INTERNATIONAL TELECOMMUNICATION UNION

ITU-T Q.754

(06/97)

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

SERIES Q: SWITCHING AND SIGNALLING

Specifications of Signalling System No. 7 – Signalling System No. 7 management

---

Signalling System No. 7 management Application Service Element (ASE) definitions

ITU-T Recommendation Q.754

(Previously CCITT Recommendation)

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.849
General	Q.700
Message transfer part (MTP)	Q.701–Q.709
Signalling connection control part (SCCP)	Q.711–Q.719
Telephone user part (TUP)	Q.720–Q.729
ISDN supplementary services	Q.730–Q.739
Data user part	Q.740–Q.749
Signalling System No. 7 management	**Q.750–Q.759**
ISDN user part	Q.760–Q.769
Transaction capabilities application part	Q.770–Q.779
Test specification	Q.780–Q.799
Q3 interface	Q.800–Q.849
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1999
BROADBAND ISDN	Q.2000–Q.2999

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION Q.754

SIGNALLING SYSTEM No. 7 MANAGEMENT APPLICATION SERVICE ELEMENT (ASE) DEFINITIONS

Summary

This Recommendation defines the application service element used by the management functions MRVT, SRVT and CVT defined in Recommendation Q.753. The ASE defines the management information used by these functions in messages across the SS No. 7 network.

The ASE interfaces with Transaction Capabilities (TCs) to provide the services used by the OMASE-User (defined in Recommendation Q.753), to allow inter-nodal SS No. 7 communication by MRVT, SRVT or CVT.

The main revisions to the 1993 version of this Recommendation are:

a) revision of the compatibility rules to allow transparent transport of unrecognized parameters;

b) definition of a new MRVT and SRVT message parameter to specify what information is required in any respective MRVR or SRVR message;

c) definition of a new MRVR and SRVR message to be used if information is required beyond that specified for the 1993 MRVR and SRVR messages;

d) definition of new MRVT and MRVR message parameters to indicate the route priorities;

e) definition of new MRVA, MRVR, SRVA and SRVR parameters to allow parameters unrecognized in the MRVT or SRVT messages to be returned;

f) definition of a new MRVT parameter requesting nodes to check if they have a route to the test initiator through the node from which they received the MRVT message (this allows checking for symmetrical routes).

Source

ITU-T Recommendation Q.754 was revised by ITU-T Study Group 11 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 5th of June 1997.

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. 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, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

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

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had/had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Recommendation Q.754

SIGNALLING SYSTEM No. 7 MANAGEMENT APPLICATION SERVICE ELEMENT (ASE) DEFINITIONS

(revised in 1997)

1 Introduction

Note that in the event of a conflict between Recommendations Q.753 and Q.754, Recommendation Q.754 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 (TCs) OPERATION and ERROR. 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 (CVTs). 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.

Messages between Signalling Points are encoded using the ASN.1 Basic Encoding Rules (BER), and octet string parameters are encoded as primitive (not constructed) elements.

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 or routeTraceNew event as a parameter.

---

[1] CMIP is defined in ISO/IEC 9596 and in Recommendation X.711.

[2] See Recommendations X.680 through X.683 and Recommendation X.690 for the description of the formal notation (also Recommendations X.208 and X.209).

testRoute is specified using the CNF-ACTION macro defined in Figure 3, routeTrace is specified using the EVENT macro defined in Figure 3. Figure 3 is the definition of the OMASE module, and defines all parameters used in the protocol.

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 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 in a sebsequent node.

testRoute CNF-ACTION	*Timer* = T1	*Class* = 1	*Code* = 00000001

See Figure 3.

2.1.1.1 testRoute Action Arguments

2.1.1.1.1 initiatingSP

The initiatingSP identifies the original requester 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.	

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 (SPs) 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

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.

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

2.1.1.1.5 routePriortyList

If the infoRequest parameter is included and requests it, as each SP is traversed it adds the priority of the route to the next SP into routePriorityList.

Parameter	*Code*
routePriorityList	10101100
Contents	
Sequence of Priority, tagged as "Priority" with the contents indicating unknown, or first choice, or second choice, etc.	

2.1.1.1.6 infoRequest

This optional parameter inserted, if at all, only by the SP initiating the test, indicates that the initiator can recognize MRVR messages occasioned by the routeTraceNew event type. The infoRequest parameter indicates which information is requested if an MRVR message should be sent to the initiator. It also can indicate which parameters should be updated as the MRVT messages traverse the network. Current values can be pointCode (bit 0 = 1), and/or pointCodeList (bit 1), and/or routePriorityList (bit 2).

Parameter	*Code*
infoRequest	10001101
Contents	
Bit string containing any or all of the indicated values.	

2.1.1.1.7 returnUnknownParams

This optional parameter is inserted, if at all, only by the SP initiating the test (and only if the infoRequest parameter is also included). It indicates which MRVT parameters a following node should return, if the following node does not recognize those parameters. The unrecognized MRVT parameters are to be copied into the (new) MRVR message (routeTraceNew) if the following node has occasion to return an MRVR (or in an MRVA message in the copyData parameter if the initiator is unknown to it). Bit 0 in returnUnknownParams indicates an MRVT parameter with tag value 15, bit 1 an MRVT parameter with tag value 16, etc.

Parameter	*Code*
returnUnknownParams	10001110
Contents	
Bit string containing any indicated values.	

2.1.1.1.8 directRouteCheck

This optional parameter indicates when TRUE that following nodes are requested to check that they have a route to the test initiator through the SP from which they received the prompting MRVT message.

Parameter	*Code*
directRouteCheck	10001111
Contents	
Boolean, TRUE indicates that the check is required.	

