Intelligent Network (IN); Intelligent Network Application Protocol (INAP); Capability Set 2 (CS2); Part 3: Test Suite Structure and Test Purposes (TSS&TP) specification for Service Switching Function (SSF); Sub-part 1: Basic capability set of CS-1 including CS-2 complements
Contents

Intellectual Property Rights ................................................................................................................................. 5

Foreword .................................................................................................................................................................. 5

1 Scope .................................................................................................................................................................. 6

2 References .......................................................................................................................................................... 6

3 Definitions and abbreviations ............................................................................................................................ 6

3.1 Definitions ......................................................................................................................................................... 6

3.2 Abbreviations .................................................................................................................................................. 7

4 Test Purpose generalities ..................................................................................................................................... 7

4.1 Introduction ....................................................................................................................................................... 7

4.2 Grouping of test purposes per elementary procedures .................................................................................. 7

4.3 Source of test purpose definitions ................................................................................................................ 8

4.4 Method used for developing TPs ................................................................................................................... 8

4.4.1 Use of MSCs generated by the SDL model of Core INAP CS-2 .................................................................... 8

4.4.2 TCAP adapter primitives .............................................................................................................................. 8

4.4.3 Generation of corresponding Test Cases ................................................................................................... 8

4.5 Method used for TP description ................................................................................................................... 9

4.5.1 Text and MSCs ........................................................................................................................................... 9

4.5.2 Test categories .......................................................................................................................................... 9

4.5.3 Test purpose naming convention ........................................................................................................... 10

4.5.4 Preambles and their naming conventions ............................................................................................... 10

4.5.5 How to interpret the parameters and their values as used in the MSCs ..................................................... 11

5 Functional configurations under test ........................................................................................................... 12

5.1 SSF basic functions ....................................................................................................................................... 12

5.2 SSF additional functions ................................................................................................................................. 13

6 TSS and TPs for CS-1 and CS-2 basic capabilities ......................................................................................... 14

6.1 Preambles and postambles used .................................................................................................................... 14

6.1.1 List of preambles and postambles for CS1 ................................................................................................ 14

6.1.2 Preamble descriptions .................................................................................................................................. 15

6.1.2.1 O_OS preamble .................................................................................................................................... 15

6.1.2.2 O_S2P preamble .................................................................................................................................. 16

6.1.2.3 T_OS preamble ..................................................................................................................................... 17

6.1.2.4 T_S2P preamble .................................................................................................................................. 18

6.1.3 Postamble descriptions ................................................................................................................................. 19

6.1.3.1 SigConA_Release postamble ................................................................................................................ 19

6.1.3.2 SigConA_Release_thenB postamble ................................................................................................... 19

6.1.3.3 ReleaseCallA postamble ....................................................................................................................... 20

6.1.3.4 ReleaseCA postamble .......................................................................................................................... 20

6.1.3.5 ReleaseCallAB_cause_00 postamble ................................................................................................. 21

6.1.3.6 ReleaseCallAB_cause_0F postamble ................................................................................................. 21

6.1.3.7 SigConB_Release postamble ................................................................................................................ 23

6.1.3.8 SigConB_Release_cause_0D postamble .............................................................................................. 23

6.1.3.9 SigConA_Release_thenB_cause10 postamble .................................................................................... 24

6.1.3.10 ReleaseCallB postamble ..................................................................................................................... 24

6.1.3.11 ReleaseCallA2 postamble ................................................................................................................... 26

6.2 Basic procedures ............................................................................................................................................ 26

6.2.1 List of procedures ....................................................................................................................................... 26

6.2.2 Definitions of the procedures .................................................................................................................... 27

6.3 Structure of the test suite (TSS) for the basic capabilities ........................................................................... 28

6.4 Test Purposes (TP) description ..................................................................................................................... 30

6.4.1 ServiceFiltering procedure ....................................................................................................................... 30

6.4.2 ActivityTest procedure .............................................................................................................................. 40

6.4.3 ApplyCharging procedure ......................................................................................................................... 44
<table>
<thead>
<tr>
<th>Section</th>
<th>Procedure Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.4</td>
<td>CallGap procedure</td>
<td>57</td>
</tr>
<tr>
<td>6.4.5</td>
<td>CallInformation procedure</td>
<td>81</td>
</tr>
<tr>
<td>6.4.6</td>
<td>Cancel procedure</td>
<td>96</td>
</tr>
<tr>
<td>6.4.7</td>
<td>CollectInformation procedure</td>
<td>106</td>
</tr>
<tr>
<td>6.4.8</td>
<td>Connect procedure</td>
<td>111</td>
</tr>
<tr>
<td>6.4.9</td>
<td>Continue procedure</td>
<td>126</td>
</tr>
<tr>
<td>6.4.10</td>
<td>FurnishChargingInformation procedure</td>
<td>129</td>
</tr>
<tr>
<td>6.4.11</td>
<td>InitialDP procedure</td>
<td>133</td>
</tr>
<tr>
<td>6.4.12</td>
<td>InitiateCallAttempt procedure</td>
<td>142</td>
</tr>
<tr>
<td>6.4.13</td>
<td>ReleaseCall procedure</td>
<td>152</td>
</tr>
<tr>
<td>6.4.14</td>
<td>RequestReportBCSMEvent procedure</td>
<td>157</td>
</tr>
<tr>
<td>6.4.15</td>
<td>SendChargingInformation procedure</td>
<td>222</td>
</tr>
<tr>
<td>6.4.16</td>
<td>RequestNotificationChargingEvent procedure</td>
<td>223</td>
</tr>
</tbody>
</table>

Annex A (informative): Description of various functional configurations .................................................230

Annex B (normative): Parameter values used in MSCs for CORE INAP primitives .........................247

Annex C (normative): Parameter values used in MSCs for TCAP primitives .................................249

Bibliography ..................................................................................................................................................250

History ..........................................................................................................................................................251
Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for ETSI members and non-members, and can be found in SR 000 314: "Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards", which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.org/ipr).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This European Standard (Telecommunications series) has been produced by ETSI Technical Committee Services and Protocol for Advanced Networks (SPAN), and is now submitted for the Voting phase of the ETSI standards Two-step Approval Procedure.

The present document is part 3, sub-part 1 of a multi-part EN covering the Intelligent Network (IN); Intelligent Network Application Protocol (INAP); Capability Set 2 (CS2), as identified below:

Part 1: "Protocol specification";
Part 2: "Protocol Implementation Conformance Statement (PICS) proforma specification";
Part 3: "Test Suite Structure and Test Purposes (TSS&TP) specification for Service Switching Function (SSF)";
   Sub-part 1: "Basic capability set of CS-1 including CS-2 complements";
   Sub-part 2: "Call Party Handling (CPH)";
   Sub-part 3: "Specialized Resource Functions (SRF)";
Part 4: "Abstract Test Suite (ATS) specification and Partial Protocol Implementation eXtra Information for Testing (PIXIT) proforma for Service Switching Function (SSF)";
Part 5: "Distributed Functional Plane (DFP) [ITU-T Recommendation Q.1224 (1997), modified]".

Proposed national transposition dates

<table>
<thead>
<tr>
<th>Proposed national transposition dates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date of latest announcement of this EN (doa): 3 months after ETSI publication</td>
</tr>
<tr>
<td>Date of latest publication of new National Standard or endorsement of this EN (dop/e): 6 months after doa</td>
</tr>
<tr>
<td>Date of withdrawal of any conflicting National Standard (dow): 6 months after doa</td>
</tr>
</tbody>
</table>
1 Scope

The present document provides the Test Suite Structure and Test Purposes (TSS&TP) for testing of the Service Switching Function (SSF) and the Specialized Resource Function (SRF) of the Intelligent Network Application Protocol (INAP) of Intelligent Network (IN) Capability Set 2 (CS2) according to EN 301 140-1 [1].

The present document relates to the basic capability set, which covers the CS-1 operations, plus the CS-2 additions related to these operations, mainly due to the test of the CS-2 additional parameters or functionalities.

The present document is completed by other parts constituting the CS-2 Core INAP specifications.

In the present version of the TP description included in tables, references to specification requirements and references to PICS in the "condition for selection" are not included, except to mention when it is a CS-2 addition.


2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

[1] ETSI EN 301 140-1 (V1.3): "Intelligent Network (IN); Intelligent Network Application Protocol (INAP); Capability Set 2 (CS2); Part 1: Protocol specification”.

[2] ETSI EN 301 140-4 (V1.1): "Intelligent Network (IN); Intelligent Network Application Protocol (INAP); Capability Set 2 (CS2); Part 4: Abstract Test Suite (ATS) specification and Partial Protocol Implementation eXtra Information for Testing (PIXIT) proforma for Service Switching Function (SSF)”.


[5] ETSI ETS 300 374-1: "Intelligent Network (IN); Intelligent Network Capability Set 1 (CS1); Core Intelligent Network Application Protocol (INAP); Part 1: Protocol specification”.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

- terms defined in EN 301 140-1 [1];
In particular, the following terms defined in ISO/IEC 9646-1 [3] apply:

- Abstract Test Suite (ATS);
- Implementation Under Test (IUT);
- System Under Test (SUT);
- Protocol Implementation Conformance Statement (PICS).

### 3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATS</td>
<td>Abstract Test Suite</td>
</tr>
<tr>
<td>BI</td>
<td>Invalid Behaviour tests</td>
</tr>
<tr>
<td>BO</td>
<td>Inopportune Behaviour tests</td>
</tr>
<tr>
<td>BV</td>
<td>Valid Behaviour tests</td>
</tr>
<tr>
<td>CA</td>
<td>Capability tests</td>
</tr>
<tr>
<td>CS</td>
<td>Call Segment</td>
</tr>
<tr>
<td>CS</td>
<td>Capability Set</td>
</tr>
<tr>
<td>EDP-R</td>
<td>Event Detection Point - Request</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>IN</td>
<td>Intelligent Network</td>
</tr>
<tr>
<td>INAP</td>
<td>Intelligent Network Application Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Intelligent Peripheral</td>
</tr>
<tr>
<td>iS</td>
<td>initiating SSF</td>
</tr>
<tr>
<td>iSSP</td>
<td>initiating SSP</td>
</tr>
<tr>
<td>IUT</td>
<td>Implementation Under Test</td>
</tr>
<tr>
<td>MSC</td>
<td>Message Sequence Chart</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>PICS</td>
<td>Protocol Implementation Conformance Statement</td>
</tr>
<tr>
<td>SCF</td>
<td>Service Control Function</td>
</tr>
<tr>
<td>SCP</td>
<td>Service Control Point</td>
</tr>
<tr>
<td>SDF</td>
<td>Service Data Function</td>
</tr>
<tr>
<td>SDL</td>
<td>Specification and Description Language</td>
</tr>
<tr>
<td>SRF</td>
<td>Specialized Resource Function</td>
</tr>
<tr>
<td>SSSF</td>
<td>Service Switching Function</td>
</tr>
<tr>
<td>SSP</td>
<td>Service Switching Point</td>
</tr>
<tr>
<td>SUT</td>
<td>System Under Test</td>
</tr>
<tr>
<td>TCAP</td>
<td>Transaction Capabilities Application Part</td>
</tr>
<tr>
<td>TP</td>
<td>Test Purpose</td>
</tr>
<tr>
<td>TSS</td>
<td>Test Suite Structure</td>
</tr>
</tbody>
</table>

### 4 Test Purpose generalities

#### 4.1 Introduction

A TP is defined for one or several conformance requirements to be tested. Each TP will result in a test case keeping the same name, specified in the ATS.

#### 4.2 Grouping of test purposes per elementary procedures

The Test Purposes are grouped by elementary procedures. A procedure groups elementary INAP operations which it is possible to test together. For each elementary procedure, are defined: how to invoke it; and what are the possible return results and return error(s) at the INAP interface.
NOTE: Some have no results at all at this INAP interface. In these cases, and to have a "visible" result, the PCO will be at the signalling control interface.

4.3 Source of test purpose definitions

The test purposes are based on the requirement documented in EN 301 140-1 [1].

4.4 Method used for developing TPs

4.4.1 Use of MSCs generated by the SDL model of Core INAP CS-2

The SDL model of INAP CS-2 is specified with object oriented SDL (SDL’92) and specifies the behaviour of the SSF. The CS-2 specification inherits the CS-1 and specifies the whole of CS-1 and CS-2. The SDL specification is the normative specification of the INAP behaviour and is contained in annex A of EN 301 140-1 [1].

The SDL model specifies precisely and unambiguously the behaviour of and the interworking between the different functional entities of the SSF. The external interfaces of the SDL model are two signalling control interfaces (SigConA and SigConB) carrying abstract primitives, and the INAP interfaces to the SCF. Mappings are provided from SigConA and SigConB to DSS.1 and ISUP. The behaviour of the SDL model thus resembles an SSP, and can be used for service emulation and the development of test purposes and test cases. MSCs delivered by this SDL model are used in the TP definition and are provided in addition to the descriptive text.

The development of the test purposes (TP) is done in two steps:

a) the descriptive text is created together with a rough MSC defined by hand. It illustrates the basic behaviour in MSC-like form which is expected from the IUT. The rough MSC does not contain all the constraints in detail. The description makes reference to a preamble and a postamble;

b) a detailed MSC is developed by simulation:

1) system level MSC for Autolink (the tool used to automatically generate the TTCN test cases based on the MSCs and the SDL model);

2) MSC for documentation of the TPs.

The reason for developing the detailed MSC by simulation is that it can be done step by step while the SDL model prompts the developer for the correct options and parameters.

The MSCs identify the different entities (SSF, SCF, SigCon A and B) involved in a given configuration and shows the different components used for a test, in term of the IUT (representing the SSF for instance) and the testers (representing the SCF and the SigCon A, B or C).

4.4.2 TCAP adapter primitives

In addition to showing the INAP protocol, and in order to ease the implementation of the test suite, the MSCs show the TCAP adapter primitives such as TC begin, TC continue, TC invoke and TC end and show using standard abbreviations the INAP operations which are embedded in the TCAP primitive, together with the operation arguments.

4.4.3 Generation of corresponding Test Cases

Using Computer Aided Test Generation techniques, TTCN test cases can be automatically generated from the SDL model. It is also possible to verify manually developed test cases against the SDL model. The clear separation of CS-1 and CS-2 in the SDL model makes it possible to use it for both CS-1 and CS-2 test case development.
4.5 Method used for TP description

4.5.1 Text and MSCs

In general, a TP is described using text presented in a table followed by an MSC.

The table describing each TP is as follows:

<table>
<thead>
<tr>
<th>Table 1a (TP name, also corresponding test case name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Purpose:</td>
</tr>
<tr>
<td>Requirement Ref:</td>
</tr>
<tr>
<td>Selection Cond:</td>
</tr>
<tr>
<td>Test preamble:</td>
</tr>
<tr>
<td>Test description</td>
</tr>
<tr>
<td>Pass criteria:</td>
</tr>
<tr>
<td>Test postamble</td>
</tr>
</tbody>
</table>

In addition to the TP name and a reference to the specification requirement, the table contains a short title of the test purpose, the condition to select and run this test case (expression in terms of PICS references), the name of the test preamble, information on the test body including for instance details on parameters which do not appear in the companion MSC, the pass criteria for a successful test and the name of the test postamble.

The MSC which follows the TP description describes the test body, as the preambles and postambles are mostly defined by a single line in the MSC.

4.5.2 Test categories

**Capability tests (CA)**

Capability testing provides a limited testing to ascertain the capabilities stated in the PICS can be observed.

**Valid Behaviour tests (BV)**

Predefined state transitions are considered as valid. The test purposes in the valid behaviour test sub group cover as far as reasonable the verification of the normal and exceptional procedures of the various Finite State Machines (FSMs), i.e. a valid behaviour test is a test where the message sequence and the message contents is considered as valid.

**Invalid Behaviour tests (BI)**

This test sub group is intended to verify that the IUT is able to react properly having received an invalid Protocol Data Unit (PDU). An invalid PDU is defined as a syntactically incorrect message.

**Inopportune Behaviour tests (BO)**

This test group is intended to verify that the IUT is able to react properly in the case an inopportune protocol event occurring. Such an event is syntactically correct but occurs when it is not expected, e.g. a correctly coded operation is received in a wrong state (the IUT may respond by sending error UnexpectedComponentSequence).
4.5.3 Test purpose naming convention

The identifier of the TP is built according to the scheme in table 1b.

<table>
<thead>
<tr>
<th>Table 1b: TP identifier naming convention scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier: IN2_&lt;i&gt;<em>&lt;sss&gt;</em>&lt;pp&gt;<em>&lt;cc&gt;</em>&lt;nn&gt;</td>
</tr>
<tr>
<td>IN2 indicates IN Capability Set 1 and 2 (CS-1 being in CS-2)</td>
</tr>
<tr>
<td>&lt;i&gt; = interface: A SSF-SCF interface</td>
</tr>
<tr>
<td>B SSF-SRF interface</td>
</tr>
<tr>
<td>C SCF-SCF interface</td>
</tr>
<tr>
<td>&lt;sss&gt; = common set BASIC Basic set for CS-1 complemented for CS-2</td>
</tr>
<tr>
<td>CPH Call Party Handling from Capability Set 2</td>
</tr>
<tr>
<td>CTM Cordless Terminal Portability from Capability Set 2</td>
</tr>
<tr>
<td>&lt;pp&gt; = procedure name like</td>
</tr>
<tr>
<td>SF ServiceFiltering</td>
</tr>
<tr>
<td>&lt;cc&gt; = test category:</td>
</tr>
<tr>
<td>CA Capability tests</td>
</tr>
<tr>
<td>BV Valid Behaviour tests</td>
</tr>
<tr>
<td>BI Invalid Behaviour tests</td>
</tr>
<tr>
<td>BO Inopportune Behaviour tests</td>
</tr>
<tr>
<td>&lt;nn&gt; = sequential number: (01-99)</td>
</tr>
</tbody>
</table>

Example of test purpose and test case name: \textbf{IN2_A_BASIC_SF_BV_02}

4.5.4 Preambles and their naming conventions

Preambles are used to bring the IUT from the initial state to the state where the test takes place. In the CS-2 scheme, the set of the preambles forms a tree, which means that in order to reach the state created by preamble P3, it is necessary to execute preamble P1 followed by preambles P2 then P3.

The naming convention used reflects the description of the connection view set by executing the preamble, in terms of nature of the legs per Call Segment (CS), starting from the stable legs then the ones on hold then the ones in transfer, with the indication of the number of legs, while the first letter indicates how this configuration was initiated.

The general form is:

\texttt{a_[stableLegsParty or onHold (legs) or transfer(legs) for CallSegment 1]_[idem for CallSegment2]_[idem for CallSegment 3]}

where:

- \texttt{a} is letter:
  - \texttt{O} for Originating (outgoing call for a user);
  - \texttt{T} for Terminating (incoming call for a user);
  - \texttt{I} for Initiate Call Attempt (initiated from the network).
The state names and their abbreviations used are:

Null

<table>
<thead>
<tr>
<th>State Name</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Party</td>
<td>IP</td>
</tr>
<tr>
<td>Originating Set-up</td>
<td>OS</td>
</tr>
<tr>
<td>Terminating Set-up</td>
<td>TS</td>
</tr>
<tr>
<td>Terminating 1 Party Set-up</td>
<td>T1P</td>
</tr>
<tr>
<td>Stable 1 Party</td>
<td>S1P</td>
</tr>
<tr>
<td>Stable 2 Party</td>
<td>S2P</td>
</tr>
<tr>
<td>Transfer (no. of passive legs)</td>
<td>TF(x)</td>
</tr>
<tr>
<td>On Hold (no. of passive legs)</td>
<td>OH(x)</td>
</tr>
<tr>
<td>Stable MultiParty (no. of passive legs)</td>
<td>S(x)P</td>
</tr>
</tbody>
</table>

The term "null" stands for "none" as in preamble O_NULL_S2P_OH3.

There can be two set of CSs with the same nature of legs present at the same time, as in the preamble name O_S2P_OH2_OH3.

4.5.5 How to interpret the parameters and their values as used in the MSCs

The MSCs show the exchanges of PDUs of the TCAP protocol, as well as the Core INAP protocol. PDUs of both protocols use parameters.

The list of the parameters for the Core INAP protocol is given in reference ETS 300 374-1 [5].

The list of parameters for the TCAP protocol is recalled here for each TCAP primitives. Note that only mandatory parameters are used.

TCAP primitives from SCF to TCAP:

   TC_InvokeReq (InvokeID, DialogueID, Class, OperationCode, Timeout);
   TC_BeginReq (DialogueID, OriginatingAddress);
   TC_ContinueReq (DialogueID, OriginatingAddress);
   TC_EndReq (DialogueID, Termination);
   TC_AbortReq (DialogueID).

TCAP primitives from TCAP to SCF:

   TC_InvokeInd (InvokeID, DialogueID, Class, OperationCode, LastComponent);
   TC_BeginInd (DialogueID, OriginatingAddress, ComponentPresent);
   TC_ContinueInd (DialogueID, OriginatingAddress, ComponentPresent);
   TC_EndInd (DialogueID, Termination, ComponentPresent);
   TC_AbortInd (DialogueID);
   TC_ErrorInd (InvokeID, DialogueID, ErrorCode, LastComponent);
   TC_ReturnResultInd (InvokeID, DialogueID, LastComponent, OperationCode, OperationArg);
TC_RejectInd (InvokeID, DialogueID).

The values of these parameters are either mandatory and imposed by the specifications, or they are informative only and chosen arbitrarily in ranges compatible with the specifications.

The list of the informative parameters, for which a value is to be assigned in particular for the execution of a test suite, is included in the PIXIT proforma. See reference EN 301 140-4 [2].

