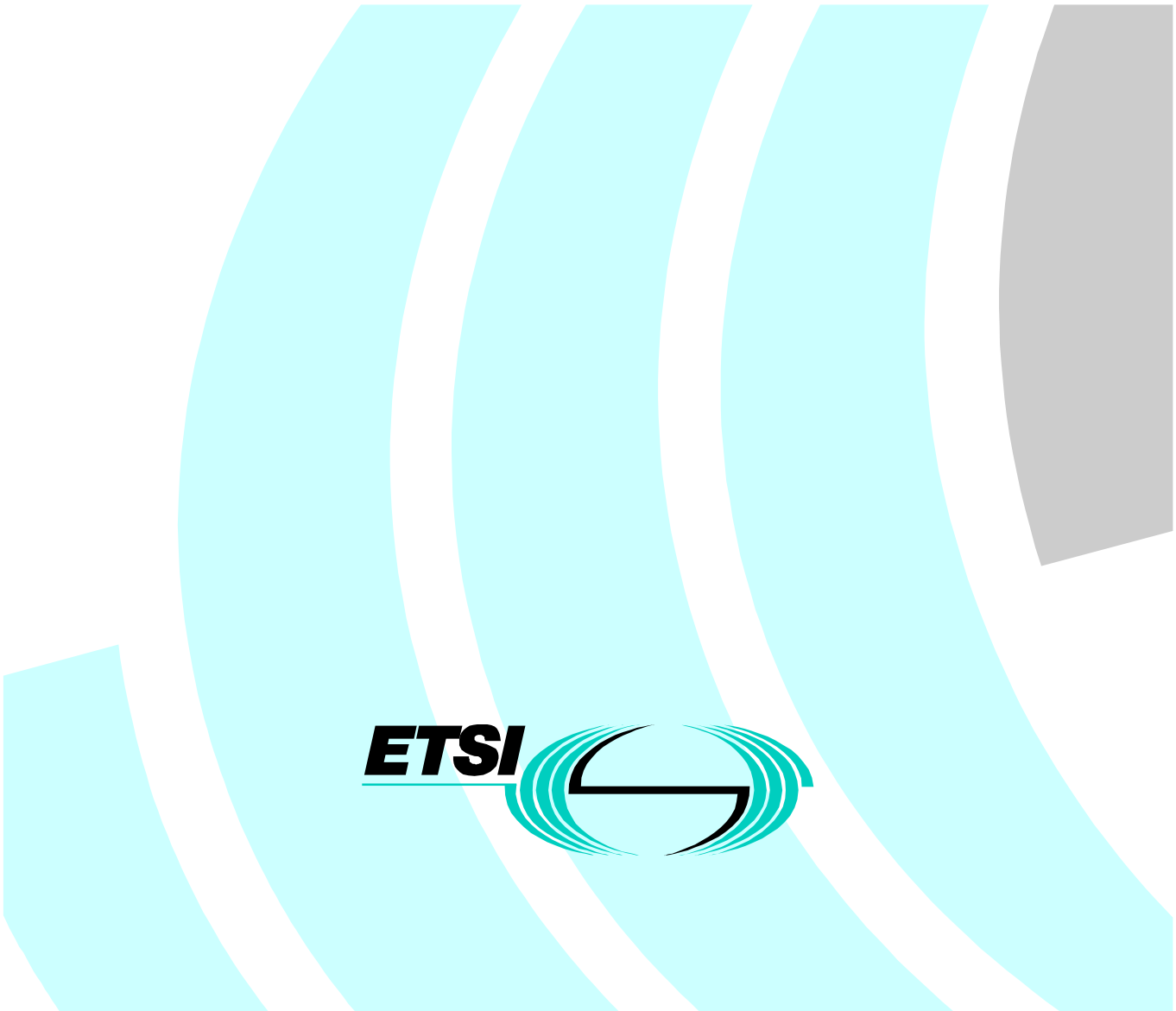
Draft EN 301 140-3-1 V1.1.2 (1999-06)

European Standard (Telecommunications series)

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



Reference
DEN/SPS-03038-3-1 (ak0r9idc.PDF)

Keywords
IN, CS2, INAP, TSS&TP

ETSI

Postal address
F-06921 Sophia Antipolis Cedex - FRANCE

Office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16
Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Internet
secretariat@etsi.fr
Individual copies of this ETSI deliverable
can be downloaded from
http://www.etsi.org
If you find errors in the present document, send your
comment to: editor@etsi.fr

Contents

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 **free of charge** 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 Public Enquiry 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 Application Protocol (INAP) capability set 2, 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	
Date of latest announcement of this EN (doa):	3 months after ETSI publication
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	6 months after doa
Date of withdrawal of any conflicting National Standard (dow):	6 months after doa

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.

ISO/IEC 9646-1 [4] and ISO/IEC 9646-2 [5] are used as the basis for the test methodology.

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

[2] 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".

[3] 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)".

[4] ISO/IEC 9646-1: "Information technology - Open systems interconnection - Conformance testing methodology and framework - Part 1: General concepts".

[5] ISO/IEC 9646-2: "Information technology - Open systems interconnection - Conformance testing methodology and framework - Part 2: Abstract test suite 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];
- terms defined in ISO/IEC 9646-1 [4] and in ISO/IEC 9646-2 [5].

In particular, the following terms defined in ISO/IEC 9646-1 [4] 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:

ATS	Abstract Test Suite
BI	Invalid Behaviour tests
BO	Inopportune Behaviour tests
bS	tests for SSP basic functions
BV	Valid Behaviour tests
CA	Capability tests
CS	Capability Set
CS	Call Segment
EDP-N	Event Detection Point - Notification
EDP-R	Event Detection Point - Request
FE	Functional Entity
FSM	Finite State Machine
IN	Intelligent Network
INAP	Intelligent Network Application Protocol
IP	Intelligent Peripheral
iS	initiating SSF
iSSP	initiating SSP
IUT	Implementation Under Test
MSC	Message Sequence Chart
PICS	Protocol Implementation Conformance Statement
PDU	Protocol Data Unit
rS	SSF relay
SCF	Service Control Function
SCP	Service Control Point
SDF	Service Data Function
SDL	Specification and Description Language
SDP	Service Data Point
SRF	Specialized Resource Function
SRSM	SRF call State Model
SSF	Service Switching Function
SSME	SSF Management Entity
SSP	Service Switching Point
SUT	System Under Test
TCAP	Transaction Capabilities Application Part
TDP	Trigger Detection Point
TP	Test Purpose
TSS	Test Suite Structure

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 that 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 1a

(TP name, also corresponding test case name)	
Test Purpose:	
Requirement Ref:	
Selection Cond:	
Test preamble:	
Test description	
Pass criteria:	
Test postamble	

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 1b: TP identifier naming convention scheme

Identifier:		**IN2_<i>_<sss>_<pp>_<cc>_<nn>**		
IN2		indicates IN Capability Set 1 and 2 (CS-1 being in CS-2)		
<i>	=	interface:	A	SSF-SCF interface
			B	SSF-SRF interface
			C	SCF-SCF interface
<sss>	=	common set	BASIC	Basic set for CS-1 complemented for CS-2
			CPH	Call Party Handling from Capability Set 2
			CTM	Cordless Terminal Portability from Capability Set 2
<pp>	=	procedure name like		
			SF	ServiceFiltering
<cc>	=	test category:		
			CA	Capability tests
			BV	Valid Behaviour tests
			BI	Invalid Behaviour tests
			BO	Inopportune Behaviour tests
<nn>	=	sequential number:		(01-99)

Example of test purpose and test case name: **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:

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

where:

a is letter:

- O for Originating (outgoing call for a user);
- T for Terminating (incoming call for a user);
- I for Initiate Call Attempt (initiated from the network).

The state names and their abbreviations used are:

Null

1 Party	1P
Originating Setup	OS
Terminating Setup	TS
Terminating 1 Party Setup	T1P
Stable 1 Party	S1P
Stable 2 Party	S2P
Transfer (no. of passive legs)	TF(x)
On Hold (no. of passive legs)	OH(x)
Stable MultiParty (no. of passive legs)	S(x)P

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 [1].

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.

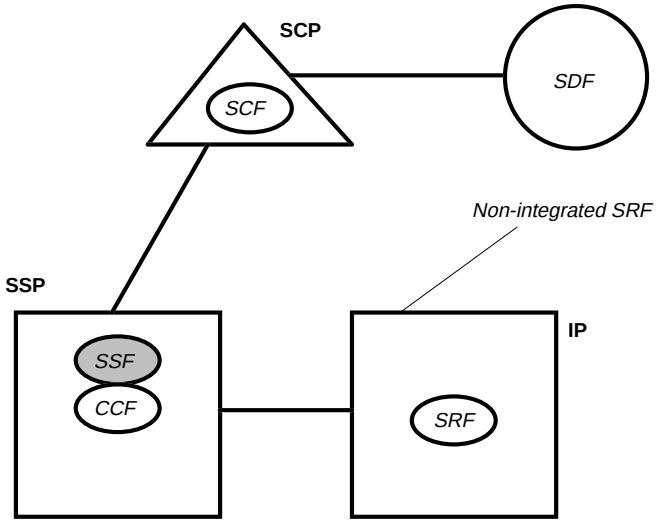
The list of the informative parameters, for which a value is to be assigned in particular for the execution of a test suite, is included in the PIXIT proforma. See reference [3].

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.

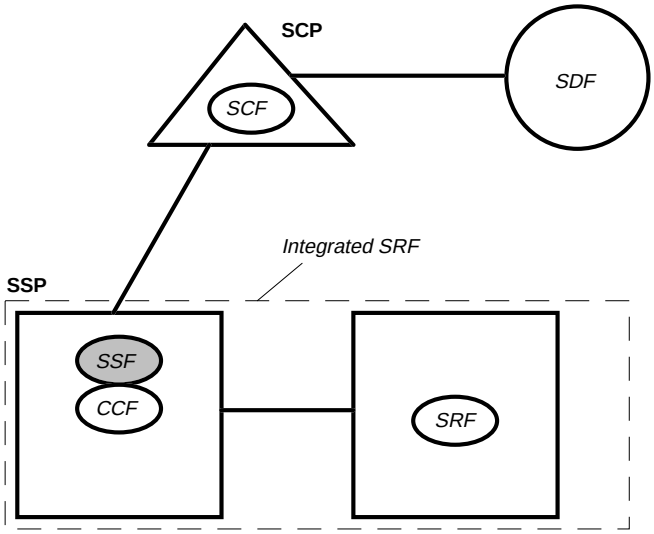
5 Functional configurations under test

5.1 SSF basic functions



SCP with single SSP

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



SCP with single SSP

Figure 2: Configuration 2: IUT= SSF (integrated with SRF)

5.2 SSF additional functions

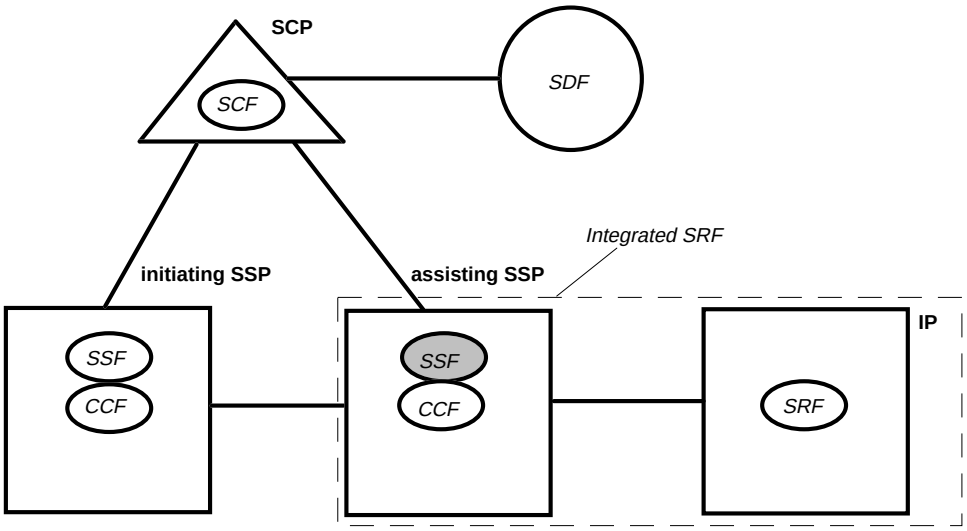


Figure 3: Configuration 3: IUT= SSF of assisting SSP (integrated SRF)

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

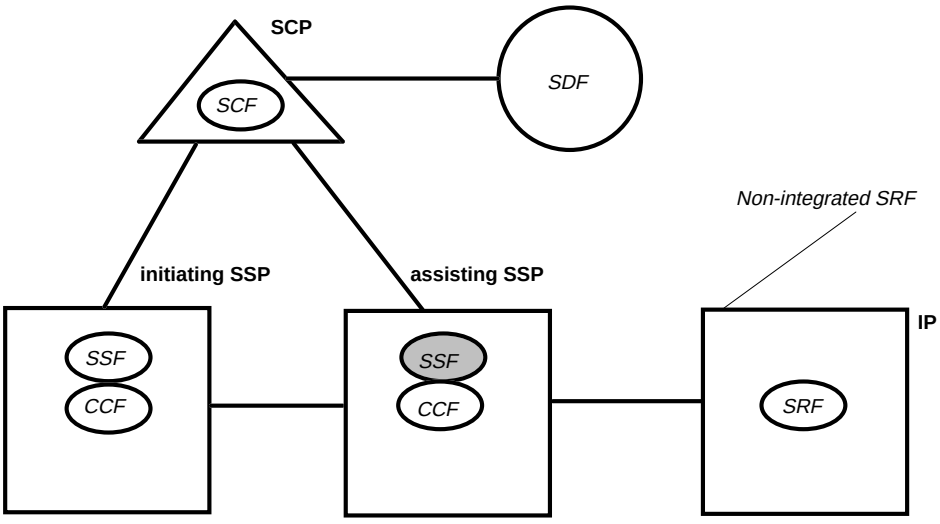


Figure 4: Configuration 4: IUT= SSF of assisting SSP (non integrated SRF)

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

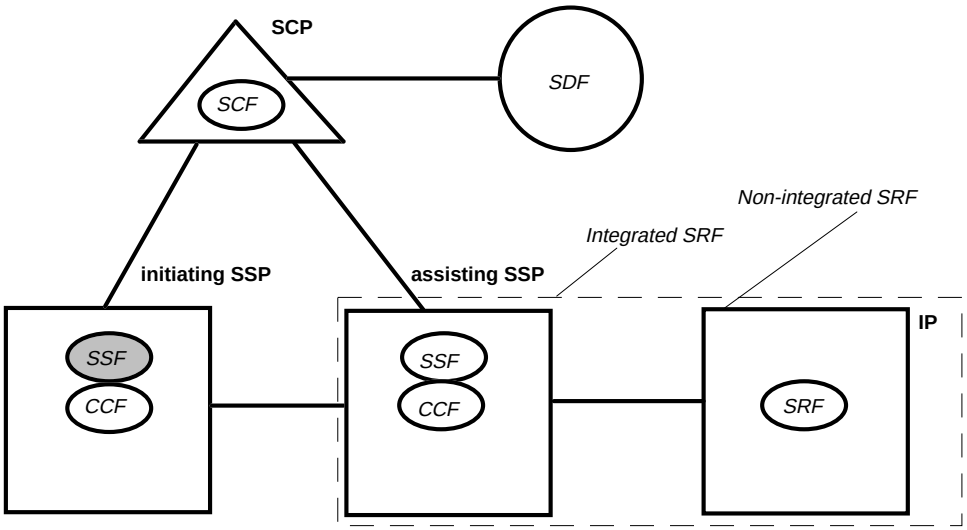


Figure 5: Configuration 5: IUT= SSF of initiating SSP

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

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;

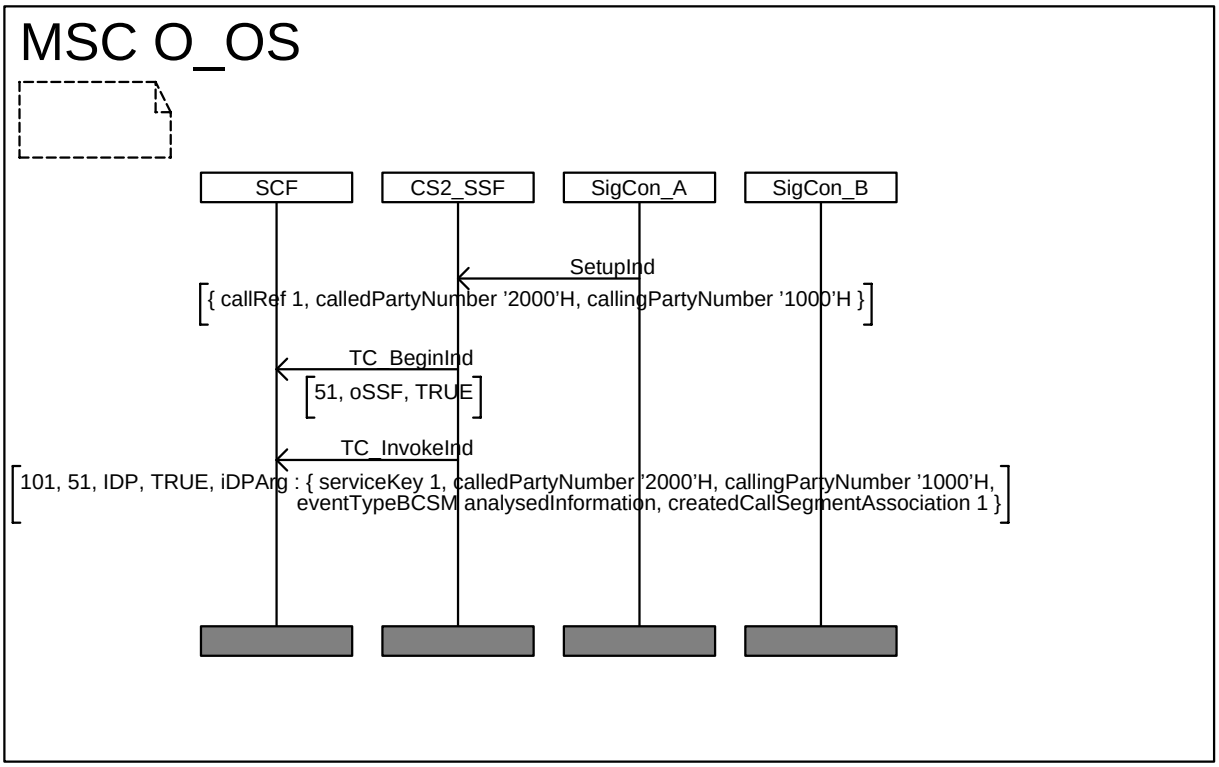
SigConB_Release.