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-CONFIRMED-ACTION 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. 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 has type BOOLEAN. The third parameter is optional, it is present if failureType is "unknownInitiating SP" traceSent is FALSE, and the prompting MRVT message contained a requestInfo or a returnUnknownParams parameter (or both).

Specific Error	*Code*
failure	00000001

Parameter	*Code*
failureType	10000000

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

Parameter	*Code*
copyData	10000100
Contents	
An OCTET STRING containing parameters requested by the prompting MRVT message.	

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

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

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

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

2.1.2.1.3 excessiveLengthRoute

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

Parameter	*Code*
excessiveLengthRoute	10100010

2.1.2.1.4 unknownDestination

If the destination is unknown, no additional information is required, since the infoRequest parameter was not included in the testRoute CNF-ACTION request.

2.1.2.1.5 routeInaccessible

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

Parameter	*Code*
routeInaccessible	10000100

2.1.2.1.6 processingFailure

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

Parameter	*Code*
processingFailure	10000101

2.1.2.1.7 unknownInitiatingSP

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

Parameter	*Code*
unknownInitiatingSP	10000110

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

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.

The value "sPNotAnSTP" of failureType can also mean that the intermediate signalling point receiving an MRVT message is not authorized to transfer messages, received from the MRVT sender, with its MTP label OPC that of the test initiator and DPC that of the test destination.

Parameter	*Code*
sPNotAnSTP	10101000

2.1.2.1.10 maxNrMRVTestsAlready

This report is used by the signalling point receiving the MRVT message if the maximum number of MRV Tests n_T are already running at the SP. It is reported as "processingFailure", see 2.1.2.1.6, if the prompting MRVT message (testRoute) did not contain the infoRequest parameter.

2.1.3 routeTraceNew

This report is used if the prompting testRoute action contained an infoRequest parameter.

routeTraceNew EVENT	*Timer* = 0	*Class* = 4	*Code* = 00000100

2.1.3.1 Event information

2.1.3.1.1 success

On successful completion, the trace of the Point Codes (one or more) of the SPs crossed are included in pointCodeList (copied from the pointCodesTraversed parameter of the testRoute action).

routePriorityList is copied from the testRoute action, if present and requested in the infoRequest parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

2.1.3.1.2 detectedLoop

If a loop is detected, the point codes (3 or more) of the SPs in the loop are included in pointCodeList.

routePriorityList is copied from the testRoute action, if present and requested in the infoRequest parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

2.1.3.1.3 excessiveLengthRoute

If this error occurs, the entire route is copied from the testRoute action pointCodesTraversed parameter into the pointCodeList parameter.

routePriorityList is copied from the testRoute action, if present and requested in the infoRequest parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

2.1.3.1.4 unknownDestination

This is equivalent to unknownDestination of 2.1.2.1.4. If the infoRequest parameter of the prompting testRoute action requested it, the pointCodesTraversed parameter of the testRoute action is copied into pointCodeList.

routePriorityList is copied from the testRoute action, if present and requested in the infoRequest parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

2.1.3.1.5 routeInaccessible

If this event is reporting just one inaccessible SP, its point code is placed in pointCode.

If the event is reporting more than one inaccessible SP (and hence the prompting testRoute action indicated that the originator could accept it), the list of all such inaccessible SPs is put into pointCodeList.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

2.1.3.1.6 processingFailure

If the test cannot be run due to local conditions, the event reports a processing failure. This includes rejection of the testRoute action by SCCP or TC at a remote SP.

If the testRoute action infoRequest parameter was present, and had bit 0 set to 1, the point code of the SP where the test failed is put into the pointCode parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

2.1.3.1.7 unknownInitiatingSP

The point code of the SP detecting the unknown initiator is returned in the pointCode parameter.

If the prompting testRoute result contained a copyData parameter, this is copied into the routeTraceNew copyData parameter.

2.1.3.1.8 timerExpired

The point codes of the SPs from which no result of the testRoute actions were received are included into pointCodeList.

2.1.3.1.9 sPNotAnSTP

This error occurs if the intermediate SP does not have the STP function, or it is known that it is not authorized to transfer messages from the test initiator to the test destination.

The pointCodesTraversed parameter of the prompting testRoute action is copied into pointCodeList.

routePriorityList is copied from the testRoute action, if present and requested in the infoRequest parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

2.1.3.1.10 indirectRoute

This report is used if the direct route test was requested and the SP receiving the MRVT message did not have a route to the test initiator directly through the MRVT message sender. The identity of the prompting MRVT message sender is included in the pointCode parameter, the identity of the SP detecting no direct route is the OPC in the MRVR message's MTP label.

2.1.3.1.11 maxNrMRVTestsAlready

This report is used by the signalling point receiving the MRVT message if the maximum number of MRV Tests n_T are already running at the SP.

If the testRoute action infoRequest parameter was present, and had bit 0 set to 1, the point code of the SP where the test failed is put into the pointCode parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

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 CNF-ACTION defined in Figure 3, routeTrace is specified using EVENT 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 in a subsequent node.

testRoute CNF-ACTION	*Timer* = T2	*Class* = 1	*Code* = 00000001

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.	

3.1.1.1.2 traceRequested

traceRequested indicates that a trace of all routes used to reach the destination should be reported. It is of type BOOLEAN.

Parameter	*Code*
traceRequested	10000001
Contents	*Meaning*
TRUE (= 1) FALSE (= 0)	trace was requested, return trace information on success and failure. 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 (TSPs) 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

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.	

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.	

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 bits 3 through 6 = GlobalTitle Indicator. Octet 2, 3, ... = Initiator Global Title.	