Annex B and Annex C of the present document contain a copy of the PIXIT proforma parameter tables of respectively the Core INAP and the TCAP protocols. These proforma tables are filled up and contain the parameter values used for the definition of the MSCs and TPs.

The preamble T_OS (and all preambles and test cases which use this preamble) contains reference to an ASP Mgt_SetTriggerTable. This does not exist in the protocol, but in the SDL model it identifies which Trigger Detection points need to be set before commencing the test case.

5 Functional configurations under test

5.1 SSF basic functions

![Diagram](image-url)

Figure 1: Configuration 1: IUT= SSF (non-integrated with SRF)
5.2 SSF additional functions

**Figure 3: Configuration 3: IUT= SSF of assisting SSP (integrated SRF)**
6 TSS and TPs for CS-1 and CS-2 basic capabilities

6.1 Preambles and postambles used

6.1.1 List of preambles and postambles for CS1

Here is a list of preambles used in the Basic CS-1 and CS-2 capabilities part:

O_OS;
O_S2P;
T_OS;
T_S2P.

Here is a list of postambles used in the Basic CS-1 and CS-2 capabilities part:

- SigConA_Release;
- SigConA_Release_thenB;
- ReleaseCallA;
- ReleaseICA;
- ReleaseCallAB_cause_00;
- ReleaseCallAB_cause_0F;
- SigConB_Release.

More preambles and postambles are defined for the complete CS-2. See clause 7.

### 6.1.2 Preamble descriptions

#### 6.1.2.1 O_OS preamble

This preamble is used to bring the IUT from the idle or Null state to the 1 party state.

![MSC O_OS diagram]

- **SetupInd**
  - **callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H**
- **TC_BeginInd**
  - **51, oSSF, TRUE**
- **TC_InvokeInd**
  - **101, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }**

"MSC O_OS"
6.1.2.2 O_S2P preamble

This preamble is used to bring the IUT from the idle or Null state to the 2 party state.

**MSC O_S2P**

```
SetupInd
  { callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
  TC_BeginInd
  [ 51, oSSF, TRUE
    TC_InvokeInd
    101, 51, IDP, TRUE, IDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H,
      eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }
  ]
  TC_InvokeReq
  1, 51, 2, RRB, short, rRBArg : { bcsmEvents { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '02'H } }
  TC_InvokeReq
  2, 51, 2, CON, short, cONArg : { destinationRoutingAddress '2001'H, callingPartyNumber '1000'H }
  TC_ContinueReq
  [ 51, oSCF
    SetupReq
    { callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
  ]
```

---

**MSC O_S2P**

```
SetupInd
  { callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
  TC_BeginInd
  [ 51, oSSF, TRUE
    TC_InvokeInd
    101, 51, IDP, TRUE, IDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H,
      eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }
  ]
  TC_InvokeReq
  1, 51, 2, RRB, short, rRBArg : { bcsmEvents { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '02'H } }
  TC_InvokeReq
  2, 51, 2, CON, short, cONArg : { destinationRoutingAddress '2001'H, callingPartyNumber '1000'H }
  TC_ContinueReq
  [ 51, oSCF
    SetupReq
    { callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
  ]
```
6.1.2.3 T_OS preamble

**MSC T_OS**

![Diagram of MSC T_OS](image)

- **Mgt_SetTriggerTable**
  - `{ { legID '01'H, serviceKey 1, eventTypeBCSM analysedInformation, monitorMode transparent } }`

- **Mgt_SetTriggerTable**
  - `{ { legID '01'H, serviceKey 1, eventTypeBCSM termAttemptAuthorized, monitorMode interrupted } }`

- **SetupInd**
  - `{ callRef 1, calledPartyNumber '2002'H, callingPartyNumber '1000'H }`

- **TC_BeginInd**
  - `51, oSSF, TRUE`

- **TC_InvokeInd**
  - `101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2002'H, callingPartyNumber '1000'H, eventTypeBCSM termAttemptAuthorized, createdCallSegmentAssociation 1 }`
6.1.2.4 T_S2P preamble

**MSC T_S2P**

- **SetupInd**
  - callRef 1, calledPartyNumber '2002'H, callingPartyNumber '1000'H
  - [51, oSSF, TRUE]
  - TC_BeginInd

- **TC_InvokeInd**
  - 101, 51, IDP, TRUE
  - IDPArg : { serviceKey 1, calledPartyNumber '2002'H, callingPartyNumber '1000'H, BCSM analysedInformation, createdCallSegmentAssociation 1 }

- **TC_InvokeReq**
  - 1, 51, 2, RRB, short
  - rRBArg : { bcsmEvents { eventTypeBCSM tDisconnect, monitorMode interrupted, legID sendingSideID : '02'H } }

- **TC_InvokeInd**
  - 2, 51, 2, CON, short
  - cONAArg : { destinationRoutingAddress {'2001'H, callingPartyNumber '1000'H }

- **TC_InvokeReq**
  - 2, 51, 2, CON, short
  - cONAArg : { destinationRoutingAddress {'2001'H, callingPartyNumber '1000'H }

- **TC_ContinueReq**
  - 51, oSCF

- **SetupReq**
  - [ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H ]
6.1.3 Postamble descriptions

Postambles are used to bring the IUT from the state where the test ends, back to the initial state.

6.1.3.1 SigConA_Release postamble

MSC SigConA_Release

6.1.3.2 SigConA_Release_thenB postamble

MSC SigConA_Release_thenB
6.1.3.3 ReleaseCallA postamble

**MSC ReleaseCallA**

- SCF
- CS2_SSF
- SigCon_A
- SigCon_B

TC_InvokeReq

100, 51, 4, RC, short, rCAArg : initialCallSegment : '00'H

TC_EndReq

S1, basic

ReleaseReq

{ callRef 1, cause '00'H }, senderRef

6.1.3.4 ReleaseICA postamble

**MSC ReleaseICA**

- SCF
- CS2_SSF
- SigCon_A
- SigCon_B

ReleaseInd

{ callRef 2, cause '00'H }
6.1.3.5 ReleaseCallAB_cause_00 postamble

MSC ReleaseCallAB_cause_00

**SCF**
- TC_InvokeReq
- TC_EndReq

**CS2_SSF**

**SigCon_A**

**SigCon_B**

100, 51, 4, RC, short, rC Arg : initialCallSegment : '00'H

TC_InvokeReq

TC_EndReq

51, basic

ReleaseReq

ReleaseReq

ReleaseReq

{ callRef 1, cause '00'H, senderRef }

{ callRef 2, cause '00'H, senderRef }
6.1.3.6 ReleaseCallAB_cause_0F postamble

MSC ReleaseCallAB_cause_0F

- SCF
- CS2_SSF
- SigCon_A
- SigCon_B

TC_InvokeReq
100, 51, 4, RC, short, rCArg : initial CallSegment : '00'H

TC_EndReq
51, basic

ReleaseReq

{ callRef 1, cause '00'H, senderRef }

{ callRef 2, cause '0F'H, senderRef }
6.1.3.7 SigConB_Release postamble

MSC SigConB_Release

```
SCF   CS2_SSF   SigCon_A   SigCon_B

ReleaseInd
[[ callRef 2, cause '00'H ]]
```

6.1.3.8 SigConB_Release_cause_0D postamble

MSC SigConB_Release_cause_0D

```
SCF   CS2_SSF   SigCon_A   SigCon_B

ReleaseInd
[[ callRef 2, cause '0D'H ]]
```
6.1.3.9 SigConA_Release_thenB_cause10 postamble

MSC SigConA_Release_thenB_cause10

[Diagram showing the sequence of events involving SCF, CS2SSF, SigConA, and SigConB, with messages and references labeled as specified.]
6.1.3.10 ReleaseCallB postamble

MSC ReleaseCallB

TC_InvokeReq
[T51, 4, RC, short, rCArg : initialCallSegment : '00'H]

TC_EndReq
[T51, basic]

ReleaseReq
[callRef 2, cause '00'H, senderRef]
6.1.3.11 ReleaseCallA2 postamble

MSC ReleaseCallA2

6.2 Basic procedures

6.2.1 List of procedures

The Test Purposes for Basic CS-1 and CS-2 functionalities are grouped according to the following procedures:

NOTE: The acronyms below are names given to a procedure (example: SF for Service Filtering), and may not be in line with the standardized ones used for invoking such a procedure (example: ASF for Activate Service Filtering). These acronyms are made of two letters only and are used when giving a name to a Test Purpose or a test case.

SF ServiceFiltering
AT ActivityTest
AC ApplyCharging
AR AssistRequestInstructions
CG CallGap
CF CallInformation
CA Cancel
CI CollectInformation
CO Connect
CR ConnectToResource
CU Continue (no specific Test Purpose)
DF DisconnectForwardConnection
EC EstablishTemporaryConnection
FC FurnishChargingInformation
DP InitialDP
IC InitiateCallAttempt
6.2.2 Definitions of the procedures

ServiceFiltering procedure (SF)
Invoke: ActivateServiceFiltering
Return Result: ServiceFilteringResponse
Return Error: ActivateServiceFiltering

ActivityTest procedure (AT)
Invoke: ActivityTest
Return Result: ActivityTest
Return Error: None

ApplyCharging procedure (AC)
Invoke: ApplyCharging
Return Result: ApplyChargingReport
Return Error: ApplyCharging

AssistRequestInstructions procedure (AR)
Invoke: AssistRequestInstructions
Return Result: None
Return Error: AssistRequestInstructions

CallGap procedure (CG)
Invoke: CallGap
Return Result: None
Return Error: None

CallInformation procedure (CF)
Invoke: CallInformationRequest
Return Result: CallInformationReport
Return Error: CallInformationRequest

Cancel procedure (CA)
Invoke: Cancel
Return Result: PlayAnnouncement(Error)
Return Error: PromptAndCollectUserInformation(Error)

CollectInformation procedure (CI)
Invoke: CollectInformation
Return Result: RequestReportBCSMEvent
Return Error: EventReportBCS

Connect procedure (CO)
Invoke: Connect
Return Result: None
Return Error: Connect

ConnectToResource procedure (CR)
Invoke: ConnectToResource
Return Result: None
Return Error: ConnectToResource
6.3 Structure of the test suite (TSS) for the basic capabilities

Table 1c shows the structure of the test suites for SSF functions and the number of Test Purposes produced.
Table 1c: Test suite structure of the SSF test

<table>
<thead>
<tr>
<th>IUT</th>
<th>Interface</th>
<th>Protocol component</th>
<th>Procedure</th>
<th>Category and number</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSF</td>
<td>SSF-SCF</td>
<td>Basic subset</td>
<td>SF</td>
<td>ServiceFiltering</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AT</td>
<td>ActivityTest</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AC</td>
<td>ApplyCharging</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CG</td>
<td>CallGap</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CF</td>
<td>CallInformation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA</td>
<td>Cancel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CI</td>
<td>CollectInformation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CO</td>
<td>Connect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CU</td>
<td>Continue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FC</td>
<td>Furnish Charging</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DP</td>
<td>InitialDP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 2</td>
<td>BV 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IC</td>
<td>InitiateCall Attempt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RC</td>
<td>ReleaseCall</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RR</td>
<td>RequestReport</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA 1</td>
<td>BV 33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SC</td>
<td>SendCharging</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CA</td>
<td>BV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BO</td>
</tr>
</tbody>
</table>
6.4 Test Purposes (TP) description

The objective is to test the INAP procedures at the Service Switching Point (SSP).

SigCon A and SigCon B are the signalling controls for users A and B, and the IUT is a SSF while the main lower tester is an SCF.

6.4.1 ServiceFiltering procedure

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>test ServiceFiltering procedure on duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td>none</td>
</tr>
<tr>
<td>Selection Cond.</td>
<td>none</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF issues ActivateServiceFiltering invoke containing mandatory parameters only, with:</td>
</tr>
<tr>
<td></td>
<td>- filteredCallTreatment including sFBillingChargingCharacteristics only,</td>
</tr>
<tr>
<td></td>
<td>- filteringCharacteristics being interval,</td>
</tr>
<tr>
<td></td>
<td>- filteringTimeOut being duration,</td>
</tr>
<tr>
<td></td>
<td>- filteringCriteria being serviceKey,</td>
</tr>
<tr>
<td></td>
<td>then a call is initiated after Characteristics being interval duration expires</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>SSF accepts the call, then</td>
</tr>
<tr>
<td></td>
<td>SSF issues ServiceFilteringResponse invoke with parameters</td>
</tr>
<tr>
<td></td>
<td>- countersValue including 1 counterAndValue,</td>
</tr>
<tr>
<td></td>
<td>- filteringCriteria being serviceKey</td>
</tr>
<tr>
<td>Postamble:</td>
<td>Release Call A.</td>
</tr>
<tr>
<td>Purpose:</td>
<td>test ServiceFiltering procedure on miscellaneous parameters</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
</tbody>
</table>
| Test description | SCF issues ActivateServiceFiltering invoke containing mandatory and optional parameters, with:  
- filteredCallTreatment including:  
  - sFBillingChargingCharacteristics,  
  - informationToSend,  
  - maximumNumberOfCounters,  
- filteringCharacteristics being numberOfCalls,  
- filteringTimeOut being stopTime,  
- filteringCriteria being addressAndService including:  
  - calledAddressValue,  
  - serviceKey,  
  - callingAddressValue,  
  - locationNumber,  
- startTime |
| Pass criteria | - Before startTime, SSF does not filter a call and passes it to SCF |
| Postamble: | SigConA_Release_thenB. |
MSC IN2m_A_BASIC_SF_BV_01

TC_InvokeReq
1, 51, 1, ASF, medium, aSFArg : {
  filteredCallTreatment { sFBillingChargingCharacteristics 'BBBB',
  informationToSend { tone : { toneID 0, duration 66 },
  maximumNumberOfCounters 14, releaseCause '00'H },
  filteringCharacteristics numberOfCalls : 1,
  filteringTimeOut Duration : 5000,
  filteringCriteria addressAndService : { calledAddressValue '2000'H, serviceKey 28,
  callingAddressValue '1000'H, locationNumber '9001'H } }

TC_BeginReq
1, 51, TRUE, ASF_R, aSFRArg : Null

StartTime_not_matched

SetupInd
{ callRef 1, calledPartyNumber '2000'H }
TC_BeginInd
51, oSCF

TC_InvokeInd
101, 52, IDP, TRUE, IDPArg : { serviceKey 1, createdCallSegmentAssociation 1 }
TC_InvokeReq
1, 52, 2, CON, short, cONArg : { destinationRoutingAddress '2001'H, callingPartyNumber '1000'H }
TC_ContinueReq
52, oSCF

SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }

SigConA_Release
This Test Purpose is not included.

<table>
<thead>
<tr>
<th>IN2_A_BASIC_SF_BV_03</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
MSC IN2m_A_BASIC_SF_BV_03

1, 51, 1, ASF, medium, aSFArg : {
filteredCallTreatment : filteredBillingCharacteristics
informationToUser : tone (toneID 0, duration 66),
maximumNumberOfCounters 14,
releaseCause '001
filteredCriteria addressAndService : (calledAddressValue '2000H, serviceKey 28,
callingAddressValue '1000H, locationNumber '9001H))

TC_InvokeReq

1, 51, 1, ASF, medium, aSFArg : {
filteredCallTreatment : filteredBillingCharacteristics
informationToUser : tone (toneID 0, duration 66),
maximumNumberOfCounters 14,
releaseCause '001
filteredCriteria addressAndService : (calledAddressValue '2000H, serviceKey 28,
callingAddressValue '1000H, locationNumber '9001H))

TC_InvokeInd 1, 51, TRUE, ASF_R, aSFRArg : Null

SetupInd

{ callRef 1, calledPartyNumber '2000'H }

ReleaseReq

{ callRef 1, cause '101'H }

TC_BeginInd

51, oSSF, TRUE

StopTime_Is_reached

TC_InvokeReq

1, 52, 2, CUE, short, cUEArg : Null

TC_EndReq

52, basic

SetupReq

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

TC_BeginInd

53, oSSF, TRUE

TC_InvokeInd 1, 53, SFR, TRUE, sFRArg : Null

SetupInd

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

StopTime_Is_reached

TC_InvokeReq

1, 52, 2, CUE, short, cUEArg : Null

TC_EndReq

52, basic

SetupReq

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

TC_BeginInd

53, oSSF, TRUE

TC_InvokeInd 1, 53, SFR, TRUE, sFRArg : Null

SetupInd

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

StopTime_Is_reached

TC_InvokeReq

1, 52, 2, CUE, short, cUEArg : Null

TC_EndReq

52, basic

SetupReq

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

TC_BeginInd

53, oSSF, TRUE

TC_InvokeInd 1, 53, SFR, TRUE, sFRArg : Null

SetupInd

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

StopTime_Is_reached

TC_InvokeReq

1, 52, 2, CUE, short, cUEArg : Null

TC_EndReq

52, basic

SetupReq

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

TC_BeginInd

53, oSSF, TRUE

TC_InvokeInd 1, 53, SFR, TRUE, sFRArg : Null

SetupInd

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

StopTime_Is_reached

TC_InvokeReq

1, 52, 2, CUE, short, cUEArg : Null

TC_EndReq

52, basic

SetupReq

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

TC_BeginInd

53, oSSF, TRUE

TC_InvokeInd 1, 53, SFR, TRUE, sFRArg : Null

SetupInd

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

StopTime_Is_reached

TC_InvokeReq

1, 52, 2, CUE, short, cUEArg : Null

TC_EndReq

52, basic

SetupReq

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

TC_BeginInd

53, oSSF, TRUE

TC_InvokeInd 1, 53, SFR, TRUE, sFRArg : Null

SetupInd

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

StopTime_Is_reached

TC_InvokeReq

1, 52, 2, CUE, short, cUEArg : Null

TC_EndReq

52, basic

SetupReq

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

TC_BeginInd

53, oSSF, TRUE

TC_InvokeInd 1, 53, SFR, TRUE, sFRArg : Null

SetupInd

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

StopTime_Is_reached

TC_InvokeReq

1, 52, 2, CUE, short, cUEArg : Null

TC_EndReq

52, basic

SetupReq

{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }

TC_BeginInd

53, oSSF, TRUE
Purpose: test ServiceFiltering procedure on missing parameters

Selection Cond.

Preamble: none

Test description
SCF issues ActivateServiceFiltering invoke with missing parameter
- filteredCallTreatment

Pass criteria
- Check that SSF sends to SCF a ActivateServiceFiltering error with the indication of missing parameter

Postamble: None.

---

MSC IN2m_A_BASIC_SF_BI_01

![Diagram]

---

Purpose: test ServiceFiltering procedure with parameter out of range

Requirement ref

Selection Cond.

Preamble: none

Test description
SCF issues ActivateServiceFiltering invoke with parameter out of range
- filteringTimeOut with duration > 86400

Pass criteria
- Check that SSF sends to SCF a ActivateServiceFiltering error with the indication of out of range parameter

Postamble: None.
MSC IN2m_A_BASIC_SF_BI_02

TC_InvokeReq 1, 51, ASF, medium, aSFArg : {
  filteredCallTreatment { sFBillingChargingCharacteristics 'BBBB' H,
    informationToSend tone : { toneID 0, duration 66 },
    maximumNumberOfCounters 14, releaseCause '00' H },
  filteringTimeOut Duration 86400,
  filteringCriteria addressAndService : { calledAddressValue '2000' H, serviceKey 28,
    callingAddressValue '1000' H, locationNumber '9001' H } }

TC_ContinueReq 51, oSCF

TC_ErrorInd 1, 51, TRUE, parameterOutOfRangePar : Null

Purpose:

Test ServiceFiltering procedure in Monitoring state

In2_A_BASIC_SF_BO_01

Purpose: test ServiceFiltering procedure in Monitoring state

Requirement ref

Selection Cond.

Preamble: O_S2P

Test description

SCF issues ActivateServiceFiltering invoke containing mandatory and optional parameters, with:
- filteredCallTreatment including:
  - sFBillingChargingCharacteristics,
  - filteringTimeOut being duration,
  - filteringCriteria being addressAndService including:
    - calledAddressValue,
    - serviceKey,
    - callingAddressValue,
    - locationNumber
- filteringCharacteristics being interval

Pass criteria

SSF issues ServiceFiltering error with unexpectedComponentSequence parameter

Postamble: ReleaseCallAB_cause_00.
MSC IN2m_A_BASIC_SF_BO_01

TC_InvokeReq

1, 51, 1, ASF, medium, sSFArg : {
  filteredCallTreatment : {
    billingCharacteristics : "BBBB'"H,
    informationToSend : {
      tone : {
        toneID : 0, duration : 66
      },
      maximumNumberOfCounters : 14, releaseCause : "00'"H
    },
    filteringTimeOut : stopTime : "980220155555'H,
    filteringCriteria : {
      addressAndService : {
        calledAddressValue : "2000'H, serviceKey : 28,
        callingAddressValue : "1000'H, locationNumber : "9001'H"}
    }
  }
}

TC_ContinueReq

51, oSCF

TC_ErrorInd

1, 51, TRUE, unexpectedComponentSequencePar : Null

ReleaseCallAB_cause_00
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>test ServiceFiltering procedure in WaitForInstruction state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_OS</td>
</tr>
</tbody>
</table>
| Test description | SCF issues ActivateServiceFiltering invoke containing mandatory and optional parameters, with:  
- filteredCallTreatment including:  
  - sFBillingChargingCharacteristics,  
- filteringTimeOut being duration,  
  - filteringCriteria being addressAndService including:  
    - calledAddressValue,  
    - serviceKey,  
    - callingAddressValue,  
    - locationNumber  
- filteringCharacteristics being interval |
| Pass criteria | SSF issues ServiceFiltering error with unexpectedComponentSequence parameter |
| Postamble: | ReleaseCallA. |
6.4.2 ActivityTest procedure

**IN2_A_BASIC_AT_CA_01**

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of ActivityTest in monitoring state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_S2P</td>
</tr>
<tr>
<td>Test description</td>
<td>ActivityTest invoke sent by SCF to SSF, with TCAP DialogueId of dialogue identical to the one used in the preamble</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>ActivityTest result sent by SSF to SCF related to the existing dialogue</td>
</tr>
<tr>
<td>Postamble:</td>
<td>ReleaseCallAB_cause_0F.</td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_AT_CA_01

Purpose:
Test of ActivityTest in WaitForInstructions state

Requirement ref

Preamble:
O_OS

Selection Cond.