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

6.1.2 Preamble descriptions

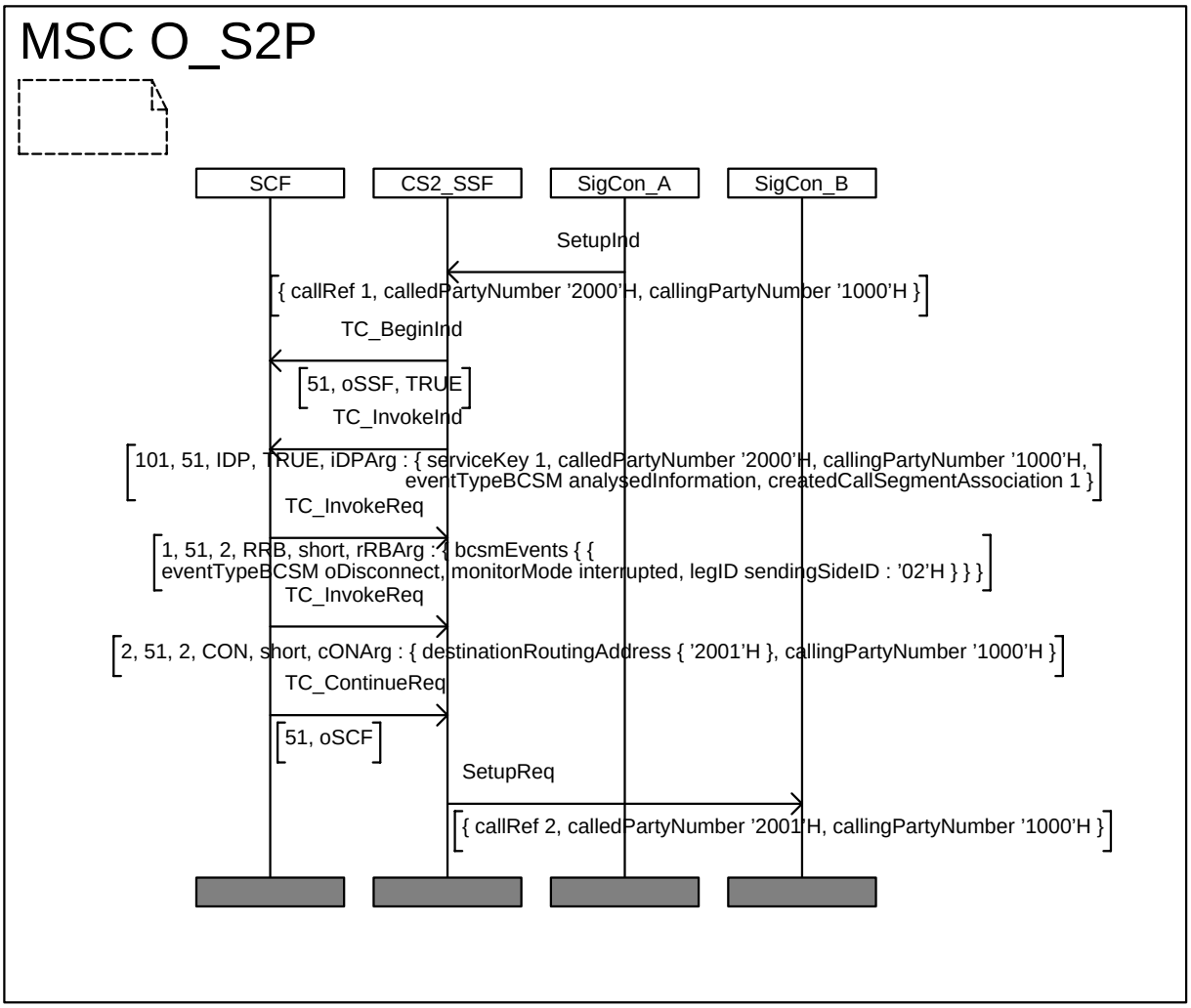
6.1.2.1 O_OS preamble

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

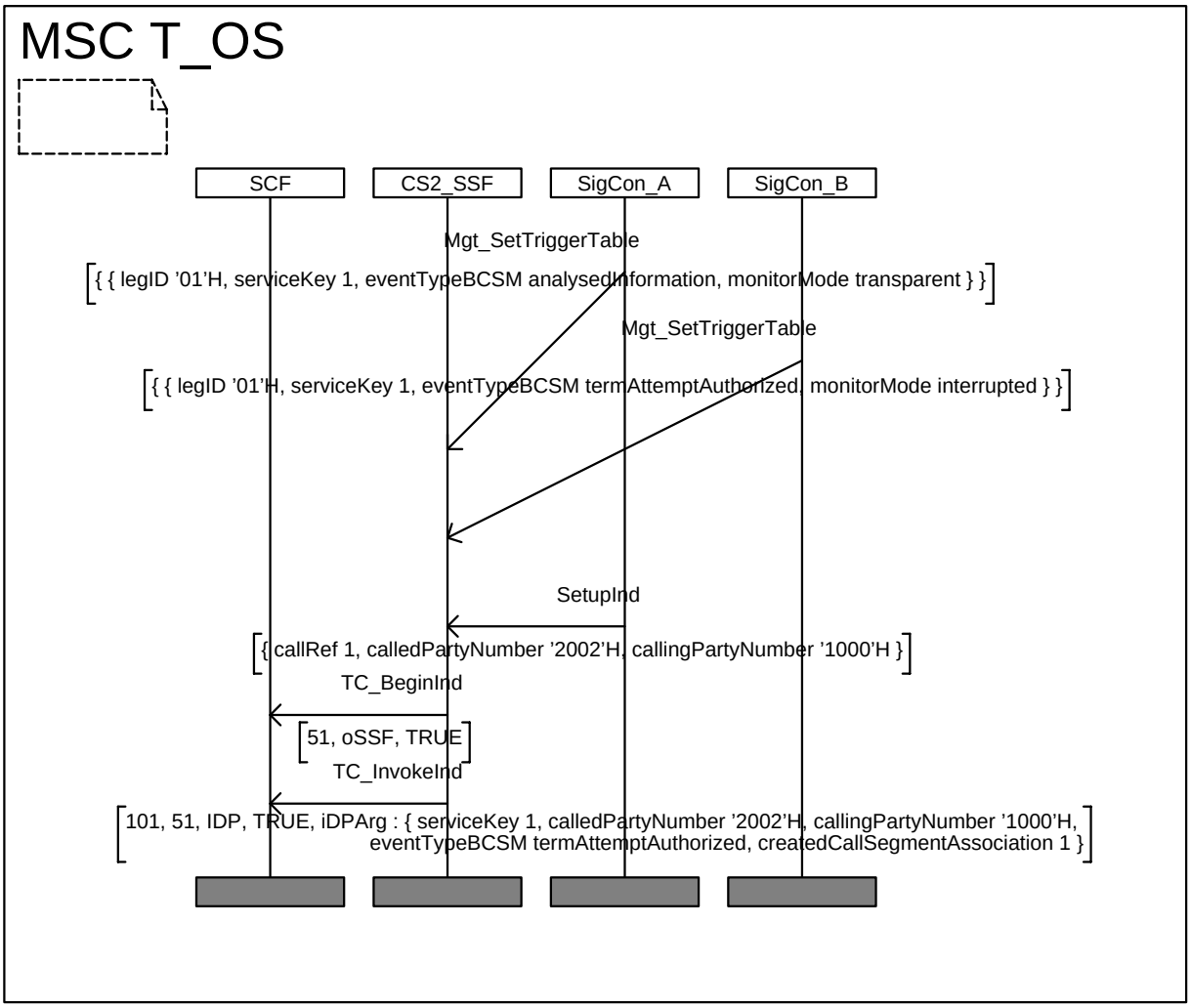


6.1.2.2 O_S2P preamble

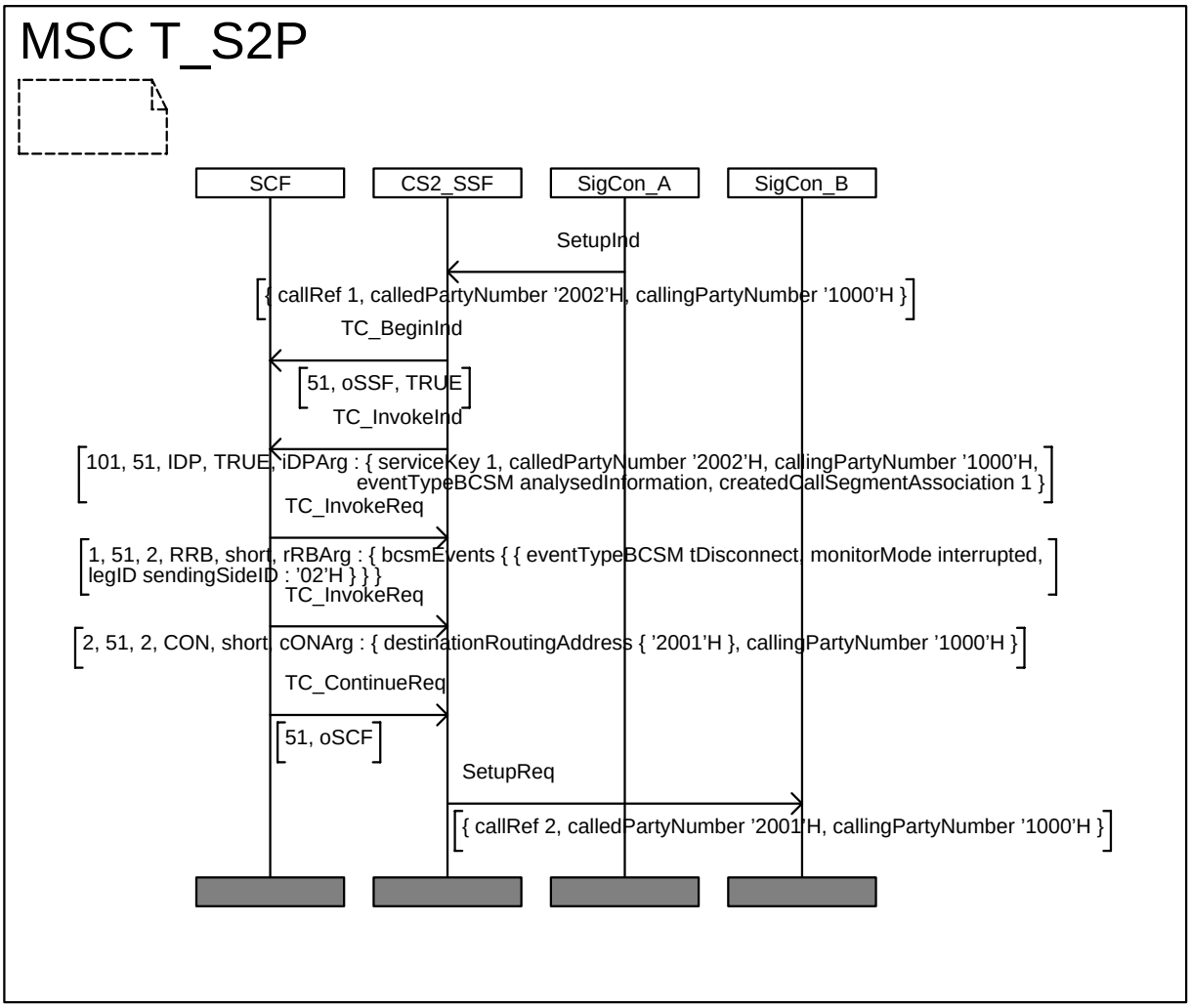
This preamble is used to bring the IUT from the idle or Null state to the 2 party state.


MSC O_S2P
SCF
CS2_SSF
SigCon_A
SigCon_B
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