3.1.1.1.8 destinationPC

The destinationPC 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.	

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 bits 3 through 6 = Global Title Indicator. Octet 2, 3, ... = Original Global Title.	

3.1.1.1.13 inputGT

The inputGT, used only in the SRVT compare form, identifies the Test GTI + GT prior to translation at a TSP. It is of type OCTET STRING.

Parameter	*Code*
inputGT	10010000
Contents	
Octet 1 bits 3 through 6 = Global Title Indicator. Octet 2, 3, ... = Input Global Title to TSP.	

3.1.1.1.14 infoRequest

This optional parameter inserted, if at all, only by the SP initiating the test, indicates that the initiator can recognize SRVR messages occasioned by the routeTraceNew event type. The infoRequest parameter indicates which information is requested if an SRVR message should be sent to the initiator. It also can indicate which parameters should be updated as the SRVT messages traverse the network. Current values can be pointCode (bit 0 = 1), and/or pointCodeList (bit 1).

Parameter	*Code*
infoRequest	10001101
Contents	
Bit string containing any or all of the indicated values.	

3.1.1.1.15 returnUnknownParams

This optional parameter is inserted, if at all, only by the SP initiating the test (and only if the infoRequest parameter is also included). It indicates which SRVT parameters a following node should return, if the following node does not recognize those parameters. The unrecognized SRVT parameters are to be copied into the (new) SRVR message (routeTraceNew) if the following node has occasion to return an SRVR (or in an SRVA message in the copyData parameter if the initiator is unknown to it). Bit 0 in returnUnknownParams indicates an SRVT parameter with tag value 15, bit 1 an SRVT parameter with tag value 16, etc.

Parameter	*Code*
returnUnknownParams	10001110
Contents	
Bit string containing any indicated values.	

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-CONFIRMED-ACTION 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

Parameter	*Code*
failureType	10000000

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

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

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

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

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

3.1.2.1.3 excessiveLengthRoute

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

Parameter	*Code*
excessiveLengthRoute	10100010

3.1.2.1.4 unknowndestination

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

Parameter	*Code*
unknownDestination	10000011

3.1.2.1.5 routeInaccessible

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

Parameter	*Code*
routeInaccessible	10000100

3.1.2.1.6 processingFailure

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

Parameter	*Code*
processingFailure	10000101

3.1.2.1.7 unknownInitiatingSP

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

Parameter	*Code*
unknownInitiatingSP	10000110

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

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

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

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

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

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

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

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

3.1.2.1.16 notRecognizedSecondary

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

Parameter	*Code*
notRecognizedSecondary	10101111

3.1.2.1.17 routingProblem

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

Parameter	*Code*
routingProblem	10110000

3.1.3 routeTraceNew Event

This report is used if the prompting testRoute action contained an infoRequest parameter.

routeTraceNew EVENT	*Timer* = 0	*Class* = 4	*Code* = 00000100

3.1.3.1 Event information

The information included is as for routeTrace, but with the "result" parameter indicating the result, and the data associated with the result included in the optional parameters. The copyData parameter contains, if present, parameters from the prompting testRoute action that were not understood, and that were requested to be returned by the returnUnknownParams in the testRoute action. One extra result over routeTrace is currently defined for routeTraceNew, this is maxNrSRVRTestsAlready.

3.1.3.1.1 success

On successful completion, the trace of the Point Codes (one or more) of the SPs crossed are included in pointCodeList (copied from the pointCodesTraversed parameter of the testRoute action).

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.2 detectedLoop

If a loop is detected, the point codes (three or more) of the SPs in the loop are included in pointCodeList.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.3 excessiveLengthRoute

If this error occurs, the entire route is copied from the testRoute action pointCodesTraversed parameter into the pointCodeList parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.4 unknownDestination

If the infoRequest parameter of the prompting testRoute action requested it, the pointCodesTraversed parameter of the testRoute action is copied into pointCodeList.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.5 routeInaccessible

If this event is reporting just one inaccessible SP, its point code is placed in pointCode.

If the event is reporting more than one inaccessible SP (and hence the prompting testRoute action indicated that the originator could accept it), the list of all such inaccessible SPs is put into pointCodeList.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.6 processingFailure

If the test cannot be run due to local conditions, the event reports a processing failure. This includes rejection of the testRoute action by SCCP or TC at a remote SP.

If the testRoute action infoRequest parameter was present, and had bit 0 set to 1, the point code of the SP where the test failed is put into the pointCode parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.7 unknownInitiatingSP

The point code of the SP detecting the unknown initiator is returned in the pointCode parameter.

If the prompting testRoute result contained a copyData parameter, this is copied into the routeTraceNew copyData parameter.

3.1.3.1.8 timerExpired

The point codes of the SPs from which no result of the testRoute actions were received are included into pointCodeList.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.9 wrongSP

The pointCodesTraversed parameter of the prompting testRoute action is copied into pointCodeList.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.10 incorrectTranslation-Primary

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

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.11 incorrectTranslation-Secondary

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

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.12 incorrectTranslation-Intermediate

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

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.13 notPrimaryDestination

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

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.14 notSecondaryDestination

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

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.15 notRecognizedPrimary

The complete list of Translation SPs traversed in the route to the secondary destination is copied into pointCodeList.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.16 notRecognizedSecondary

The complete list of Translation SPs traversed in route to the primary destination is copied into pointCodeList.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.17 routingProblem

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

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

3.1.3.1.18 maxNrSRVTestsAlready

This report is used by the signalling point receiving the SRVT message if the maximum number of SRV Tests n_{SRVT} are already running at the SP.

