPartyAddress	M
DialogueID	M
InvokelD	M
ManagedObjectClass	M
ManagedObjectInstance	M
EventType	M
EventTime	O
EventInfo	O

Parameter definitions

CallingPartyAddress: As defined in the calling address of 2.2/Q.711.

CalledPartyAddress: As defined in the called address of 2.2/Q.711. The above addresses serve to identify OMAP at the calling and called SP respectively. For MRVT they can both be in the form of point code plus (OMAP) subsystem number, for SRVT they are in a form suitable for the type of SCCP routing applied in the test.

DialogueID: As defined in Recommendations Q.771-Q.775. It maps to Transaction ID. defined in Recommendation Q.772.

InvokeID: As defined in Recommendation Q.772.

ManagedObjectClass: Identifies the class of objects for which this event is defined.

ManagedObjectInstance: Identifies the object instance on which the event is reported.

EventType: Specifies the particular event that is being reported by the object instance.

EventTime: Specifies the time at which the event was generated.

EventInfo: Provides additional event specific information.

6.2.2.2 Event reporting procedure

6.2.2.2.1 Receipt of OM-EVENT-REPORT request

The event reporting procedure is initiated by receipt of the OM-EVENT-REPORT request primitive. When this occurs, the OMPM constructs the APDU requesting the eventReport operation, and transmits the APDU using the TC-INVOKE and TC-BEGIN service.

The TC-INVOKE request primitive contains the following parameters and values:

- Dialogue ID – Defined by the OMASE-User.
- Invoke ID – Defined by the OMASE-User.
- Operation – Set to eventReport.
- Class – Set to 4.
- Parameters – Those following the word "PARAMETER" in the definition of eventReport. The value of the parameter eventType specifies which action is to be performed – it should indicate routeTrace for the procedures defined at present.
- Timeout – Set to 0 for both MRVT and SRVT.

The TC-BEGIN request primitive uses the following parameters and values:

- Destination address – As received in the OM-EVENT-REPORT request primitive CalledPartyAddress.
- Originating address – As received in the OM-EVENT-REPORT request primitive CallingPartyAddress.
- Dialogue ID – As in the TC-INVOKE.

The N-UNITDATA request primitive ultimately issued to the SCCP due to the receipt of these TC request primitives should contain the Sequence control parameter set to indicate "sequence guaranteed", and the Return option parameter should be set to indicate "discard message on error". See 2.2.2/Q.711.

After transmission of the APDU, the OMPM terminates the dialogue using the TC-END request primitive with parameters Dialogue ID and Termination, the latter indicating "prearranged end".

6.2.2.2.2 Receipt of TC-BEGIN with TC-INVOKE indication

On receipt of a well-formed APDU requesting the eventReport operation from the TC-BEGIN and TC-INVOKE indication primitives, the OMPM issues an OM-EVENT-REPORT indication primitive. If the APDU is not well-formed, the OMPM discards it.

The OMPM terminates the dialogue with a TC-END request primitive with parameters Dialogue ID and Termination, the latter indicates "prearranged end".

6.2.2.2.3 Receipt of TC-BEGIN with TC-L-REJECT indication

In this case, the OMPM issues a TC-END request primitive with parameters Dialogue ID and Termination, the latter indicating "prearranged end".

6.2.2.2.4 Receipt of TC-P-ABORT indication

In this case, the OMPM ignores the TC-P-ABORT.

6.2.3 OM-CONFIRMED-ACTION

6.2.3.1 Service primitive

The OM-CONFIRMED-ACTION service is shown in Table 2. The specific action to be performed is interpreted in the context of the object class specified. This service is a confirmed service (a report of success or failure is always sent).

TABLE 2/Q.754

OM-CONFIRMED-ACTION Service

Parameter Name	Req/Ind	Res/Con
CallingPartyAddress	M	M
CalledPartyAddress	M	M
DialogueID	M	M
InvokeID	M	M
AccessControl	O	–
BaseManagedObjectClass	M	–
BaseManagedObjectInstance	M	–
ActionInfo	M	–
ActionResult	–	M[a)]
ActionError	–	M[b)]
Timer	M[c)]	–

a) Mandatory in Return Result component (may be empty).

b) Mandatory in Return Error component.

c) This parameter is only in the request primitive.

Parameter definitions

CallingPartyAddress: See Table 1.

CalledPartyAddress: See Table 1.

DialogueID: Mapped by TCAP into transaction ID as defined in Recommendation Q.772.

InvokeID: As defined in Recommendation Q.772.

AccessControl: Information to be used as input to access control functions.

BaseManagedObjectClass: Identifies the class of objects for which this action is defined.

BaseManagedObjectInstance: Identifies the object instance on which the action is to be performed.