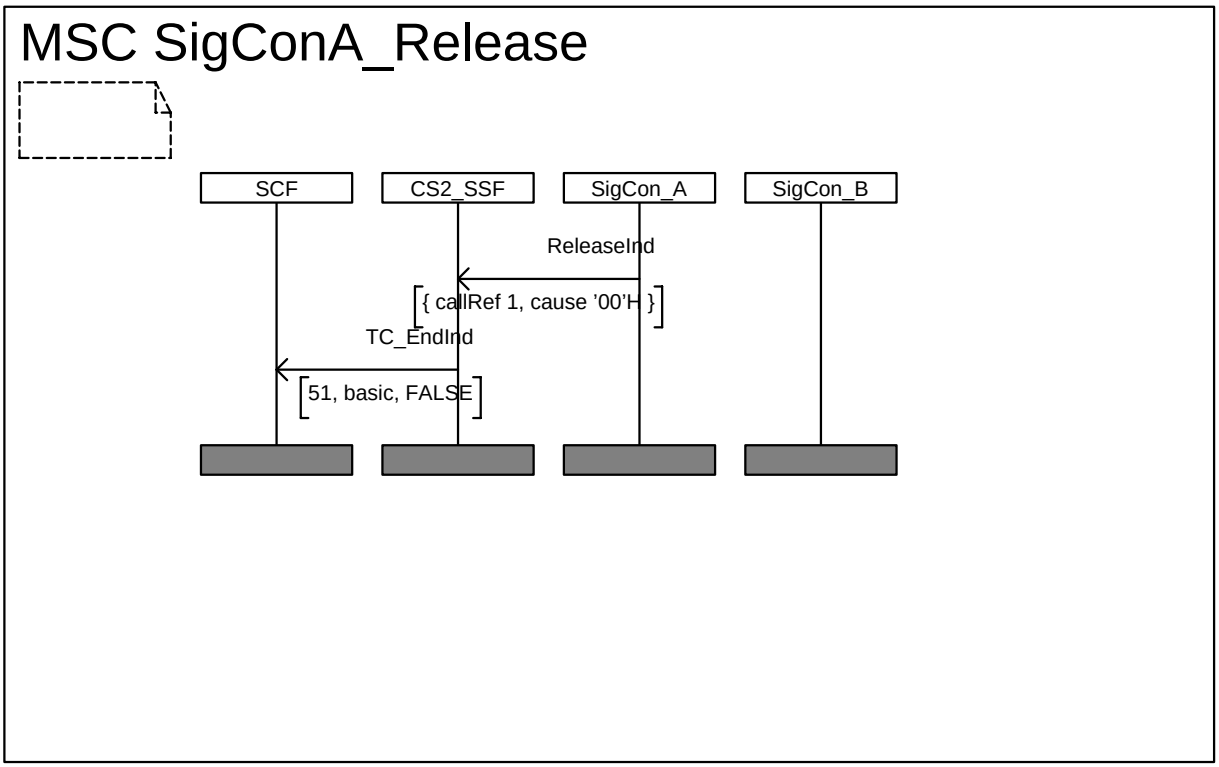
6.1.2.4 T_S2P preamble



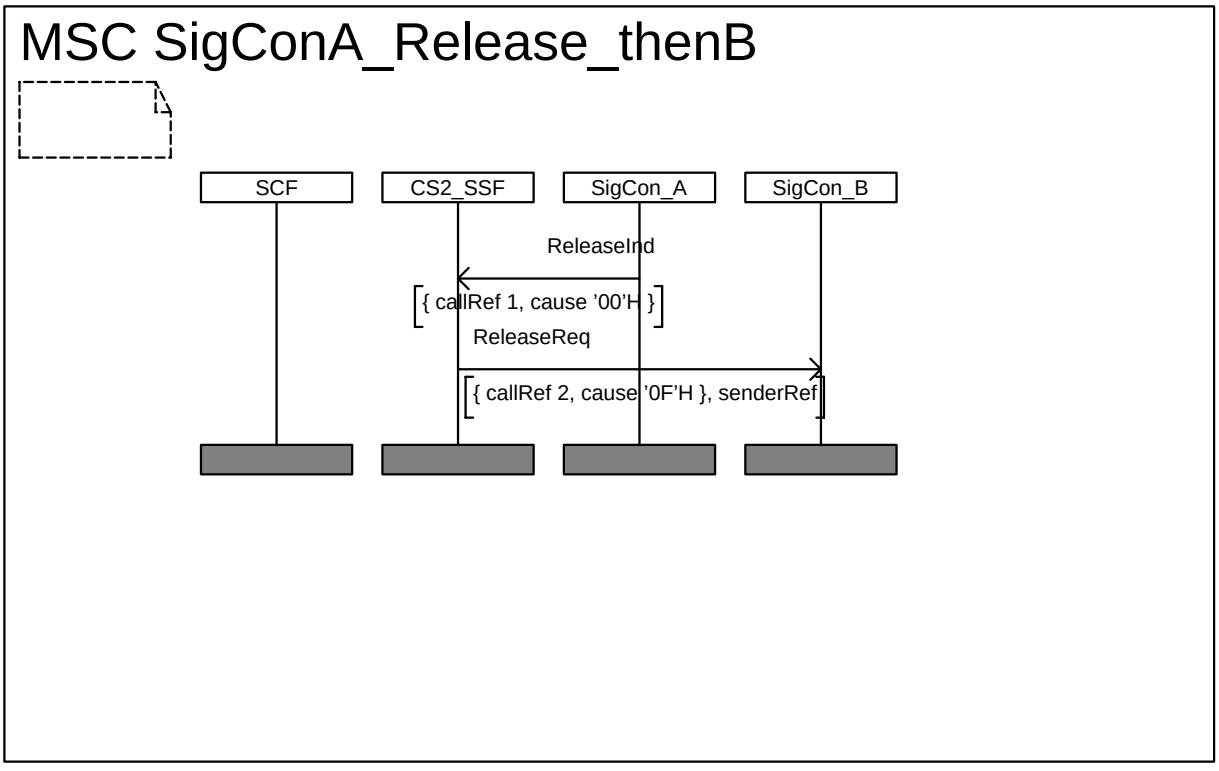
6.1.3 Postamble descriptions

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

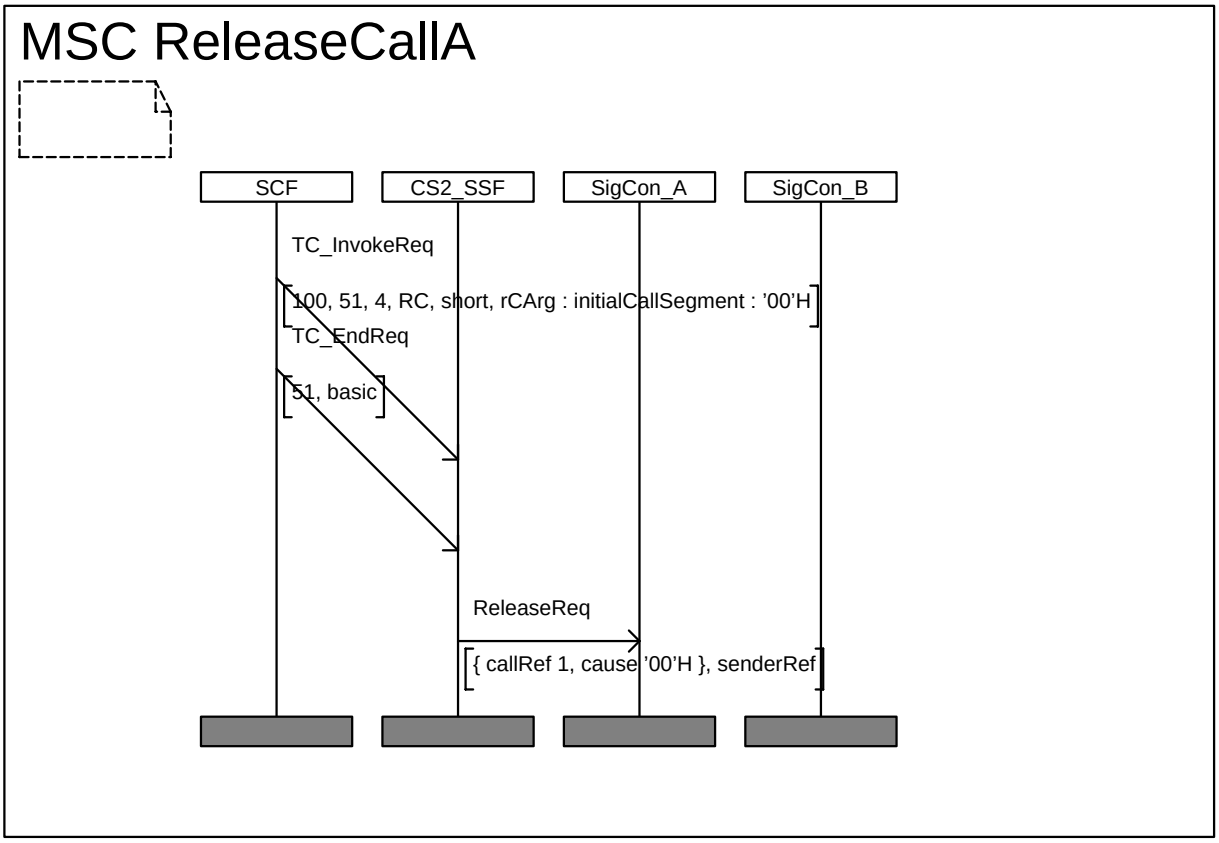
6.1.3.1 SigConA_Release postamble



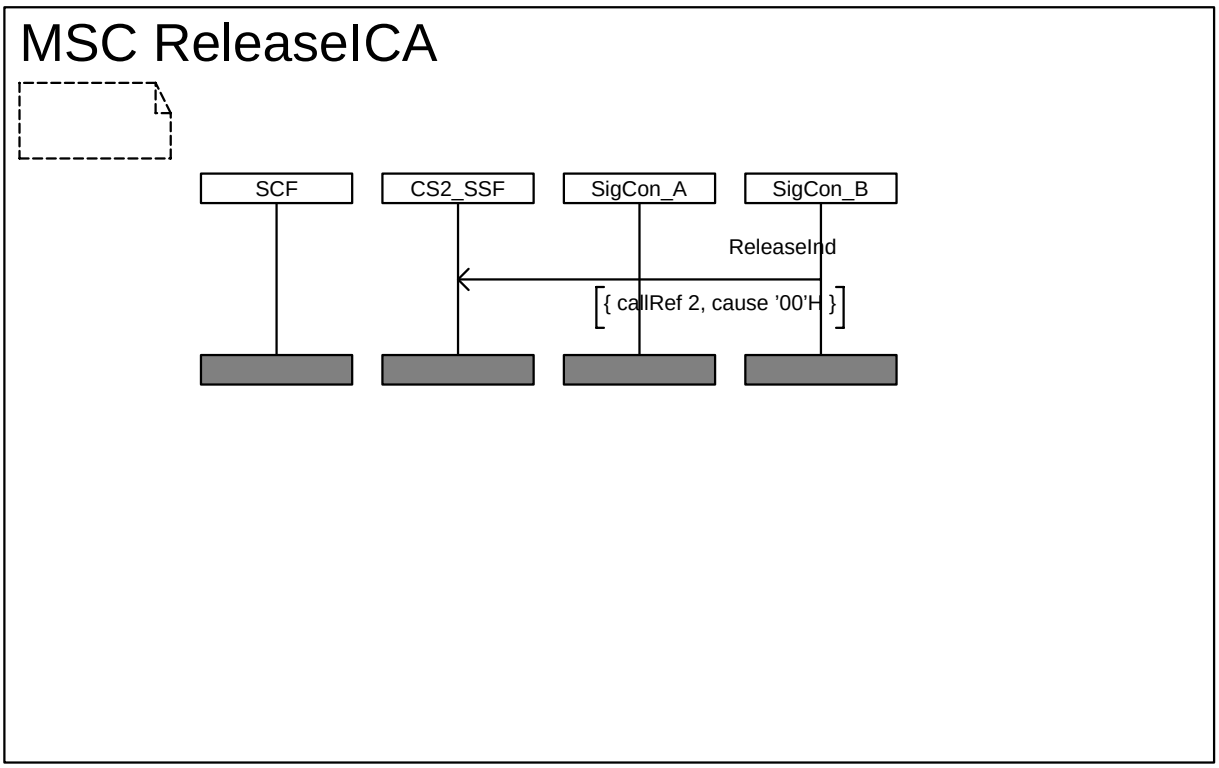
6.1.3.2 SigConA_Release_thenB postamble



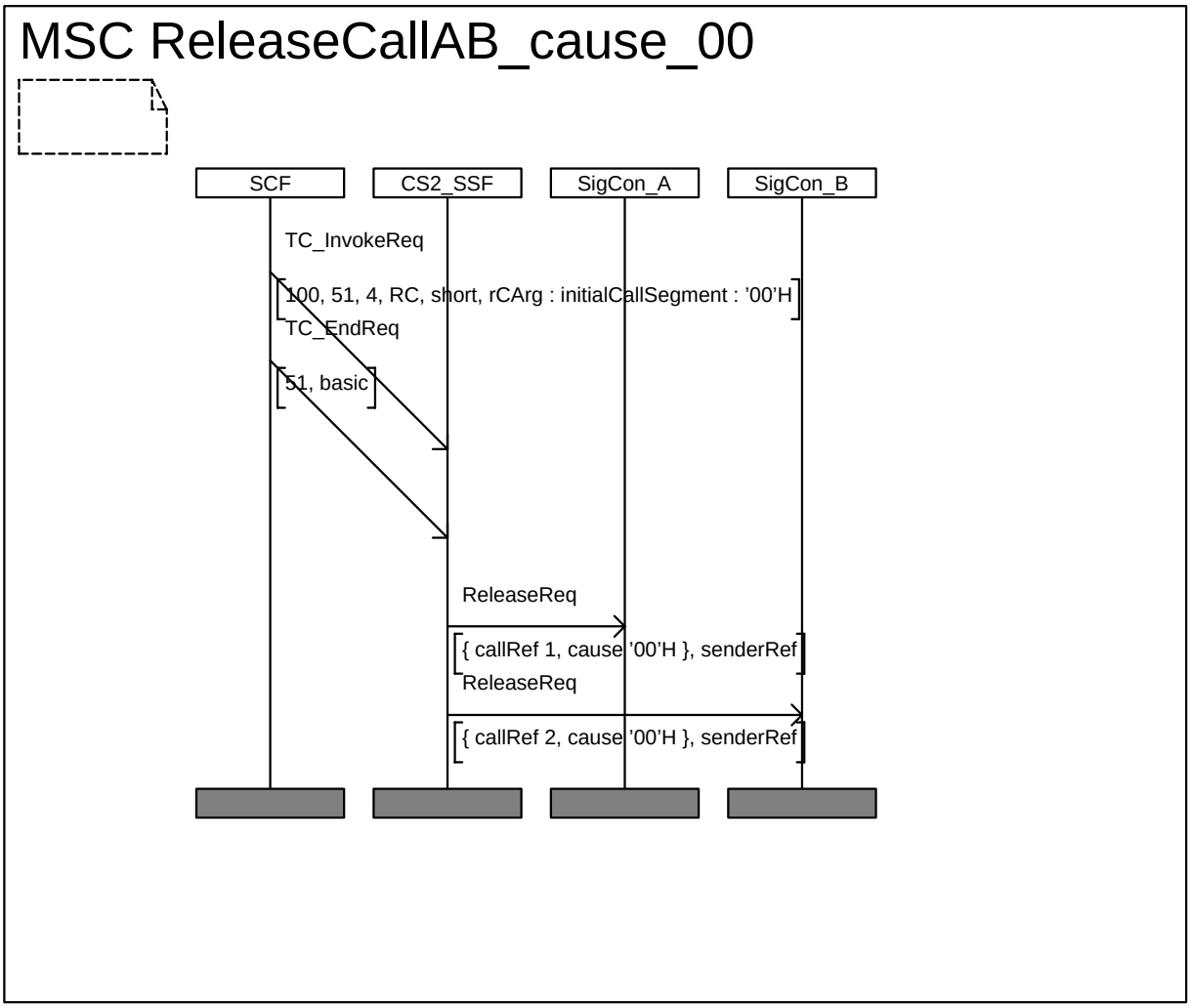
6.1.3.3 ReleaseCallA postamble



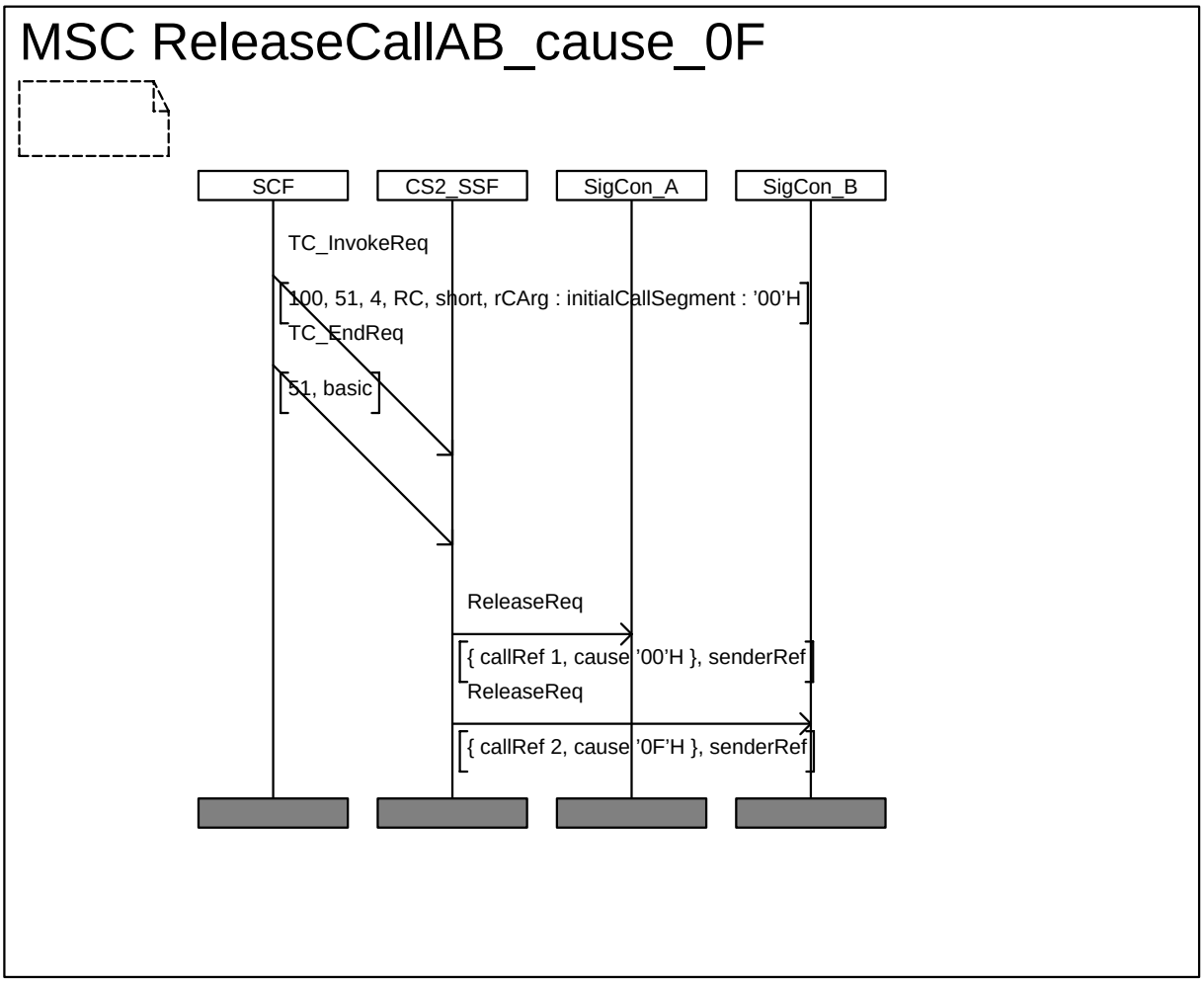
6.1.3.4 ReleaseICA postamble



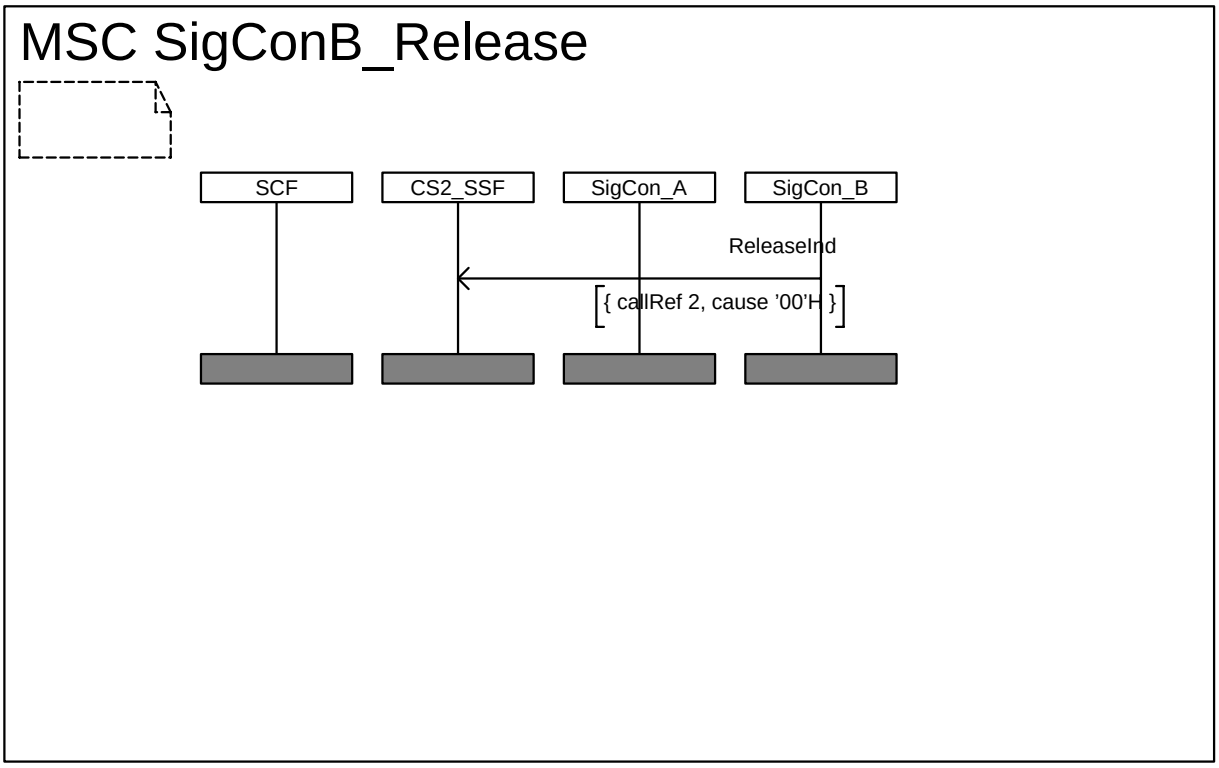
6.1.3.5 ReleaseCallAB_cause_00 postamble



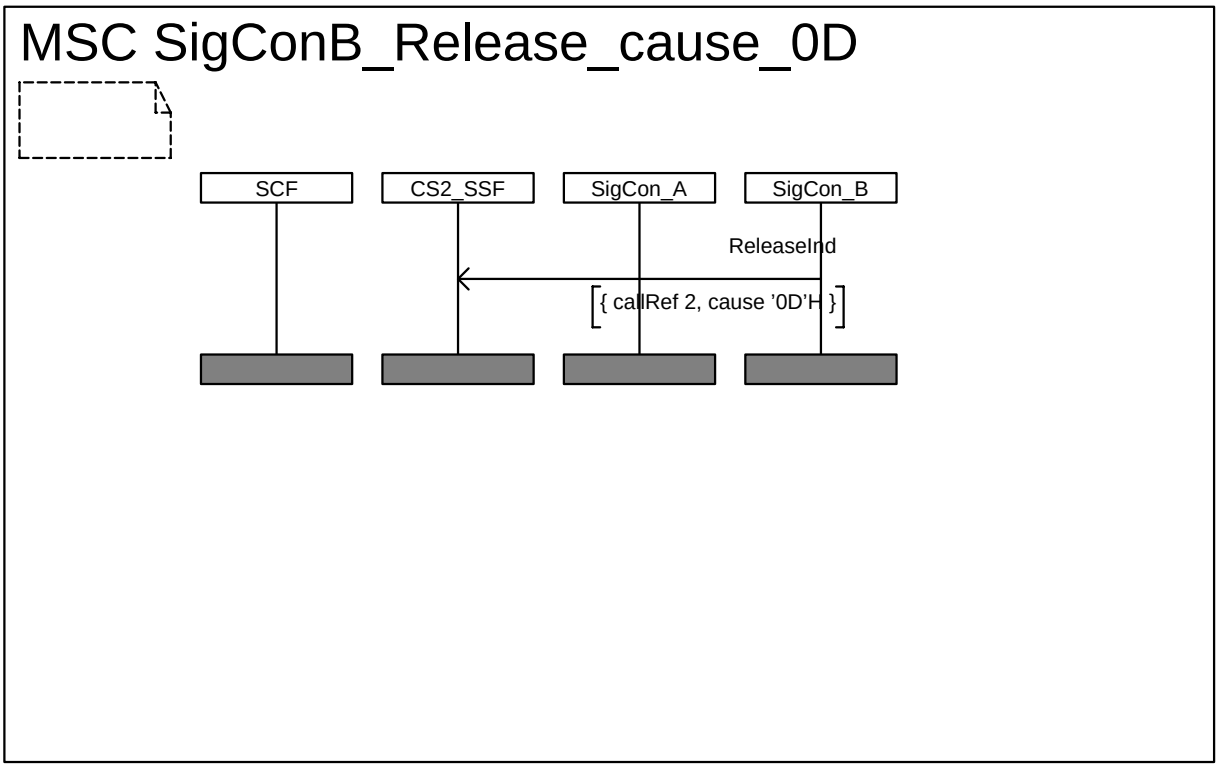
6.1.3.6 ReleaseCallAB_cause_0F postamble



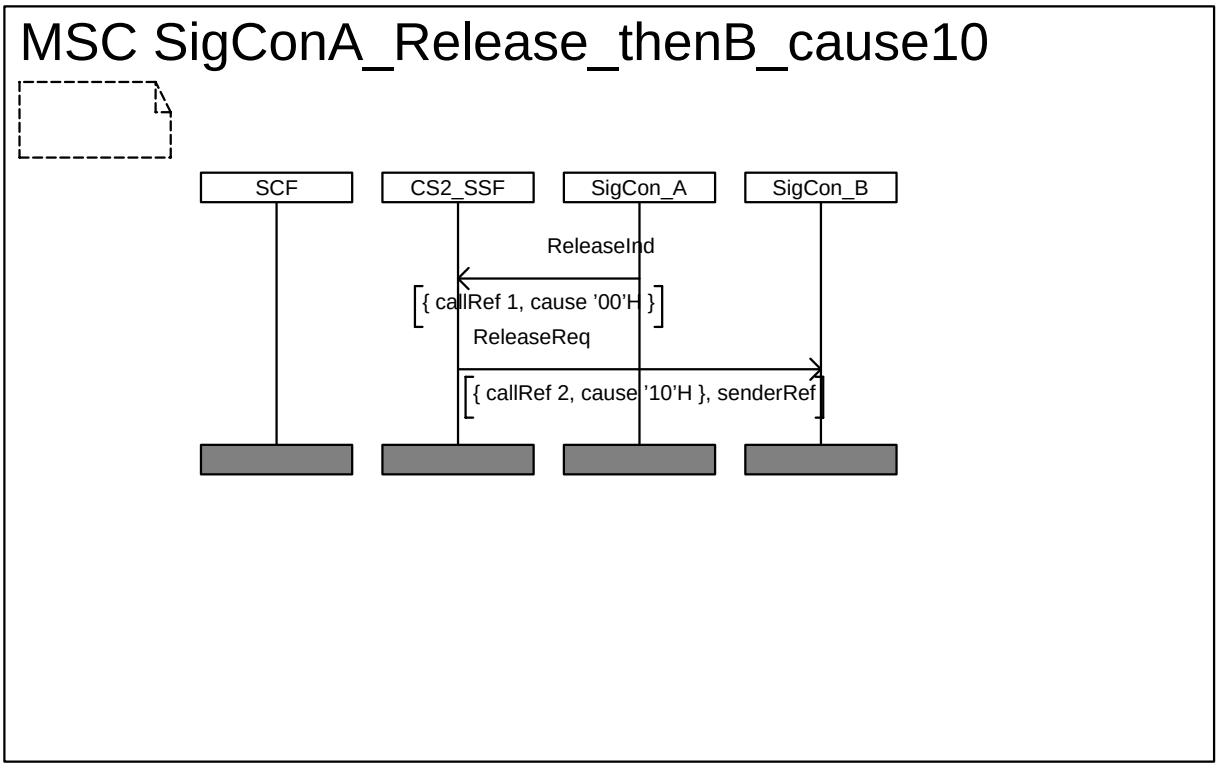
6.1.3.7 SigConB_Release postamble



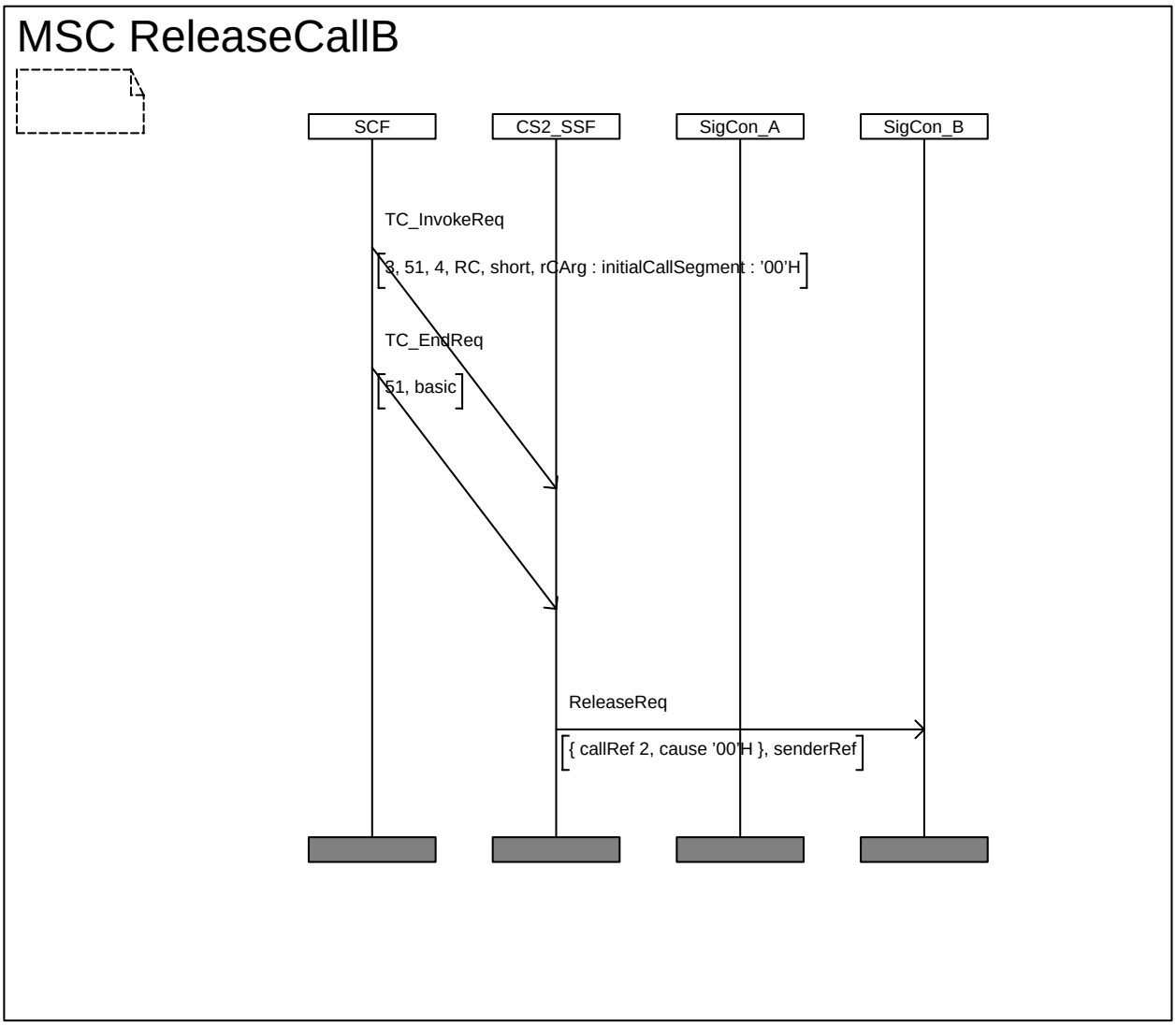
6.1.3.8 SigConB_Release_cause_0D postamble