If the testRoute action infoRequest parameter was present, and had bit 0 set to 1, the point code of the SP where the test failed is put into the pointCode parameter.

If there are parameters in the prompting testRoute action that are not understood, and testRoute contains a returnUnknownParams parameter which requests them, they are copied into the copyData parameter.

4 Circuit management

4.1 Circuit Validation Test (CVT) ASE

The Circuit Validation Test ASE provides the services accessed via OM-CONFIRMED-ACTION 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 predefined 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.

cktValidTestCNF-ACTION	*Timer* = T_c	*Class* = 1	*Code* = 00000001

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 }
        CODE                3
        }
    -- Timer = Tc , class = 1
```

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,
                …}
        CODE        3
        }
```

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.

The response in OMAP to local broadcast of N-STATE and N-PCSTATE initiated by SCCP is also 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 :: = { itu-t 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 Recommendations X.680 and X.690.

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

Currently Defined Operations	
0	eventReport
7	confirmedAction

Figure 1/Q.754 – OMAP operations derived from CMIP

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 passed on transparently by the OMPM to the OMASE-User.

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.

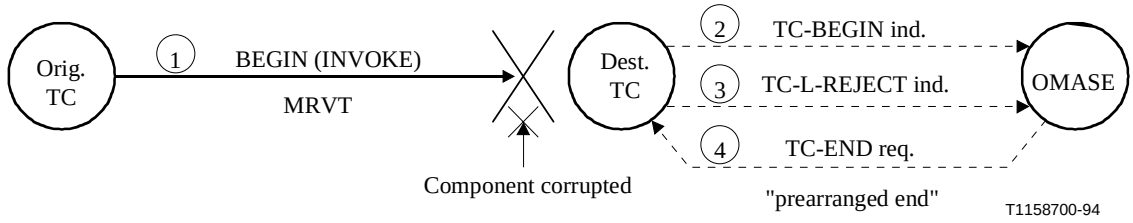

Figure 2a/Q.754

---

[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.

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 "processingFailure". This is illustrated in Figure 2b.

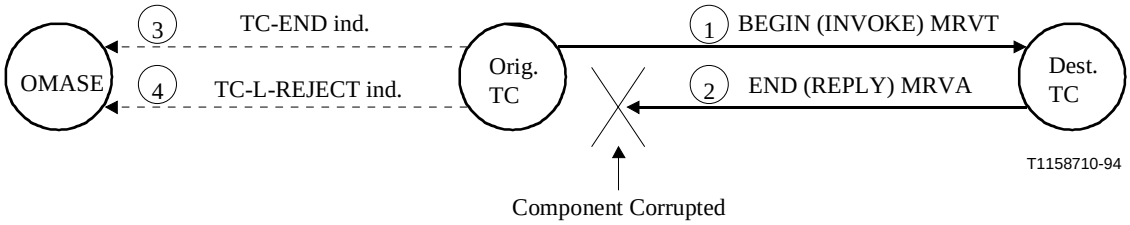

Figure 2b/Q.754

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 "processingFailure".

6.2.3.2.9 Receipt of TC-P-ABORT indication

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

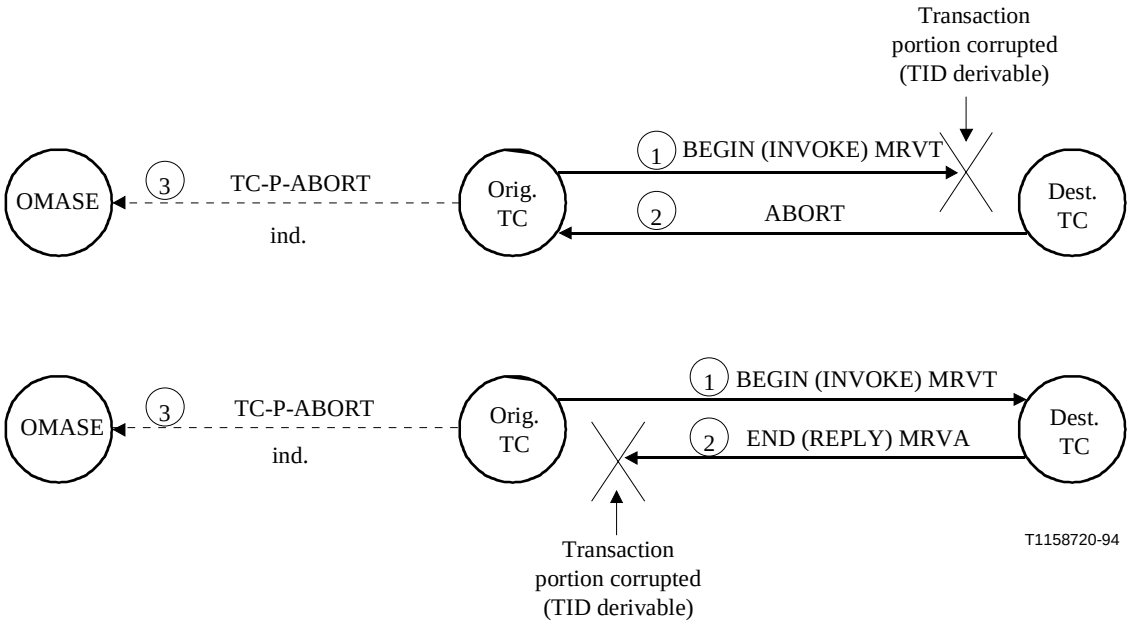

Figure 2c/Q.754

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 "processingFailure".

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 "processingFailure".