ActionInfo: Is a sequence of ActionType and (optional) ActionInfoArg. ActionType is defined by the CNF-ACTION macro, and specifies a particular action that is to be performed on the object instance. ActionInfoArg contains the parameters for the action to be performed.

ActionResult: This field contains the result of the successful action performed, as appropriate.

ActionError: This field indicates error or problem status information if the action did not successfully complete.

Timer: This parameter contains the particular value for the timeout period waiting for the response. It is set to T_1 for MRVT, T_2 for SRVT, or T_c for CVT.

The value is given in Recommendation Q.753.

6.2.3.2 Procedures for confirmed action

6.2.3.2.1 Receipt of OM-CONFIRMED-ACTION request

The confirmedAction procedure is initiated by the receipt of the OM-CONFIRMED-ACTION request primitive. In this case, the OMPM constructs an APDU requesting the confirmedAction operation, and transmits the APDU using the TC-INVOKE and TC-BEGIN service.

The TC-INVOKE request primitive contains the following parameters and values:

- Operation – Takes the value of confirmedAction.
- Class – Value is 1.
- Parameters – Corresponds with the parameters of confirmedAction as defined after the keyword "PARAMETER" of the operation definition. The value "testRoute" is obtained by derivation from CNF-ACTION of the localForm of ActionTypeID from ActionType of ActionInfo.
- Timeout – Is copied from the parameter "Timer" in the OM-CONFIRMED-ACTION request.
- The Invoke ID and Dialogue ID are copied from the OM-CONFIRMED-ACTION request.

The TC-BEGIN request primitive uses the following parameters and values:

- Dialogue ID – As in the TC-INVOKE.
- Destination address – The CalledPartyAddress of the OM-CONFIRMED-ACTION request.
- Originating address – The CallingPartyAddress of the OM-CONFIRMED-ACTION request.

The N-UNITDATA request primitive ultimately issued to the SCCP due to the receipt of these TC request primitives should contain the Sequence control parameter set to indicate "sequence not guaranteed", and the Return option parameter should be set to indicate "return message on error". See 2.2.2/Q.711.

6.2.3.2.2 Receipt of TC-BEGIN with TC-INVOKE indication

In this case, if the APDU is well-formed and requests the confirmedAction operation, the OMPM issues an OM-CONFIRMED-ACTION indication primitive to the OMASE-User.

If the APDU is not well formed, the OMPM ignores the TC indications.

An implementation-dependent mechanism allied to that defined in 3.3.4/Q.774 should cater for any local problems.

If the APDU contains extra parameters, they are ignored by the OMPM.

6.2.3.2.3 Receipt of OM-CONFIRMED-ACTION response

The OM-CONFIRMED-ACTION response primitive can contain either the ActionResult parameter, or the ActionError parameter.

The ActionResult parameter indicates that the execution of the operation was successful, and the OMPM issues a TC-RESULT-L request primitive. If the test was a CVT, the following parameters are included in the TC-RESULT-L:

- Operation – Has the value of confirmedAction.
- Parameters – Corresponds to the parameter Success of ACTIONRESULT for the CVT.

The presence of the ActionError parameter indicates that the operation was unsuccessful, and the OMPM issues a TC-U-ERROR request primitive with the following parameters:

- Error – Takes the appropriate error value from the set defined after the word "ERRORS" of the operation definition.
- Parameters – Corresponds with the parameters defined after the word "PARAMETER" of the definition of the error.

The result of the operation is transmitted by the OMPM issuing a TC-END request with parameters Dialogue ID and Termination, the latter indicating "basic end".

The N-UNITDATA request primitive ultimately issued to the SCCP due to the receipt of these TC request primitives should contain the Sequence control parameter set to indicate "sequence guaranteed", and the Return option parameter should be set to indicate "discard message on error". See 2.2.2/Q.711.

6.2.3.2.4 Receipt of TC-END with TC-RESULT-L indication

In this case, the OMPM if the APDU is well-formed, issues an OM-CONFIRMED-ACTION confirm primitive with parameter ActionResult to the OMASE-User (including the Dialogue ID).

If the APDU is not well formed, the OMPM ignores the TC primitives[3)].

6.2.3.2.5 Receipt of TC-END with TC-U-ERROR indication

If the APDU is well-formed, the OMPM issues an OM-CONFIRMED-ACTION confirm primitive with parameter ActionError (and Dialogue ID) to the OMASE-User.

If the APDU is not well formed, the OMPM ignores the TC primitives[3)].

6.2.3.2.6 Receipt of TC-L-CANCEL indication

This occurs if the invocation timer expires.

In this case, the OMPM issues an OM-CONFIRMED-ACTION confirm primitive, with the specific error "failure" for CNF-ACTION, and if the operation invoked was testRoute, the parameter failureType indicates timerExpired.