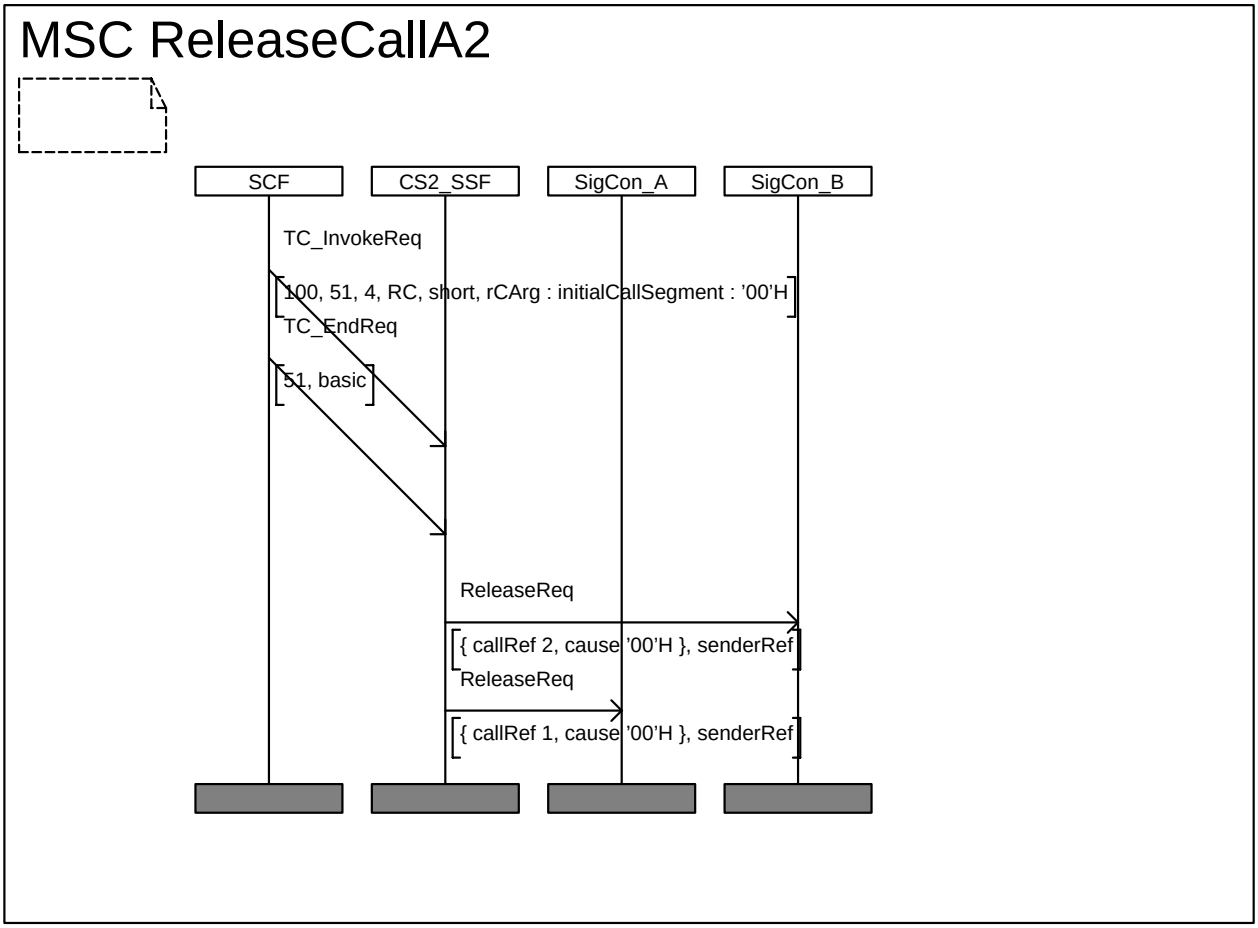
6.1.3.9 SigConA_Release_thenB_cause10 postamble



6.1.3.10 ReleaseCallB postamble



6.1.3.11 ReleaseCallA2 postamble



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

PA	PlayAnnouncement (no specific Test Purpose)
PC	PromptAndCollectUserInformation
RC	ReleaseCall
RN	RequestNotificationChargingEvent
RR	RequestReportBCSMEvent
SC	SendChargingInformation

6.2.2 Definitions of the procedures

ServiceFiltering procedure (SF)

```
Invoke:             ActivateServiceFiltering
Return Result:      ServiceFilteringResponse
                    ActivateServiceFiltering
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
                    ApplyChargingReport
```

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)
                    PromptAndCollectUserInformation(Error)
Return Error:       Cancel
```

CollectInformation procedure (CI)

```
Invoke:             CollectInformation
                    RequestReportBCSMEvent
Return Result:      EventReportBCSM
Return Error:       CollectInformation
                    RequestReportBCSMEvent
```

Connect procedure (CO)

```
Invoke:             Connect
Return Result:      None
Return Error:       Connect
```

ConnectToResource procedure (CR)

```
Invoke:             ConnectToResource
Return Result:      None
Return Error:       ConnectToResource
```

Continue (CU)

```
Invoke:             Continue
Return Result:      None
Return Error:       None
```

DisconnectForwardConnection procedure (DF)

```
Invoke:             DisconnectForwardConnection
Return Result:      None
Return Error:       DisconnectForwardConnection
```

EstablishTemporaryConnection procedure (EC)

```
Invoke:             EstablishTemporaryConnection
Return Result:      None
Return Error:       EstablishTemporaryConnection
```

FurnishChargingInformation procedure (FC)

```
Invoke:             FurnishChargingInformation
Return Result:      None
Return Error:       FurnishChargingInformation
```

InitialDP procedure (DP)

```
Invoke:             SetupInd (Signalling Control interface)
Return Result:      InitialDP
Return Error:       InitialDP
```

InitiateCallAttempt procedure (IC)

```
Invoke:             InitialCallAttempt
Return Result:      None
Return Error:       InitiateCallAttempt
```

PlayAnnouncement procedure (PA)

```
Invoke:             PlayAnnouncement
Return Result:      SpecializedResourceReport
Return Error:       PlayAnnouncement
```

PromptAndCollectUserInformation procedure (PC)

```
Invoke:             PromptAndCollectUserInformation
Return Result:      PromptAndCollectUserInformation
Return Error:       PromptAndCollectUserInformation
```

ReleaseCall procedure (RC)

```
Invoke:             ReleaseCall
Return Result:      None
Return Error:       None
```

RequestNotificationChargingEvent procedure (RN)

```
Invoke:             RequestNotificationChargingEvent
Return Result:      EventNotificationCharging
Return Error:       RequestNotificationChargingEvent
```

RequestReportBCSMEvent procedure (RR)

```
Invoke:             RequestReportBCSMEvent
Return Result:      EventReportBCSM
Return Error:       RequestReportBCSMEvent
```

SendChargingInformation procedure (SC)