Test description
ActivityTest invoke sent by SCF to SSF with TCAP DialogueID of dialogue identical to the one used in the preamble

Pass criteria
ActivityTest result sent by SSF to SCF related to the existing dialogue

Postamble:
SigConA_Release.
Purpose: Test of ActivityTest in Idle state

Requirement ref

Selection Cond.

Preamble: none

Test description ActivityTest invoke sent by SCF to SSF with TCAP DialogueId of dialogue which is not existing

Pass criteria SSF rejects the invoke or aborts the dialogue (TCAP)

Postamble: none
MSC IN2m_A_BASIC_AT_BV_02

IN2_A_BASIC_AT_BI_01

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of invalid ActivityTest invoke</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_S2P</td>
</tr>
<tr>
<td>Test description</td>
<td>ActivityTest invoke with argument sent by SCF to SSF during preamble</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>SSF issues TC_error indicating unexpectedParameter</td>
</tr>
<tr>
<td>Postamble:</td>
<td>SigConA_release_then_B</td>
</tr>
</tbody>
</table>
6.4.3  ApplyCharging procedure

Charging related aspects in IN are network operator specific. Therefore, it is not possible to define useful test purposes for charging procedures using a network operator independent approach. TP specification has to be done by network operators, using INAP procedures themselves. The TPs could be specified by combining ApplyCharging, FurnishChargingInformation and SendChargingInformation procedures.

Examples:

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Test of <strong>ApplyCharging</strong> base procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Preamble</td>
<td>O_OS</td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Test description</td>
<td><strong>ApplyCharging</strong> invoke sent by SCF to SSF, containing mandatory parameters only, with: - aChBillingChargingCharacteristics, followed by <strong>Connect</strong> invoke containing mandatory parameters only - destinationRoutingAddress, As a consequence, SSF sends a <strong>SetupReq</strong> to SigCon B</td>
</tr>
<tr>
<td>Pass criteria</td>
<td><strong>ApplyCharging</strong> report sent by SSF to SCF upon reception of a <strong>Release</strong> indication from SigCon B</td>
</tr>
<tr>
<td>Postamble</td>
<td>none</td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_AC_CA_01

<table>
<thead>
<tr>
<th>SCF</th>
<th>CS2SSF</th>
<th>SigCon_A</th>
<th>SigCon_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_OS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TC_InvokeReq

1. 51, 2, AC, short, aCAArg : { aChBillingCharacteristics: "44'H, sendCalculationToSCPIndication FALSE } }

TC_InvokeReq

2. 51, 2, CON, short, cONArg : { destinationRoutingAddress: "2001'H } }

TC_ContinueReq

51, oSCF

SetupReq

[ { callRef 2, calledPartyNumber: "2001'H, callingPartyNumber: '1000'H } ]

ReleaseInd

[ { callRef 2, cause: '0D'H } ]

ReleaseReq

[ { callRef 1, cause: '0D'H, senderRef } ]

TC_EndInd

51, basic, TRUE

TC_InvokeInd

102, 51, ACR, TRUE, aCRArg : "H"
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_AC_BV_01</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of <strong>ApplyCharging</strong> procedure with optional parameter</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> O_OS</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
MSC IN2m_A_BASIC_AC_BV_01

TC_InvokeReq

1, 51, 2, AC, short, aCArg : \{ aChBillingChargingCharacteristics '44'H, partyToCharge SendingSideID : '01'H \}

TC_InvokeReq

2, 51, 2, CON, short, cONArg : \{ destinationRoutingAddress '2001'H \}

TC_ContinueReq

51, oSCF

SetupReq

\{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H \}

SetupConf

SetupResp

\{ callRef 2 \}

\{ callRef 1 \}

Triggering_of_conditions_for_apply_charging

TC_ContinueInd

51, oSSF, TRUE

TC_InvokeInd

102, 51, ACR, TRUE, aCRArg : ''

ReleaseCallAB_cause_00
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_AC_BV_02</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
MSC IN2m_A_BASIC_AC_BV_02

TC_InvokeReq
1, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '44'H, partyToCharge SendingSideID : '01'H }

TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

TC_ContinueReq
51, oSCF

SetupReq
SetupConf
SetupResp

Triggering_of_conditions_for_charging_report

SSF_has_detected_an_error_during_processing_of_applyChargingReport

TC_EndInd
51, basic, TRUE
TC_ErrorInd
1, 51, TRUE

SigConA_Release_thenB
| Purpose: | Test of ApplyCharging and Connect in same transaction |
| Requirement ref | |
| Selection Cond. | |
| Preamble: | O_OS |
| Test description | ApplyCharging invoke sent by SCF to SSF, containing mandatory parameters only, with: - aChBillingChargingCharacteristics, followed by Connect invoke containing mandatory parameters only - destinationRoutingAddress, As a consequence, SSF sends a SetupReq to SigCon B |
| Pass criteria | ApplyCharging report sent by SSF to SCF upon reception of a Release indication from SigCon A |
| Postamble: | none |
MSC IN2_A_BASIC_AC_BV_03

TC_InvokeReq
1, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '44'H, sendCalculationToSCPIndication FALSE }

TC_InvokeReq
3, 51, 2, CON, short, cONArg : { destinationRoutingAddress '2001'H }

TC_ContinueReq
51, oSCF

SetupReq

SetupResp

SetupConf

ReleaseInd

ReleaseReq
102, 51, ACR
TRUE, aCRArg :

TC_InvokeInd
51, basic, TRUE
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC_AC_BV_04</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of <strong>ApplyCharging</strong> when Sigcon B answers</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>
| **Test description** | **ApplyCharging** invoke sent by SCF to SSF, containing mandatory parameters only, with:  
- aChBillingChargingCharacteristics,  
followed by **Connect** invoke containing mandatory parameters only  
- destinationRoutingAddress,  
As a consequence, SSF sends a **SetupReq** to SigCon B  
SigCon B sends **SetupConf** |
| **Pass criteria** | **ApplyCharging** report sent by SSF to SCF upon reception of a **Release** indication from SigCon A |
| **Postamble:** | none |
MSC IN2_A_BASIC_AC_BV_04

SCF  CS2_SSF  SigCon_A  SigCon_B

SetupInd

\{ callRef 1, calledPartyNumber '2000' H, callingPartyNumber '1000' H \}
TC_BeginInd

51, oSSF, TRUE
TC_InvokeInd

\[ \text{callingPartyNumber '1000' H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1} \]
TC_InvokeReq

1, 51, AC, short, aCArg : \{ aChBillingChargingCharacteristics '44' H, sendCalculationToSCPIndication FALSE \}

TC_InvokeConf

2, 51, 2, CON, short, cONArg : \{ destinationRoutingAddress \{ '2001' H \} \}
TC_InvokeReq

51, oSCF

SetupResp

\{ callRef 1 \}
TC_EndInd

\{ callRef 2 \}
TC_InvokeInd

102, 51, ACR, TRUE, aCRArg : '-'
ReleaseReq

\{ callRef 2, cause '10' H \}
senderRef

ReleaseInd

\{ callRef 1, cause '00' H \}
TC_EndInd
**Purpose:** Test of **ApplyCharging** procedure with missing parameter

**Requirement ref**

**Selection Cond.**

**Preamble:** O_OS

**Test description**
SCF sends to SSF **ApplyCharging** invoke without the parameter - aChBillingChargingCharacteristics,

**Pass criteria**
SSF sends to SCF **TC_ErrorIndication** containing missingParameter error

**Postamble:** SigConA_Release

---

**MSC IN2m_A_BASIC_AC_BI_01**

```
<table>
<thead>
<tr>
<th>SCF</th>
<th>CS2_SSF</th>
<th>SigCon_A</th>
<th>SigCon_B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
O_OS

TC_InvokeReq

2, 51, 2, AC, short,
TC_ContinueReq

51, oSCF
TC_ContinueInd

[51, oSSF, TRUE]
TC_ErrorInd

[2, 51, TRUE, missingParameterPar : Null]

SigConA_Release
```
Purpose: Test of **ApplyCharging** procedure error with unknown LegID

Selection Cond.: CS-2 only

Preamble: O_OS

Test description:
- SCF sends to SSF **ApplyCharging** invoke with - partyToCharge being a not existing legid

Pass criteria:
- SSF sends to SCF **TC_errorInd** containing unknownLegID

Postamble: SigConA_Release
### IN2_A_BASIC_AC_BO_01

**Purpose:** Test of `ApplyCharging` procedure in wrong state

**Requirement ref:** none

**Selection Cond.:** none

**Preamble:** none

**Test description:** SCF sends to SSF `ApplyCharging` invoke from idle state

**Pass criteria:** SSF sends to SCF a `TC-ABORT`

**Postamble:** none

---

**MSC IN2m_A_BASIC_AC_BO_01**

![Call Flow Diagram]

- SCF
- CS2_SSF
- SigCon_A
- SigCon_B

- `TC_InvokeReq`
  - Parameters: `{ aChBillingChargingCharacteristics '44'H, sendCalculationToSCPIndication FALSE }

- `TC_ContinueReq`
  - Parameters: `[51, aSCF]`

- `TC_AbortInd`
  - Parameters: `[51]`
### 6.4.4 CallGap procedure

**IN2_A_BASIC_CG_CA_01**

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of CallGap base procedure and serviceKey parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF a CallGap invoke containing mandatory parameters only, with:</td>
</tr>
<tr>
<td></td>
<td>- gapCriteria:</td>
</tr>
<tr>
<td></td>
<td>- gapOnService with any valid value for serviceKey,</td>
</tr>
<tr>
<td></td>
<td>- gapIndicators</td>
</tr>
<tr>
<td></td>
<td>- duration being a duration value in seconds</td>
</tr>
<tr>
<td></td>
<td>- gapInterval being an interval value in seconds</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>- Check that SSF releases a call when callGapping is active for the service key used in SetupInd</td>
</tr>
<tr>
<td></td>
<td>- Check that SSF sends to SCF an InitialDP invoke as callgapping is NOT active when the service key in the SetupInd is different</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of interval, check that SSF sends an InitialDP invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>- cGEncountered</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of duration, check that SSF sends an InitialDP invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>but without &quot;cGEncountered&quot;</td>
</tr>
<tr>
<td>Postamble:</td>
<td>none</td>
</tr>
<tr>
<td>Purpose:</td>
<td>Test of CallGap procedure and calledAddressValue parameter</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Requirement ref</td>
<td>none</td>
</tr>
<tr>
<td>Selection Cond.</td>
<td>none</td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF a CallGap invoke containing mandatory parameters only, with:</td>
</tr>
<tr>
<td></td>
<td>- gapCriteria:</td>
</tr>
<tr>
<td></td>
<td>calledAddressValue with any valid value,</td>
</tr>
<tr>
<td></td>
<td>- gapIndicators</td>
</tr>
<tr>
<td></td>
<td>duration being a duration value in seconds</td>
</tr>
<tr>
<td></td>
<td>gapInterval being an interval value in seconds</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>- Check that SSF releases a call when callgapping is active for the called address used in SetupInd</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of interval, check that SSF sends an InitialDP invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>- cGEncountered</td>
</tr>
<tr>
<td></td>
<td>- Check that SSF releases a call coming within the second interval, as callgapping is still active for the called address used in SetupInd</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of duration, check that SSF sends an InitialDP invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>- and without &quot;cGEncountered&quot;</td>
</tr>
<tr>
<td>Postamble:</td>
<td>none</td>
</tr>
</tbody>
</table>
**MSC IN2m_A_BASIC.CG_BV_01**

- **Start interval and duration**
  - `TC_InvokeReq`
  - `TC_BeginReq`
  - `SetupInd`
  - `ReleaseReq`

- **Expiration of interval**
  - `TC_InvokeInd`
  - `TC_InvokeReq`
  - `TC_EndReq`

- **Expiration of duration and start of second interval**
  - `TC_InvokeInd`
  - `TC_InvokeReq`
  - `TC_EndReq`
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of CallGap procedure and calledAddressAndService parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF a CallGap invoke containing mandatory parameters only, with:</td>
</tr>
<tr>
<td></td>
<td>- gapCriteria:</td>
</tr>
<tr>
<td></td>
<td>calledAddressAndService with any valid value for serviceKey</td>
</tr>
<tr>
<td></td>
<td>- gapIndicators</td>
</tr>
<tr>
<td></td>
<td>duration being a duration value in seconds</td>
</tr>
<tr>
<td></td>
<td>gapInterval being an interval value in seconds</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>- Check that SSF releases a call when callgapping is active for the called address and service key used in SetupInd</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of interval, check that SSF sends an InitialDP invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>- cGEncountered</td>
</tr>
<tr>
<td></td>
<td>- Check that SSF releases a call coming within the second interval, as callgapping is still active for the called address and service key used in SetupInd</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of duration, check that SSF sends an InitialDP invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>- and without &quot;cGEncountered&quot;</td>
</tr>
<tr>
<td>Postamble:</td>
<td>none</td>
</tr>
</tbody>
</table>
Purpose: Test of CallGap procedure and optional parameter callingAddressAndService

Requirement ref

Selection Cond.

Preamble: none

Test description

SCF sends to SSF a CallGap invoke containing mandatory parameters with:
- gapCriteria: callingAddressAndService with any valid value,
- gapIndicators
  duration being a duration value in seconds
  gapInterval being an interval value in seconds

Pass criteria
- Check that SSF releases a call when callGaping is active for the calling address and service key used in SetupInd
- When a SetupInd comes with service key used in CallGap but not the calling address, check that SSF sends an InitialDP invoke containing all mandatory parameters
- When a SetupInd comes after expiration of interval, check that SSF sends an InitialDP invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:
  - serviceKey,
  - cGEncountered
- Check that SSF releases a call coming within the second interval, as callGaping is still active for the calling address and service key used in SetupInd
- When a SetupInd comes after expiration of duration, check that SSF sends an InitialDP invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:
  - serviceKey,
  but without "cGEncountered"

Postamble: none
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of <strong>CallGap</strong> procedure and optional parameters callingAddressAndService and gapTreatment via inbandInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td>none</td>
</tr>
<tr>
<td>Selection Cond.</td>
<td>none</td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
</tbody>
</table>
| Test description | SCF sends to SSF a **CallGap** invoke containing mandatory and optional parameters with:  
  - gapCriteria:  
    - callingAddressAndService with any valid value  
  - gapIndicators  
    duration being a duration value in seconds  
    gapInterval being an interval value in seconds  
  - gapTreatment  
    informationToSend being inbandInfo |
| Pass criteria | - Check that SSF releases a call when callgapping is active for the calling address and service key used in SetupInd. The inband information has to be sent to the calling party.  
  - When a SetupInd comes after expiration of interval, check that SSF sends an **InitialDP** invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:  
    - serviceKey,  
    - cGEncountered  
  - Check that SSF releases a call coming within the second interval, as callgapping is still active for the calling address and service key used in SetupInd. The inband information has to be sent to the calling party.  
  - When a SetupInd comes after expiration of duration, check that SSF sends an **InitialDP** invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:  
    - serviceKey,  
    but without "cGEncountered" |
| Postamble: | none |
MSC IN2m_A_BASIC.CG_BV_04
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC.CG_BV_05</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of CallGap procedure and optional parameters callingAddressAndService and gapTreatment with releaseCause</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> none</td>
</tr>
<tr>
<td><strong>Test description</strong> SCF sends to SSF a CallGap invoke containing mandatory and optional parameters with:</td>
</tr>
<tr>
<td>- gapCriteria:</td>
</tr>
<tr>
<td>callingAddressAndService</td>
</tr>
<tr>
<td>- gapIndicators</td>
</tr>
<tr>
<td>duration being a duration value in seconds</td>
</tr>
<tr>
<td>gapInterval being an interval value in seconds</td>
</tr>
<tr>
<td>- gapTreatment</td>
</tr>
<tr>
<td>releaseCause being any value except default value</td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td>- Check that SSF releases a call when callgapping is active for the calling address and service key used in SetupInd. The cause value has to be sent to the calling party</td>
</tr>
<tr>
<td>- When a SetupInd comes after expiration of interval, check that SSF sends an InitialDP invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:</td>
</tr>
<tr>
<td>- serviceKey,</td>
</tr>
<tr>
<td>- cGEncountered</td>
</tr>
<tr>
<td>- Check that SSF releases a call coming within the second interval, as callgapping is still active for the calling address and service key used in SetupInd. The cause value has to be sent to the calling party</td>
</tr>
<tr>
<td>- When a SetupInd comes after expiration of duration, check that SSF sends an InitialDP invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:</td>
</tr>
<tr>
<td>- serviceKey,</td>
</tr>
<tr>
<td>but without &quot;cGEncountered&quot;</td>
</tr>
<tr>
<td><strong>Postamble:</strong> none</td>
</tr>
<tr>
<td>IN2_A_BASIC.CG_BV_06</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
Start interval and duration

TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapCriteria callingAddressAndService : { callingAddressValue '1000H', serviceKey 27 }, gapIndicators duration 66, gapInterval 33 }, controlType manuallyInitiated

TC_BeginReq
51, sSCF

SetupInd
callRef 1, calledPartyNumber '2000H', callingPartyNumber '1000H'

ReleaseReq
{ callRef 1, cause '10H' }, senderRef

Expiration of interval

SetupInd
callRef 1, calledPartyNumber '2002H', callingPartyNumber '1002H'

ReleaseReq
{ callRef 1, cause '10H' }, senderRef

Expiration of duration

SetupInd
callRef 1, calledPartyNumber '2000H', callingPartyNumber '1000H'

ReleaseReq
{ callRef 1, cause '10H' }, senderRef
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC.CG_BV_07</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of CallGap procedure and serviceKey parameter</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
<td>SCF sends to SSF a CallGap invoke containing mandatory parameters only, with:</td>
</tr>
<tr>
<td></td>
<td>- gapCriteria:</td>
</tr>
<tr>
<td></td>
<td>- gapOnService with any valid value for serviceKey,</td>
</tr>
<tr>
<td></td>
<td>- gapIndicators</td>
</tr>
<tr>
<td></td>
<td>- duration being a duration value in seconds</td>
</tr>
<tr>
<td></td>
<td>- gapInterval being an interval value in seconds set to 0</td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Check that SSF releases a call when callGapping is active for the service key used in SetupInd</td>
</tr>
<tr>
<td></td>
<td>- Check that SSF sends to SCF an InitialDP invoke as callgapping is NOT active when the service key in the SetupInd is different</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of interval, check that SSF sends an InitialDP invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>- cGEncountered</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of duration, check that SSF sends an InitialDP invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>but without &quot;cGEncountered&quot;</td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of <strong>CallGap</strong> procedure and <strong>serviceKey</strong> parameter</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>

**Test description**

SCF sends to SSF a **CallGap** invoke containing mandatory parameters only, with:
- **gapCriteria:**
  - **gapOnService** with any valid value for **serviceKey**,  
  - **gapIndicators**
    - duration being a duration value in seconds set to 0  
    - **gapInterval** being an interval value in seconds

**Pass criteria**

- Check that SSF releases a call when callGapping is active for the service key used in **SetupInd**
- Check that SSF sends to SCF an **InitialDP** invoke as callgapping is NOT active when the service key in the **SetupInd** is different
- When a **SetupInd** comes after expiration of interval, check that SSF sends an **InitialDP** invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:
  - **serviceKey**,  
  - **cGEncountered**
- When a **SetupInd** comes after expiration of duration, check that SSF sends an **InitialDP** invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:
  - **serviceKey**,  
  but without "cGEncountered"