The OMPM terminates the dialogue with a TC-END request primitive, with the Termination parameter indicating "prearranged end".

6.2.3.2.7 Receipt of TC-BEGIN or TC-END with TC-L-REJECT indications

Receipt of TC-BEGIN with TC-L-REJECT indication is illustrated in Figure 2a.

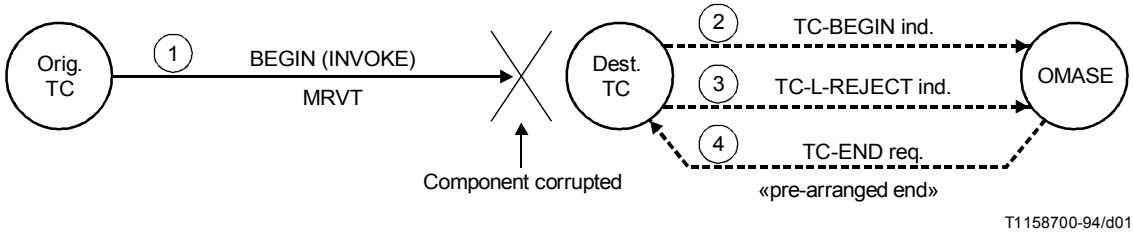

FIGURE 2a/Q.754

If the OMPM receives a TC-L-REJECT indication with a TC-BEGIN indication, the OMPM terminates the dialogue by issuing a TC-END request primitive with the Termination parameter indicating "prearranged end".

If the OMPM receives a TC-L-REJECT indication with a TC-END indication, it issues an OM-CONFIRMED-ACTION confirm primitive with the specific error "failure" of CNF-ACTION, and if testRoute was invoked, the parameter failureType in the confirm primitive indicates "routeInaccessible". This is illustrated in the Figure 2b.

6.2.3.2.8 Receipt of TC-END with TC-R-REJECT indication

In this case, the OMPM issues an OM-CONFIRMED-ACTION confirm primitive with the specific error "failure" of CNF-ACTION, and if testRoute was invoked, the parameter failureType in the confirm primitive indicates "routeInaccessible".

6.2.3.2.9 Receipt of TC-P-ABORT indication

This is illustrated by the two diagrams of Figure 2c.

---

3) The use of the overall guard timer in the OMASE-User at the test initiator node enables the test to fail gracefully in this circumstance.