```
Invoke:             SendChargingInformation
Return Result:      None
Return Error:       SendChargingInformation
```

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

IUT	Interface	Protocol component	Procedure	Category and number
SSF	SSF-SCF	Basic subset	SF ServiceFiltering	CA 1
				BV 3
				BI 2
				BO 2
			AT ActivityTest	CA 1
				BV 2
				BI 1
				BO
			AC ApplyCharging	CA 1
				BV 4
				BI 2
				BO 1
			CG CallGap	CA 1
				BV 11
				BI 1
				BO
			CF CallInformation	CA 1
				BV 5
				BI 1
				BO 2
			CA Cancel	CA 1
				BV 3
				BI 1
				BO 1
			CI CollectInformation	CA 1
				BV
				BI 1
				BO 3
			CO Connect	CA 1
				BV 9
				BI 1
				BO 1
			CU Continue	CA 1
				BV 1
				BI
				BO
			FC Furnish Charging Information	CA 1
				BV 1
				BI
				BO
			DP InitialDP	CA 2
				BV 5
				BI 2
				BO
			IC InitiateCall Attempt	CA 1
				BV 2
				BI
				BO 2
			RC ReleaseCall	CA 1
				BV 2
				BI
				BO 1
			RR RequestReport BCSMEvent	CA 1
				BV 33
				BI 2
				BO 1
			SC SendCharging Information	CA
				BV
				BI
				BO

IUT	Interface	Protocol component	Procedure	Category and number
			RN	CA 1
			Request	BV 2
			Notification	BI 1
			ChargingEvent	BO 1

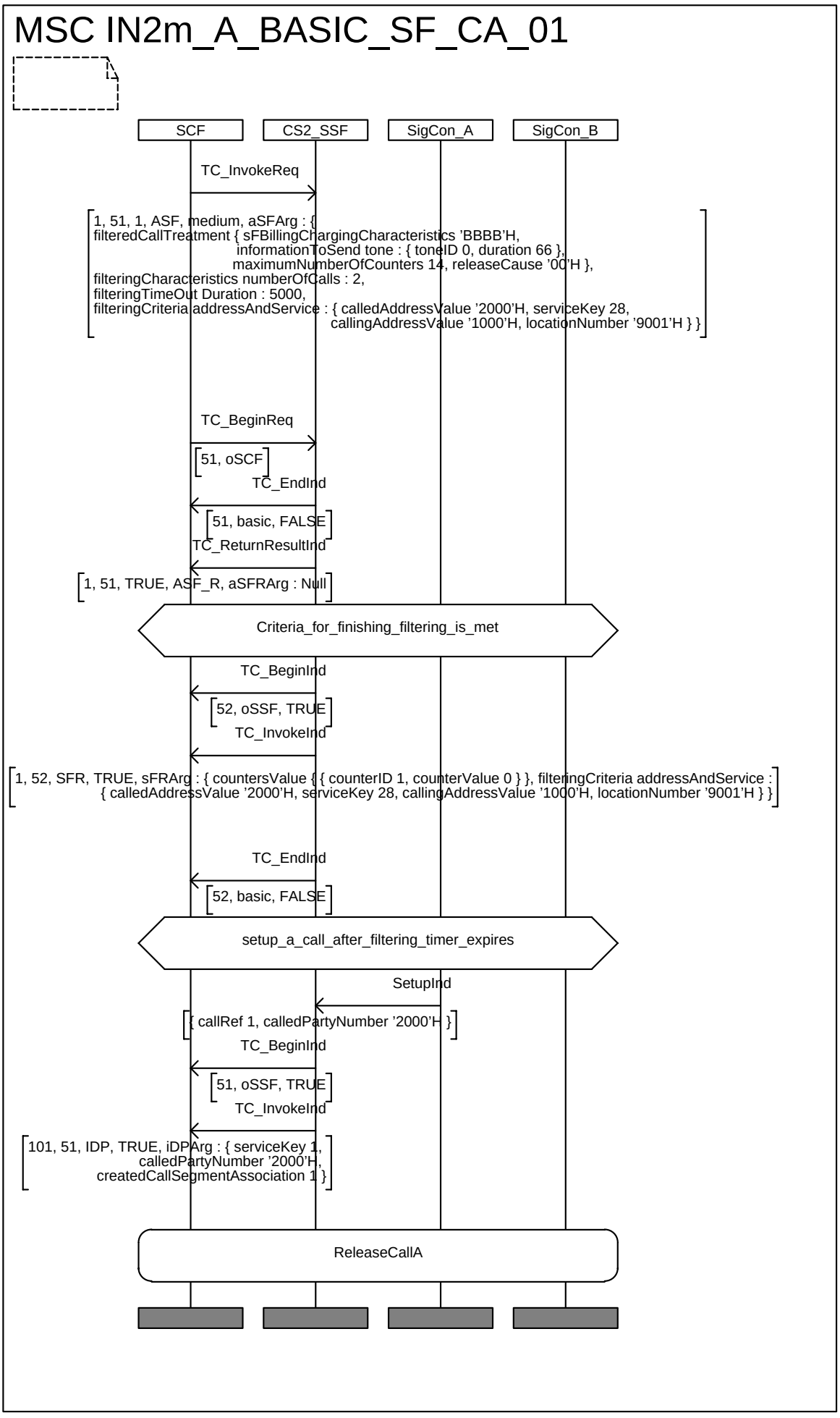
6.4 Test Purposes (TP) description

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

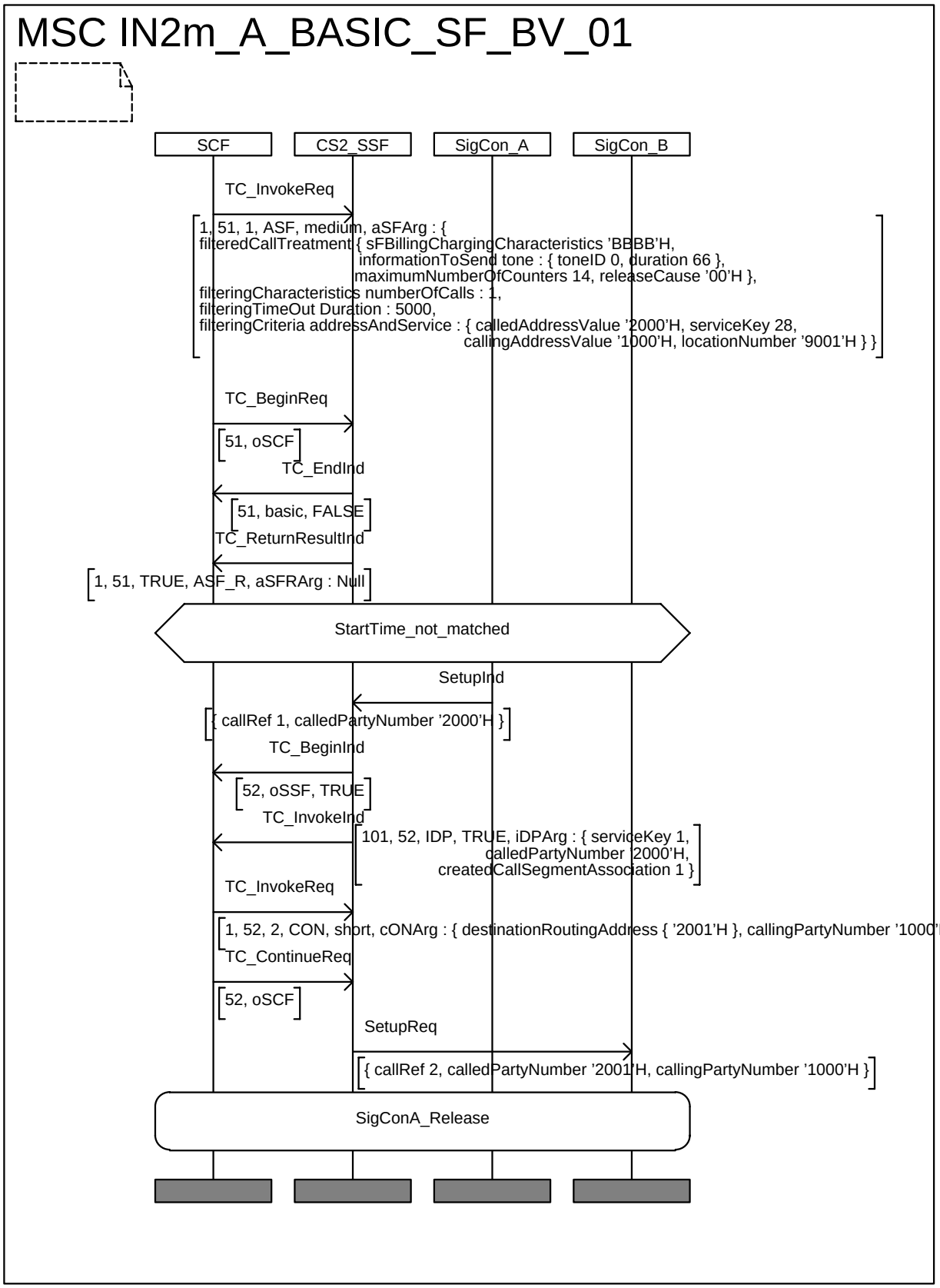
SigCon A and SigCon B are the signalling controls for users A and B, and the IUT is a SSF while the main lower tester is an SCF.

6.4.1 ServiceFiltering procedure

IN2_A_BASIC_SF_CA_01	
Purpose:	test **ServiceFiltering** procedure on duration
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF issues **ActivateServiceFiltering** invoke containing mandatory parameters only, with: - filteredCallTreatment including sFBillingChargingCharacteristics only, - filteringCharacteristics being interval, - filteringTimeOut being duration, - filteringCriteria being serviceKey, then a call is initiated after Characteristics being interval duration expires
Pass criteria	SSF accepts the call, then SSF issues **ServiceFilteringResponse** invoke with parameters - countersValue including 1 counterAndValue, - filteringCriteria being serviceKey
Postamble:	Release Call A.


MSC IN2m_A_BASIC_SF_CA_01
SCF
CS2_SSF
SigCon_A
SigCon_B
TC_InvokeReq
1, 51, 1, ASF, medium, aSFArg : {
filteredCallTreatment { sFBillingChargingCharacteristics 'BBBB'H,
informationToSend tone : { toneID 0, duration 66 },
maximumNumberOfCounters 14, releaseCause '00'H },
filteringCharacteristics numberOfCalls : 2,
filteringTimeOut Duration : 5000,
filteringCriteria addressAndService : { calledAddressValue '2000'H, serviceKey 28,
callingAddressValue '1000'H, locationNumber '9001'H } }
TC_BeginReq
51, oSCF
TC_EndInd
51, basic, FALSE
TC_ReturnResultInd
1, 51, TRUE, ASF_R, aSFRArg : Null
Criteria_for_finishing_filtering_is_met
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
1, 52, SFR, TRUE, sFRArg : { countersValue { { counterID 1, counterValue 0 } }, filteringCriteria addressAndService :
{ calledAddressValue '2000'H, serviceKey 28, callingAddressValue '1000'H, locationNumber '9001'H } }
TC_EndInd
52, basic, FALSE
setup_a_call_after_filtering_timer_expires
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 }
ReleaseCallA

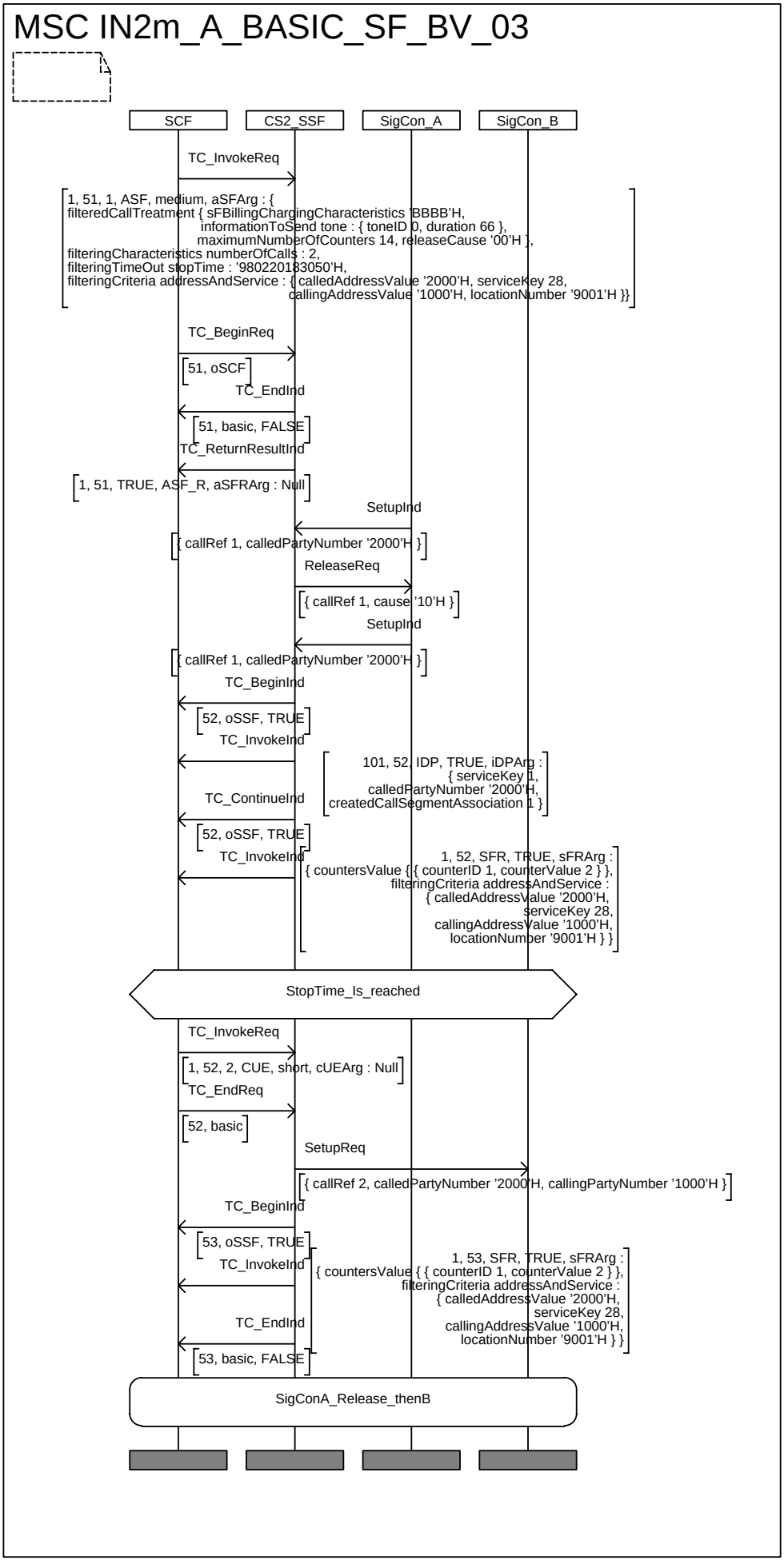
IN2_A_BASIC_SF_BV_01	
Purpose:	test **ServiceFiltering** procedure on miscellaneous parameters
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF issues **ActivateServiceFiltering** invoke containing mandatory and optional parameters, with: - filteredCallTreatment including: - sFBillingChargingCharacteristics, - 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
SCF
CS2_SSF
SigCon_A
SigCon_B
TC_InvokeReq
1, 51, 1, ASF, medium, aSFArg : {
filteredCallTreatment { sFBillingChargingCharacteristics 'BBBB'H,
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
51, oSCF
TC_EndInd
51, basic, FALSE
TC_ReturnResultInd
1, 51, TRUE, ASF_R, aSFRArg : Null
StartTime_not_matched
SetupInd
{ callRef 1, calledPartyNumber '2000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1,
calledPartyNumber '2000'H,
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

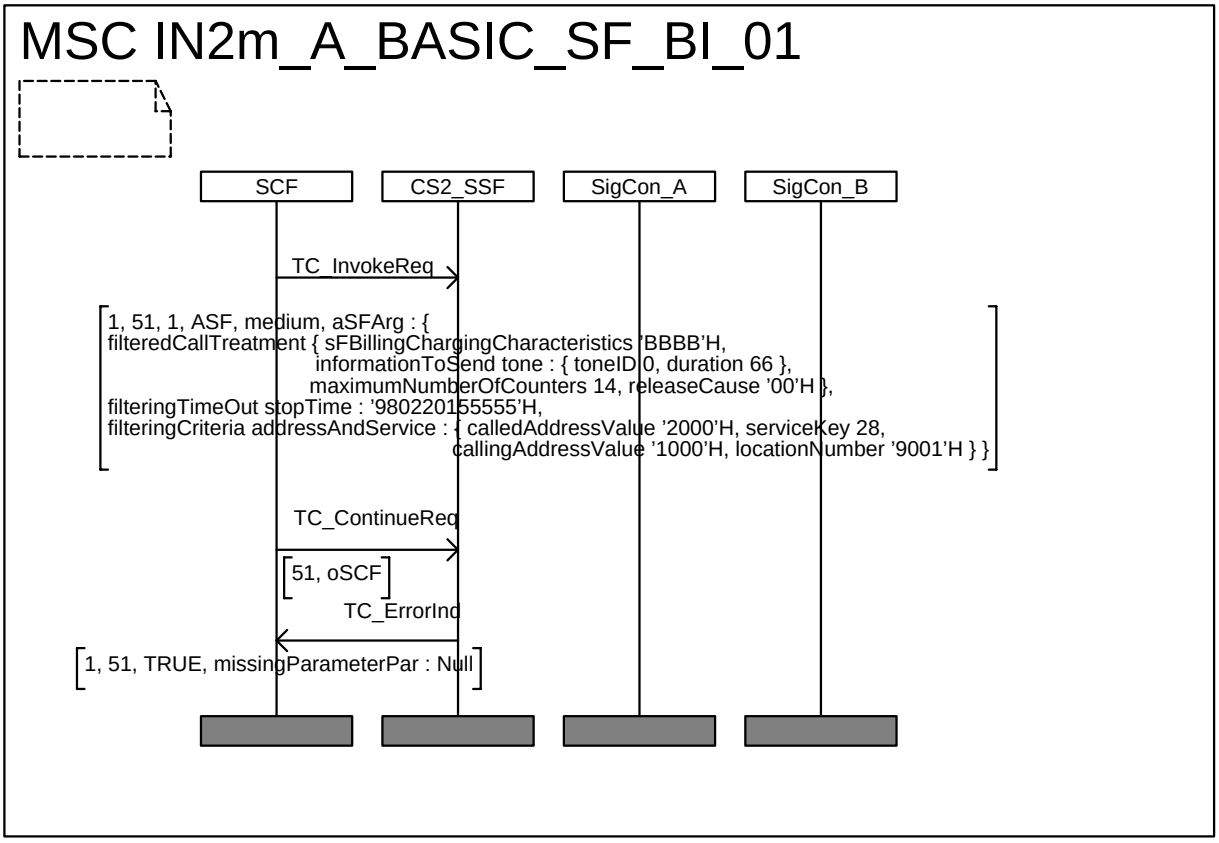
IN2_A_BASIC_SF_BV_02

This Test Purpose is not included.

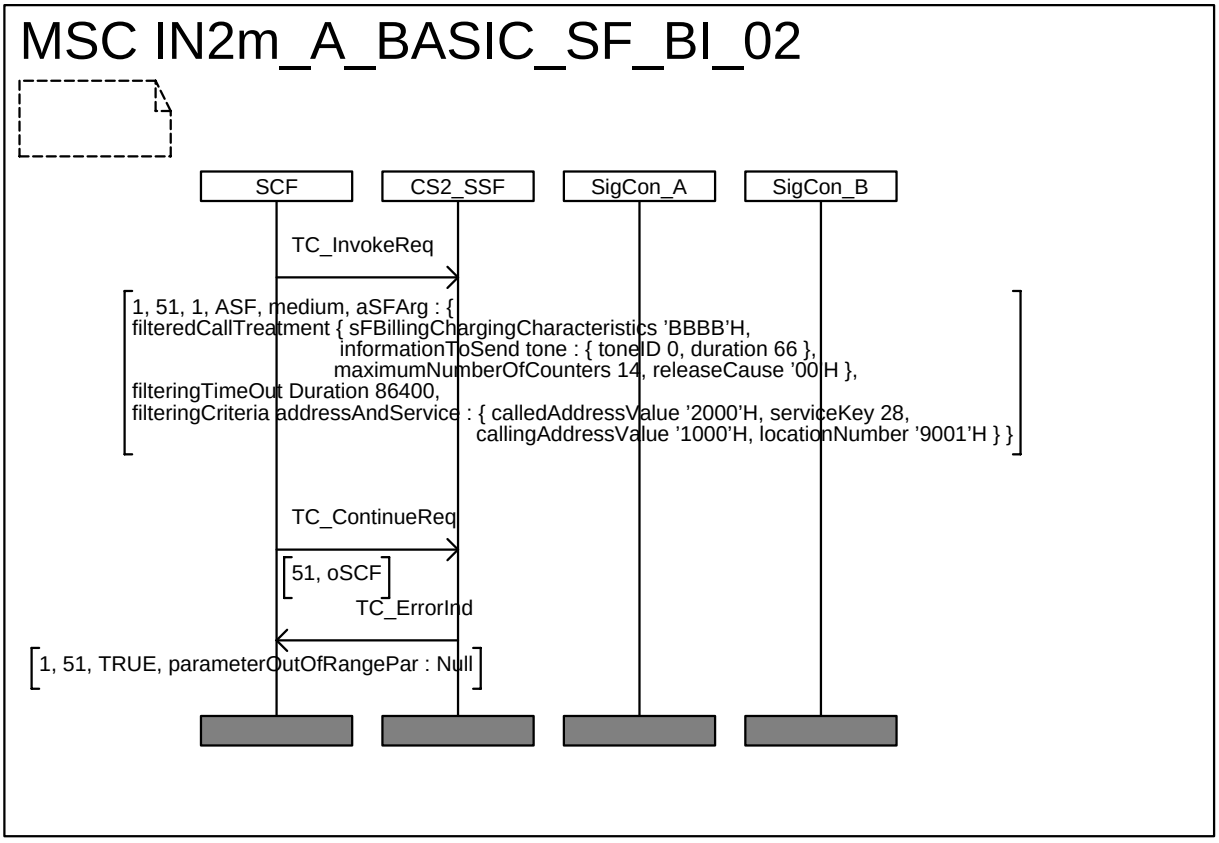
IN2_A_BASIC_SF_BV_03	
Purpose:	test **ServiceFiltering** procedure on miscellaneous parameters
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF issues **ActivateServiceFiltering** invoke containing mandatory and optional parameters, with: - filteredCallTreatment including: - sFBillingChargingCharacteristics, - informationToSend, - maximumNumberOfCounters, - filteringCharacteristics being numberOfCalls, - filteringTimeOut being stopTime, - filteringCriteria being addressAndService including: - calledAddressValue, - serviceKey, - callingAddressValue, - locationNumber, - startTime