**Postamble:**

none
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC.CG_BV_09</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of <strong>CallGap</strong> procedure in <strong>WaitForInstruction</strong> state</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>O_OS</td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF a **CallGap** invoke containing mandatory parameters only, with:  
- gapCriteria:  
  - gapOnService with any valid value for serviceKey,  
- gapIndicators  
  - duration being a duration value in seconds  
  - gapInterval being an interval value in seconds |
| **Pass criteria** | - Check that SSF releases a call when callgapping is active for the service key used in SetupInd  
- Check that SSF sends to SCF an **InitialDP** invoke as callgapping is NOT active when the service key in the SetupInd is different  
- When a SetupInd comes after expiration of interval, check that SSF sends an **InitialDP** invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:  
  - serviceKey,  
  - cGEncountered  
- When a SetupInd comes after expiration of duration, check that SSF sends an **InitialDP** invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:  
  - serviceKey,  
  but without "cGEncountered" |
<p>| <strong>Postamble:</strong> | none |</p>
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of CallGap procedure in Monitoring state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_S2P</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF a CallGap invoke containing mandatory parameters only, with:</td>
</tr>
<tr>
<td></td>
<td>- gapCriteria:</td>
</tr>
<tr>
<td></td>
<td>gapOnService with any valid value for serviceKey,</td>
</tr>
<tr>
<td></td>
<td>gapIndicators</td>
</tr>
<tr>
<td></td>
<td>duration being a duration value in seconds</td>
</tr>
<tr>
<td></td>
<td>gapInterval being an interval value in seconds</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>- Check that SSF releases a call when callGapping is active for the service key used in SetupInd</td>
</tr>
<tr>
<td></td>
<td>- Check that SSF sends to SCF an InitialDP invoke as callgapping is NOT active when the service key in the SetupInd is different</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of interval, check that SSF sends an InitialDP invoke containing all mandatory parameters and indicating call gapping encountered, with at least the parameters:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>- cGEncountered</td>
</tr>
<tr>
<td></td>
<td>- When a SetupInd comes after expiration of duration, check that SSF sends an InitialDP invoke containing all mandatory parameters without indicating call gapping encountered, with at least the parameter:</td>
</tr>
<tr>
<td></td>
<td>- serviceKey,</td>
</tr>
<tr>
<td></td>
<td>but without &quot;cGEncountered&quot;</td>
</tr>
<tr>
<td>Postamble:</td>
<td>none</td>
</tr>
<tr>
<td><strong>IN2_A_BASIC.CG_BI_01</strong></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of CallGap procedure and missing parameter</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>none</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
<td>SCF sends to SSF a CallGap invoke with missing mandatory parameter gapCriteria</td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
<td>Check that SSF rejects the invoke</td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
<td>none</td>
</tr>
</tbody>
</table>
### CallInformation procedure

<table>
<thead>
<tr>
<th>IN2_A_BASIC_CF_CA_01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of CallInformation procedure</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> O_OS</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Postamble:</strong> none</td>
</tr>
</tbody>
</table>
MSC IN2m_A_BASIC_CF_CA_01

1, 51, 2, CIRQ, short: cIRQArg : { requestedInformationTypeList ( releaseCause ) }

2, 51, 2, CON, short: cONArg : { destinationRoutingAddress ( '2001'H ) }

{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }

{ callRef 2 }

{ callRef 1 }

{ callRef 1, cause '10'H }

51, oSSF, TRUE

51, oSSF

51, oSSF

51, basic, FALSE

null

null

null

null

null
<table>
<thead>
<tr>
<th>IN2_A_BASIC_CF_BV_01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
O_OS

TC_InvokeReq

1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { releaseCause } }

TC_InvokeReq

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress {'2001'} } 

SetupReq

{ callRef 2, calledPartyNumber {'2001'}, callingPartyNumber {'1000'} } 

ReleaseInd

TC_InvokeInd

102, 51, CIR, TRUE, cIRArg : { requestedInformationList { { requestedInformationType, requestedInformationValue : {'06'} } } }

ReleaseInd

{ callRef 2, cause {'06'} }

51, basic, FALSE 

TC_EndInd
<table>
<thead>
<tr>
<th><strong>Purpose:</strong></th>
<th>Test of CallInformation procedure and time parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **Preamble:** | O_OS  
In addition, user B is declared not answering within timeout |
| **Test description** | SCF sends to SSF a CallInformationRequest invoke, containing mandatory parameters only and indicating a single information type, with at least the parameters:  
- requestedInformationTypeList including:  
  - requestedInformationType (callAttemptElapsedTime),  
  followed by a Continue to establish a Connection with SigConB  
But the connection is not established, as B does not answer |
| **Pass criteria** | - Check that upon detection of SSF timer expiration, SSF sends CallInformationReport with at least the parameters  
  - requestedInformationList including:  
    - requestedInformationType (callAttemptElapsedTime)  
    - requestedInformationValue being callAttemptElapsedTimeValue, |
| **Postamble:** | none |
TC_InvokeReq
1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { callAttemptElapsedTime } }
TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001' } }
SetupReq
{ callRef 2, calledPartyNumber '2001', callingPartyNumber '1000' }
ReleaseInd
{ callRef 2, cause '09' }
TC_InvokeInd
102, 51, CIR, TRUE, cIRArg : { requestedInformationList { { requestedInformationType callAttemptElapsedTime, requestedInformationValue callAttemptElapsedTimeValue : 70 } } }
TC_EndInd
51, basic, FALSE
ReleaseReq
{ callRef 2, cause '09', senderRef }
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of CallInformation procedure and elapsed time parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_OS</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF a CallInformationRequest invoke, containing mandatory parameters and indicating a multiple information type, with at least the parameters:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationTypeList including:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationType (callAttemptElapsedTime),</td>
</tr>
<tr>
<td></td>
<td>also including:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationType (callStopTime),</td>
</tr>
<tr>
<td></td>
<td>also including:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationType (callConnectedElapsedTime),</td>
</tr>
<tr>
<td></td>
<td>and including:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationType (calledAddress),</td>
</tr>
<tr>
<td></td>
<td>followed by a Continue to establish a Connection with SigConB</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>- Check that upon detection of a release from SigConA, SSF sends CallInformationReport to SCF and indicating a multiple information type, with at least the parameters:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationList including:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationType (callAttemptElapsedTime),</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationValue being callAttemptElapsedTimeValue,</td>
</tr>
<tr>
<td></td>
<td>also including:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationType (callStopTime),</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationValue being callStopTimeValue,</td>
</tr>
<tr>
<td></td>
<td>also including:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationType (callConnectedElapsedTime),</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationValue being callConnectedElapsedTimeValue,</td>
</tr>
<tr>
<td></td>
<td>and including:</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationType (calledAddress),</td>
</tr>
<tr>
<td></td>
<td>- requestedInformationValue being calledAddressValue</td>
</tr>
<tr>
<td>Postamble:</td>
<td>none</td>
</tr>
</tbody>
</table>
TC_InvokeReq
1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { callAttemptElapsedTime, callStopTime, callConnectedElapsedTime, calledAddress } }

SetupReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { ’2001’H } }

TC_ContinueReq
51, oSCF

SetupConf

ReleaseInd
[ { callRef 1 } ]

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeReq
102, 51, 4, CUE, medium, cUEArg : Null

TC_ContinueReq
51, oSSF

TC_EndInd
51, basic, FALSE

ReleaseReq
[ { callRef 2, cause ’10’H }, senderRef ]
## Purpose:
Test of **CallInformation** procedure combined with **RequestReportBCSMEvent** procedure.

## Requirement ref

## Selection Cond.

## Preamble:
O_OS

## Test description
SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters
- eventTypeBCSM= oDisconnect
- monitoringMode=interrupted

then SCF sends a **CallInformationRequest** invoke, containing mandatory parameters only, with at least the parameters:
- requestedInformationTypeList including:
- requestedInformationType being releaseCause,
followed by a **Connect** invoke and SSF establishes the call
The call is answered (SigCon B sends **SetupConf**)
SigCon A (calling party) clears the call after it is answered (ReleaseInd sent)

## Pass criteria
- Check that upon detection of call release, SSF sends **CallInformationReport** with at least the parameters
- requestedInformationList including:
- requestedInformationType being releaseCause,
- requestedInformationValue being releaseCauseValue used
then Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM**= oDisconnect

## Postamble:
none
### Purpose:
Test of **CallInformation** procedure combined with **RequestReportBCSMEvent** procedure.

### Requirement ref

### Selection Cond.

### Preamble:
O_OS

### Test description
SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters
- **eventTypeBCSM= oAnswer**
- **monitoringMode=interrupted**
then SCF sends a **CallInformationRequest** invoke, containing mandatory parameters only, with at least the parameters:
  - **requestedInformationTypeList including:**
  - **requestedInformationType being releaseCause,**
followed by a **Connect** invoke and SSF establishes the call
The call is answered (SigCon B sends **SetupConf**) SIGCon A (calling party) clears the call after it is answered (ReleaseInd sent)

### Pass criteria
- Check that when SigConB is answering, SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM= oAnswer**
- Check that upon detection of call release from SigConA, SSF sends **CallInformationReport** with at least the parameters
  - **requestedInformationList including:**
  - **requestedInformationType being releaseCause,**
  - **requestedInformationValue being releaseCauseValue used**

### Postamble:
none
Purpose: Test of CallInformation procedure with invalid parameters

Requirement ref

Selection Cond.

Preamble: O_OS

Test description: SCF sends to SSF a CallInformationRequest invoke, with
- RequestedInformationTypeList, containing an invalid parameter
  followed by a Continue to establish a Connection with SigConB

Pass criteria: Check that SSF sends back CallInformationRequest error to SCF
  indicating: requestedInfoError

Postamble: SigConA_Release

---

MSC IN2m_A_BASIC_CF_BI_01

[Diagram of the MSC IN2m_A_BASIC_CF_BI_01 with nodes SCF, CS2_SSF, SigCon_A, SigCon_B, and transitions TC_InvokeReq, TC_InvokeReq, TC_ContinueReq, TC_ContinueInd, TC_ErrorInd, and SigConA_Release]
<table>
<thead>
<tr>
<th><strong>Purpose:</strong></th>
<th>Test of CallInformation procedure in wrong state (idle)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
<td>SCF sends CallInformationRequest invoke to SSF</td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
<td>Check that SSF sends to SCF a TC_ABORT</td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

**MSC IN2m_A_BASIC_CF_BO_01**

```
SCF         CS2_SSF  SigCon_A  SigCon_B
     ↓       ↓         ↓         ↓
TC_InvokeReq  | 1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { releaseCause } }
     ↓       ↓         ↓         ↓
TC_BeginReq   51, oSCF
     ↓         ↓         ↓         ↓
TC_AbortInd   51
```
IN2_A_BASIC_CF_BO_02

| Purpose: | Test of CallInformation procedure in wrong state (monitoring) |
| Requirement ref | |
| Selection Cond. | |
| Preamble: | O_S2P |
| Test description | SCF sends CallInformationRequest invoke to SSF |
| Pass criteria | Check that SSF sends to SCF a CallInformationRequest error with an indication of UnexpectedComponentSequence |
| Postamble: | SigConA_Release_thenB |

MSC IN2m_A_BASIC_CF_BO_02

```
[3, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { callStopTime } }]
```

```
[51, oSCF]
```

```
[51, oSCF, true]
```

```
[3, 51, TRUE, unexpectedComponentSequence Par : Null]
```

SigConA_Release

```
```

SCF CS2_SSF SigCon_A SigCon_B

O_S2P

TC_InvokeReq

TC_ContinueReq

TC_ContinueInd

TC_ErrorInd

SigConA_Release

```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```
6.4.6 Cancel procedure

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of Cancel base procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_S2P</td>
</tr>
<tr>
<td>Test description</td>
<td>Cancel invoke sent by SCF to SSF, containing allRequests</td>
</tr>
</tbody>
</table>
| Pass criteria | - Check that SSF returns to idle state  
                - To ensure that SSF is now in idle state, SCF sends ActivityTest invoke to SSF with Dialogueld used in InitialDP. SSF rejects or aborts the invoke as dialogue is not used any more |
| Postamble: | SigConA_Release_thenB |
MSC IN2_A_BASIC_CA_CA_01

TC_InvokeReq
5, 51, 4, CAN, short, cANArg : allRequests : Null

TC_EndReq
51, basic

TC_InvokeReq
6, 51, 1, AT, short, aTArg : Null

TC_ContinueReq
51, oSCF

TC_AbortInd
51

SigConA_Release_thenB
This test purpose was dropped.

### IN2_A_BASIC_CA_BV_02

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Test of <strong>Cancel</strong> procedure on RequestNotificationChargingEvent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_OS</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF <strong>RequestNotificationChargingEvent</strong> invoke containing mandatory parameters only, with:</td>
</tr>
<tr>
<td></td>
<td>- eventTypeCharging,</td>
</tr>
<tr>
<td></td>
<td>- monitorMode (interrupted)</td>
</tr>
<tr>
<td></td>
<td>After triggering of charging event from SigConA, SSF sends to SCF an <strong>EventNotificationCharging</strong> invoke with the indication of eventTypeCharging</td>
</tr>
<tr>
<td></td>
<td>SCF sends to SSF <strong>Continue</strong> invoke then a new <strong>RequestNotificationChargingEvent</strong> invoke containing mandatory parameters only, with:</td>
</tr>
<tr>
<td></td>
<td>- eventTypeCharging,</td>
</tr>
<tr>
<td></td>
<td>- monitorMode (interrupted)</td>
</tr>
<tr>
<td></td>
<td>followed by a <strong>Cancel</strong> invoke containing allRequests</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>- Check that SSF cancels the request for an <strong>EventNotificationCharging</strong> and <strong>does</strong> not send it to SCF when the calling party (SigConA) triggers the charging event.</td>
</tr>
<tr>
<td>Postamble:</td>
<td>SigConA_Release_thenB</td>
</tr>
</tbody>
</table>
### IN2_A_BASIC_CA_BV_03

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of <strong>Cancel</strong> procedure on <strong>ApplyCharging</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td><strong>O_OS</strong></td>
</tr>
<tr>
<td><strong>Test description</strong></td>
<td>SCF sends to SSF <strong>ApplyCharging</strong> invoke containing mandatory parameter <strong>aChBillingChargingCharacteristics</strong> followed by a <strong>Connect</strong> to establish a <strong>Connection</strong> with <strong>SigConB</strong> When the connection is established, <strong>Cancel</strong> invoke is sent by SCF to SSF, containing allRequests</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>- Check that SSF cancels the request for an <strong>ApplyChargingReport</strong> and <strong>does not</strong> send it to SCF when the calling party (<strong>SigConA</strong>) releases the call.</td>
</tr>
<tr>
<td>Postamble:</td>
<td>none</td>
</tr>
</tbody>
</table>
SCF  CS2_SSF  SigCon_A  SigCon_B

O_OS

1, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '44'H, sendCalculationToSCPIndication FALSE }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress '2001'H }

51, oSCF

SetupReq

callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H

SetupConf

SetupResp

{ callRef 2 }

TC_InvokeReq

3, 51, 4, CAN, short, cANArg : allRequests : Null

TC_EndReq

51, basic

ReleaseInd

{ callRef 1, cause '10'H }

ReleaseReq

{ callRef 2, cause '10'H, senderRef }

MSC IN2_A_BASIC_CA_BV_03
<table>
<thead>
<tr>
<th>IN2_A_BASIC_CA_BV_04</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_CA_BV_04

1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { callAttemptElapsedTime } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

51, oSCF

{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }

SetupConf

{ callRef 2 }

{ callRef 1 }

3, 51, 4, CAN, short, cANArg : { InvokeID : 1 }

TC_InvokeReq

TC_ContinueReq

TC_InvokeReq

TC_InvokeReq

51, basic

SigConA_Release_thenB_cause10
**Purpose:** Test of **Cancel** error procedure with **cancelFailed**

** Requirement ref**

**Selection Cond.**

**Preamble:** O_OS

**Test description** **Cancel** invoke is sent by SCF to SSF, containing invokeID being not existing operation invokeId

**Pass criteria** - Check that SSF sends to SCF **Cancel** with error **cancelFailed**

**Postamble:** SigConA_Release

---

### MSC IN2m_A_BASIC_CA_BI_01

```
O_OS

TC_InvokeReq
[2, 51, 2, CAN, short, cANArg : invokeID : 45 }]

TC_ContinueReq
[51, oSCF]
TC_ContinueInd
[51, oSCF, true]
TC_ErrorInd
[2, 51, TRUE, cancelFailedPar : { problem unknownOperation, operation 2 }]

SigConA_Release
```
### IN2_A_BASIC_CA_BO_01

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Test of <strong>Cancel</strong> procedure in wrong (idle) state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
<tr>
<td>Test description</td>
<td><strong>Cancel</strong> invoke sent by SCF to SSF, containing invokeID</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>Check that SSF sends a <strong>TC-ABORT</strong></td>
</tr>
<tr>
<td>Postamble:</td>
<td>none</td>
</tr>
</tbody>
</table>

#### MSC IN2m_A_BASIC_CA_BO_01

```
SCF  | CS2SSF  | SigCon_A | SigCon_B |
TC_InvokeReq | 1, 51, 2, CAN, short, cANArg : InvokeID : 45 |
TC_BeginReq   | 51, oSCF |
TC_AbortInd    | [51]     |
```
### 6.4.7 CollectInformation procedure

<table>
<thead>
<tr>
<th><strong>Purpose:</strong></th>
<th>Test of CollectInformation base procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **Preamble:** | O_OS  
Preamble contains an InitialDP without complete digits for CalledPartyNumber |
| **Test description** | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM=collectedInfo  
- monitoringMode=interrupted  
followed by CollectInformation invoke  
then the calling party sends the remaining digits (after CallProgressReq is received and SubsequentAddressInd and AddressEndInd is sent) |
| **Pass criteria** | Check that SSF sends to SCF an EventReportBCSM with the indication of eventTypeBCSM=collectedInfo, together with the remaining called party digits |
| **Postamble:** | SigConA_Release |
### IN2_A_BASIC_CI_BI_01

**Purpose:** Test of CollectInformation procedure and unknown CallsegmentId

**Requirement ref**

**Selection Cond.** CS-2 only

**Preamble:** O_OS

**Test description**
- SCF - SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
  - eventTypeBCSM= collectedInfo
  - monitoringMode=interrupted
- followed by CollectInformation invoke with
  - callSegmentID being a not existing call segment

**Pass criteria**
- Check that SSF sends to SCF a TC_ErrorInd with unexpectedParameter

**Postamble:** SigConA_Release

---

### MSC IN2m_A_BASIC_CI_BI_01

```
SCF       CS2_SSF   SigCon_A     SigCon_B

O_OS

TC_InvokeReq
[1, 51, 2, RRB, short, rRBArg : { bcsmeEvents { { eventTypeBCSM collectedInfo, monitorMode interrupted } } ]

TC_InvokeReq
[2, 51, 2, Ci, medium,]

TC_ContinueReq
[51, oSCF]

TC_ContinueInd
[51, oSSF, TRUE]

TC_ErrorInd
[2, 51, TRUE, unexpectedParameterPar : Null]

SigConA_Release
```

---
### IN2_A_BASIC_CI_BO_01

**Purpose:** Test of `CollectInformation` procedure in wrong sequence

**Requirement ref**

**Selection Cond.**

**Preamble:**
- `O_OS` 
  - Preamble contains an InitialDP without complete digits for CalledPartyNumber

**Test description**

- SCF sends `CollectInformation` invoke to SSF without sending before any `RequestReportBCSMEvent` invoke

**Pass criteria**

- Check that SSF sends to SCF a `CollectInformation` error with an indication of UnexpectedComponentSequence

**Postamble:** ReleaseCallA

---

### MSC IN2m_A_BASIC_CI_BO_01

```
+-----------------+         +-----------------+         +-----------------+         +-----------------+
<table>
<thead>
<tr>
<th>SCF</th>
<th>CS2_SSF</th>
<th>SigCon_A</th>
<th>SigCon_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC_InvokeReq</td>
<td></td>
<td>TC_ContinueReq</td>
<td></td>
</tr>
<tr>
<td>2, 51, 2, Cl, medium, cIArg : {}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_ContinueReq</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[51, oSSF]</td>
<td></td>
<td>TC_ContinueInd</td>
<td></td>
</tr>
<tr>
<td>[51, oSSF, TRUE]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_ErrorInd</td>
<td></td>
<td>ReleaseCallA</td>
<td></td>
</tr>
<tr>
<td>[2, 51, TRUE, unexpectedComponentSequencePar : Null]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

---
### IN2_A_BASIC_CI_BO_02

**Purpose:** Test of `CollectInformation` procedure in wrong state (idle state)

**Requirement ref**

**Selection Cond.**

**Preamble:** none

**Test description** SCF sends `CollectInformation` invoke to SSF from idle state

**Pass criteria** Check that SSF sends to SCF a `TC-ABORT`

**Postamble:** None

---

### MSC IN2m_A_BASIC_CI_BO_02

![Diagram](image)

---

### IN2_A_BASIC_CI_BO_03

**Purpose:** Test of `CollectInformation` procedure in wrong state (monitoring state)

**Requirement ref**

**Selection Cond.**

**Preamble:** O_S2P

**Test description** SCF sends `CollectInformation` invoke to SSF from Monitoring state

**Pass criteria** Check that SSF sends to SCF a `CollectInformation` error with an indication of UnexpectedComponentSequence

**Postamble:** `ReleaseCallAB_cause_00`
6.4.8 Connect procedure

<table>
<thead>
<tr>
<th>IN2_A_BASIC_CO_CA_01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
**Purpose:**
Test of **Connect** procedure with alertingPattern parameter

**Requirement ref**

**Selection Cond.**

**Preamble:**
O_OS

**Test description**
SCF sends to SSF **Connect** invoke with mandatory and optional parameters, with destinationRoutingAddress alertingPattern
SSF sends a SetupRequest to B side

**Pass criteria**
Check that the connect operation is not rejected

**Postamble:**
SigConA_Release_thenB
**Purpose:** Test of Connect procedure with cutAndPaste parameter

**Requirement ref**

**Selection Cond.**

**Preamble:** O_OS

**Test description**

SCF sends to SSF Connect invoke with mandatory and optional parameters, with
destinationRoutingAddress

cutAndPaste

SSF sends a SetupRequest to B side

**Pass criteria**

Check that the Connect operation is not rejected

**Postamble:** SigConA_Release_thenB

---

**MSC IN2_A_BASIC_CO_BV_01**

**IN2_A_BASIC_CO_BV_02**

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of Connect procedure with cutAndPaste parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_OS</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF Connect invoke with mandatory and optional parameters, with destinationRoutingAddress cutAndPaste SSF sends a SetupRequest to B side</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>Check that the Connect operation is not rejected</td>
</tr>
<tr>
<td>Postamble:</td>
<td>SigConA_Release_thenB</td>
</tr>
</tbody>
</table>
**MSC IN2_A_BASIC_CO_BV_02**

**Purpose:** Test of **Connect** procedure with supplementary services parameters

**Requirement ref**

**Selection Cond.**

**Preamble:** O_OS

**Test description** SCF sends to SSF **Connect** invoke containing parameters related to supplementary services, with:
- destinationRoutingAddress
- originalCalledPartyID
- redirectingPartyID
- redirectionInformation

SSF sends a SetupRequest to B side

**Pass criteria** Check that the above parameters are mapped from Connect into the Set-up request:
- destinationRoutingAddress -> calledPartyNumber
- originalCalledPartyID -> originalCalledNumber
- redirectingPartyID -> redirectingNumber
- redirectionInformation

**Postamble:** SigConA_Release_thenB

---

**IN2_A_BASIC_CO_BV_03**

**Purpose:** Test of **Connect** procedure with supplementary services parameters

**Requirement ref**

**Selection Cond.**

**Preamble:** O_OS

**Test description** SCF sends to SSF **Connect** invoke containing parameters related to supplementary services, with:
- destinationRoutingAddress
- originalCalledPartyID
- redirectingPartyID
- redirectionInformation

SSF sends a SetupRequest to B side

**Pass criteria** Check that the above parameters are mapped from Connect into the Set-up request:
- destinationRoutingAddress -> calledPartyNumber
- originalCalledPartyID -> originalCalledNumber
- redirectingPartyID -> redirectingNumber
- redirectionInformation

**Postamble:** SigConA_Release_thenB
Purpose: Test of Connect procedure with optional parameters related to the calling party

Requirement ref

Selection Cond.