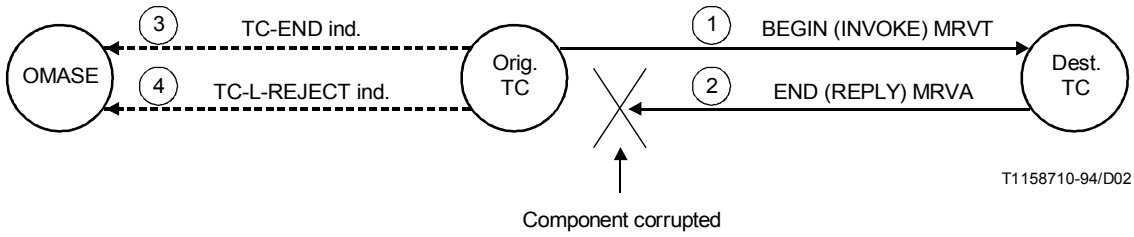

FIGURE 2b/Q.754

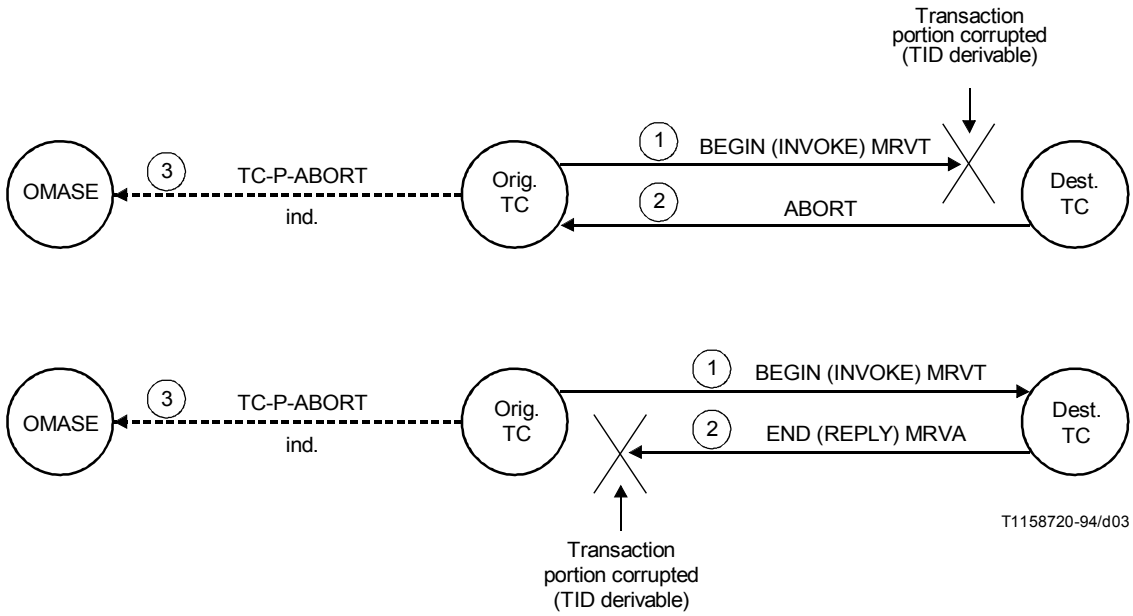

FIGURE 2c/Q.754

In this case, the OMPM issues an OM-CONFIRMED-ACTION confirm primitive with the specific error "failure" of CNF-ACTION, and if testRoute was invoked, the parameter failureType in the confirm primitive indicates "routeInaccessible".

6.2.3.2.10 Receipt of TC-NOTICE

In this case, the OMPM issues an OM-CONFIRMED-ACTION confirm primitive with the specific error "failure" of CNF-ACTION, and if testRoute was invoked, the parameter failureType in the confirm primitive indicates "routeInaccessible".

Error definitions

A number of errors have been referred-to in the definition of the two OM-services. These errors are defined in this subclause.

Definitions

noSuchObjectClass: The object class in the Invoke APDU is not recognised by the receiving end.

noSuchObjectInstance: While the object class in the Invoke APDU is recognised, there is no corresponding object instance of that class at the receiving end.

accessDenied: Access to the resource is not allowed.

processingFailure: A failure occurred while processing a specific action or event. The failure indicators are action or event specific.

noSuchAction: The action type is not supported by or known to the receiving end.

noSuchArgument: The argument specified is not known to or supported by the receiving end.

invalidArgumentValue: The argument value is not appropriate for the receiving end.

6.3 Abstract syntax of the OMASE protocol

```
-- OMASE  protocol
OMASE { ccitt recommendation q 754 omase(0) version1(1) }

DEFINITIONS ::=
BEGIN
-- TCAP definitions
EXPORTS EVENT, CNF-ACTION, SPECIFIC-ERROR;

IMPORTS OPERATION, ERROR FROM TCAPMessage { ccitt recommendation q 773 modules(0) messages(1)
version2(2) };
-- OMASE operations

        eventReport OPERATION

        PARAMETER SEQUENCE {

            managedObjectClass      ObjectClass,
            managedObjectInstance   ObjectInstance,
            eventTime               [5] IMPLICIT GeneralizedTime OPTIONAL,
            eventType               [7] IMPLICIT EVENT,
            eventInfo               [8] ANY DEFINED BY eventType OPTIONAL }

        ::= localValue 0

        confirmedAction OPERATION

        PARAMETER SEQUENCE {

                COMPONENTS OF       BaseManagedObjectId,
                accessControl       [5] AccessControl OPTIONAL,
                actionInfo          [12] IMPLICIT ActionInfo }

        RESULT actionResult    ActionResult

        ERRORS { accessDenied, invalidArgumentValue,
                noSuchAction, noSuchArgument,
                noSuchObjectClass, noSuchObjectInstance,
                processingFailure }

        ::= localValue 7
```

FIGURE 3/Q.754 (sheet 1 of 5)

Formal syntax of OMASE services

-- The om-service error definitions are as follows:

noSuchObjectClass ERROR
PARAMETER ObjectClass
::= localValue 0

noSuchObjectInstance ERROR
PARAMETER ObjectInstance
::= localValue 1

accessDenied ERROR
::= localValue 2

noSuchAction ERROR
PARAMETER NoSuchAction
::= localValue 9

processingFailure ERROR
PARAMETER ProcessingFailure -- *optional*
::= localValue 10

noSuchArgument ERROR
PARAMETER NoSuchArgument
::= localValue 14

invalidArgumentValue ERROR
PARAMETER InvalidArgumentValue
::= localValue 15

FIGURE 3/Q.754 (sheet 2 of 5)