Pass criteria	1- SSF filters the first call 2- When Characteristics being numberOfCalls is reached, SSF passes a call to SCF then SSF issues **ServiceFilteringResponse** invoke with parameters: countersValue including counterValue, filteringCriteria being addressAndService 3- When stopTime is reached, SSF issues a final **ServiceFilteringResponse** with counterID remaining to previous value (no new call)
Postamble:	SigConA_Release_then_B .


MSC IN2m_A_BASIC_SF_BV_03
SCF
CS2_SSF
SigCon_A
SigCon_B
TC_InvokeReq
1, 51, 1, ASF, medium, aSFArg : {
filteredCallTreatment { sFBillingChargingCharacteristics 'BBBB'H,
informationToSend tone : { toneID 0, duration 66 },
maximumNumberOfCounters 14, releaseCause '00'H },
filteringCharacteristics numberOfCalls : 2,
filteringTimeOut stopTime : '980220183050'H,
filteringCriteria addressAndService : { calledAddressValue '2000'H, serviceKey 28,
callingAddressValue '1000'H, locationNumber '9001'H }}
TC_BeginReq
51, oSCF
TC_EndInd
51, basic, FALSE
TC_ReturnResultInd
1, 51, TRUE, ASF_R, aSFRArg : Null
SetupInd
{ callRef 1, calledPartyNumber '2000'H }
ReleaseReq
{ callRef 1, cause '10'H }
SetupInd
{ callRef 1, calledPartyNumber '2000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg :
{ serviceKey 1,
calledPartyNumber '2000'H,
createdCallSegmentAssociation 1 }
TC_ContinueInd
52, oSSF, TRUE
TC_InvokeInd
1, 52, SFR, TRUE, sFRArg :
{ countersValue {{ counterID 1, counterValue 2 } },
filteringCriteria addressAndService :
{ calledAddressValue '2000'H,
serviceKey 28,
callingAddressValue '1000'H,
locationNumber '9001'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 :
{ countersValue {{ counterID 1, counterValue 2 } },
filteringCriteria addressAndService :
{ calledAddressValue '2000'H,
serviceKey 28,
callingAddressValue '1000'H,
locationNumber '9001'H } }
TC_EndInd
53, basic, FALSE
SigConA_Release_thenB

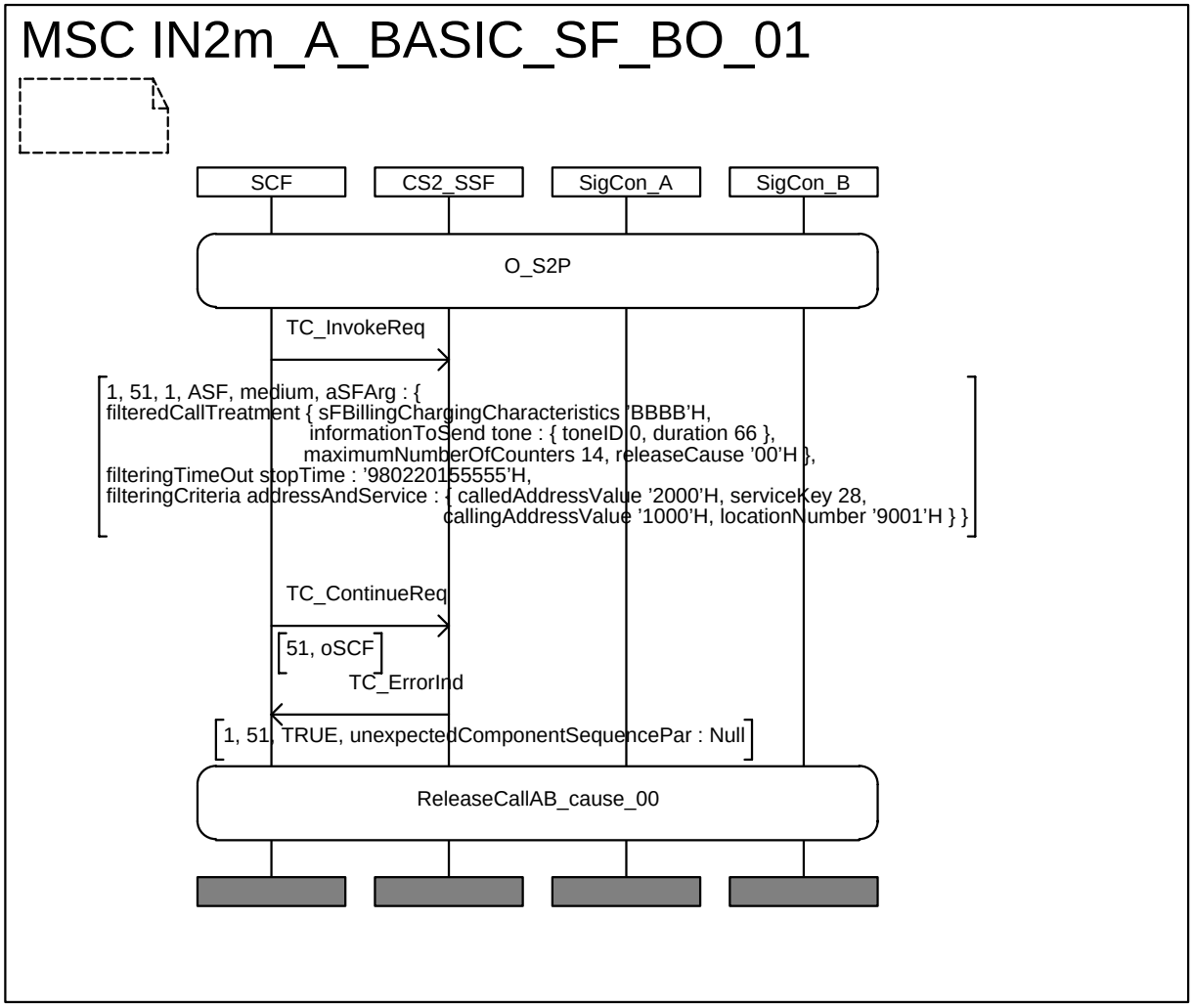
IN2_A_BASIC_SF_BI_01	
Purpose:	test **ServiceFiltering** procedure on missing parameters
Selection Cond.	
Preamble:	none
Test description	SCF issues **ActivateServiceFiltering** invoke with missing parameter - filteredCallTreatment
Pass criteria	- Check that SSF sends to SCF a **ActivateServiceFiltering** error with the indication of **missing parameter**
Postamble:	None.



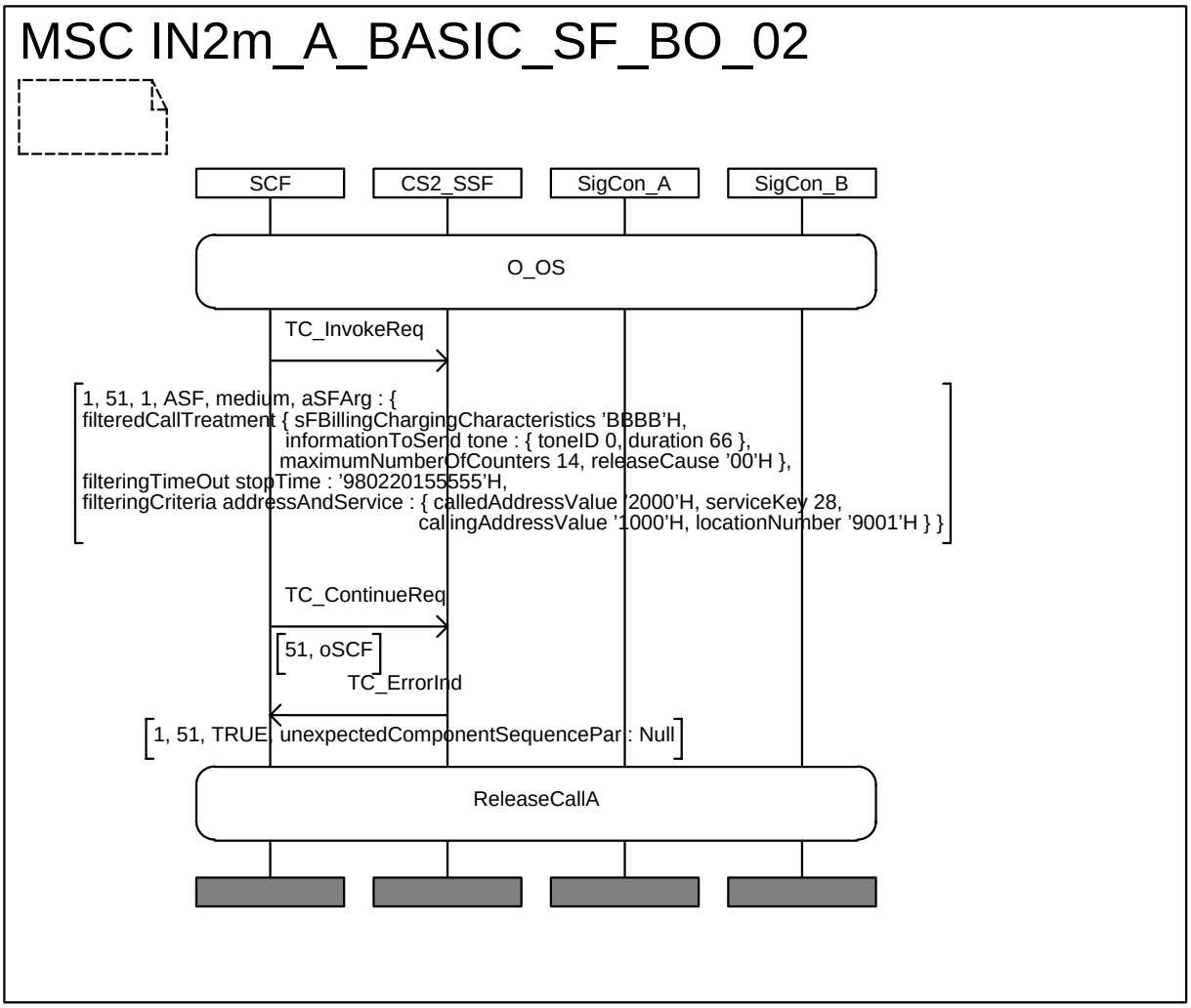
IN2_A_BASIC_SF_BI_02	
Purpose:	test ServiceFiltering procedure with parameter out of range
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF issues ActivateServiceFiltering invoke with parameter out of range - filteringTimeOut with duration > 86400
Pass criteria	- Check that SSF sends to SCF a ActivateServiceFiltering error with the indication of out of range parameter
Postamble:	None.



XXXX	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 .

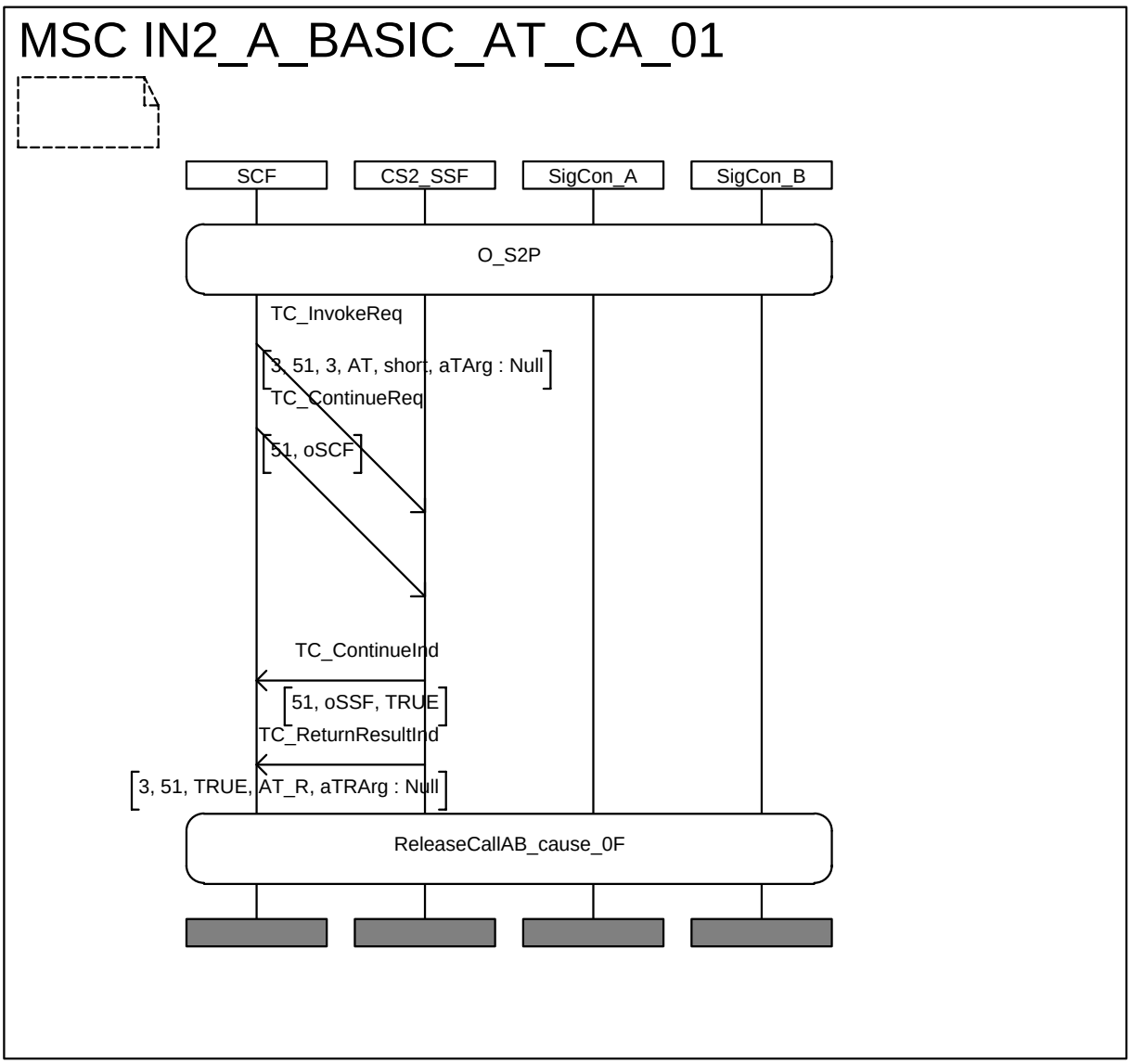


XXXX	IN2_A_BASIC_SF_BO_02
Purpose:	test **ServiceFiltering** procedure in WaitForInstruction state
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF issues **ActivateServiceFiltering** invoke containing mandatory and optional parameters, with: - filteredCallTreatment including: sFBillingChargingCharacteristics, - 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.

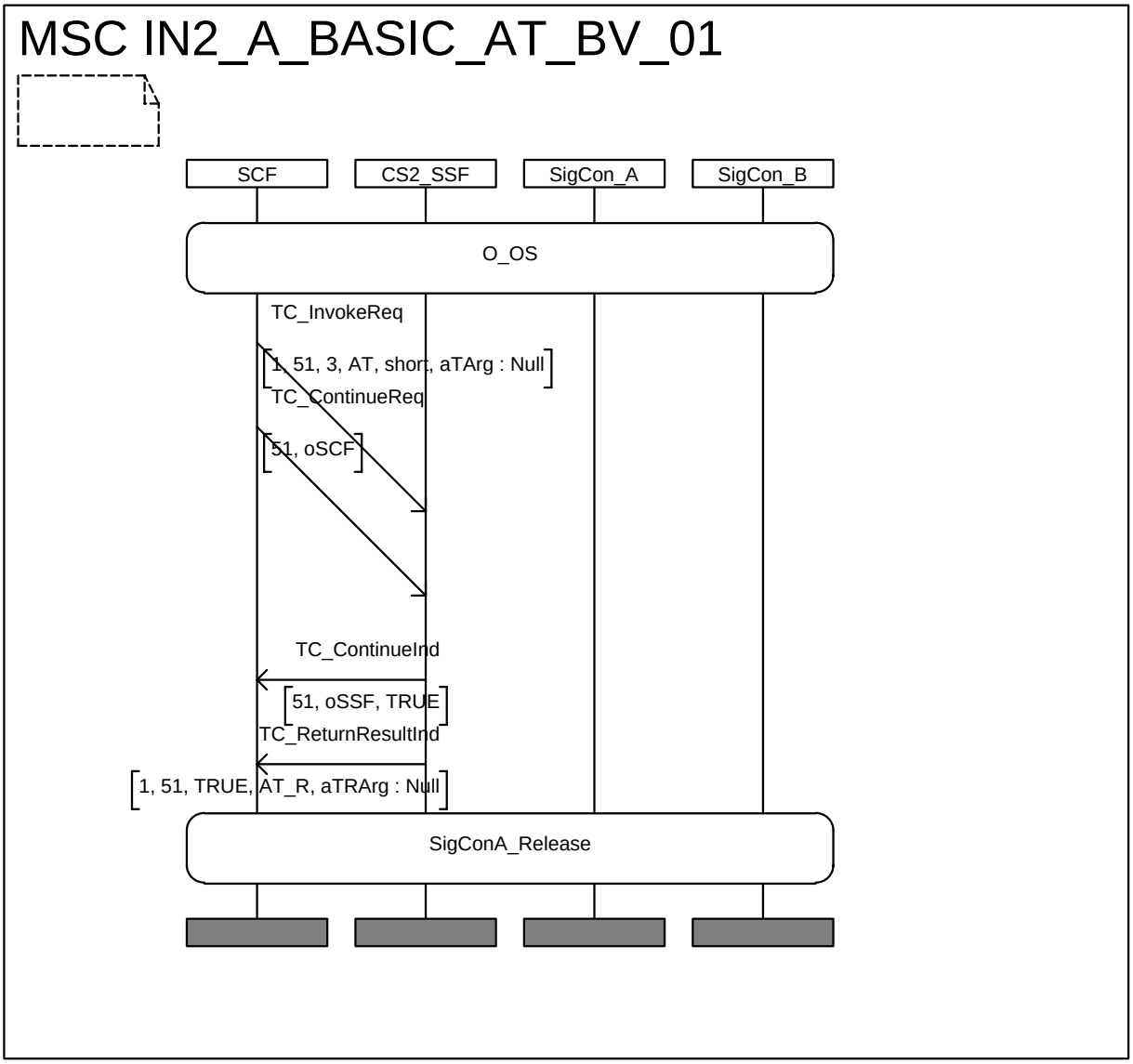


6.4.2 ActivityTest procedure

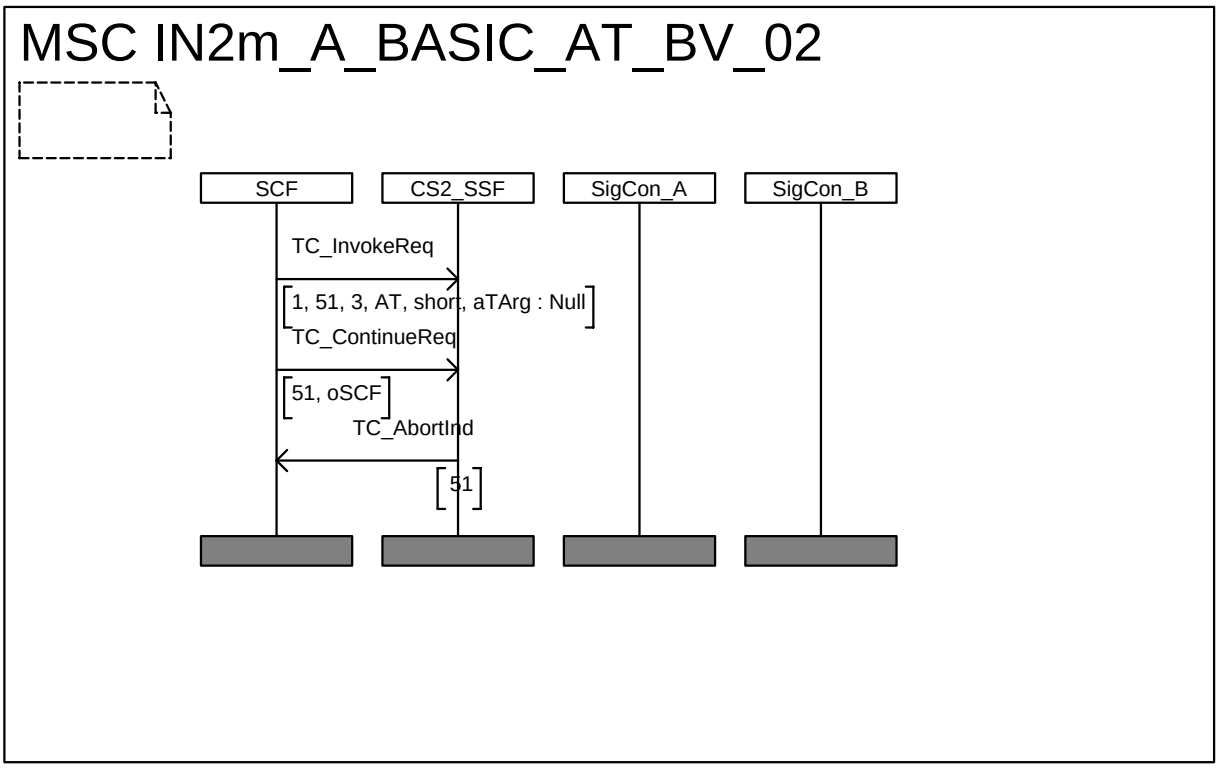
	IN2_A_BASIC_AT_CA_01
Purpose:	Test of **ActivityTest** in monitoring state
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
Test description	**ActivityTest** invoke sent by SCF to SSF, with TCAP DialogueId of dialogue identical to the one used in the preamble
Pass criteria	**ActivityTest** result sent by SSF to SCF related to the existing dialogue
Postamble:	ReleaseCallAB_cause_0F .



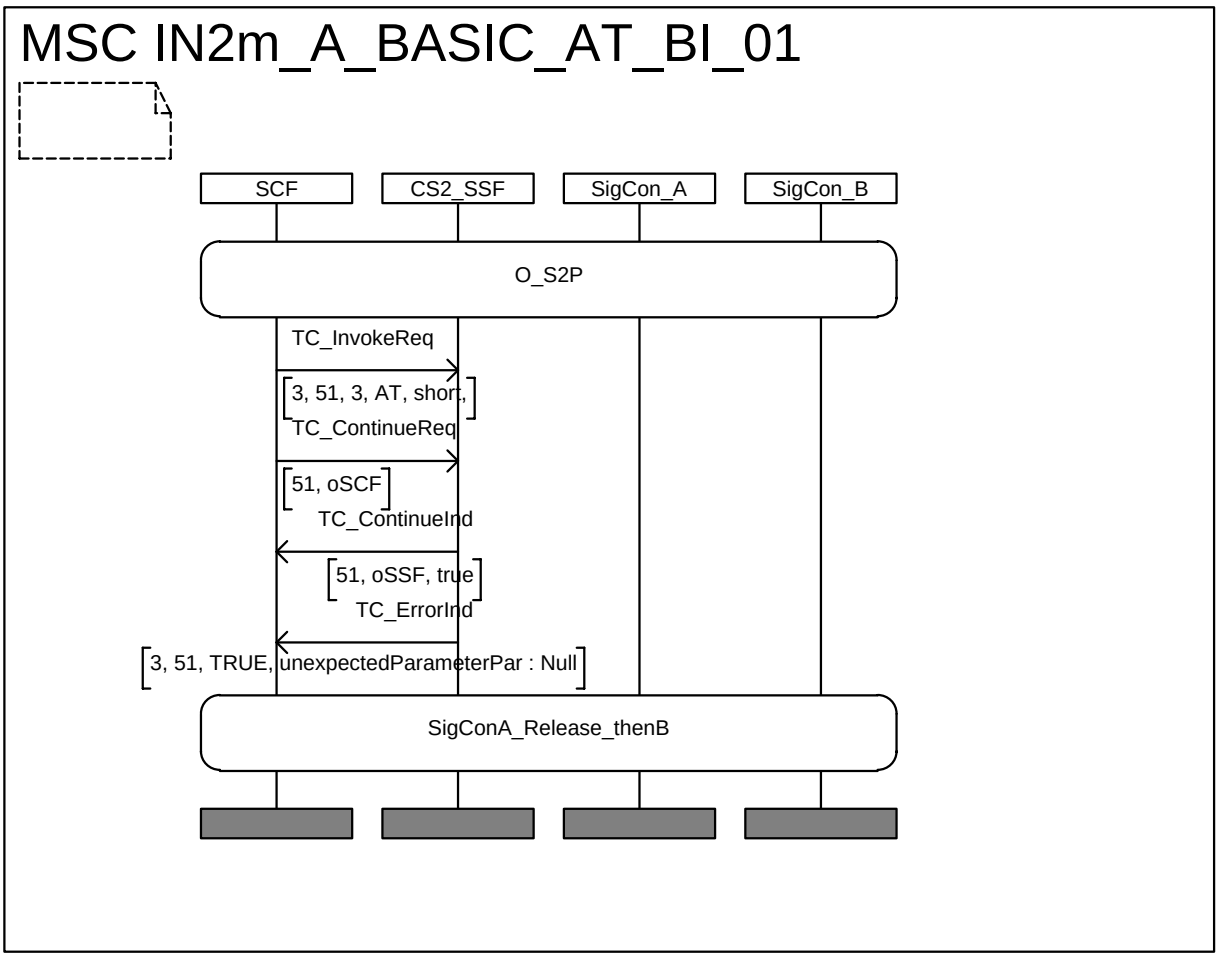
IN2_A_BASIC_AT_BV_01	
Purpose:	Test of **ActivityTest** in WaitForInstructions state
Requirement ref	
Preamble:	O_OS
Selection Cond.	
Test description	**ActivityTest** invoke sent by SCF to SSF with TCAP DialogueId of dialogue identical to the one used in the preamble
Pass criteria	**ActivityTest** result sent by SSF to SCF related to the existing dialogue
Postamble:	SigConA_Release .



IN2_A_BASIC_AT_BV_02	
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



IN2_A_BASIC_AT_BI_01	
Purpose:	Test of invalid **ActivityTest** invoke
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
Test description	**ActivityTest** invoke with argument sent by SCF to SSF during preamble
Pass criteria	SSF issues TC_error indicating unexpectedParameter