Preamble: O_OS

Test description
- SCF sends to SSF Connect invoke containing mandatory and optional parameters
  - destinationRoutingAddress
  - callingPartyNumber
  - callingPartysCategory

SSF sends a SetupRequest to B side

Pass criteria
- Check that the above parameters are mapped from Connect into the Set-up request:
  - destinationRoutingAddress -> calledPartyNumber
  - callingPartyNumber -> callingPartyNumber
  - callingPartysCategory -> callingPartysCategory

Postamble: SigConA_Release_thenB
MSC IN2_A_BASIC_CO_BV_04

---

### Purpose:
Test of Connect procedure with optional parameter routeList

### Requirement ref

### Selection Cond.

### Preamble:
O_Os

### Test description
SCF sends to SSF Connect invoke containing mandatory and optional parameters:
- destinationRoutingAddress
- routeList (with two different routes)

SSF sends a SetupRequest to B side.
B Side sends a RelInd with release cause being routeSelectFailure.

### Pass criteria
Check that if the first route fails the IUT sends a second SetUpReq to B side using the second route.

### Postamble:
SigConA_Release_thenB
**Purpose:**
Test of Connect procedure with optional parameter serviceInteractionIndicators

**Requirement ref**

**Selection Cond.**

**Preamble:**
O_OS

**Test description**
SCF sends to SSF Connect invoke containing mandatory and optional parameters
- destinationRoutingAddress,
- serviceInteractionIndicators
SSF sends a SetupRequest to B side

**Pass criteria**
Check that the above parameters are mapped from Connect into the Set-up request:
- destinationRoutingAddress\(\rightarrow\) calledPartyNumber,
- serviceInteractionIndicators\(\rightarrow\) serviceIndicators

**Postamble:**
SigConA_Release_thenB
**Purpose:** Test of Connect procedure received on each allowed DP of O_BCSM

**Requirement ref**

**Selection Cond.**

**Preamble:** O_S2P

**Test description**
- SCF sends a RRB operation to arm routeSelectFailure event
- SigConB sends a ReleaseInd with release cause being routeSelectFailure
- SCF sends to SSF Connect invoke containing mandatory parameter - destinationRoutingAddress.

**Pass criteria**
- Check that SSF sends a SetupReq to B side

**Postamble:** ReleaseCallAB_cause_0F
3, 51, 2, RRB, short, rRBArg : { bsCallEvents { { eventTypeBCSM routeSelectFailure, monitorMode interrupted } } }

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM routeSelectFailure, eventSpecificInformationBCSM routeSelectFailureSpecificInfo :{ failureCause '0C'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

4, 51, 2, CON, medium, cONArg : { destinationRoutingAddress '2003'H }
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of <strong>Connect</strong> procedure in response to <strong>oNoAnswer</strong> DP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td><strong>O_OS</strong></td>
</tr>
</tbody>
</table>
| Test description | SCF sends a RRB operation to arm **oNoAnswer** event  
SCF sends a **Connect** operation with mandatory parameters  
**SigConB** sends a **ReleaseInd** with release cause being **bPtyNoAnswer**  
**Scs SCF** sends to **SSF** **Connect** invoke containing mandatory parameters  
- **destinationRoutingAddress**, |
| Pass criteria | Check that **SSF** sends a **SetupReq** to **B side** |
| Postamble: | **SigConA_Release_thenB** |
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { 2001'H } }

1, 0, SCF

SetupReq

{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }

ReleaseInd

{ callRef 2, cause '09'H }

51, oSSF, TRUE

TC_ContinueInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

5, 51, 2, CON, short, cONArg : { destinationRoutingAddress { 2003'H } }

TC_EndReq

51, basic

SetupReq

{ callRef 2, calledPartyNumber '2003'H, callingPartyNumber '1000'H }

SigConA_Release_thenB
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC_CO_BV_09</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of Connect procedure in response to oCalledPartyBusy DP.</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>O_OS</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
<td>SCF sends a RRB operation to arm oCalledPartyBusy event SC SCF sends a Connect operation with mandatory parameters SigConB sends a ReleaseInd with release cause being bPtyBusy_UDUB SCF sends to SSF Connect invoke containing mandatory parameter - destinationRoutingAddress,</td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
<td>Check that SSF sends a SetupReq to B side</td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
<td>SigConA_Release_thenB</td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_CO_BV_09

1, 51, 2, RRB, short, rRBArg : { bcsmeEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

51, oSCF

SetupReq

1, 51, 2, RRB, short, rRBArg : { bcsmeEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

51, oSCF

TC_InvokeReq

ReleaseInd

51, oSCF

TC_InvokeInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oCalledPartyBusy, eventSpecificInformationBCSM oCalledPartyBusySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2003'H } }

TC_EndReq

51, basic

SetupReq

1, 51, 2, RRB, short, rRBArg : { bcsmeEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

51, oSCF

TC_InvokeReq

ReleaseInd

51, oSCF

TC_InvokeInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oCalledPartyBusy, eventSpecificInformationBCSM oCalledPartyBusySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2003'H } }

TC_EndReq

51, basic

SetupReq

1, 51, 2, RRB, short, rRBArg : { bcsmeEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

51, oSCF

TC_InvokeReq

ReleaseInd

51, oSCF

TC_InvokeInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oCalledPartyBusy, eventSpecificInformationBCSM oCalledPartyBusySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2003'H } }

TC_EndReq

51, basic

SetupReq

1, 51, 2, RRB, short, rRBArg : { bcsmeEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

51, oSCF

TC_InvokeReq

ReleaseInd

51, oSCF

TC_InvokeInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oCalledPartyBusy, eventSpecificInformationBCSM oCalledPartyBusySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2003'H } }

TC_EndReq

51, basic

SetupReq

1, 51, 2, RRB, short, rRBArg : { bcsmeEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

51, oSCF

TC_InvokeReq

ReleaseInd

51, oSCF

TC_InvokeInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oCalledPartyBusy, eventSpecificInformationBCSM oCalledPartyBusySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2003'H } }

TC_EndReq

51, basic

SetupReq

1, 51, 2, RRB, short, rRBArg : { bcsmeEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

51, oSCF

TC_InvokeReq

ReleaseInd

51, oSCF

TC_InvokeInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oCalledPartyBusy, eventSpecificInformationBCSM oCalledPartyBusySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2003'H } }

TC_EndReq

51, basic

SetupReq
### IN2_A_BASIC_CO_BI_01

**Purpose:** Test of Connect procedure with invalid parameter

**Requirement ref**

**Selection Cond.**

**Preamble:** O_OS

**Test description**
SCF sends to SSF Connect invoke containing mandatory parameters with an invalid value in
- destinationRoutingAddress.

**Pass criteria**
Check that SSF sends back Connect error, with error parameter UnexpectedDataValue

**Postamble:** SigConA_Release

### MSC IN2m_A_BASIC_CO_BI_01

<table>
<thead>
<tr>
<th>SCF</th>
<th>CS2_SSF</th>
<th>SigCon_A</th>
<th>SigCon_B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
O_OS

TC_InvokeReq

3, 51, 2, CON, short, cONArg : { destinationRoutingAddress : 'AA20' } } ]

TC_ContinueReq

51, oSCF

TC_ContinueInd

51, oSSF, TRUE

TC_ErrorInd

3, 51, TRUE, unexpectedDataValue

SigConA_Release
```

---

**ETSI**
### IN2_A_BASIC_CO_BO_01

**Purpose:** Test of `Connect` procedure initiated from wrong state

**Requirement ref**

**Selection Cond.**

**Preamble:** O_S2P

**Test description** SCF sends to SSF `Connect` invoke containing mandatory parameters, while in monitoring state

**Pass criteria** Check that SSF sends back `Connect` error, with error parameter `UnexpectedComponentSequence`

**Postamble:** ReleaseCallAB_cause_00

---

**MSC IN2m_A_BASIC_CO_BO_01**

<table>
<thead>
<tr>
<th>SCF</th>
<th>CS2_SSF</th>
<th>SigCon_A</th>
<th>SigCon_B</th>
<th>O_S2P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TC_InvokeReq</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3, 51, 2, CON, short, cONArg : { destinationRoutingAddress: '2003'H }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TC_ContinueReq</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>51, oSCF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TC_ContinueInd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>51, oSSF, TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TC_ErrorInd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3, 51, TRUE, unexpectedComponentSequencePar : Null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ReleaseCallAB_cause_00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.4.9 Continue procedure

<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_CU_CA_01</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of Continue procedure</td>
</tr>
<tr>
<td><strong>Requirement ref:</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.:</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> O_OS</td>
</tr>
<tr>
<td><strong>Test description:</strong> SCF sends to SSF Continue invoke</td>
</tr>
<tr>
<td><strong>Pass criteria:</strong> Check that SSF continues call processing, i.e. SetupReq is detected at SigConB</td>
</tr>
<tr>
<td><strong>Postamble:</strong> SigConA_Release_thenB</td>
</tr>
</tbody>
</table>

MSC IN2_A_BASIC_CU_CA_01

```
+-------------------+-------------------+-------------------+-------------------+
| SCF              | CS2_SSF           | SigCon_A          | SigCon_B          |
+-------------------+-------------------+-------------------+-------------------+
| O_OS              |
+-------------------+-------------------+-------------------+-------------------+
| TC_InvokeReq      |
| 51, 2, CUE, short, cUEArg : Null |
| TC_EndReq         |
| 51, basic         |
| SetupReq          |
| { callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H } |
+-------------------+-------------------+-------------------+-------------------+
| SigConA_Release_thenB |
+-------------------+-------------------+-------------------+-------------------+
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_CU_BV_01</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of <em>Continue</em> procedure when 2 outstanding EDP-Rs are reported</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> O_OS</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td>1. SCF sends to SSF <em>RequestReportBCSMEvent</em> invoke with parameters (oMidCall, interrupted, leg1) and (oDisconnect, interrupted, leg2)</td>
</tr>
<tr>
<td>2. SCF sends to SSF <em>Connect</em> invoke with a called party number to establish a call to the B-party.</td>
</tr>
<tr>
<td>3. After SetupReq is detected at SigConB, the call is answered by SetupConf at SigConB</td>
</tr>
<tr>
<td>4. A party issues <em>ServiceFeatureInd</em> at SigConA</td>
</tr>
<tr>
<td>5. SSF sends to SCF <em>EventReportBCSMEvent</em> invoke with parameter (oMidCall, leg1)</td>
</tr>
<tr>
<td>6. B party issues <em>ReleaseInd</em> at SigConB</td>
</tr>
<tr>
<td>7. SSF sends to SCF <em>EventReportBCSMEvent</em> invoke with parameter (oDisconnect, leg2)</td>
</tr>
<tr>
<td>8. SCF sends to SSF <em>Continue</em> invoke (note that the release should <em>not</em> be continued)</td>
</tr>
<tr>
<td>9. SCF sends to SSF a 2nd <em>Continue</em> invoke</td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td>Check that SSF sends <em>ReleaseReq</em> to SigConA</td>
</tr>
<tr>
<td><strong>Postamble:</strong> none</td>
</tr>
</tbody>
</table>
6.4.10 FurnishChargingInformation procedure

Charging related aspects in IN are network operator specific. Therefore, it is not possible to define useful test purposes for charging procedures using a network operator independant approach. TP specification has to be done by network operators, using INAP procedures themselves. TP could be specified by combining ApplyCharging, FurnishChargingInformation and SendChargingInformation procedures.

Following some examples:

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of FurnishChargingInformation procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_OS</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF FurnishChargingInformation invoke containing parameter - FCIBillingChargingCharacteristics (with completeChargingrecord) followed by ApplyCharging with mandatory parameters Then a call is established and remains for a given time to obtain a charging record</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>Check that upon release of the call, SSF sends to SCF an ApplyChargingReport invoke and that the call is released</td>
</tr>
<tr>
<td>Postamble:</td>
<td>none</td>
</tr>
</tbody>
</table>
MSC IN2m_A_BASIC_FC_CA_01

TC_InvokeReq
1, 51, 2, FCI, short, fCIArg : fCIBCCcs1 : 'BBBB'H

TC_InvokeReq
2, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '44'H, sendCalculationToSCPIndication FALSE }

TC_InvokeReq
3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
<table>
<thead>
<tr>
<th><strong>Purpose:</strong></th>
<th>Test of <strong>FurnishChargingInformation</strong> procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td><strong>T_OS</strong></td>
</tr>
</tbody>
</table>
| **Test description** | **SCF** sends to **SSF** **FurnishChargingInformation** invoke containing parameter  
  **FCIBillingChargingCharacteristics** (with **completeChargingRecord**)  
  followed by **ApplyCharging** with mandatory parameters  
  Then a call is established and remains for a given time to obtain a charging record |
| **Pass criteria** | **Check that upon release of the call, SSF sends to SCF an **ApplyChargingReport** invoke and that the call is released** |
| **Postamble:** | **none** |
MSC IN2m_A_BASIC_FC_BV_01

TC_InvokeReq 1, 51, 2, FCI, short, fCIArg : fCIBCCcs1 : 'BBBB'H

TC_InvokeReq 2, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '44'H, sendCalculationToSCPIndication FALSE }

TC_InvokeReq 3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

TC_ContinueReq 51, oSCF

SetupReq

SetupConf

SetupResp

ReleaseInd

ReleaseReq

TC_EndInd

TC_InvokeInd 102, 51, ACR, TRUE, aCRArg : '44'H
### 6.4.11 InitialDP procedure

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of InitialDP procedure and its parameter calledPartyNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
<tr>
<td>Test description</td>
<td>SigConA sends to SSF a SetupInd containing at least the parameter: - calledPartyNumber</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>Check that SSF sends to SCF an InitialDP invoke containing the parameter related to the called party: - calledPartyNumber</td>
</tr>
<tr>
<td>Postamble:</td>
<td>SigConA_Release</td>
</tr>
</tbody>
</table>

---

**MSC IN2_A_BASIC_DP_CA_01**

![MSC Diagram](chart)
Purpose: Test of InitialDP procedure and its parameter callingPartyNumber

Requirement ref

Selection Cond.

Preamble: none

Test description SigConA sends to SSF a SetupInd containing at least the parameter:
- callingPartyNumber

Pass criteria Check that SSF sends to SCF an InitialDP invoke containing the parameter related to the calling party:
- callingPartyNumber

Postamble: SigConA_Release

### MSC IN2_A_BASIC_DP_CA_02

```
<table>
<thead>
<tr>
<th>SCF</th>
<th>CS2_SSF</th>
<th>SigCon_A</th>
<th>SigCon_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>(callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_BeginInd</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[51, oSSF, TRUE]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_InvokeInd</td>
<td>101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SigConA_Release</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Purpose:
Test of InitialDP procedure and its parameter callingPartysCategory

Requirement ref

Selection Cond.

Preamble:
none

Test description
SigConA sends to SSF a SetupInd containing at least the parameter:
- callingPartysCategory

Pass criteria
Check that SSF sends to SCF an InitialDP invoke containing the parameter related to the calling party category:
- callingPartysCategory

Postamble:
SigConA_Release

MSC IN2_A_BASIC_DP_BV_01
**Purpose:**
Test of InitialDP procedure and its parameter locationNumber

**Requirement ref**

**Selection Cond.**

**Preamble:**
none

**Test description**
SigConA sends to SSF a SetupInd containing at least the parameter:
- locationNumber

**Pass criteria**
Check that SSF sends to SCF an InitialDP invoke containing the parameter related to the location information:
- locationNumber

**Postamble:**
SigConA_Release

---

**MSC IN2_A_BASIC_DP_BV_02**

```
[ 101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber ’2000’H, locationNumber ’9001’H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 } ]
```
### IN2_A_BASIC_DP_BV_03

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of InitialDP procedure and its parameter originalCalledPartyID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
</tbody>
</table>
| Test description| Signalling Control A (SigConA) sends to SSF a SetupInd containing at least the parameter:  
|                | - originalCalledPartyID                                                     |
| Pass criteria  | Check that SSF sends to SCF an InitialDP invoke containing the parameter related to the original called party number:  
|                | - originalCalledPartyID                                                     |
| Postamble:     | SigConA_Release                                                             |

### MSC IN2_A_BASIC_DP_BV_03

```plaintext
SCF -> CS2_SSF -> SigCon_A -> SigCon_B

SetupInd
{ callRef 1, calledPartyNumber '2000'H, originalCalledNumber '2211'H }  
TC_BeginInd 51, oSSF, TRUE
TC_InvokeInd 101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, originalCalledPartyID '2211'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }

SigConA_Release
```

---

ETSII Final draft ETSI EN 301 140-3-1 V1.1.3 (2000-02)
**Purpose:** Test of InitialDP procedure and its parameter redirectingPartyID

**Requirement ref**

**Selection Cond.**

**Preamble:** none

**Test description** SigConA sends to SSF a SetupInd containing at least the parameter:
- redirectingPartyID

**Pass criteria** Check that SSF sends to SCF an InitialDP invoke containing the parameter related to redirecting party number:
- redirectingPartyID

**Postamble:** SigConA_Release

---

**MSC IN2_A_BASIC_DP_BV_04**

```
{ callRef 1, calledPartyNumber '2000'H, locationNumber '9001'H, redirectingNumber '3000'H }

TC_BeginInd

51, oSSF, TRUE

TC_InvokeInd

101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, locationNumber '9001'H, eventTypeBCSM analysedInformation, redirectingPartyID '3000'H, createdCallSegmentAssociation 1 }

SigConA_Release
```
Purpose: Test of InitialDP procedure with timer expiration

Requirement ref: none

Selection Cond.:

Test description:
- SigConA sends a SetupInd containing at least the mandatory parameters
- SSF sends to SCF an InitialDP invoke
- SCF does not send to SSF an operation, so timer Tssf expires

Pass criteria: SSF sends release to SigConA to free all resources involved

Postamble:

MSC IN2m_A_BASIC_DP_BV_05

SCF → CS2_SSF → SigCon_A → SigCon_B

SetupInd

{ callRef 1, calledPartyNumber '2000'H }

TC_BeginInd

[ 51, oSSF, TRUE ]

TC_InvokeInd

[ 101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, createdCallSegmentAssociation 1 } ]

Tssf times out, as SCF did not initiate an operation

ReleaseReq

{ callRef 1, cause '0C' }
**IN2_A_BASIC_DP_BI_01**

**Purpose:** Test of InitialDP procedure with error

**Requirement ref**

**Selection Cond.**

**Preamble:** none

**Test description**
- SigConA sends a SetupInd containing at least the mandatory parameters
- SSF sends to SCF an InitialDP invoke
- SCF sends to SSF an InitialDP error containing parameter: missingCustomerRecord

**Pass criteria** SSF goes to the idle state.

**Postamble:** none

---

**MSC IN2m_A_BASIC_DP_BI_01**

![MSC Diagram](image-url)
Purpose: Test of InitialDP procedure with error

Requirement ref

Selection Cond.

Preamble: none

Test description
- SigConA sends a SetupInd containing at least the mandatory parameters
- SSF sends to SCF an InitialDP invoke
- SCF sends to SSF an InitialDP error containing parameter: taskRefused

Pass criteria SSF goes to the idle state.

Postamble: none

---

**MSC IN2m_A_BASIC_DP_BI_02**

```
<table>
<thead>
<tr>
<th></th>
<th>SCF</th>
<th>CS2_SSF</th>
<th>SigCon_A</th>
<th>SigCon_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetupInd</td>
<td>[callRef 1, calledPartyNumber '2000'H]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_BeginInd</td>
<td>51, oSSF, TRUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_InvokeInd</td>
<td>101, 51, IDP, TRUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[serviceKey 1, calledPartyNumber '2000'H, createdCallSegmentAssociation 1]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_ErrorReq</td>
<td>101, 51, taskRefusedPar : congestion</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC_EndReq</td>
<td>51, basic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReleaseReq</td>
<td>[callRef 1, cause '0C'H], senderRef</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
### 6.4.12 InitiateCallAttempt procedure