Formal syntax of OMASE services

-- The following gives the supporting type definitions:

```
ActionArgument          ::= SEQUENCE  { COMPONENTS OF      BaseManagedObjectId,
                                        accessControl      [5] AccessControl OPTIONAL,
                                        actionInfo         [12] IMPLICIT ActionInfo }

ActionInfo              ::= SEQUENCE  { actionType         ActionTypeId,
                                        actionInfoArg      [4] ANY DEFINED BY actionType
                                                           OPTIONAL }

ActionResult            ::= SEQUENCE  { managedObjectClass ObjectClass
                                                           OPTIONAL,
                                        managedObjectInstance
                                                           ObjectInstance OPTIONAL,
                                        currentTime        [5] IMPLICIT GeneralizedTime
                                                           OPTIONAL,
                                        actionReply        [6] IMPLICIT ActionReply
                                                           OPTIONAL }

ActionTypeId            ::= CHOICE    {
                                        -- globalForm ...
                                        localForm          [3] IMPLICIT CNF-ACTION }

BaseManagedObjectId     ::= SEQUENCE  { baseManagedObjectClass
                                                           ObjectClass,
                                        baseManagedObjectInstance
                                                           ObjectInstance }

EventReportArgument     ::= SEQUENCE  { managedObjectClass ObjectClass,
                                        managedObjectInstance
                                                           ObjectInstance,
                                        eventTime          [5] IMPLICIT GeneralizedTime
                                                           OPTIONAL,
                                        eventType          EventTypeId,
                                        eventInfo          [8] ANY DEFINED BY eventType
                                                           OPTIONAL }

EventTypeId             ::= CHOICE    {
                                        -- globalForm ...
                                        localForm          [7] IMPLICIT EVENT }
```

FIGURE 3/Q.754 (sheet 3 of 5)

Formal syntax of OMASE services

```
InvalidArgumentValue    ::= CHOICE      { actionValue                   [0] IMPLICIT ActionInfo,
                                          eventValue                    [1] IMPLICIT SEQUENCE {
                                          eventType                     EventTypeId,
                                          eventInfo                     [8] ANY DEFINED BY eventType
                                                                        OPTIONAL }}

NoSuchAction            ::= SEQUENCE    { managedObjectClass            ObjectClass,
                                          actionType                    ActionTypeId }

NoSuchArgument          ::= CHOICE      { actionId                      0] IMPLICIT SEQUENCE {
                                          managedObjectClass            ObjectClass OPTIONAL,
                                          actionType                    ActionTypeId },
                                          eventId                       [1] IMPLICIT SEQUENCE {
                                          managedObjectClass            ObjectClass OPTIONAL,
                                          eventType                     EventTypeID }}

ObjectClass             ::= CHOICE      { globalForm                    [0] IMPLICIT
                                                                        OBJECT IDENTIFIER,

                                          -- ...
                                          }

ObjectInstance          ::= CHOICE      {
                                          -- ...
                                          nonSpecificForm               [3] IMPLICIT OCTET STRING,
                                          -- ...
                                          }

ProcessingFailure       ::= SEQUENCE    { managedObjectClass            ObjectClass
                                                                        OPTIONAL,
                                          managedObjectInstance         ObjectInstance OPTIONAL,
                                          specificErrorInfo             [5] IMPLICIT SpecificErrorInfo }

SpecificError           ::= INTEGER     -- defined by object class

SpecificErrorInfo       ::= SEQUENCE    { errorType                     [0] IMPLICIT SpecificError,
                                          errorParm                     [1] ANY DEFINED BY errorType
                                                                        OPTIONAL }

Timer                   ::= INTEGER     -- seconds
```

FIGURE 3/Q.754 (sheet 4 of 5)

Formal syntax of OMASE services

```
-- Specific event reports are categorised by object class. The protocol uses may be described
-- by the EVENT MACRO below.

        EVENT MACRO ::=

        BEGIN

            TYPE NOTATION       ::=   EventInfo
            VALUE NOTATION      ::=   value(VALUE INTEGER)
            EventInfo           ::=   "EVENTINFO" NamedType | empty
            NamedType           ::=   identifier type | type

        END
-- Specific Actions are categorised by object class. The protocol uses may be described
-- by the CNF-ACTION macro below.

        CNF-ACTION MACRO ::=

        BEGIN

            TYPE NOTATION       ::=   ActionArg ActionResult SpecificErrors
            VALUE NOTATION      ::=   value(VALUE INTEGER)
            ActionArg           ::=   "ACTIONARG" NamedType | empty
            ActionResult        ::=   "ACTIONRESULT" NamedType | empty
            SpecificErrors      ::=   "SPECIFICERRORS" "{" SpecificErrorList"}" | empty
            NamedType           ::=   identifier type | type
            SpecificErrorList   ::=   SpecificError | SpecificErrorList","SpecificError
            SpecificError       ::=   value(SPECIFIC-ERROR)

        END
-- Errors that are action or event specific are defined using the SPECIFIC-ERROR macro below.

            SPECIFIC-ERROR MACRO ::=

            BEGIN

                TYPE NOTATION       ::=   ProcessingErrorParm
                VALUE NOTATION      ::=   value(VALUE INTEGER)
                ProcessingErrorParm ::=   «PARAMETER» NamedType | empty
                NamedType           ::=   identifier type | type

            END

END -- protocole OMASE
```