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 recognized by the receiving end.

noSuchObjectInstance: While the object class in the Invoke APDU is recognized, 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

See Figure 3.

```
-- OMASE  protocol --
OMASE { itu-t(0) recommendation q 754 omase(0) version2(2) }

DEFINITIONS EXPLICIT TAGS::=
BEGIN
-- TCAP definitions --
EXPORTS EVERYTHING;

-- the OPERATION and ERROR information objects defined here are equivalent to the respective MACROs in
-- TCAPMessages {ccitt recommendation q 773 modules(2) messages(1) version2(2) } of Rec. Q.773 (1993) --

OPERATION ::= CLASS
{
        &ArgumentType       OPTIONAL,
        &ResultType         OPTIONAL,
        &Errors             ERROR OPTIONAL,
        &Linked             OPERATION OPTIONAL,
        &operationCode      Code UNIQUE OPTIONAL
}
WITH SYNTAX
{
        [PARAMETER    &ArgumentType]
        [RESULT       &ResultType]
        [ERRORS       &Errors]
        [LINKED       &Linked]
        [CODE         &operationCode]
}

ERROR ::= CLASS
{
        &ParameterType      OPTIONAL,
        &errorCode          Code UNIQUE OPTIONAL
}
WITH SYNTAX
{
        [PARAMETER    &ParameterType]
        [CODE         &errorCode]
}

Code ::= CHOICE
{
        localValue          INTEGER.
        GlobalValue         OBJECT IDENTIFIER
}
```

Figure 3/Q.754 (sheet 1 of 11) – Formal syntax of OMASE services

```
-- OMASE operators --

eventReport OPERATION ::=
{
PARAMETER eventReportArgument          EventReportArgument

CODE       localValue:0
}

confirmedAction OPERATION ::=
{
PARAMETER actionArgument          ActionArgument
RESULT     actionResult          ActionResult
ERRORS     { accessDenied | invalidArgumentValue |
             noSuchAction | noSuchArgument |
             noSuchObjectClass | noSuchObjectInstance |
             processingFailure }
CODE       localValue:7
}
```

Figure 3/Q.754 (sheet 2 of 11) – Formal syntax of OMASE services

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

```
noSuchObjectClass       ERROR ::=
        {
        PARAMETER       ObjectClass
        CODE            localValue:0
        }

noSuchObjectInstance    ERROR ::=
        {
        PARAMETER       ObjectInstance
        CODE            localValue:1
        }

accessDenied            ERROR ::=
        {
        CODE            localValue:2
        }

noSuchAction            ERROR ::=
        {
        PARAMETER       NoSuchAction
        CODE            localValue:9
        }

processingFailure       ERROR ::=
        {
        PARAMETER       ProcessingFailure -- optional --
        CODE            localValue:10
        }

noSuchArgument          ERROR ::=
        {
        PARAMETER       NoSuchArgument
        CODE            localValue:14
        }

invalidArgumentValue    ERROR ::=
        {
        PARAMETER       InvalidArgumentValue
        CODE            localValue:15
        }
```

Figure 3/Q.754 (sheet 3 of 11) – 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          [3] IMPLICIT CNF-ACTION.&operationCode,
                        actionInfoArg       [4] CNF-ACTION. &ActionArgType(@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               [7] IMPLICIT EVENT. &operationCode
                        eventInfo               [8] EVENT. &EventInfo Type (@eventType)
                                                OPTIONAL}

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

ActionReply         ::= SEQUENCE    {
                        actionType              [3] IMPLICIT CNF-ACTION.
                                                &operationCode,
                        actionReplyInfo         [4]
                                    CNF-ACTION.&ActionResultType(@actionType)
                        }

AccessControl       ::= EXTERNAL
-- AccessControl syntax is to be compatible with that defined in CMIP coded X.209 --
```

Figure 3/Q.754 (sheet 4 of 11) – Formal syntax of OMASE services

```
InvalidArgumentValue    ::= CHOICE {
                            actionValue     [0] IMPLICIT ActionInfo,
                            eventValue      [1] IMPLICIT SEQUENCE {
                            eventType       [7] IMPLICIT EVENT.&operationCode,
                            eventInfo       [8] EVENT.&EventInfoType(@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.&errorCode,
                            errorParm       [1] IMPLICIT SPECIFIC-ERROR. &ProcessingError
                                            ParmType(@errorType) OPTIONAL }

Timer                   ::= INTEGER     -- seconds –
```

Figure 3/Q.754 (sheet 5 of 11) – Formal syntax of OMASE services

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

    EVENT ::= CLASS
    {
        &EventInfoType          OPTIONAL,
        &operationCode          INTEGER UNIQUE OPTIONAL
    }
    WITH SYNTAX
    {
        [EVENTINFO              &EventInfoType]
        [CODE                   &operationCode]
    }

-- Specific Actions are categorized by object class. The protocol uses may be described
-- by the CNF-ACTION INFORMATION OBJECT below. --

    CNF-ACTION ::= CLASS
    {
        &ActionArgType          OPTIONAL,
        &ActionResultType       OPTIONAL,
        &SpecificErrors         SPECIFIC-ERROR OPTIONAL,
        &operationCode          INTEGER UNIQUE OPTIONAL
    }
    WITH SYNTAX
    {
        [ACTIONARG              &ActionArgType]
        [ACTIONRESULT           &ActionResultType]
        [SPECIFICERRORS         &SpecificErrors]
        [CODE                   &operationCode]
    }

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

        SPECIFIC-ERROR ::= CLASS
        {
            &ProcessingErrorParmType        OPTIONAL,
            &errorCode                      INTEGER UNIQUE OPTIONAL
        }
        WITH SYNTAX
        {
            [PARAMETER          &ProcessingErrorParmType]
            [CODE               &errorCode]
        }
```