Postamble:	SigConA_release_then_B

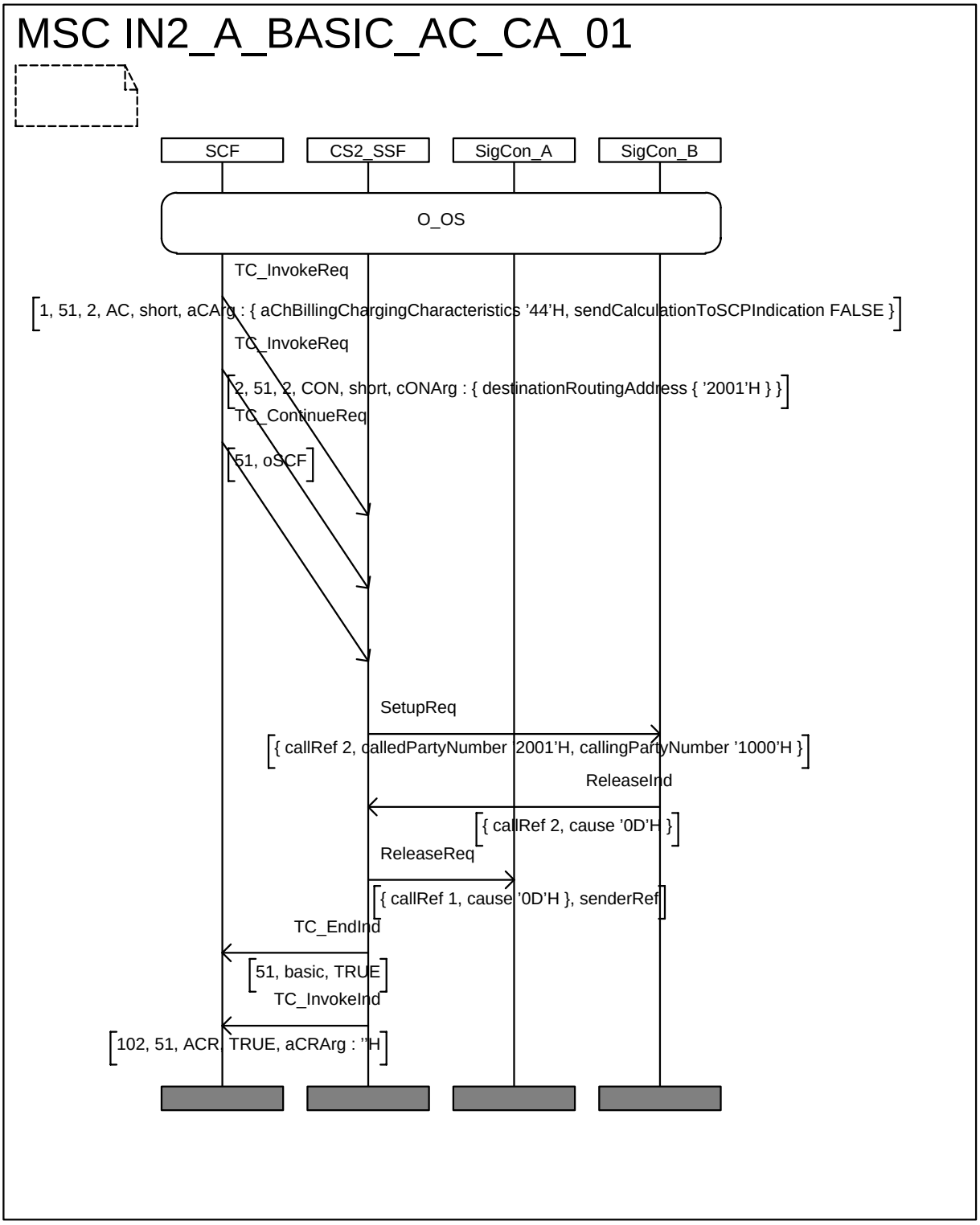


6.4.3 ApplyCharging procedure

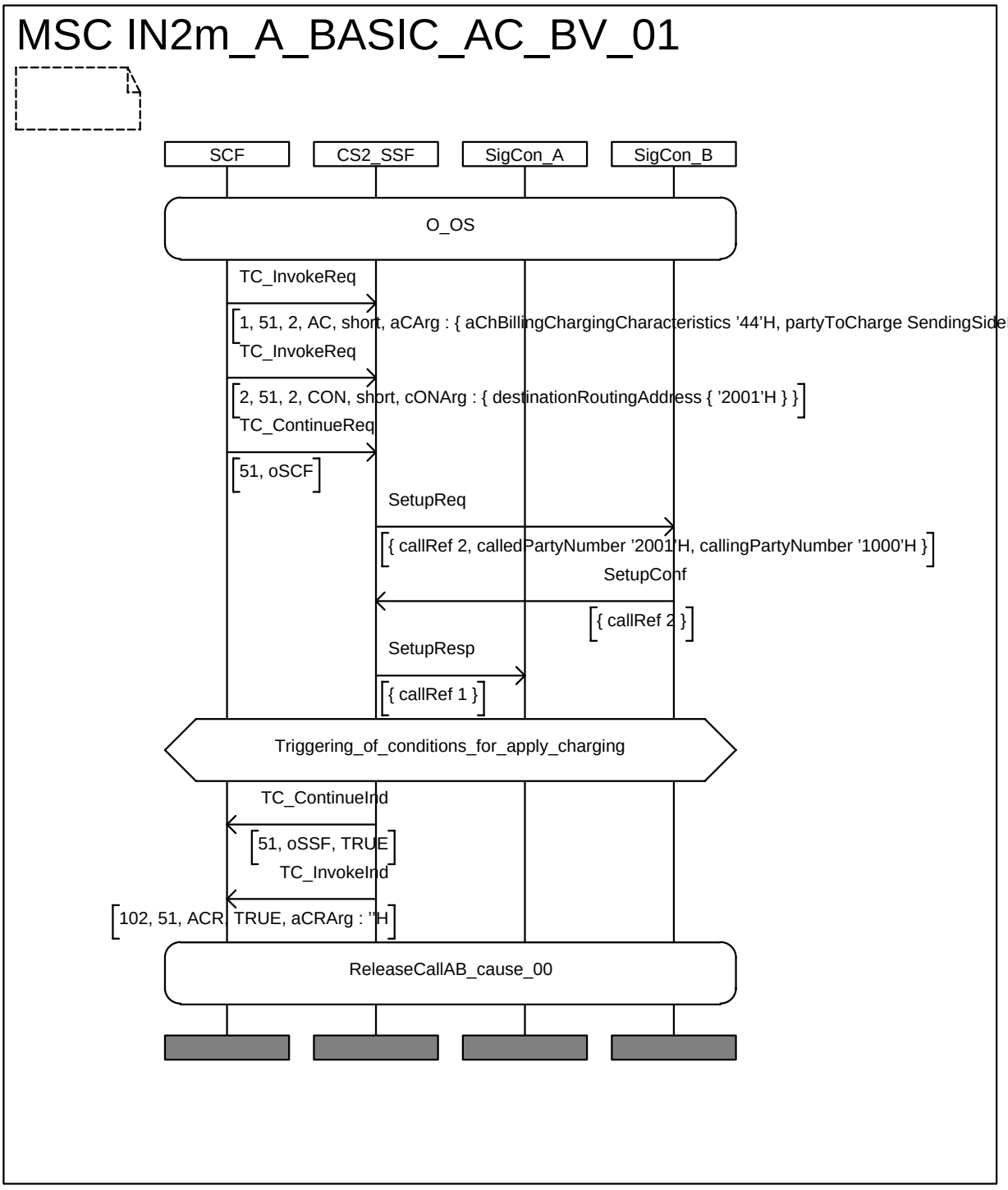
Charging related aspects in IN are network operator specific. Therefore, it is not possible to define useful test purposes for charging procedures using a network operator independent approach. TP specification has to be done by network operators, using INAP procedures themselves. The TPs could be specified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

Examples:

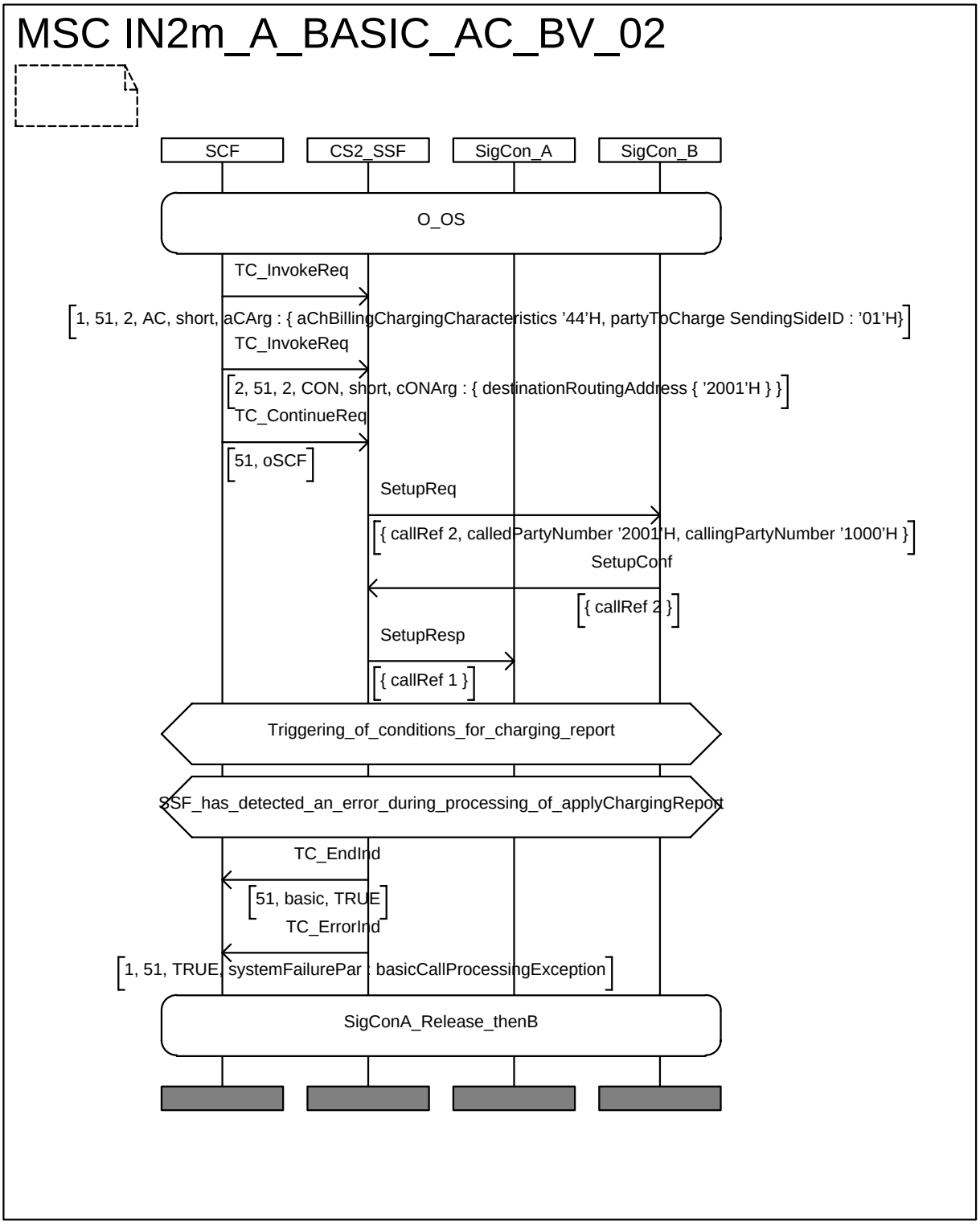
IN2_A_BASIC_AC_CA_01	
Purpose:	Test of **ApplyCharging** base procedure
Requirement ref	
Preamble:	O_OS
Selection Cond.	
Test description	**ApplyCharging** invoke sent by SCF to SSF, containing mandatory parameters only, with: - aChBillingChargingCharacteristics, followed by **Connect** invoke containing 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 B
Postamble:	none


MSC IN2_A_BASIC_AC_CA_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '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

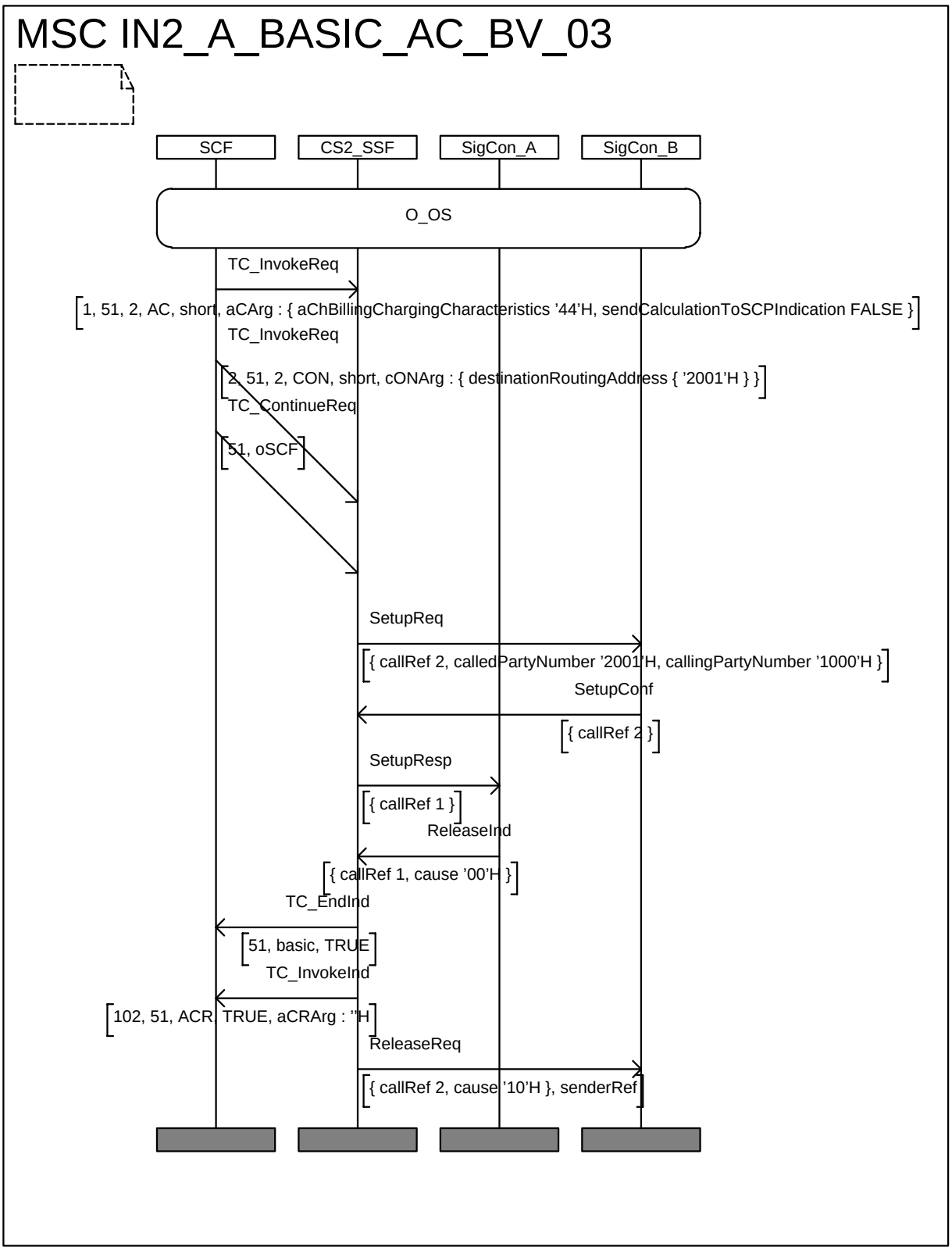
IN2_A_BASIC_AC_BV_01	
Purpose:	Test of **ApplyCharging** procedure with optional parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	**ApplyCharging** invoke sent by SCF to SSF, containing mandatory and optional parameters, with: - aChBillingChargingCharacteristics, - partyToCharge being sendingSideID, followed by **Connect** invoke containing mandatory parameters only As a consequence, SSF sends a **SetupReq** to SigCon B. SigCon B answers.
Pass criteria	upon detection of conditions for charging report, SSF sends to SCF an **ApplyChargingReport** invoke, with: - callResult
Postamble:	ReleaseCallAB_cause_00


MSC IN2m_A_BASIC_AC_BV_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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
{ callRef 2 }
SetupResp
{ callRef 1 }
Triggering_of_conditions_for_apply_charging
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, ACR, TRUE, aCRArg : ''H
ReleaseCallAB_cause_00

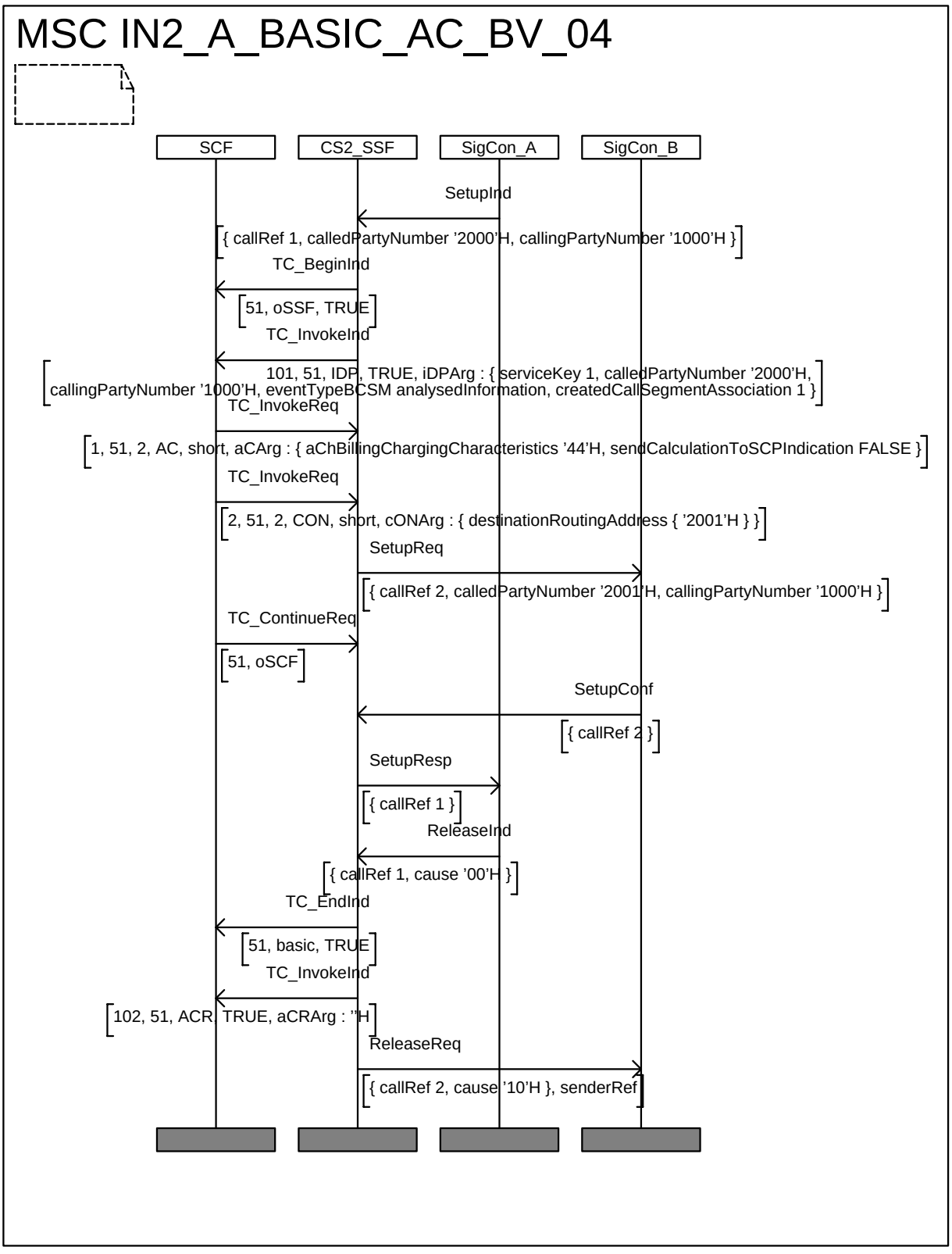
IN2_A_BASIC_AC_BV_02	
Purpose:	Test of **ApplyCharging** error procedure
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	**ApplyCharging** invoke sent by SCF to SSF, containing mandatory and optional parameters, with: - aChBillingChargingCharacteristics, - partyToCharge being sendingSideID, followed by **Connect** invoke containing mandatory parameters only As a consequence, SSF sends a **SetupReq** to SigCon B. SigConB answers upon detection of conditions for charging report, SSF detects an error.
Pass criteria	- SSF sends a TC_ErrorInd with systemFailurePar: basicCallProcessingException.
Postamble:	SigConA_Release_thenB


MSC IN2m_A_BASIC_AC_BV_02
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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
{ callRef 2 }
SetupResp
{ callRef 1 }
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, systemFailurePar : basicCallProcessingException
SigConA_Release_thenB

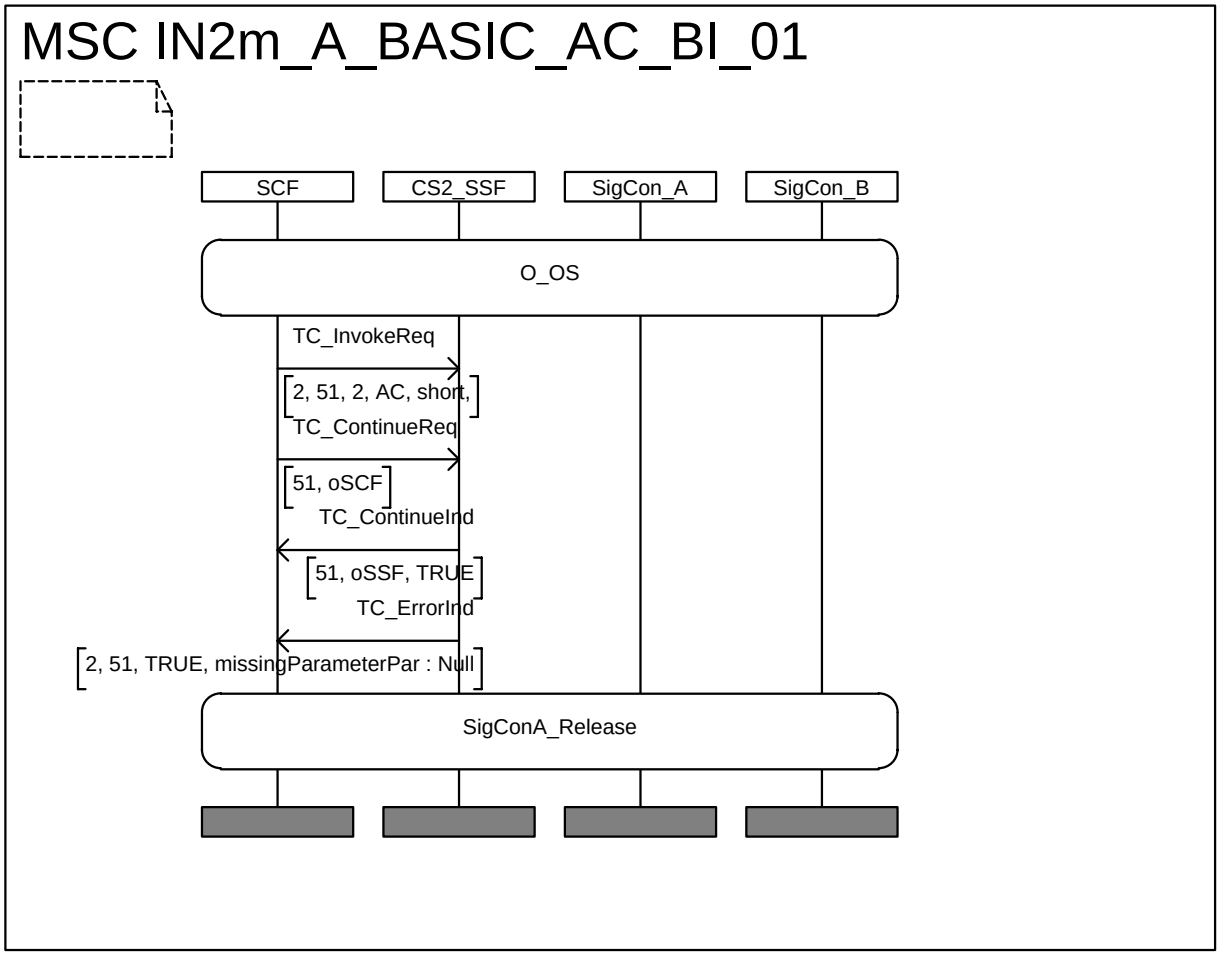
XXXX	IN2_A_BASIC_AC_BV_03
Purpose:	Test of **ApplyCharging** and Connect in same transaction
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	**ApplyCharging** invoke sent by SCF to SSF, containing mandatory parameters only, with: - aChBillingChargingCharacteristics, followed by **Connect** 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
[1, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '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 }]
SetupConf
[{ callRef 2 }]
SetupResp
[{ callRef 1 }]
ReleaseInd
[{ callRef 1, cause '00'H }]
TC_EndInd
[51, basic, TRUE]
TC_InvokeInd
[102, 51, ACR, TRUE, aCRArg : ''H]
ReleaseReq
[{ callRef 2, cause '10'H }, senderRef]

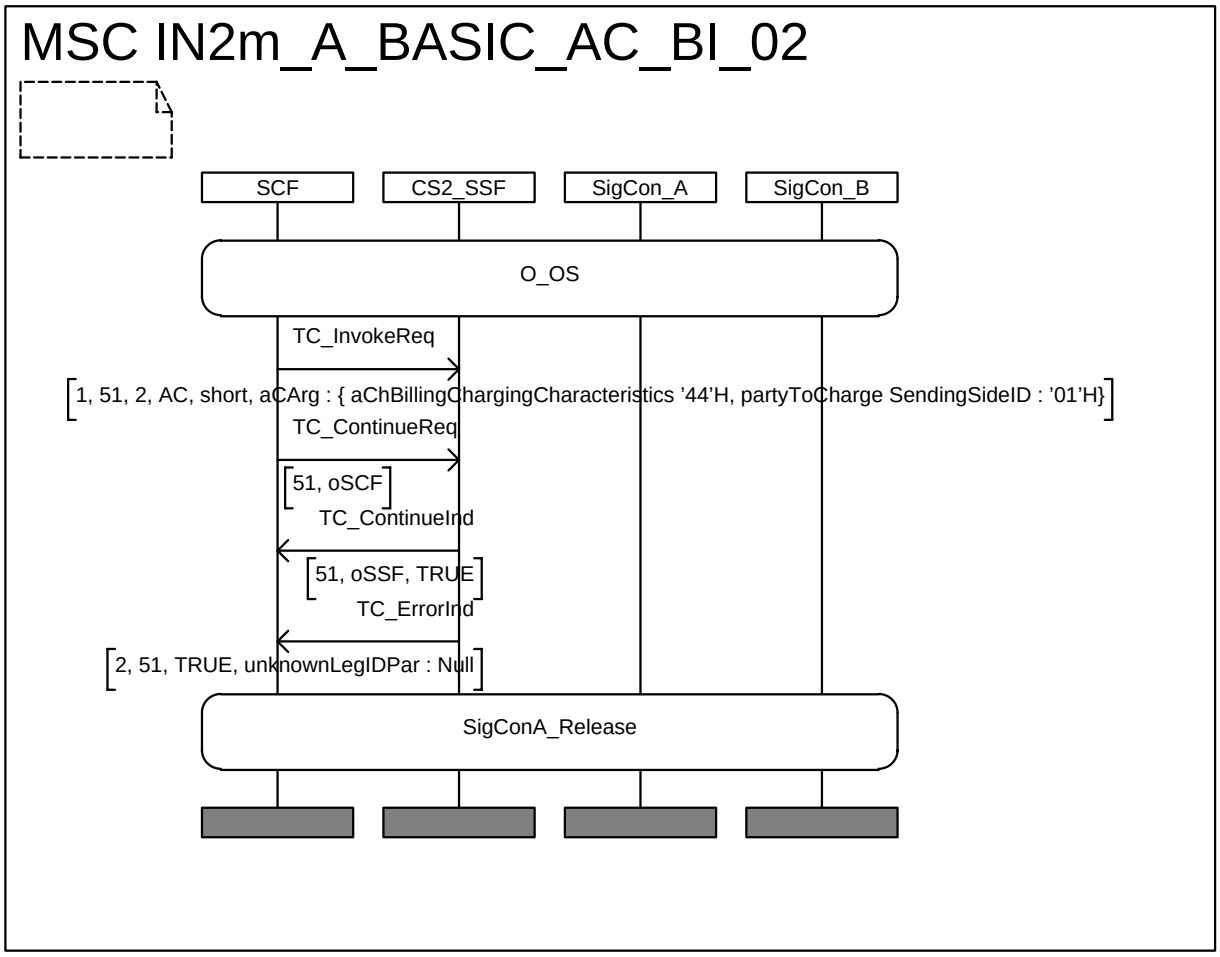
XXXX	IN2_A_BASIC_AC_BV_04
Purpose:	Test of **ApplyCharging** when Sigcon B answers
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	**ApplyCharging** invoke sent by SCF to SSF, containing mandatory parameters only, with: - aChBillingChargingCharacteristics, followed by **Connect** invoke 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
101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }
TC_InvokeReq
1, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '44'H, sendCalculationToSCPIndication FALSE }
TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
TC_ContinueReq
51, oSCF
SetupConf
{ callRef 2 }
SetupResp
{ callRef 1 }
ReleaseInd
{ callRef 1, cause '00'H }
TC_EndInd
51, basic, TRUE
TC_InvokeInd
102, 51, ACR, TRUE, aCRArg : ''H
ReleaseReq
{ callRef 2, cause '10'H }, senderRef

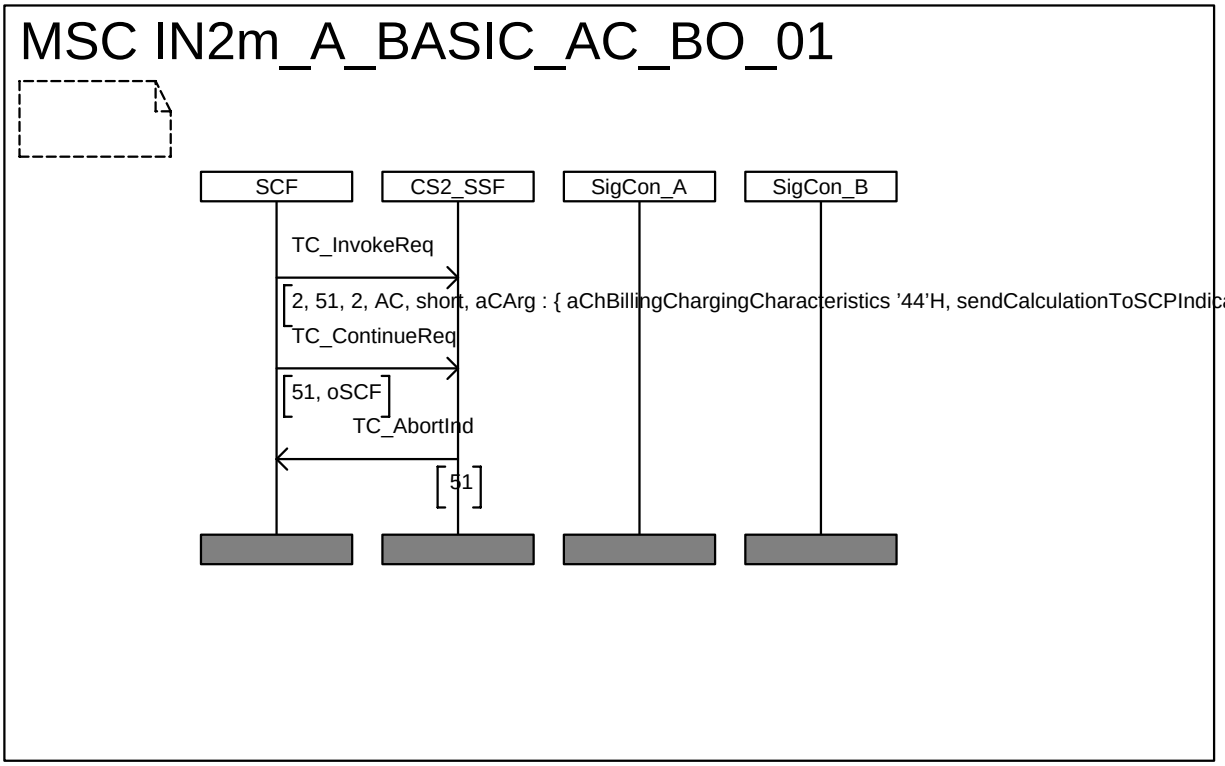
IN2_A_BASIC_AC_BI_01	