FIGURE 3/Q.754 (sheet 5 of 5)

Formal syntax of OMASE services

Annex A

(This annex forms an integral part of this Recommendation)

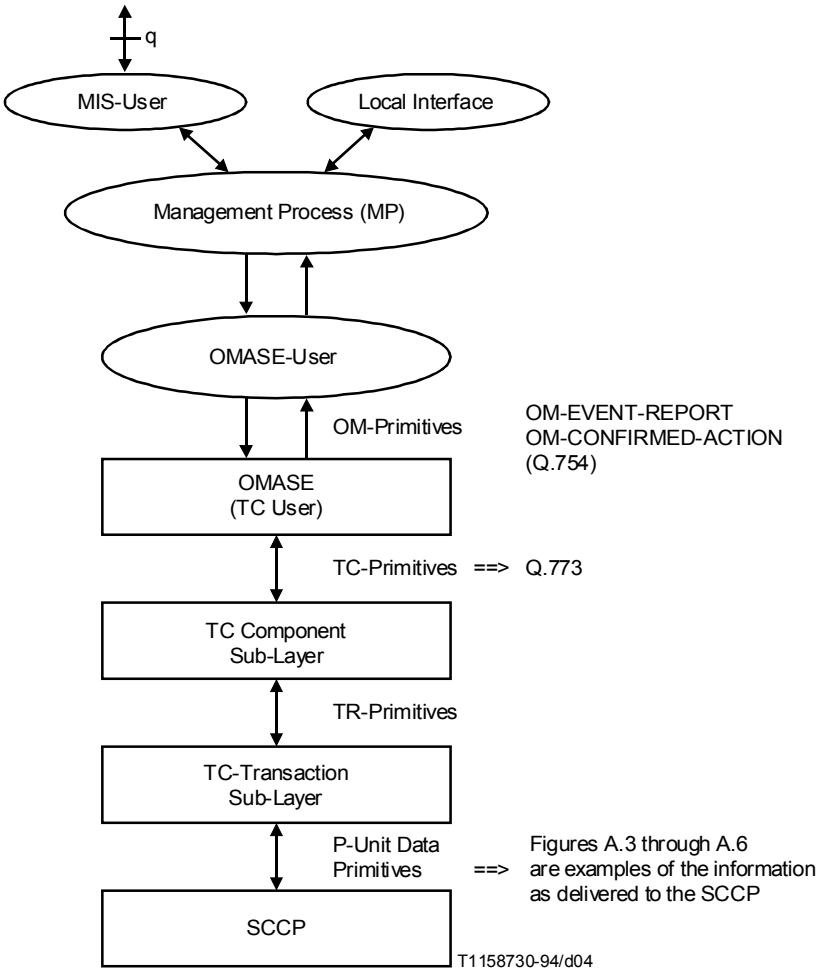

FIGURE A.1/Q.754

Primitive Interfaces

Figure A.2 illustrates the use of the primitives in an MRV Test. The OMASE-User at the origin, on receiving a "sendMRVT" request from the Management Process (MP – see Recommendation Q.753 for the model used), constructs an OM-CONFIRMED-ACTION request. The sequence is then as shown by the primitive and message sequence, up to number 5. At this point, if the node is not the tested destination, the OMASE-User receiving the OM-CONFIRMED-ACTION indication requests OM-CONFIRMED-ACTION of OMASE, to send out MRVT messages on all routes to the tested destination in the routing table. When all MRVA messages are received (seen in the OMASE-User as OM-CONFIRMED-ACTION confirm primitives), the OMASE-User issues the OM-CONFIRMED-ACTION response primitive as at 6.