Figure 3/Q.754 (sheet 6 of 11) – Formal syntax of OMASE services

```
-- specific OMASE constructs follow --

testRoute CNF-ACTION ::=
        {
                ACTIONARG SEQUENCE{
                                        initiating SP           [0] IMPLICIT PointCode,
                                        traceRequested          [1] IMPLICIT BOOLEAN,
                                        threshold               [2] IMPLICIT INTEGER,
                                        pointCodesTraversed     [3] IMPLICIT PointCodeList,
                                        formindicator           [4] IMPLICIT Formindicator OPTIONAL,
-- formIndicator is required in SRVT, but not used in MRVT --

                                        mtpBackwardRoutingRequested
                                                                [5] IMPLICIT BOOLEAN OPTIONAL,
-- mtpBackwardRoutingRequested is required in SRVT, but not in MRTV --

                                        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,
                                        inputGT                 [16] IMPLICIT GlobalTitle OPTIONAL,
-- parameters with tags 4 through 12 can only be used in SRVT, not MRVT --

                                        routePriorityList       [12] IMPLICIT RoutePriorityList
                                                                        OPTIONAL,
-- routePriorityList can only be used in MRVT, and only if the infoRequest parameter is present. --

                                        infoRequest             [13] IMPLICIT BIT STRING {
                                                                pointCode(0),
                                                                pointCodeList(1),
                                                                routePriorityList(2),
                                                                …} OPTIONAL,
-- infoRequest is used to indicate that the test initiator node can accept a route TraceNew
-- RVR message, and also asks for particular parameters to be returned in it, if it is sent. This
-- parameter can only be inserted at the initiator node, but it can be copied into regenerated MRVTs. --

                                        returnUnknownParams     [14] IMPLICIT BIT STRING {
                                                                tag15(0),
                                                                tag16(1),
                                                                …} OPTIONAL,
-- returnUnknownParams is used to indicate which parameters that a node does not understand
-- should be returned in an RVR if one is sent (or in an RVA message in the copyData field
-- if the test initiator is unknown). Bit 0 represents an RVT parameter with tag value 15, bit 1
-- an RVT parameter with tag value 16, etc. To avoid confusion in the copyData field, when
-- defining a new parameter in the RVR message, the tag should have the same value as it has
-- in the RVT message. This parameter can only be present if infoRequest is present. --

                                        directRouteCheck        [15] IMPLICIT BOOLEAN OPTIONAL,
-- directRouteCheck can only be used in MRVT. --

                … }

SPECIFICERRORS          { failure | partialSuccess }

CODE                    1

}

-- TC timer = T1 for MRVT, = T2 for SRVT, Class = 1--
```

Figure 3/Q.754 (sheet 7 of 11) – Formal syntax of OMASE services

```
PointCode ::= OCTET STRING
PointCodeList ::= SEQUENCE OF PointCode
RoutePriorityList ::= SEQUENCE OF Priority

Priority ::= INTEGER{
                        unknown(0),
                        firstChoice(1),
                        secondChoice(2),
                        thirdChoice(3),
                        …} (0..255)

FormIndicator ::= INTEGER
        { compare (0),
        noCompare (1) } (0..1)

GlobalTitle ::= OCTET STRING
        -- the GlobalTitle here consists of the SCCP GTI + GT, the GTI should be encoded exactly as in
        -- 3.4.1/Q.713, and the GT as in 3.4.2.1 to 3.4.2.4/Q.713 as appropriate. --

SubsystemNumber ::= OCTET STRING

failure     SPECIFIC-ERROR ::=
        {
        PARAMETER SEQUENCE          {failureType    [0] IMPLICIT FailureString,
                                    traceSent       [1] IMPLICIT BOOLEAN,
                                    copyData        [2] IMPLICIT CopyData OPTIONAL,
        -- copyData might be present if failureType is unknownInitiatingSp, traceSent is FALSE,
        -- and the prompting RVT message contained a requestInfo parameter, or returnUnknownParams
        -- was in the RVT message. --
                                    … }
        CODE        1
        }

FailureString ::= BIT STRING
                { detectedLoop(0),
                excessiveLengthRoute(1),
                unknownDestination(2),
                routeInaccessible(3),
                processingFailure(4),
                unknownInitiatingSP(5),
                timerExpired(6),
                sPNotAnSTP(7),
        -- wrongSp is a synonym, used in SRVT, of sPNotAnSTP. --
                incorrectTranslation-Primary (8),
                incorrectTranslation-Secondary (9),
                incorrectTranslation-Intermediate (10),
                notPrimaryDestination (11),
                notSecondaryDestination (12),
                notRecognizedPrimary (13),
                notRecognizedSecondary (14),
                routingProblem (15),
        -- bits 8 through 15 might only be set in SRVT, not MRVT. --
                maxNrMRVTestsAlready(16),
        -- maxNrSRVTestsAlready is a synonym, used in SRVT, of maxNrMRVTestsAlready. --
                indirectRoute(17),
        -- indirectRoute might only be set in MRVT, not SRVT. --
                … }

CopyData ::= OCTET STRING
```