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of <code>InitiateCallAttempt</code> base procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF an <code>InitiateCallAttempt</code> with mandatory parameters: <code>destinationRoutingAddress</code> followed by a <code>Continue</code> invoke</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>Check that SSF sends a SetupReq to the proper SigCon according to the <code>InitiateCallAttempt</code></td>
</tr>
<tr>
<td>Postamble:</td>
<td><code>SigConB_Release</code></td>
</tr>
</tbody>
</table>
1, 1, 2, ICA, short, ICAArg : \{ destinationRoutingAddress \{ '2100'H \},
legToBeCreated sendingSideID : '02'H, newCallSegment 1 \}
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of <code>InitiateCallAttempt</code> procedure with parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF an <code>InitiateCallAttempt</code> with mandatory and optional parameters destinationRoutingAddress callingPartyNumber followed by a <code>Continue</code> invoke</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>Check that SSF sends a SetupReq to the proper SigCon according to the <code>InitiateCallAttempt</code></td>
</tr>
<tr>
<td>Postamble:</td>
<td>SigConB_Release</td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_IC_BV_01

TC_InvokeReq
1, 1, 2, ICA, short, iCAArg : { destinationRoutingAddress '2100'H, callingPartyNumber '1000'H, legToBeCreated sendingSideID '02'H, newCallSegment 1 }

TC_BeginReq
1, oSCF

TC_InvokeReq
2, 1, 4, CUE, short, cUEArg : Null

TC_EndReq
1, prearranged

SetupReq
{ callRef 2, calledPartyNumber '2100'H, callingPartyNumber '1000'H }

SigConB_Release_cause_0D

ETSI
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of <code>InitiateCallAttempt</code> procedure with parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>none</td>
</tr>
</tbody>
</table>
| Test description    | SCF sends to SSF an `InitiateCallAttempt` with mandatory and optional parameters  
|                     | `destinationRoutingAddress`                           |
|                     | `alertingPattern`                                     |
|                     | followed by a `Continue` invoke                       |
| Pass criteria       | Check that SSF sends a SetupReq to the proper SigCon according to the `InitiateCallAttempt` and check the special tone indicated in `alertingPattern` |
| Postamble:          | SigConB_Release                                        |
MSC IN2_A_BASIC_IC_BV_02

TC_InvokeReq
1, 1, 2, ICA, short, iCAArg : { destinationRoutingAddress : '2100'H, alertingPattern : '123'H, legToBeCreated : 02'H, newCallSegment : 1 }
<table>
<thead>
<tr>
<th>IN2_A_BASIC_IC_BO_01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
MSC IN2m_A_BASIC_IC_BO_01

TC_InvokeReq 1, 51, 2, ICA, short, iCAArg : { destinationRoutingAddress { '2100'H }, callingPartyNumber '1000'H, legToBeCreated sendingSideID : '02'H, newCallSegment 1 }

TC_InvokeReq 2, 51, 3, AT, short, aTArg : NULL

TC_InvokeReq 2, 51, TRUE, AT_R, aTRArg : NULL

TC_InvokeReq 1, 51, true, unexpectedComponentSequencePar : Null

TC_InvokeReq 51, oSCF

TC_InvokeReq 51, oSSF, TRUE

TC_InvokeInd 51, oSCF

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE

TC_InvokeInd 51, oSSF, TRUE
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of <code>InitiateCallAttempt</code> procedure in wrong state (monitoring)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_S2P</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF an <code>InitiateCallAttempt</code> in a wrong state (Monitoring)</td>
</tr>
</tbody>
</table>
| Pass criteria | Check that SSF returns `InitiateCallAttempt` error with parameter `UnexpectedComponentSequence` and remains in the same state. To check SSF stays in monitoring state,  
- SCF sends to SSF an `ActivityTest` invoke with DialogID used in previous `InitiateCallAttempt`  
- SSF answers with `ActivityTest` result because DialogID is still active |
| Postamble: | ReleaseCallAB_cause_00 |
MSC IN2m_A_BASIC_IC_BO_02

```
TC_InvokeReq
   [1, 51, 2, ICA, short, iCAArg : { destinationRoutingAddress { '2100'H } }]
TC_ContinueReq
   [51, oSCF]
TC_ContinueInd
   [51, oSSF, TRUE]
TC_ErrorInd
   [1, 51, true, unexpectedComponentSequencePar : Null]
TC_InvokeReq
   [2, 51, 3, AT, short, aTArg : NULL]
TC_ContinueReq
   [51, oSCF]
TC_ContinueInd
   [51, oSSF, TRUE]
TC_ReturnResultInd
   [2, 51, TRUE, AT_R, aTRArg : NULL]
ReleaseCallAB_cause_00
```
6.4.13 ReleaseCall procedure

<table>
<thead>
<tr>
<th>IN2_A_BASIC_RC_CA_01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of ReleaseCall base procedure</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> O_OS</td>
</tr>
<tr>
<td><strong>Test description:</strong> SCF sends to SSF ReleaseCall invoke, with: - initialCallSegment (cause)</td>
</tr>
<tr>
<td><strong>Pass criteria:</strong> Check that SSF releases the call (ReleaseReq received by SigConA)</td>
</tr>
<tr>
<td><strong>Postamble:</strong> none</td>
</tr>
</tbody>
</table>

**MSC IN2_A_BASIC_RC_CA_01**

SCF → CS2_SSF

SigCon_A → SigCon_B

O_OS →

TC_InvokeReq

TC_EndReq

ReleaseReq

3, 51, 4, RC, short, rCArg : initialCallSegment : '00'H

{ callRef 1, cause '00'H, senderRef }
Purpose: Test of ReleaseCall procedure with two parties

Requirement ref

Selection Cond.

Preamble: O_S2P

Test description SCF sends to SSF ReleaseCall invoke, with:
- initialCallSegment (cause)

Pass criteria Check that SSF releases the call (ReleaseReq received by SigConA and SigConB)

Postamble: none
<table>
<thead>
<tr>
<th>IN2_A_BASIC_RC_BV_04</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of <em>ReleaseCall</em> procedure in combination with <em>CallInformation</em> and <em>RequestReportBCSMEvent</em> procedures</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> O_OS</td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF *RequestReportBCSMEvent* with *eventTypeBCSM*=routeSelectFailure followed by a *CallInformationRequest* invoke, with at least the parameters:  
- *requestedInformationTypeList* including:  
  - *requestedInformationType* (releaseCause),  
Then SCF releases the call using *ReleaseCall* invoke with:  
- *initialCallSegment* (cause) |
| **Pass criteria** | - Check that upon detection of call release, SSF sends *CallInformationReport* with at least the parameters  
  - *requestedInformationList* including:  
    - *requestedInformationType* (releaseCause),  
    - *requestedInformationValue* being *releaseCauseValue* used  
and check that no *EventReportBCSMEvent* is sent  
Check that SigConA receives a ReleaseReq |
| **Postamble:** none |
MSC IN2_A_BASIC_RC_BV_04

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode interrupted } } }

TC_InvokeReq
2, 51, 2, CIRQ, medium, cIRQArg : { requestedInformationTypeList { releaseCause } }

TC_ContinueReq
51, oSSF

TC_InvokeReq
3, 51, 4, RC, short, rCArg : initialCallSegment : '00'H

TC_ContinueReq
51, oSSF

ReleaseReq
{ callRef 1, cause '00'H }, senderRef

TC_EndInd
51, basic, TRUE

TC_InvokeInd
102, 51, CIR, TRUE, cIRArg : { requestedInformationList { } }
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_RC_BO_01</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of <strong>ReleaseCall</strong> procedure form wrong state (idle)</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> none</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td><strong>Postamble:</strong> none</td>
</tr>
</tbody>
</table>
6.4.14 RequestReportBCSMEvent procedure

<table>
<thead>
<tr>
<th>IN2_A_BASIC_RR_CA_01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of RequestReportBCSMEvent base procedure</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> O_OS</td>
</tr>
<tr>
<td><strong>Test description</strong> SCF sends to SSF RequestReportBCSMEvent invoke containing parameters</td>
</tr>
<tr>
<td>- eventTypeBCSM=oAbandon</td>
</tr>
<tr>
<td>- monitoringMode=interrupted</td>
</tr>
<tr>
<td>then the calling party abandons the call before the call is answered (SigCon A to send ReleaseInd)</td>
</tr>
<tr>
<td><strong>Pass criteria</strong> Check that SSF sends to SCF an EventReportBCSM with the indication of</td>
</tr>
<tr>
<td>- eventTypeBCSM=oAbandon</td>
</tr>
<tr>
<td><strong>Postamble:</strong> none</td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_RR_CA_01

Purpose: Test of RequestReportBCSMEvent procedure and tAbandon indication

Requirement ref

Selection Cond.

Preamble: T_OS

Test description
- SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
  - eventTypeBCSM=tAbandon
  - monitoringMode=interrupted
  - then the calling party abandons the call before the call is answered (SigCon A to send ReleaseInd)

Pass criteria: Check that SSF sends to SCF an EventReportBCSM with the indication of eventTypeBCSM=tAbandon

Postamble: none
MSC IN2_A_BASIC_RR_BV_01

1. 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM tAbandon, monitorMode interrupted } } }

51, oSCF

ReleaseInd

51, oSSF, TRUE

TC_InvokeInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tAbandon, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

TC_InvokeReq

100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H

TC_EndReq

51, basic
<table>
<thead>
<tr>
<th><strong>Purpose:</strong></th>
<th>Test of <code>RequestReportBCSMEvent</code> procedure and <code>collectedInfo</code> indication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
</tbody>
</table>
| **Preamble:** | O_OS  
Preamble contains an InitialDP without complete digits for CalledPartyNumber |
| **Test description** | SCF sends to SSF `RequestReportBCSMEvent` invoke containing parameters  
- `eventTypeBCSM=collectedInfo`  
- `monitoringMode=interrupted`  
- SCF sends a `CollectInformation` operation  
then the calling party sends the remaining digits (using `CallProgressInd`) |
| **Pass criteria** | Check that SSF sends to SCF an `EventReportBCSM` with the indication of  
`eventTypeBCSM=collectedInfo` |
| **Postamble:** | `SigConA_Release` |
MSC IN2_A_BASIC_RR_BV_02

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM collectedInfo, monitorMode interrupted } } }

TC_InvokeReq
2, 51, 2, CI, medium, cIArg : {}

TC_ContinueReq
51, oSCF

CallProgressReq
{ callRef 1, cause '1F'H }

SubsequentAddressInd
{ callRef 1, digits '02'H }

AddressEndInd
{ callRef 1 }

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM collectedInfo, eventSpecificInformationBCSM collectedInfoSpecificInfo : { calledPartynumber '200002'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }

SigConA_Release
**Purpose:** Test of RequestReportBCSMEvent procedure and analysedInfo indication

**Requirement ref**

**Selection Cond.**

**Preamble:** O_OS
Preamble contains an InitialDP without complete digits for CalledPartyNumber

**Test description**
SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
- eventTypeBCSM=analysedInfo
- monitoringMode=interrupted
then the calling party sends the remaining digits (after CallProgressReq is received and SubsequentAddressInd and AddressEndInd is sent)

**Pass criteria**
Check that SSF sends to SCF an EventReportBCSM with the indication of
eventTypeBCSM=analysedInfo

**Postamble:** SigConA_Release
MSC IN2_A_BASIC_RR_BV_03

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM analysedInformation, monitorMode interrupted } } }

TC_InvokeReq
2, 51, 2, CI, medium, cIArg : {}

TC_ContinueReq
51, oSCF

CallProgressReq
callRef 1, cause '1F' H

SubsequentAddressInd
callRef 1, digits '02'h

AddressEndInd
callRef 1, digits '02'h

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : {eventTypeBCSM analysedInformation, eventSpecificInformationBCSM analysedInfoSpecificInfo : calledPartynumber '200002'H, legID receivingSideID '01'H, miscCallInfo { messageType request } }

SigConA_Release
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_RR_BV_04</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM=routeSelectFailure  
- monitoringMode=interrupted  
followed by a Connect invoke  
Then SSF sends a SetupReq to SigCon B  
SigCon B releases the call (ReleaseInd) because of error: routeFailure2 ("oc"H) |
| **Pass criteria** | Check that SSF sends to SCF an EventReportBCSM with the indication of  
eventTypeBCSM= routeSelectFailure |
| **Postamble:** | SigConA_Release |
MSC IN2_A_BASIC_RR_BV_04

1. 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode interrupted } } }

2. 51, 2, CON, short, cONArg : { destinationRoutingAddress ('2001'H) }

51. oSCF

[ SetupReq
  { callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H } ]

ReleaseInd
  { callRef 2, cause '0C'H }

102, 51, ERB, TRUE, eRBArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, eventSpecificInformationBCSM routeSelectFailureSpecificInfo : { failureCause '0C'H }, legID receivingSideID '02'H, miscCallInfo { messageType request } } } }

SigConA_Release
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_RR_BV_05</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of <code>RequestReportBCSMEvent</code> procedure and <code>oCalledPartyBusy</code> indication.</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
</tbody>
</table>
| **Preamble:** O_OS  
In addition, user B is declared busy |
| **Test description**  
SCF sends to SSF `RequestReportBCSMEvent` invoke containing parameters  
- `eventTypeBCSM=0CalledPartyBusy`  
- `monitoringMode=interrupted`  
followed by a `Connect` invoke  
Then SSF sends a SetupReq to SigCon B  
SigCon B releases the call (`ReleaselnD`) with `bPtyBusy_UDUB` |
| **Pass criteria**  
Check that SSF sends to SCF an `EventReportBCSM` with the indication of  
`eventTypeBCSM=0CalledPartyBusy` |
| **Postamble:** SigConA_Release |
MSC IN2_A_BASIC_RR_BV_05

TC_InvokeReq

1, 51, 2, RRB, short, rRBArg : { bcsmevents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted } } }

TC_InvokeReq

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

TC_ContinueReq

51, oSCF

SetupReq

{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }

ReleaseInd

{ callRef 2, cause '0D'H }

TC_ContinueInd

51, oSSF, TRUE

TC_InvokeInd

legID receivingSideID : '02'H, miscCallInfo { messageType request }

SigConA_Release

ETSi
### IN2_A_BASIC_RR_BV_06

<table>
<thead>
<tr>
<th><strong>Purpose:</strong></th>
<th>Test of RequestReportBCSMEvent procedure and oNoAnswer indication.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>O_OS</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
<td>SCF sends to SSF RequestReportBCSMEvent invoke containing parameters - eventTypeBCSM=oNoAnswer - monitoringMode=interrupted followed by a Connect invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call (ReleaseInd) because user B does not answer</td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
<td>Check that SSF sends to SCF an EventReportBCSM with the indication of eventTypeBCSM= oNoAnswer</td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
<td>SigConA_Release</td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_RR_BV_06

SCF  CS2_SSF  SigCon_A  SigCon_B

O_OS

TC_InvokeReq

1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted } } }

TC_InvokeReq

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

TC_ContinueReq

51, oSCF

SetupReq

{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }

ReleaseInd

{ callRef 2, cause '09'H }

TC_ContinueInd

51, oSSF, TRUE

TC_InvokeInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

SigConA_Release
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_RR_BV_07</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of RequestReportBCSMEvent procedure and oAnswer indication.</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> O_OS</td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
- eventTypeBCSM=oAnswer
- monitoringMode=interrupted
followed by a Connect invoke
Then SSF sends a SetupReq to SigCon B
SigCon B answers the call (SetupConf from SigConB to SSF) |
| **Pass criteria** | Check that SSF sends to SCF an EventReportBCSM with the indication of
eventTypeBCSM= oAnswer |
| **Postamble:** ReleaseCallAB_cause_00 |
MSC IN2_A_BASIC_RR_BV_07

1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

SetupReq

[ { callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H } ]

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

ReleaseCallAB_cause_00
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_RR_BV_08</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM= oMidCall  
- monitoringMode=interrupted  
followed by a Connect invoke  
Then SSF sends a SetupReq to SigCon B. SetupConf from SigConB is received by SSF which issues SetupResp to SigConA.  
SigConA calling party initiates a service (ServiceFeatureInd sent to SSF) and oMidCall DP is reached |
| **Pass criteria** | Check that SSF sends to SCF an EventReportBCSM with the indication of eventTypeBCSM= oMidCall |
| **Postamble:** | ReleaseCallAB_cause_00 |
**MSC IN2_A_BASIC_RR_BV_08**

- **TC_InvokeReq**: 1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oMidCall, monitorMode interrupted } } }
- **TC_InvokeReq**: 2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001' } }
- **TC_ContinueReq**: 51, oSSF
- **SetupReq**: { callRef 2, calledPartyNumber '2001', callingPartyNumber '1000' }
- **SetupResp**: { callRef 2 }
- **ServiceFeatureInd**: { callRef 1 }
- **TC_ContinueInd**: 51, oSSF, TRUE
- **TC_InvokeInd**: 102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oMidCall, legID receivingSideID : '01', miscCallInfo { messageType request } }
- **ReleaseCallAB_cause_00**
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of RequestReportBCSMEvent procedure and oDisconnect indication.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_Os</td>
</tr>
</tbody>
</table>
| Test description | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM= oDisconnect  
- monitoringMode=interrupted  
followed by a Connect invoke  
Then SSF establishes the call (a SetupReq to SigCon B, SetupConf from SigConB to SSF, then SetupResp to SigConA)  
SigCon A (calling party) clears the call after it is answered (ReleaseInd sent) |
| Pass criteria | Check that SSF sends to SCF an EventReportBCSM with the indication of  
eventTypeBCSM= oDisconnect  
SCF sends a Continue operation, check that the B side receives a RelReq |
<p>| Postamble: | none |</p>
<table>
<thead>
<tr>
<th>IN2_A_BASIC_RR_BV_10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Test description</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_RR_BV_10

SCF   CS2_SSF   SigCon_A   SigCon_B

T_OS

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsrEvents { { eventTypeBCSM tBusy, monitorMode interrupted } } }

TC_InvokeReq
2, 51, 2, CUE, short, cUEArg : Null

TC_ContinueReq
51, oSCF

SetupReq
{ callRef 2, calledPartyNumber '2002'H, callingPartyNumber '1000'H }

ReleaseInd
{ callRef 2, cause '0D'H }

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tBusy, eventSpecificInformationBCSM tBusySpecificInfo :
{ busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

ReleaseInd
{ callRef 1, cause '00'H }

ReleaseReq
{ callRef 2, cause '0F'H, senderRef }

TC_EndInd
51, basic, FALSE
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of RequestReportBCSMEvent procedure and tNoAnswer indication.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>T_OS</td>
</tr>
<tr>
<td>Test description</td>
<td>SCF sends to SSF RequestReportBCSMEvent invoke containing parameters</td>
</tr>
<tr>
<td></td>
<td>- eventTypeBCSM=tNoAnswer</td>
</tr>
<tr>
<td></td>
<td>- monitoringMode=interrupted</td>
</tr>
<tr>
<td></td>
<td>followed by a Continue invoke</td>
</tr>
<tr>
<td></td>
<td>Then SSF sends a SetupReq to SigCon B</td>
</tr>
<tr>
<td></td>
<td>SigCon B releases the call (ReleaseInd sent) because user B does not answer</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>Check that SSF sends to SCF an EventReportBCSM with the indication of</td>
</tr>
<tr>
<td></td>
<td>eventTypeBCSM= tNoAnswer</td>
</tr>
<tr>
<td>Postamble:</td>
<td>ReleaseCallA</td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_RR_BV_11

SCF  |  CS2 SSF  |  SigCon_A  |  SigCon_B

T_OS

 TC_InvokeReq
[1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM tNoAnswer, monitorMode interrupted } } }]

 TC_InvokeReq
[2, 51, 2, CUE, short, cUEArg : Null]

 TC_InvokeReq
[51, oSSF]

 TC_InvokeReq
[51, oSCF]

 SetupReq

 TC_InvokeReq
[51, oSSF, TRUE]

 ReleaseCallA
<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_RR_BV_12</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Test of RequestReportBCSMEvent procedure and tAnswer indication.</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong> T_OS</td>
</tr>
<tr>
<td><strong>Test description</strong> Sufficient sends to SSF RequestReportBCSMEvent invoke containing parameters</td>
</tr>
<tr>
<td>- eventTypeBCSM=tAnswer</td>
</tr>
<tr>
<td>- monitoringMode=interrupted</td>
</tr>
<tr>
<td>followed by a Continue invoke</td>
</tr>
<tr>
<td>Then Sufficient sends a SetupReq to SigCon B</td>
</tr>
<tr>
<td>SigCon B answers the call (SetupConf from SigConB to Sufficient)</td>
</tr>
<tr>
<td><strong>Pass criteria</strong> Check that Sufficient sends to SCF an EventReportBCSM with the indication of</td>
</tr>
<tr>
<td>eventTypeBCSM=tAnswer</td>
</tr>
<tr>
<td><strong>Postamble:</strong> ReleaseCallA</td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_RR_BV_12

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmevents { { eventTypeBCSM tAnswer, monitorMode interrupted } } }

TC_InvokeReq
2, 51, 2, CUE, short, cUEArg : Null

TC_ContinueReq
51, oSSF

SetupReq
{ callRef 2, calledPartyNumber '2002' H, callingPartyNumber '1000' H }

SetupConf
{ callRef 2 }

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tAnswer, legID receivingId '02' H, miscCallInfo { messageType request } }

ReleaseCallA2
<table>
<thead>
<tr>
<th>IN2_A_BASIC_RR_BV_13</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM= tMidCall  
- monitoringMode=interrupted  
followed by a Continue invoke  
Then SSF sends a SetupReq to SigCon B. SetupConf from SigConB is received by SSF which issues SetupResp to SigConA.  
SigConB called party initiates a service (ServiceFeatureInd sent to SSF) and tMidCall DP is reached |
| **Pass criteria** | Check that SSF sends to SCF an EventReportBCSM with the indication of  
eventTypeBCSM= tMidCall |
| **Postamble:** | ReleaseCallA |
MSC IN2_A_BASIC_RR_BV_13

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM tMidCall, monitorMode interrupted } } }

TC_InvokeReq
2, 51, 2, CUE, short, cUEArg : Null

TC_ContinueReq
51, oSCF

SetupReq
{ callRef 2, calledPartyNumber '2002'H, callingPartyNumber '1000'H }

SetupConf

SetupResp
{ callRef 1 }

ServiceFeatureInd

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tMidCall, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

ReleaseCallA2
### Purpose:
Test of `RequestReportBCSMEvent` procedure and tDisconnect indication.