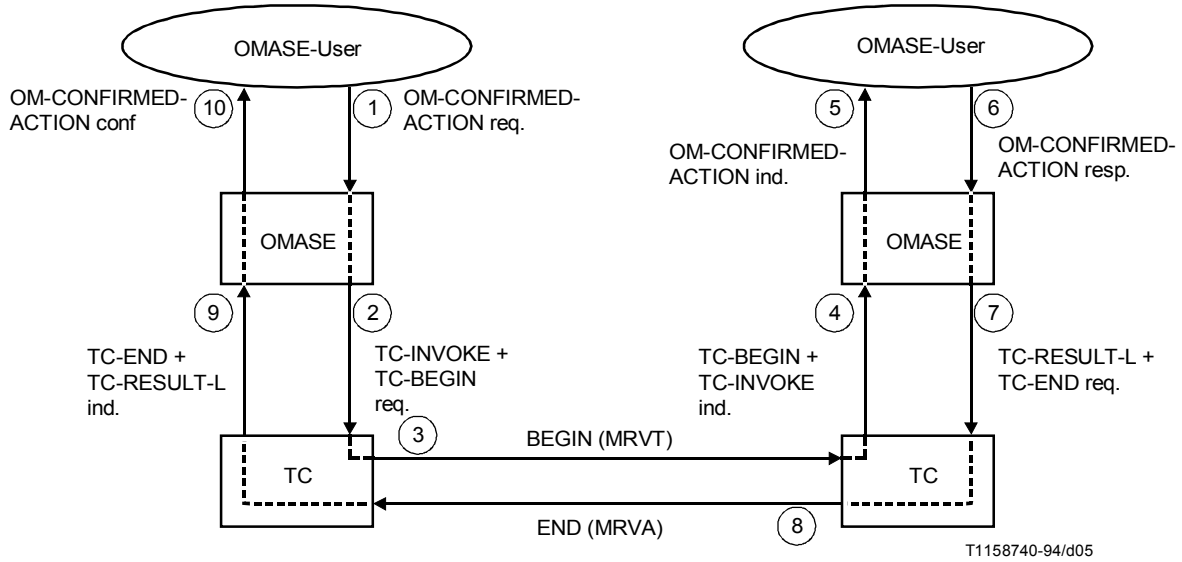

FIGURE A.2/Q.754

Example use of primitive interfaces

Field Name	Bit Encoding	Reference/Explanation
Message Type Tag	01100010	= Begin (Table 8/Q.773)
Message Length	00110010	50 octets following TC part
Transaction ID Tag	01001000	= Originating (Table 10/Q.773)
Length	00000100	4 octets
Transaction ID Value	xxxxxxxx	TCAP based on a dialogue
	xxxxxxxx	at the user level
	xxxxxxxx	
	xxxxxxxx	
Component Portion Tag	01101100	(Table 14/Q.773)
Length	00101010	All 42 octets below here
Component Type Tag	10100001	= Invoke (Table 19/Q.773)
Length	00101000	All 40 octets below here
Component ID Tag	00000010	= Invoke ID (Table 20/Q.773)
Length	00000001	1 octet
Invoke ID Value	xxxxxxxx	OMAP PROVIDED
Operation Code Tag	00000010	= Local (Table 22/Q.773)
Length	00000001	1 octet
Operation Code	00000111	= Confirmed Action (Figure 3/Q.754)
Parameter Sequence Tag	00110000	= Sequence Tag (Table 23/Q.773)
Length	00100000	All 32 octets below here
Object Class Tag	10000000	globalForm X.711 & X.209
Length	00000101	5 octets
Value-MTP Routing Tables	00000000	CCITT, Rec.
	00010001	q
	10000101	85 => 754
	01110010	72 =>
	00000000	MTP Routing Tables 1992
Object Instance Tag	10000011	nonSpecificForm X.711 & X.209
Length	00000010	2 octets
Object Instance Value	xxxxxxxx	(OMAP) Tested destination
	xxxxxxxx	
Action Info Tag	10101100	X.711 & X.209
Length	00010011	All 19 octets below here
Action Type Tag	10000011	localForm X.711 & X.209
Length	00000001	1 octet
CNF-ACTION	00000001	= testRoute (8.1.1/Q.754)
Action Info Arg Tag	10100100	X.711 & X.209
Length	00001110	All 14 octets below here
Parameter Seq. Tag	00110000	= Sequence Tag (Table 23/Q.773)
Length	00001100	All 12 octets below here
Initiating SP Tag	10000000	8.1.1.1.1/Q.754, X.209
Length	00000010	2 octets
Initiating SP Value	xxxxxxxx	(OMAP) test initiator
	xxxxxxxx	
Trace Req. Tag	10000001	8.1.1.1.2/Q.754, X.209
Length	00000001	1 octet
Value	00000001	= TRUE
Threshold Tag	10000010	= threshold (8.1.1.1.3/Q.754)
Length	00000001	1 octet
Value of threshold	xxxxxxxx	OMAP PROVIDED
Point Code Trav. Tag	10100011	8.1.1.1.4/Q.754
Length	00000000	empty Point Code list

FIGURE A.3/Q.754

Example of an MRVT message delivered to the SCCP

Field Name	Bit Encoding	Reference/Explanation
Message Type Tag	01100100	= END (Table 8/Q.773)
Message Length	00001101	13 octets following in TC part
Transaction ID Tag	01001001	= Destination (Table 10/Q.773)
Length	00000100	4 octets
Transaction ID Value	xxxxxxxx	Same as in BEGIN (MRVT msg)
	xxxxxxxx	
	xxxxxxxx	
	xxxxxxxx	
Component Portion Tag	01101100	(Table 14/Q.773)
Length	00000101	All 5 octets below here
Component Type Tag	10100010	= Ret. Res.(L) (Table 19/Q.773)
Length	00000011	All 3 octets below here
Component ID Tag	00000010	= Invoke ID (Table 20/Q.773)
Length	00000001	1 octet
Invoke ID Value	xxxxxxxx	Same as MRVT message (Correlation)