Figure 3/Q.754 (sheet 8 of 11) – Formal syntax of OMASE services

```
partialSuccess      SPECIFIC-ERROR ::=
        {
                PARAMETER SEQUENCE      {failureType        [0] IMPLICIT FailureString,
                                         traceSent          [1] IMPLICIT BOOLEAN,
                                         copyData           [4] IMPLICIT CopyData OPTIONAL,

        -- copyData might be present if failureType is unknownInitiatingSP, traceSent is FALSE,
        -- and the prompting RVT message contained a requestInfo parameter,
        -- or returnUnknownParams was in the RVT message. --

                                                            … }
                CODE        2
        }

routeTrace          EVENT ::=
                {
                EVENTINFO CHOICE {
                        success                     [0] IMPLICIT PointCodeList,
                        detectedLoop                [1] IMPLICIT PointCodeList,
                        excessiveLengthRoute        [2] IMPLICIT PointCodeList,
                        unknownDestination          [3] IMPLICIT NULL,
                        routeInaccessible           [4] IMPLICIT PointCode,
                        processingFailure           [5] IMPLICIT NULL,
                        unknownInitiatingSP         [6] IMPLICIT PointCode,
                        timerExpired                [7] IMPLICIT PointCodeList,
                        sPNotAnSTP          [8] IMPLICIT PointCodeList,
                -- wrongSP is a synonym, used in SRVT, for sPNotAnSTP. --

                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
                -- the choices with tags 9 through 16 can only be used in SRVT. --

                        }
                CODE        2
                }

        -- TC Timer = 0, Class = 4
```

Figure 3/Q.754 (sheet 9 of 11) – Formal syntax of OMASE services

```
routeTraceNew EVENT ::=
        {
                EVENTINFO SEQUENCE {
                        result                  [0] IMPLICIT ErrorTag,
                        pointCode               [1] IMPLICIT PointCode OPTIONAL,
                        pointCodeList           [2] IMPLICIT PointCodeList OPTIONAL,
                        routePriorityList       [3] IMPLICIT RoutePriorityList OPTIONAL,
                        copyData                [4] IMPLICIT CopyData OPTIONAL,
        -- copyData allows any parameters included in an RVA message, when the
        -- test initiator is unknown, to be copied into the RVR, without enhancing it. It also
        -- allows new OPTIONAL RVT parameters not understood by the node generating the
        -- RVR from the RVA message to be returned, when requested by the test initiator.
        -- Note that a new parameter defined in routeTraceNew should, if it is also defined in
        -- testRoute, have the same tag value as in testRoute.
        -- One RVR message should be sent for each error detected (no error diagnostics should be
        -- "or'd" together). --
                                                        … }
                CODE        4

        }
        -- TC Timer = 0, Class = 4
```

Figure 3/Q.754 (sheet 10 of 11) – Formal syntax of OMASE services

```
ErrorTag ::=INTEGER {
                success(0),
                detectedLoop(1),
                excessiveLengthRoute(2),
                unknownDestination(3),
                routeInaccessible(4)
                processingFailure(5)
                unknownInitiatingS(6),
                timerExpired(7),

        -- wrongSP is a synonym, used in SRVT, of sPNotAnSTP. --

        incorrectTranslation-Primary(9),
        incorrectTranslation-Secondary(10),
        incorrectTranslation-Intermediate(11),
        notPrimaryDestination(12),
        notSecondaryDestination(13),
        notRecognizedPrimary(14),
        notRecognizedSecondary(15),
                routingProblem(16)
        -- values 9 through 16 are applicable only in SRVT, not in MRVT. --

                        maxNrMRVTestsAlready(17),
        -- maxNrSRVTestsAlready is a synonym, used in SRVT, of maxNrMRVTestsAlready. --

                        indirectRoute(18),
        -- value 18 is applicable only in MRVT, not SRVT. --

                                … } (0..255)

END -- OMASE protocol –
```

Figure 3/Q.754 (sheet 11 of 11) – Formal syntax of OMASE services

ANNEX A

Use of primitive interfaces

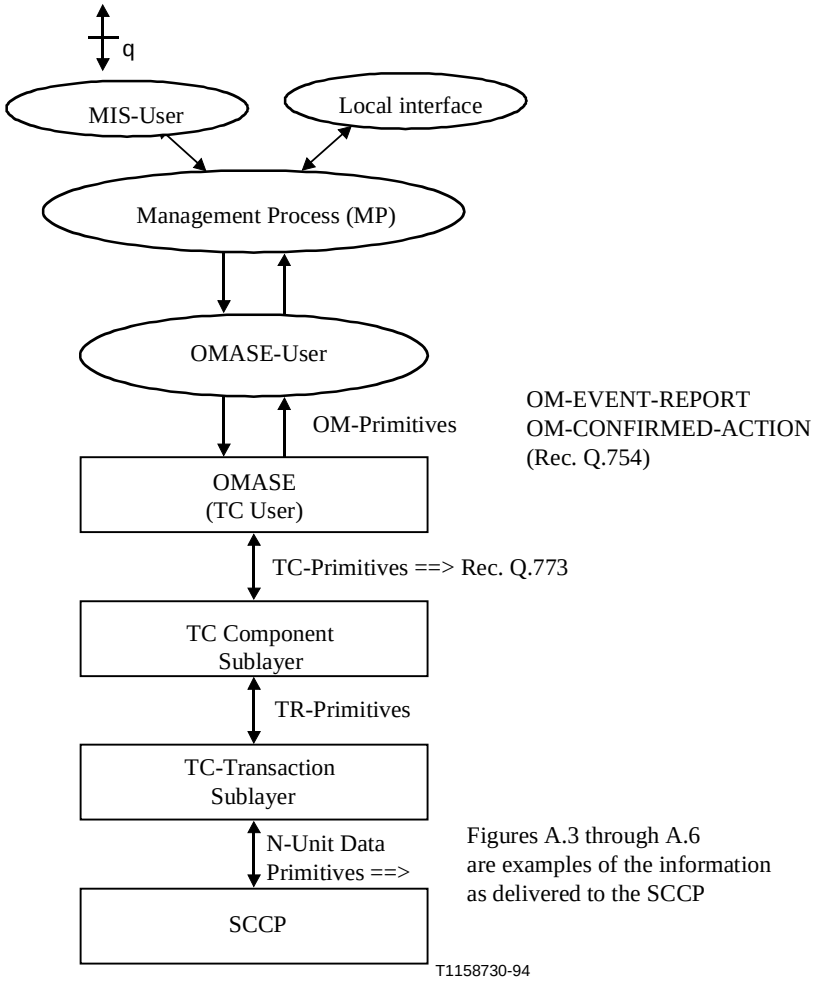

Figure A.1/Q.754 – Primitive interfaces

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.