### Requirement ref

### Selection Cond.

### Preamble:
T_OS

### Test description
SCF sends to SSF `RequestReportBCSMEvent` invoke containing parameters
- `eventTypeBCSM= tDisconnect`
- `monitoringMode=interrupted`
followed by a `Continue` invoke
Then SSF establishes the call (a SetupReq to SigCon B, SetupConf from SigConB to SSF which sends SetupResp to SigConA)
SigCon A (calling party) clears the call after it is answered (ReleaseInd sent)

### Pass criteria
Check that SSF sends to SCF an `EventReportBCSM` with the indication of `eventTypeBCSM= tDisconnect`

### Postamble:
ReleaseCallB
MSC IN2_A_BASIC_RR_BV_14

T_OS

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents : { eventTypeBCSM : tDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } }

TC_InvokeReq
2, 51, 2, CUE, short, cUEArg : Null

TC_ContinueReq
51, oSCF

SetupReq

TC_InvokeReq
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM : tDisconnect, eventSpecificInformationBCSM : tDisconnectSpecificInfo : { releaseCause : '10'H, legID receivingSideID : '01'H, miscCallInfo : { messageType : request } } }

ReleaseCallB

ETS1
**Purpose:** Test of RequestReportBCSMEvent procedure and tDisconnect indication.

**Requirement ref**

**Selection Cond.**

**Preamble:** T_OS

**Test description**

<table>
<thead>
<tr>
<th>SCF sends to SSF RequestReportBCSMEvent invoke containing parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>- eventTypeBCSM= tDisconnect</td>
</tr>
<tr>
<td>- monitoringMode=notifyAndContinue</td>
</tr>
<tr>
<td>followed by a Continue invoke</td>
</tr>
<tr>
<td>The IUT establishes the call, sends a SetUpReq to B side</td>
</tr>
<tr>
<td>Then SigCon A (calling party) clears the call after it is answered (ReleaseInd sent)</td>
</tr>
</tbody>
</table>

**Pass criteria**

- Check that SSF sends to SCF an EventReportBCSM with the indication of eventTypeBCSM= tDisconnect
- Check that SigConB is receiving a ReleaseReq to continue clearing the call

**Postamble:** none
MSC IN2_A_BASIC_RR_BV_15

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents : { eventTypeBCSM : tDisconnect, monitorModeNotifyAndContinue, legID : '01' } }
This TP has been deleted.

<table>
<thead>
<tr>
<th>IN2_A_BASIC_RR_BV_17</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref:</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.:</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td><strong>Test description:</strong></td>
</tr>
<tr>
<td><strong>Pass criteria:</strong></td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
</tr>
</tbody>
</table>
MSC IN2_A_BASIC_RR_BV_17

1. TC_InvokeReq 1, 51, RRB, short, eRBArg : { bcsmevent { { eventTypeBCSM oAnswer, monitorMode interrupted}, { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID: '02' } } }

2. TC_ContinueReq
3. TC_InvokeReq 51, oSCF

4. SetupReq
5. TC_ContinueInd
6. TC_InvokeInd 102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oAnswer, legID receivingSideID: '02' }, miscCallInfo { messageType request } }

7. ReleaseInd
8. TC_ContinueInd
9. TC_InvokeInd 103, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oDisconnect, eventSpecificInformationBCSM oDisconnectSpecificInfo : { releaseCause '10' }, legID receivingSideID: '02', miscCallInfo { messageType request } } }

SigConA_Release
Purpose:
Test of RequestReportBCSMEvent procedure and oAbandon

Requirement ref
Selection Cond.

Preamble:
O_OS

Test description
SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
- eventTypeBCSM=\text{\textit{oAbandon}}
- monitoringMode=\text{\textit{notifyAndContinue}}
then the calling party abandons the call before the call is answered (SigCon A to send ReleaseInd)

Pass criteria
Check that SSF sends to SCF an EventReportBCSM with the indication of eventTypeBCSM=\text{\textit{oAbandon}}

Postamble: none
Purpose: Test of RequestReportBCSMEvent procedure and tAbandon

Requirement ref

Selection Cond.

Preamble: T_OS

Test description: SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
- eventTypeBCSM=tAbandon
- monitoringMode=notifyAndContinue
then the calling party abandons the call before the call is answered (SigCon A to send ReleaseInd)

Pass criteria: Check that SSF sends to SCF an EventReportBCSM with the indication of eventTypeBCSM=tAbandon

Postamble: none
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC_RR_BV_20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose:</td>
<td>Test of RequestReportBCSMEvent procedure and collectedInfo indication</td>
</tr>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_OS Preamble contains an InitialDP without complete digits for CalledPartyNumber</td>
</tr>
</tbody>
</table>
| Test description | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM=collectedInfo  
- monitoringMode= notifyAndContinue  
then the calling party sends the remaining digits (using CallProgressInd) |
| Pass criteria | Check that SSF sends to SCF an EventReportBCSM with the indication of  
eventTypeBCSM=collectedInfo |
| Postamble: | SigConA_Release_thenB |
MSC IN2_A_BASIC_RR_BV_20

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM collectedInfo, monitorMode notifyAndContinue } } }

TC_InvokeReq
2, 51, 2, CI, medium, cIArg : {}

TC_ContinueReq
51, oSCF

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM collectedInfo, eventSpecificInformationBCSM collectedInfoSpecificInfo : { calledPartyNumber ’200002’H, legID receivingSideID : ’01’H, miscCallInfo { messageType request } } }

TC_EndInd
51, basic, FALSE

SetupReq
{ callRef 2, calledPartyNumber ’2000’H, callingPartyNumber ’1000’H }
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of RequestReportBCSMEvent procedure and analysedInfo indication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
</tbody>
</table>
| Preamble: | O_OS  
Preamble contains an InitialDP without complete digits for CalledPartyNumber |
| Test description | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
  - eventTypeBCSM=analysedInfo  
  - monitoringMode= notifyAndContinue  
  followed by CollectInformation operation  
  then the calling party sends the remaining digits (after CallProgressReq is received and SubsequentAddressInd and AddressEndInd is sent) |
| Pass criteria | Check that SSF sends to SCF an EventReportBCSM with the indication of  
  - eventTypeBCSM=analysedInfo |
| Postamble: | SigConA_Release_thenB |
MSC_IN2_A_BASIC_RR_BV_21

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg :  
{ bcsmEvents { { eventTypeBCSM analysedInformation, monitorMode notifyAndContinue } } }

TC_InvokeReq
2, 51, 2, CI, medium, cIArg : {}

TC_ContinueReq
51, oSCF

CallProgressReq
{ callRef 1, cause '1FH' }

SubsequentAddressInd
{ callRef 1, digits '02H' }

AddressEndInd
{ callRef 1 }

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : 
{ eventTypeBCSM analysedInformation, eventSpecificInformationBCSM analysedInfoSpecificInfo :
  { calledPartynumber '2000H', legID receivingSideID '01H', miscCallInfo { messageType request } } }

TC_EndInd
51, basic, FALSE

SetupReq
{ callRef 2, calledPartyNumber '2000H', callingPartyNumber '1000H' }

SigConA_Release_thenB
Purpose: Test of RequestReportBCSMEvent procedure and routeSelectFailure indication

Requirement ref

Selection Cond. O_OS

Preamble: SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
- eventTypeBCSM=routeSelectFailure
- monitoringMode= notifyAndContinue
followed by a Connect invoke
Then SSF sends a SetupReq to SigCon B
SigCon B releases the call (ReleaseInd) with cause routeFailure2

Pass criteria Check that SSF sends to SCF an EventReportBCSM with the indication of
eventTypeBCSM= routeSelectFailure

Postamble: none

MSC IN2_A_BASIC_RR_BV_22

![MSC Diagram]
<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of RequestReportBCSMEvent procedure and oCalledPartyBusy indication.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
</tbody>
</table>
| Preamble: | O_OS  
In addition, user B is declared busy |
| Test description | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM=oCalledPartyBusy  
- monitoringMode= notifyAndContinue  
followed by a Connect invoke 
Then SSF sends a SetupReq to SigCon B  
SigCon B releases the call (ReleaseInd) with cause bPtyBusy _UDUB  |
| Pass criteria | Check that SSF sends to SCF an EventReportBCSM with the indication of  
eventTypeBCSM= oCalledPartyBusy |
| Postamble: | none |
MSC IN2_A_BASIC_RR_BV_23

1, 51, 2, RRB, short, RRBArg : { bcsmevents { { eventTypeBCSM oCalledPartyBusy, monitorMode notifyAndContinue } } }

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

51, oSCF

SetupReq

[ { callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H } ]

ReleaseInd

[ { callRef 2, cause '0D'H } ]

51, oSSF, TRUE

TC_InvokeInd

102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oCalledPartyBusy, eventSpecificInformationBCSM oCalledPartyBusySpecificInfo : { busyCause '0D'H }, legsID receivingSideID : '02'H, miscCallInfo { messageType request } }

51, basic, FALSE

ReleaseReq

[ { callRef 1, cause '0D'H }, senderRef ]
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC_RR_BV_24</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of RequestReportBCSMEvent procedure and oNoAnswer indication.</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>O_OS</td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM=oNoAnswer  
- monitoringMode= notifyAndContinue  
followed by a Connect invoke  
Then SSF sends a SetupReq to SigCon B  
SigCon B releases the call (ReleaseInd) because user B does not answer |
| **Pass criteria** | Check that SSF sends to SCF an EventReportBCSM with the indication of  
eventTypeBCSM= oNoAnswer |
| **Postamble:** | none |
MSC IN2_A_BASIC_RR_BV_24

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmevents { { eventTypeBCSM oNoAnswer, monitorMode notifyAndContinue } } }
TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001' } }
TC_ContinueReq
51, oSCF

SetupReq
51, oSSF, TRUE

ReleaseReq
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oNoAnswer, legID receivingSideID : '02' , miscCallInfo { messageType request } }

ReleaseInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oNoAnswer, legID receivingSideID : '02' , miscCallInfo { messageType request } }

TC_ContinueInd
51, oSSF, TRUE
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC_RR_BV_25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose:</td>
<td>Test of RequestReportBCSMEvent procedure and oAnswer indication.</td>
</tr>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_OS</td>
</tr>
</tbody>
</table>
| Test description | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM=oAnswer  
- monitoringMode= notifyAndContinue  
followed by a Connect invoke  
Then SSF sends a SetupReq to SigCon B  
SigCon B answers the call (SetupConf from SigConB to SSF) |
| Pass criteria | Check that SSF sends to SCF an EventReportBCSM with the indication of  
eventTypeBCSM= oAnswer |
| Postamble: | SigConA_Release-thenB_cause10 |
MSC IN2_A_BASIC_RR_BV_25

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode notifyAndContinue } } }

TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

TC_ContinueReq
51, oSCF

SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }

SetupConf
{ callRef 2 }

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

SetupResp
{ callRef 1 }

TC_EndInd
51, basic, FALSE

SigConA_Release_thenB_cause10
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC_RR_BV_26</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of RequestReportBCSMEvent procedure and oMidCall indication.</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>O_OS</td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM= oMidCall  
- monitoringMode= notifyAndContinue  
followed by a Connect invoke  
Then SSF sends a SetupReq to SigCon B. SetupConf from SigConB is received by SSF which issues SetupResp to SigConA.  
SigConA calling party initiates a service (ServiceFeatureInd sent to SSF) and oMidCall DP is reached |
| **Pass criteria** | Check that SSF sends to SCF an EventReportBCSM with the indication of  
eventTypeBCSM= oMidCall |
| **Postamble:** | SigConA_Release_thenB_cause10 |
MSC IN2_A_BASIC_RR_BV_26

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oMidCall, monitorMode notifyAndContinue } } }

TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress '2001'H }

TC_ContinueReq
51, oSCF

SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }

SetupResp
{ callRef 1 }

ServiceFeatureInd
{ callRef 1 }

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oMidCall, legID receivingSideID '01'H, miscCallInfo { messageType request } }

TC_EndInd
51, basic, FALSE

SigConA_Release_thenB_cause10
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC_RR_BV_27</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of RequestReportBCSMEvent procedure and oDisconnect indication.</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>O_OS</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
<td>SCF sends to SSF RequestReportBCSMEvent invoke containing parameters</td>
</tr>
<tr>
<td></td>
<td>- eventTypeBCSME= oDisconnect</td>
</tr>
<tr>
<td></td>
<td>- monitoringMode= notifyAndContinue</td>
</tr>
<tr>
<td></td>
<td>- legID=sendingSideID: &quot;01&quot;H</td>
</tr>
<tr>
<td></td>
<td>followed by a Connect invoke</td>
</tr>
<tr>
<td></td>
<td>Then SSF establishes the call ( a SetupReq to SigCon B, SetupConf from SigConB to SSF, then SetupResp to SigConB)</td>
</tr>
<tr>
<td></td>
<td>SigCon A (calling party) clears the call after it is answered (ReleaseInd sent)</td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
<td>Check that SSF sends to SCF an EventReportBCSM with the indication of</td>
</tr>
<tr>
<td></td>
<td>eventTypeBCSM= oDisconnect</td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
<td>none</td>
</tr>
<tr>
<td>Purpose:</td>
<td>Test of RequestReportBCSMEvent procedure and tBusy indication.</td>
</tr>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
</tbody>
</table>
| Preamble: | T_OS  
In addition, user B is declared busy |
| Test description | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM=tBusy  
- monitoringMode= notifyAndContinue  
followed by a Continue invoke  
Then SSF sends a SetupReq to SigCon B  
SigCon B releases the call (ReleaseInd sent) because user B is busy (UDUB="0D"H) |
| Pass criteria | Check that SSF sends to SCF an EventReportBCSM with the indication of  
eventTypeBCSM= tBusy |
<p>| Postamble: | none |</p>
<table>
<thead>
<tr>
<th><strong>XXXX</strong></th>
<th><strong>IN2_A_BASIC_RR_BV_29</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of RequestReportBCSMEvent procedure and tNoAnswer indication.</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>T_OS</td>
</tr>
<tr>
<td><strong>Test description</strong></td>
<td>SCF sends to SSF RequestReportBCSMEvent invoke containing parameters</td>
</tr>
<tr>
<td></td>
<td>- eventTypeBCSM=tNoAnswer</td>
</tr>
<tr>
<td></td>
<td>- monitoringMode= notifyAndContinue</td>
</tr>
<tr>
<td></td>
<td>followed by a Continue invoke</td>
</tr>
<tr>
<td></td>
<td>Then SSF sends a SetupReq to SigCon B</td>
</tr>
<tr>
<td></td>
<td>SigCon B releases the call (ReleaseInd sent) because user B does not answer</td>
</tr>
<tr>
<td><strong>Pass criteria</strong></td>
<td>Check that SSF sends to SCF an EventReportBCSM with the indication of</td>
</tr>
<tr>
<td></td>
<td>eventTypeBCSM= tNoAnswer</td>
</tr>
<tr>
<td><strong>Postamble:</strong></td>
<td>none</td>
</tr>
<tr>
<td>Purpose:</td>
<td>Test of <code>RequestReportBCSMEvent</code> procedure and <code>tAnswer</code> indication.</td>
</tr>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>T_OS</td>
</tr>
</tbody>
</table>
| Test description | SCF sends to SSF `RequestReportBCSMEvent` invoke containing parameters  
- `eventTypeBCSM = tAnswer`  
- `monitoringMode = notifyAndContinue`  
followed by a `Continue` invoke  
Then SSF sends a SetupReq to SigCon B  
SigCon B answers the call (SetupConf from SigConB to SSF) |
| Pass criteria | Check that SSF sends to SCF an `EventReportBCSM` with the indication of `eventTypeBCSM = tAnswer` |
| Postamble: | SigConA_Release_thenB_cause10 |
MSC IN2_A_BASIC_RR_BV_30

T(OS)

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsnEvents { { eventTypeBCSM tAnswer, monitorMode notifyAndContinue } } }

TC_InvokeReq
2, 51, 2, CUE, short, cUEArg : Null

TC_ContinueReq
51, oSCF

SetupReq
{ callRef 2, calledPartyNumber '2002'H, callingPartyNumber '1000'H }

SetupConf
{ callRef 2 }

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }

TC_EndInd
51, basic, FALSE

SetupResp
{ callRef 1 }

SigConA_Release_thenB_cause10
Purpose: Test of RequestReportBCSMEvent procedure and tMidCall indication.

Requirement ref

Selection Cond.

Preamble: T_OS

Test description

- SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
  - eventTypeBCSM= tMidCall
  - monitoringMode= notifyAndContinue
  followed by a Continue invoke

Then SSF sends a SetupReq to SigCon B. SetupConf from SigConB is received by SSF which issues SetupResp to SigConA.

SigConB called party initiates a service (ServiceFeatureInd sent to SSF) and tMidCall DP is reached

Pass criteria

- Check that SSF sends to SCF an EventReportBCSM with the indication of eventTypeBCSM= tMidCall

Postamble: SigConA_Release_thenB_cause10
MSC IN2_A_BASIC_RR_BV_31

TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM tMidCall, monitorMode notifyAndContinue } } }

TC_InvokeReq
2, 51, 2, CUE, short, cUEArg : Null

TC_ContinueReq
51, oSCF

SetupReq
{ callRef 2, calledPartyNumber '2002'H, callingPartyNumber '1000'H }

SetupConf
{ callRef 2 }

SetupResp
{ callRef 1 }

ServiceFeatureInd
{ callRef 2 }

TC_ContinueInd
51, oSSF, TRUE

TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tMidCall, messageType request }

TC_EndInd
51, basic, FALSE

SigConA_Release_thenB_cause10
Purpose: Test of RequestReportBCSMEvent procedure and oAbandon - transparent

Requirement ref

Selection Cond.

Preamble: O_OS

Test description
- SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
  - eventTypeBCSM=oAbandon
  - monitoringMode=interrupted then
- SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
  - eventTypeBCSM=oAbandon
  - monitoringMode=transparent then the calling party abandons the call before the call is answered (SigCon A to send ReleaseInd)

Pass criteria Check that SSF does not send to SCF an EventReportBCSM

Postamble: none
<table>
<thead>
<tr>
<th>XXXX</th>
<th>IN2_A_BASIC_RR_BV_33</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>Test of RequestReportBCSMEvent procedure and oAnswer - transparent</td>
</tr>
<tr>
<td><strong>Requirement ref</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
<td>O_OS</td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM=oAnswer  
- monitoringMode=interrupted  
SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM=oAnswer  
- monitoringMode=transparent  
followed by a Connect invoke  
then SSF sends a SetupReq to SigCon B  
SigCon B answers the call (SetupConf from SigCon B to SSF) |
| **Pass criteria** | Check that SSF does not send to SCF an EventReportBCSM |
| **Postamble:** | SigConA_Release_thenB |
| Purpose: | Test of RequestReportBCSMEvent procedure and oDisconnect indication - transparent |
| Requirement ref | |
| Selection Cond. | |
| Preamble: | O_Os |
| Test description | SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM= oDisconnect  
- monitoringMode= interrupted  
then SCF sends to SSF RequestReportBCSMEvent invoke containing parameters  
- eventTypeBCSM= oDisconnect  
- monitoringMode= transparent  
followed by a Connect invoke  
Then SSF establishes the call ( a SetupReq to SigCon B, SetupConf from SigConB to SSF, then SetupResp to SigConB)  
SigCon A (calling party) clears the call after it is answered (ReleaseInd sent) |
| Pass criteria | Check that SSF does not send to SCF an EventReportBCSM |
| Postamble: | none |
[MSC IN2_A_BASIC_RR_BV_34]

SCF  |  CS2_SSF  |  SigCon_A  |  SigCon_B

O_Os

TC_InvokeReq

1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted } } }

TC_ContinueReq

51, oSCF

TC_InvokeReq

2, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode transparent } } }

TC_InvokeReq

2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }

TC_ContinueReq

51, oSCF

SetupReq

[{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }]

SetupConf

[{ callRef 2 }]

SetupResp

[{ callRef 1 }]

ReleaseInd

[{ callRef 1, cause '00'H }]

ReleaseReq

[{ callRef 2, cause '10'H, senderRef }]

ReleaseReq
Purpose: Test of RequestReportBCSMEvent procedure and out of range parameter

Requirement ref

Selection Cond.

Preamble: O_OS

Test description

- SCF - SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
  - eventTypeBCSM= oMidCall
  - monitoringMode=interrupted
  - legID=invalid value

Pass criteria

- Check that SSF sends to SCF a RequestReportBCSMEvent error with the indication of out of range parameter
- When call Set-up is established, check that SSF is not sending to SCF any EventReportBCSMEvent

Postamble: SigConA_Release
Purpose: Test of RequestReportBCSMEvent procedure and missing parameter

Selection Cond.