FIGURE A.4/Q.754

Example of an MRVA (success) message delivered to the SCCP

Field Name	Bit Encoding	Reference/Explanation
Message Type Tag	01100100	= END (Table 8/Q.773)
Message Length	00100000	32 octets following in TC part
Transaction ID Tag	01001001	= Destination (Table 10/Q.773)
Length	00000100	4 octets
Transaction ID Value	xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx	Same as in BEGIN (MRVT msg)
Component Portion Tag	01101100	(Table 14/Q.773)
Length	00011000	All 24 octets below here
Component Type Tag	10100011	= Ret. Error (Table 19/Q.773)
Length	00010110	All 22 octets below here
Component ID Tag	00000010	= Invoke ID (Table 20/Q.773)
Length	00000001	1 octet
Invoke ID Value	xxxxxxxx	Same as MRVT message (Correlation)
Error Code Tag	00000010	Table 24/Q.773 (local)
Length	00000001	1 octet
Processing Failure	00001010	Figure 3/Q.754
Parameter Sequence Tag	00110000	= Sequence Tag (Table 23/Q.773)
Length	00001110	All 14 octets below here
Specific Error Info Tag	10100101	Figure 3/Q.754
Length	00001100	all 12 octets below here
Error Type Tag	10000000	Figure 3/Q.754
Length	00000001	1 octet
Failure	00000001	2.1.1.3.1/Q.754
Error Parameters	10100001	2.1.1.3.1/Q.754
Length	00000111	All 7 octets below here
Failure Type Tag	10000000	2.1.1.3.1/Q.754
Length	00000010	2 octets
Unused Bits	00000000	no bits
Failure String	xxxxxxxx	Depends on type failure (2.1.1.3.1/Q.754)
Trace Sent Tag	10000001	2.1.1.3.1/Q.754
Length	00000001	1 octet
Trace Sent Value	0000000x	True = 1, False = 0 (2.1.1.3.1/Q.754)

FIGURE A.5/Q.754

Example of an MRVA (failure) message delivered to the SCCP

Field Name	Bit Encoding	Reference/Explanation
Message Type Tag	01100010	= BEGIN (Table 8/Q.773)
Message Length	00101100	44 octets following TC part
Transaction ID Tag	01001000	= Originating (Table 10/Q.773)
Length	00000100	4 octets
Transaction ID Value	xxxxxxxx	TCAP based on a dialogue at
	xxxxxxxx	the user level
	xxxxxxxx	
	xxxxxxxx	
Component Portion Tag	01101100	(Table 14/Q.773)
Length	00100100	All 36 octets below here
Component Type Tag	10100001	= Invoke (Table 19/Q.773)
Length	00100010	All 34 octets below here
Component ID Tag	00000010	= Invoke ID (Table 20/Q.773)
Length	00000001	1 octet
Invoke ID Value	xxxxxxxx	OMAP PROVIDED
Operation Code Tag	00000010	= Local (Table 22/Q.773)
Length	00000001	1 octet
Operation Code	00000000	= Event Report (Figure 3/Q.754)
Parameter Sequence Tag	00110000	= Sequence Tag (Table 23/Q.773)
Length	00011010	All 26 octets below here
Object Class Tag	10000000	(Figure 3/Q.754)
Length	00000101	5 octets
Value-MTP Routing Tables	00000000	CCITT, Rec.
	00010001	q
	10000101	85 => 754
	01110010	72 =>
	00000000	MTP Routing Tables 1992
Object Instance Tag	10000011	(Figure 3/Q.754)
Length	00000010	2 octets
Object Instance Value	xxxxxxxx	Terminating PC (OMAP)
	xxxxxxxx	<Tested Destination>
Event Type Tag	10000111	Figure 3/Q.754
Length	00000001	1 octet
Event Type	00000010	= routeTrace (2.1.2/Q.754)
Event Info Type Tag	10101000	Figure 3/Q.754
Length	00001010	All 10 octets below here
Success Identifier	10100000	2.1.2.1.1/Q.754
Length	00001000	All 8 octets below here
Point Code Tag	00000100	= OCTET STRING
Length	00000010	2 octets
Point Code	xxxxxxxx	
	xxxxxxxx	
Point Code Tag	00000100	= OCTET STRING
Length	00000010	2 octets
Point Code	xxxxxxxx	
	xxxxxxxx	

FIGURE A.6/Q.754

Example of an MRVR (success) message delivered to the SCCP

Printed in Switzerland

Geneva, 1994