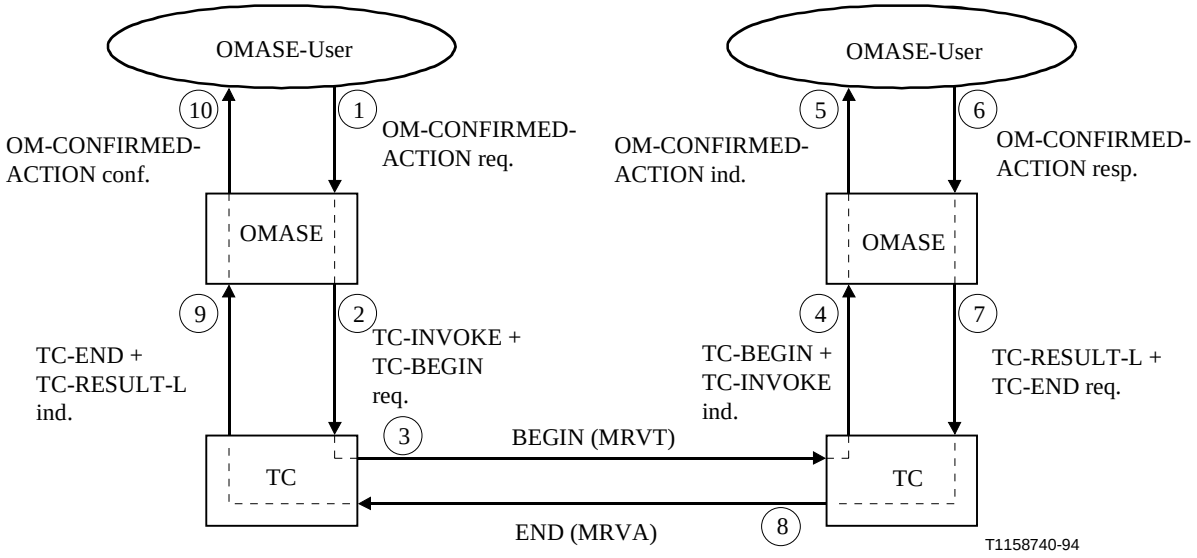

Figure A.2/Q.754 – Example use of primitive interfaces

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 and X.690
Length	00000101	5 octets
Value-MTP Routing Tables	00000000	ITU-T Rec.
	00010001	q
	10000101	85 => 754
	01110010	72 =>
	00000000	MTP Routing Tables 1992
Object Instance Tag	10000011	nonSpecificForm X.711 and X.690
Length	00000010	2 octets
Object Instance Value	xxxxxxxx	(OMAP) Tested destination
	xxxxxxxx	
Action Info Tag	10101100	Recs. X.711 and X.690
Length	00010011	All 19 octets below here
Action Type Tag	10000011	localForm X.711 and X.690
Length	00000001	1 octet
CNF-ACTION	00000001	= testRoute (Figure 3/Q.754)
Action Info Arg Tag	10100100	Recs. X.711 and X.690
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	Figure 3/Q.754, Rec. X.690
Length	00000010	2 octets
Initiating SP Value	xxxxxxxx	(OMAP) test initiator
	xxxxxxxx	
Trace Req. Tag	10000001	Figure 3/Q.754, Rec. X.690
Length	00000001	1 octet
Value	00000001	= TRUE
Threshold Tag	10000010	= threshold (Figure 3/Q.754)
Length	00000001	1 octet
Value of threshold	xxxxxxxx	OMAP PROVIDED
Point Code Trav. Tag	10100011	Figure 3/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	Same as in BEGIN (MRVT msg)
	xxxxxxxx	
	xxxxxxxx	
	xxxxxxxx	
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	Figure 3/Q.754
Error Parameters	10100001	Figure 3/Q.754
Length	00000111	All 7 octets below here
Failure Type Tag	10000000	Figure 3/Q.754
Length	00000010	2 octets
Unused Bits	00000000	no bits
Failure String	xxxxxxxx	Depends on type failure
Trace Sent Tag	10000001	Figure 3/Q.754
Length	00000001	1 octet
Trace Sent Value	0000000x	True = 1, False = 0, Fig. 3/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 xxxxxxxx xxxxxxxx xxxxxxxx	TCAP based on a dialogue at the user level
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 00010001 10000101 01110010 00000000	ITU-T Rec. q 85 => 754 72 => MTP Routing Tables 1992
Object Instance Tag	10000011	(Figure 3/Q.754)
Length	00000010	2 octets
Object Instance Value	xxxxxxxx xxxxxxxx	Terminating PC (OMAP) <Tested Destination>
Event Type Tag	10000111	Figure 3/Q.754
Length	00000001	1 octet
Event Type	00000010	= routeTrace (Figure 3/Q.754)
Event Info Type Tag	10101000	Figure 3/Q.754
Length	00001010	All 10 octets below here
Success Identifier	10100000	Figure 3/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

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	**Switching and signalling**
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communication
Series Z	Programming languages