Preamble: O_OS

Test description
SCF - SCF sends to SSF RequestReportBCSMEvent invoke containing parameters
- eventTypeBCSM= oMidCall
- monitoringMode=none

Pass criteria
- Check that SSF rejects the RequestReportBCSMEvent

Postamble: SigConA_Release

MSC IN2m_A_BASIC_RR_BI_02

SCF -> CS2_SSF -> SigCon_A -> SigCon_B

O_OS

TC_InvokeReq

2, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oMidCall } } }

TC_ContinueReq

[51, oSCF]

TC_ContinueInd

[51, oSSF, TRUE]

TC_ErrorInd

[2, 51, TRUE, missingParameterPar : Null]

SigConA_Release
6.4.15 SendChargingInformation procedure

Charging related aspects in IN are network operator specific. Therefore, it is not possible to define useful test purposes for charging procedures using a network operator independant approach. TP specification has to be done by network operators, using INAP procedures themselves. SendChargingInformation TP could be specified in combination with ApplyCharging and FurnishChargingInformation procedures.
### 6.4.16 RequestNotificationChargingEvent procedure

<table>
<thead>
<tr>
<th><strong>IN2_A_BASIC_RN_CA_01</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
</tr>
<tr>
<td><strong>Requirement ref:</strong></td>
</tr>
<tr>
<td><strong>Preamble:</strong></td>
</tr>
<tr>
<td><strong>Selection Cond.</strong></td>
</tr>
</tbody>
</table>

**Test description**

SCF sends to SSF RequestNotificationChargingEvent invoke containing mandatory parameters only, with:

- ChargingEvent
  - eventTypeCharging
  - monitorMode (interrupted)

**Pass criteria**

After triggering of charging event from SigConA, check that SSF sends to SCF an EventNotificationCharging with the indication of eventTypeCharging.

**Postamble:**

ReleaseCallAB_cause_00
### IN2_A_BASIC_RN_BV_01

<table>
<thead>
<tr>
<th>Purpose:</th>
<th>Test of <code>RequestNotificationChargingEvent</code> procedure with legID parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement ref</td>
<td></td>
</tr>
<tr>
<td>Selection Cond.</td>
<td></td>
</tr>
<tr>
<td>Preamble:</td>
<td>O_Os</td>
</tr>
</tbody>
</table>
| **Test description** | SCF sends to SSF `RequestNotificationChargingEvent` invoke containing mandatory parameters only, with:  
- ChargingEvent  
eventTypeCharging,  
monitorMode (interrupted)  
legID being sendingSideID |
| **Pass criteria** | After triggering of charging event from SigConA, check that SSF sends to SCF an `EventNotificationCharging` with the indication of eventTypeCharging and legID being sendingSideID |
| **Postamble:** | ReleaseCallAB_cause_00 |
MSC IN2m_A_BASIC_RN_BV_01

TC_InvokeReq

2, 51, 2, RNC, short, RNCArg : { { eventTypeCharging 'AAAA'H, monitorMode interrupted, legID sendingSide : '02'H } } TC_InvokeReq

3, 51, 4, CUE, medium, cUEArg : Null

SetupReq

{ calledPartyNumber '2000'H, callingPartyNumber '1000'H }

SetupConf

SetupResp

{ callRef 2 } ChargingEventInd

{ callRef 1 } ChargingEventInd

ReleaseCallAB_cause_00
**IN2_A_BASIC_RN_BV_02**

**Purpose:** Test of `RequestNotificationChargingEvent` procedure with legID parameter

**Requirement ref**

**Selection Cond.**

**Preamble:** O_S2P

**Test description**
SCF sends to SSF `RequestNotificationChargingEvent` invoke containing mandatory parameters only, with:
- `eventTypeCharging`
- `monitorMode (notifyAndContinue)`
- `legID` being `receivingSideID`

**Pass criteria**
After triggering of charging event from SigConB, check that SSF sends to SCF an `EventNotificationCharging` with the indication of `eventTypeCharging` and `legID` being `receivingSideID`

**Postamble:** SigConA_Release_thenB

---

**MSC IN2m_A_BASIC_RN_BV_02**

![Diagram of MSC IN2m_A_BASIC_RN_BV_02]
### Purpose:
Test of `RequestNotificationChargingEvent` procedure with missing parameter

### Requirement ref

### Selection Cond.

### Preamble:
O.OS

### Test description
SCF sends to SSF `RequestNotificationChargingEvent` invoke **without** mandatory parameters

### Pass criteria
check that SSF sends to SCF an `RequestNotificationChargingEvent` error with the indication of missingParameter

### Postamble:
SigConA_Release

---

### MSC IN2m_A_BASIC_RN_BI_01

```
SCF   CS2_SSF   SigCon_A   SigCon_B

O.OS

<table>
<thead>
<tr>
<th>TC_InvokeReq</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, 51, 2, RNC, medium,</td>
</tr>
<tr>
<td>TC_ContinueReq</td>
</tr>
<tr>
<td>[51, oSCF]</td>
</tr>
<tr>
<td>TC_ContinueInd</td>
</tr>
<tr>
<td>[51]</td>
</tr>
<tr>
<td>TC_ErrorInd</td>
</tr>
<tr>
<td>2, 51, TRUE, missingParameterPar : Null</td>
</tr>
</tbody>
</table>

SigConA_Release
```

---

**ETSI**
### IN2_A_BASIC_RN_BO_01

**Purpose:** Test of RequestNotificationChargingEvent procedure from wrong state

**Requirement ref**

**Selection Cond.**

**Preamble:** none

**Test description**

SCF sends to SSF RequestNotificationChargingEvent invoke containing mandatory parameters only, with:
- ChargingEvent
  - eventTypeCharging,
  - monitorMode (interrupted)

**Pass criteria**

Check that SSF sends to SCF a TC-ABORT

**Postamble:** none

### MSC IN2m_A_BASIC_RN_BO_02

```
SCF ----> CS2_SSF
       | TC_InvokeReq
       | [1, 51, 2, RNC, short, rNCArg : { eventTypeCharging
       |   AAAA'H, monitorMode notifyAndContinue }]
       | TC_ContinueReq
       | [51, oSCF]

SigCon_A ----> SigCon_B
       | TC_AbortInd
       | [51]
```

---

**ETSI**
Annex A (informative):
Description of various functional configurations

In these various configurations, the shaded area represents the implementation under test (IUT).

**Functional Configuration 1: Example for SCP with single SSP Non-Integrated or Integrated SRF**

![Diagram](image-url)

**Figure A.1: Configuration 1_1: IUT= SDF**

![Diagram](image-url)

**Figure A.2: Configuration 1_2: IUT= SCF**
Figure A.3: Configuration 1_3: IUT= SSF (non integrated with SRF)

Figure A.4: Configuration 1_4: IUT= SSF (integrated with SRF)
Functional Configuration 2: Example for direct path SCP-IP
Figure A.7: Configuration 2_1: IUT = SDF

Figure A.8: Configuration 2_2: IUT = SCF

Figure A.9: Configuration 2_3: IUT = SSF
Figure A.10: Configuration 2_4: IUT = SRF

Functional Configuration 3: Example for SSP Assist/Hand-off (assisting SSP with relay)

Figure A.11: Configuration 3_1: IUT = SDF
SSP Assist/Hand-off (assisting SSP with relay)

Figure A.12: Configuration 3_2: IUT= SCF

SSP Assist/Hand-off (assisting SSP with relay)

Figure A.13: Configuration 3_3: IUT= SSF of initiating SSP
SSP Assist/Hand-off (assisting SSP with relay)

Figure A.14: Configuration 3_4: IUT= SSF of assisting SSP (integrated SRF)

SSP Assist/Hand-off (assisting SSP with relay)

Figure A.15: Configuration 3_5: IUT= SRF (integrated with assisting SSF)
Figure A.16: Configuration 3_6: IUT= SRF (non integrated with assisting SSF)

Figure A.17: Configuration 3_7: IUT= SSF of assisting SSP (non integrated SRF)
Functional Configuration 4: Example for SSP Assist/Hand-off (initiating SSP with relay)

SSP Assist/Hand-off (initiating SSP with relay)

Figure A.18: Configuration 4_1: IUT= SDF

SSP Assist/Hand-off (initiating SSP with relay)

Figure A.19: Configuration 4_2: IUT= SCF
Figure A.20: Configuration 4_3: IUT= SSF of initiating SSP (integrated SRF)

Figure A.21: Configuration 4_4: IUT= SSF of initiating SSP (non integrated SRF)
SSP Assist/Hand-off (initiating SSP with relay)

Figure A.22: Configuration 4_5: IUT= SRF of initiating SSP (non integrated SRF)

SSP Assist/Hand-off (initiating SSP with relay)

Figure A.23: Configuration 4_6: IUT= SRF of initiating SSP (integrated SRF)
SSP Assist/Hand-off (initiating SSP with relay)

**Figure A.24: Configuration 4_7: IUT= SRF as IP**

Functional Configuration: Example for SSP Assist/Hand-off (initiating and assisting SSP with relay)

SSP Assist/Hand-off (initiating and assisting SSP with relay)

**Figure A.25: Configuration 5_1: IUT= SDF**
Figure A.26: Configuration 5_2: IUT= SCF

Figure A.27: Configuration 5_3: IUT= SSF of assisting SSP(with integrated SRF)
SSP Assist/Hand-off (initiating and assisting SSP with relay)

Figure A.28: Configuration 5_4: IUT= SSF of initiating SSP (with integrated SRF)

SSP Assist/Hand-off (initiating and assisting SSP with relay)

Figure A.29: Configuration 5_5: IUT= SSF of assisting SSP (with non-integrated SRF)
SSP Assist/Hand-off (initiating and assisting SSP with relay)

Figure A.30: Configuration 5_6: IUT= SSF of initiating SSP(with non integrated SRF)

SSP Assist/Hand-off (initiating and assisting SSP with relay)

Figure A.31: Configuration 5_7: IUT= SRF of initiating SSP(non integrated SRF)
SSP Assist/Hand-off (initiating and assisting SSP with relay)

Figure A.32: Configuration 5_8: IUT= SRF of initiating SSP(integrated SRF)

SSP Assist/Hand-off (initiating and assisting SSP with relay)

Figure A.33: Configuration 5_9: IUT= SRF of assisting SSP(non integrated SRF)
SSP Assist/Hand-off (initiating and assisting SSP with relay)

Figure A.34: Configuration 5_10: IUT = SRF of assisting SSP (integrated SRF)
Annex B (normative):
Parameter values used in MSCs for CORE INAP primitives

The following table is an abstract from the PIXIT for CORE INAP CS1, showing the values of the parameters of CORE INAP primitives used to design the MSCs.

**Table B.1**

<table>
<thead>
<tr>
<th>Item</th>
<th>Parameter</th>
<th>Parameter type</th>
<th>Explanation/Format</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIX_AChBillingChargingCharacteristics</td>
<td>AChBillingChargingCharacteristics</td>
<td>&quot;xx&quot;H</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>PIX_AlertingPattern</td>
<td>AlertingPattern</td>
<td>&quot;xxx&quot;H</td>
<td>123</td>
<td></td>
</tr>
<tr>
<td>PIX_AlertingPattern_ICA</td>
<td>AlertingPattern</td>
<td>&quot;xxx&quot;H</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>PIX_APtyAbandonCause</td>
<td>Cause</td>
<td>&quot;xx&quot;H</td>
<td>0F</td>
<td></td>
</tr>
<tr>
<td>PIX_APtyDiscCause</td>
<td>Cause</td>
<td>&quot;xx&quot;H</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>PIX_AssistingSSPIRoutingAddress</td>
<td>AssistingSSPIRoutingAddress</td>
<td>&quot;xxx&quot;H</td>
<td>7755</td>
<td></td>
</tr>
<tr>
<td>PIX_BPtyBusy_UDUBCause</td>
<td>Cause</td>
<td>&quot;xx&quot;H</td>
<td>0D</td>
<td></td>
</tr>
<tr>
<td>PIX_BPtyNoAnswerCause</td>
<td>Cause</td>
<td>&quot;xx&quot;H</td>
<td>09</td>
<td></td>
</tr>
<tr>
<td>PIX_CalledPartyNumber1_CON</td>
<td>CalledPartyNumber</td>
<td>LegId 2</td>
<td>&quot;xxxx&quot;H</td>
<td>2001</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber2_CON</td>
<td>CalledPartyNumber</td>
<td>LegId 3</td>
<td>&quot;xxxx&quot;H</td>
<td>2003</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber3_CON</td>
<td>CalledPartyNumber</td>
<td>LegId 4</td>
<td>&quot;xxxx&quot;H</td>
<td>2005</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber4_CON</td>
<td>CalledPartyNumber</td>
<td>LegId 5</td>
<td>&quot;xxxx&quot;H</td>
<td>2007</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber5_CON</td>
<td>CalledPartyNumber</td>
<td>LegId 6</td>
<td>&quot;xxxx&quot;H</td>
<td>2009</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber6_CON</td>
<td>CalledPartyNumber</td>
<td>LegId 7</td>
<td>&quot;xxxx&quot;H</td>
<td>2011</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber7_CON</td>
<td>CalledPartyNumber</td>
<td>LegId 8</td>
<td>&quot;xxxx&quot;H</td>
<td>2013</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber8_CON</td>
<td>CalledPartyNumber</td>
<td>LegId 9</td>
<td>&quot;xxxx&quot;H</td>
<td>2015</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber1_ICA</td>
<td>CalledPartyNumber</td>
<td>LegId 2</td>
<td>&quot;xxxx&quot;H</td>
<td>2100</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber2_ICA</td>
<td>CalledPartyNumber</td>
<td>LegId 3</td>
<td>&quot;xxxx&quot;H</td>
<td>2101</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber3_ICA</td>
<td>CalledPartyNumber</td>
<td>LegId 4</td>
<td>&quot;xxxx&quot;H</td>
<td>2102</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber4_ICA</td>
<td>CalledPartyNumber</td>
<td>LegId 5</td>
<td>&quot;xxxx&quot;H</td>
<td>2103</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber5_ICA</td>
<td>CalledPartyNumber</td>
<td>LegId 6</td>
<td>&quot;xxxx&quot;H</td>
<td>2104</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber6_ICA</td>
<td>CalledPartyNumber</td>
<td>LegId 7</td>
<td>&quot;xxxx&quot;H</td>
<td>2105</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber7_ICA</td>
<td>CalledPartyNumber</td>
<td>LegId 8</td>
<td>&quot;xxxx&quot;H</td>
<td>2106</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber1_SetupInd</td>
<td>CalledPartyNumber</td>
<td>LegId 2</td>
<td>&quot;xxxx&quot;H</td>
<td>2000</td>
</tr>
<tr>
<td>PIX_CalledPartyNumber2_SetupInd</td>
<td>CalledPartyNumber</td>
<td>LegId 3</td>
<td>&quot;xxxx&quot;H</td>
<td>2002</td>
</tr>
<tr>
<td>PIX_CallingPartyNumber1</td>
<td>CallingPartyNumber</td>
<td>&quot;xxxx&quot;H</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>PIX_CallingPartyNumber2</td>
<td>CallingPartyNumber</td>
<td>&quot;xxxx&quot;H</td>
<td>1002</td>
<td></td>
</tr>
<tr>
<td>PIX_CallingPartysCategory_CON</td>
<td>CallingPartysCategory</td>
<td>&quot;xx&quot;H</td>
<td>BB</td>
<td></td>
</tr>
<tr>
<td>PIX_CallingPartysCategory_SetupInd</td>
<td>CallingPartysCategory</td>
<td>&quot;xx&quot;H</td>
<td>CC</td>
<td></td>
</tr>
<tr>
<td>PIX_DateAndTime</td>
<td>DateAndTime</td>
<td>YYYYMMDDHHMMSS</td>
<td>980115123030</td>
<td></td>
</tr>
<tr>
<td>PIX_Duration</td>
<td>Duration</td>
<td>Seconds</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>PIX_EventTypeCharging1</td>
<td>EventTypeCharging</td>
<td>&quot;AAAA&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIX_EventTypeCharging2</td>
<td>EventTypeCharging</td>
<td>&quot;CCCC&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIX_FCIBillingChargingCharacteristics</td>
<td>FCIBillingChargingCharacteristics</td>
<td>55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIX_InbandInfo_message</td>
<td>InbandInfo</td>
<td>InformationToSend</td>
<td>&quot;AABB&quot;</td>
<td></td>
</tr>
<tr>
<td>PIX_Interval</td>
<td>Integer</td>
<td>Seconds</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>PIX_IPRoutingAddress</td>
<td>IPRoutingAddress</td>
<td>&quot;xxx&quot;H</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>PIX_LocationNumber</td>
<td>LocationNumber</td>
<td>&quot;xxxx&quot;H</td>
<td>9001</td>
<td></td>
</tr>
<tr>
<td>PIX_MaximumNumberOfCounters</td>
<td>MaximumNumberOfCounters</td>
<td>&quot;xx&quot;H</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>PIX_NumberOfCalls</td>
<td>Integer</td>
<td>xx</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>PIX_OriginalCalledPartyNumber</td>
<td>CalledPartyNumber</td>
<td>&quot;xxxx&quot;H</td>
<td>2211</td>
<td></td>
</tr>
<tr>
<td>PIXRedirectingPartyNumber</td>
<td>CalledPartyNumber</td>
<td>&quot;xxxx&quot;H</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>PIX_RedirectionInformation</td>
<td>RedirectionInformation</td>
<td>&quot;xx&quot;H</td>
<td>AA</td>
<td></td>
</tr>
<tr>
<td>PIX_ReleaseCause</td>
<td>Cause</td>
<td>&quot;xx&quot;H</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>PIX_RouteSelectFailure1Cause</td>
<td>Cause</td>
<td>&quot;xx&quot;H</td>
<td>0B</td>
<td></td>
</tr>
<tr>
<td>PIX_RouteSelectFailure2Cause</td>
<td>Cause</td>
<td>&quot;xx&quot;H</td>
<td>0C</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Parameter</td>
<td>Parameter type</td>
<td>Explanation/Format</td>
<td>Value</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>----------------</td>
<td>--------------------</td>
<td>-------</td>
</tr>
<tr>
<td>PIX_ScfID</td>
<td>ScfID</td>
<td><em>xxxx</em>H</td>
<td></td>
<td>8881</td>
</tr>
<tr>
<td>PIX_ServiceInteractionIndicators</td>
<td>ServiceInteractionIndicators</td>
<td><em>xx</em>H</td>
<td></td>
<td>22</td>
</tr>
<tr>
<td>PIX_ServiceKey1</td>
<td>ServiceKey</td>
<td><em>xx</em>H</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>PIX_ServiceKey2</td>
<td>ServiceKey</td>
<td><em>xx</em>H</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>PIX_SFBillingChargingCharacteristics</td>
<td>SFBillingChargingCharacteristics</td>
<td><em>xxxx</em>H</td>
<td></td>
<td>BBBB</td>
</tr>
<tr>
<td>PIX_StartTime</td>
<td>DateAndTime</td>
<td>YYMMDDHHMMSS</td>
<td></td>
<td>971128113015</td>
</tr>
<tr>
<td>PIX_StopTime</td>
<td>DateAndTime</td>
<td>YYMMDDHHMMSS</td>
<td></td>
<td>971212113015</td>
</tr>
<tr>
<td>PIX_ElementaryMessageID</td>
<td>integer</td>
<td>xxx</td>
<td></td>
<td>191</td>
</tr>
<tr>
<td>PIX_CorrelationId</td>
<td>correlationID</td>
<td><em>xxx</em>H</td>
<td></td>
<td>AAA</td>
</tr>
<tr>
<td>PIX.UiScriptID1</td>
<td>integer</td>
<td>xxx</td>
<td></td>
<td>202</td>
</tr>
<tr>
<td>PIX.UiScriptID2</td>
<td>integer</td>
<td>xxx</td>
<td></td>
<td>203</td>
</tr>
<tr>
<td>PIX.UiScriptIDInvalid</td>
<td>integer</td>
<td>xxx</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td>PIX.UiScriptResult</td>
<td>UiScriptResult</td>
<td><em>xxxx</em>H</td>
<td></td>
<td>5110</td>
</tr>
<tr>
<td>PIX.UiScriptSpecificInfo</td>
<td>UiScriptSpecificInfo</td>
<td><em>xxxx</em>H</td>
<td></td>
<td>5220</td>
</tr>
</tbody>
</table>
Annex C (normative): Parameter values used in MSCs for TCAP primitives

The following table is an abstract from the PIXIT for CORE INAP CS1, showing the values of the parameters of TCAP primitives used to design the MSCs.

Table C.1: Parameter values

<table>
<thead>
<tr>
<th>Item</th>
<th>Parameter</th>
<th>Parameter type</th>
<th>Explanation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIX_Invokeld</td>
<td>InvokeIdType</td>
<td></td>
<td>Direction SCF -&gt; SSF</td>
<td>1-100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Direction SSF -&gt; SCF</td>
<td>101-200</td>
</tr>
<tr>
<td>PIX_Dialogueld</td>
<td>DialogueIdType</td>
<td></td>
<td>Direction SCF -&gt; SSF</td>
<td>1-50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Direction SSF -&gt; SCF</td>
<td>51-100</td>
</tr>
</tbody>
</table>
Bibliography

The following material, though not specifically referenced in the body of the present document (or not publicly available), gives supporting information.

- ETSI EN 301 140-2: "Intelligent Network (IN); Intelligent Network Application Protocol (INAP); Capability Set 2 (CS2); Part 2: Protocol Implementation Conformance Statement (PICS) proforma specification".
## History

<table>
<thead>
<tr>
<th>Document history</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>V1.1.3</strong></td>
</tr>
</tbody>
</table>