Purpose:	Test of **ApplyCharging** procedure with missing parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **ApplyCharging** invoke without the parameter - aChBillingChargingCharacteristics,
Pass criteria	SSF sends to SCF **TC_ErrorIndication** containing missingParameter error
Postamble:	SigConA_Release



IN2_A_BASIC_AC_BI_02	
Purpose:	Test of **ApplyCharging** procedure error with unknown LegID
Requirement ref	
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

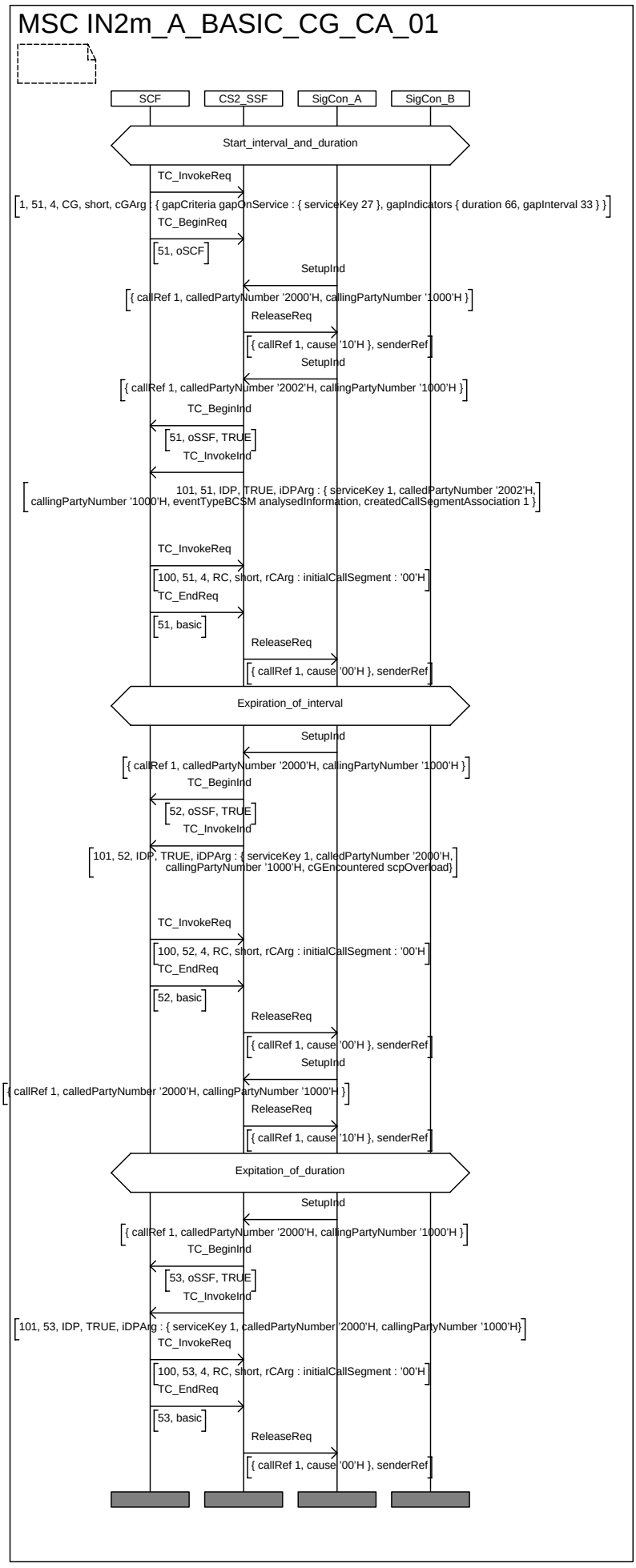


IN2_A_BASIC_AC_BO_01	
Purpose:	Test of **ApplyCharging** procedure in wrong state
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **ApplyCharging** invoke from idle state
Pass criteria	SSF sends to SCF **a TC-ABORT**
Postamble:	none

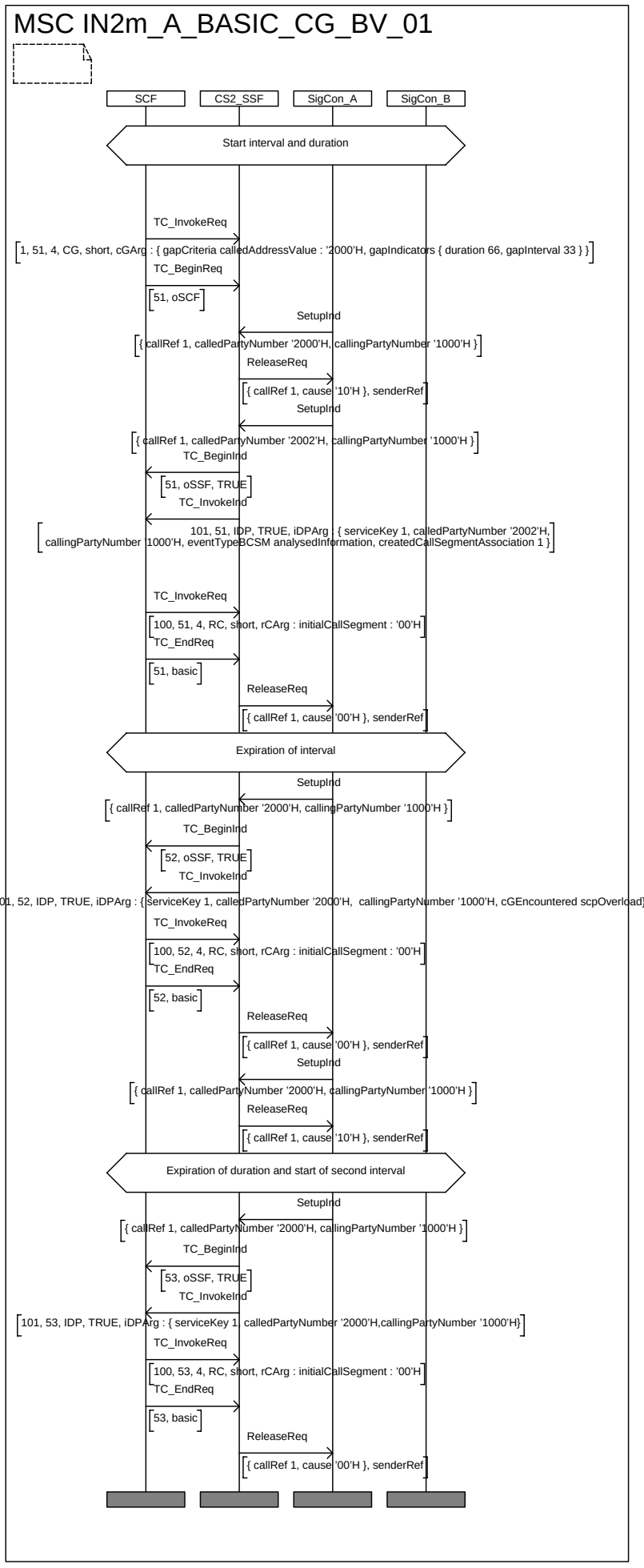


6.4.4 CallGap procedure

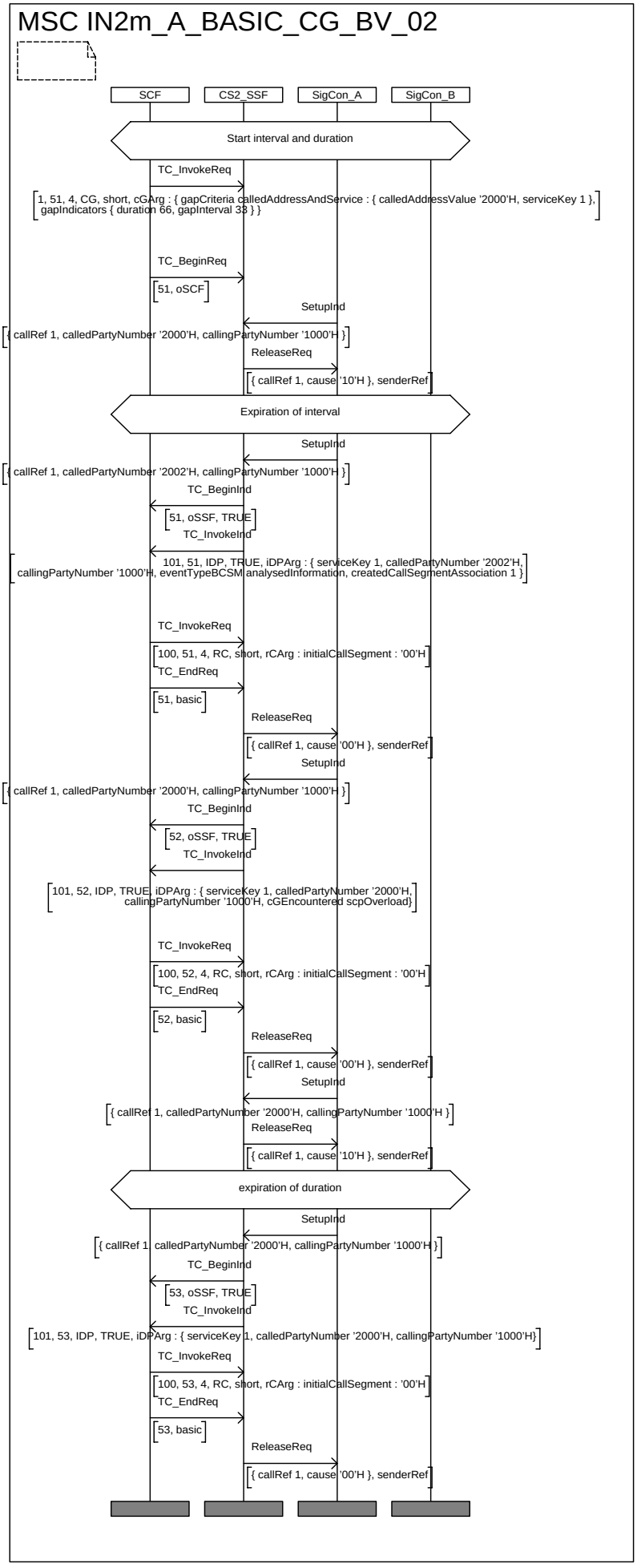
IN2_A_BASIC_CG_CA_01	
Purpose:	Test of **CallGap** base procedure and serviceKey parameter
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory parameters only, with: - gapCriteria: gapOnService with any valid value for serviceKey, - 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"
Postamble:	none


MSC IN2m_A_BASIC_CG_CA_01
SCF
CS2_SSF
SigCon_A
SigCon_B
Start_interval_and_duration
TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapCriteria gapOnService : { serviceKey 27 }, gapIndicators { duration 66, gapInterval 33 } }
TC_BeginReq
51, oSCF
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
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 analysedInformation, createdCallSegmentAssociation 1 }
TC_InvokeReq
100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
51, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
Expiration_of_interval
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload}
TC_InvokeReq
100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
52, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
Expitation_of_duration
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
53, oSSF, TRUE
TC_InvokeInd
101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}
TC_InvokeReq
100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
53, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef

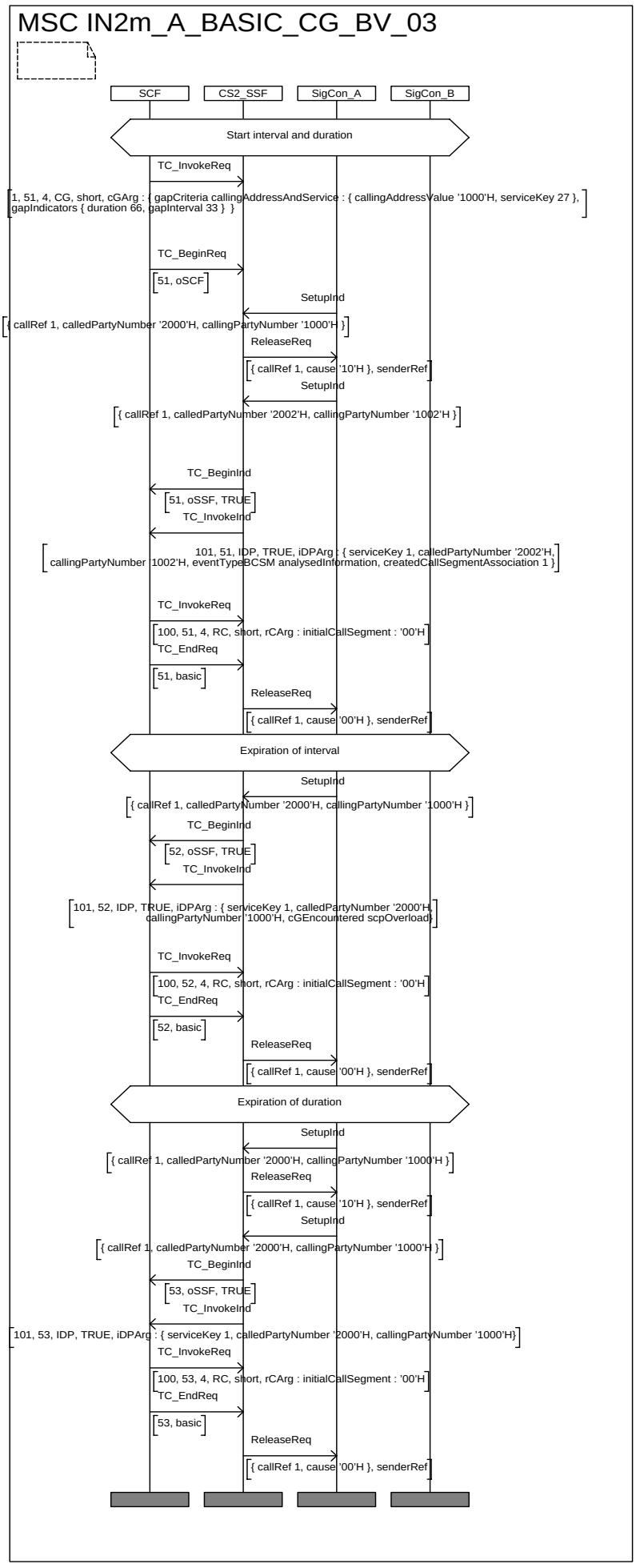
IN2_A_BASIC_CG_BV_01	
Purpose:	Test of **CallGap** procedure and calledAddressValue parameter
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory parameters only, with: - gapCriteria: calledAddressValue with any valid value, - gapIndicators duration being a duration value in seconds gapInterval being an interval value in seconds
Pass criteria	- Check that SSF releases a call when callgapping is active for the called address used in SetupInd - 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 called address 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, - and without "cGEncountered"
Postamble:	none


MSC IN2m_A_BASIC_CG_BV_01
SCF
CS2_SSF
SigCon_A
SigCon_B
Start interval and duration
TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapCriteria calledAddressValue : '2000'H, gapIndicators { duration 66, gapInterval 33 } }
TC_BeginReq
51, oSCF
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
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 analysedInformation, createdCallSegmentAssociation 1 }
TC_InvokeReq
100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
51, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
Expiration of interval
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload}
TC_InvokeReq
100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
52, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
Expiration of duration and start of second interval
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
53, oSSF, TRUE
TC_InvokeInd
101, 53, IDP, TRUE, iDPArg : { serviceKey 1 calledPartyNumber '2000'H,callingPartyNumber '1000'H}
TC_InvokeReq
100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
53, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef

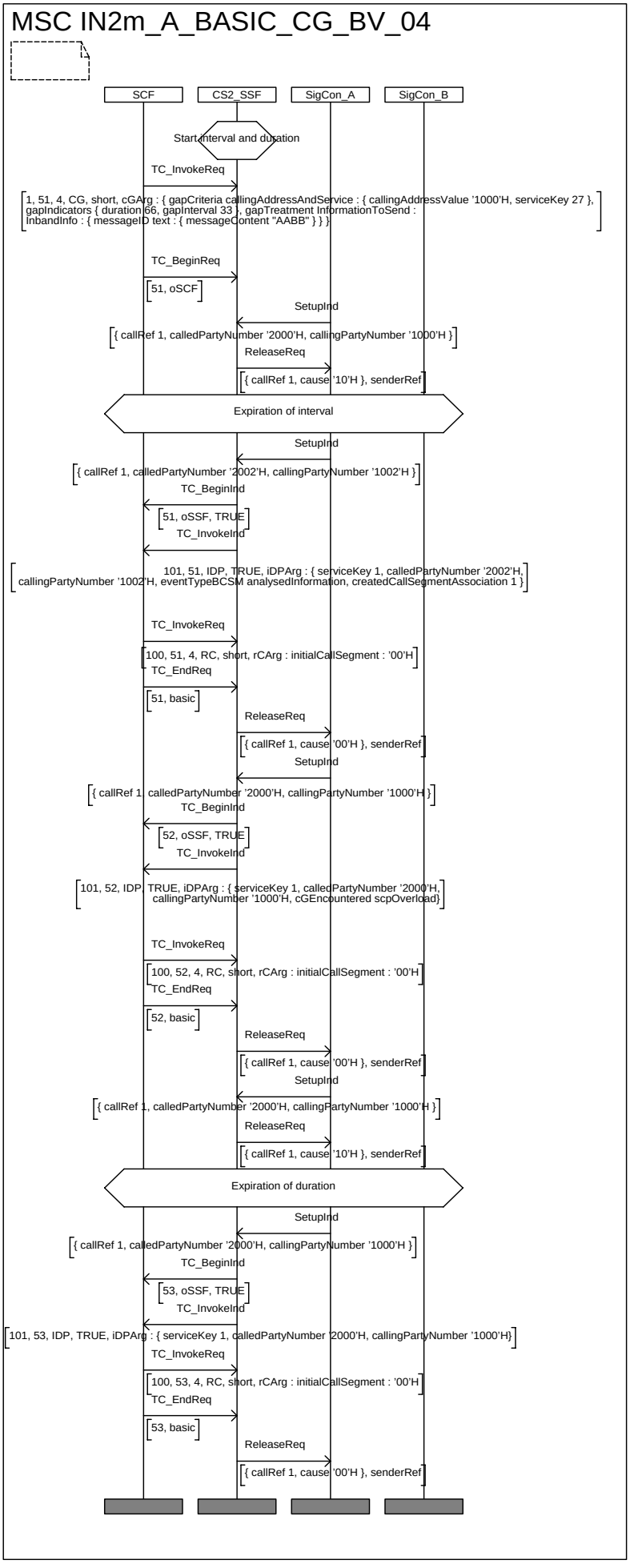
IN2_A_BASIC_CG_BV_02	
Purpose:	Test of **CallGap** procedure and calledAddressAndService parameter
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory parameters only, with: - gapCriteria: calledAddressAndService with any valid 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 called address and service key used in SetupInd - 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 called 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, - and without "cGEncountered"
Postamble:	none


MSC IN2m_A_BASIC_CG_BV_02
SCF
CS2_SSF
SigCon_A
SigCon_B
Start interval and duration
TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapCriteria calledAddressAndService : { calledAddressValue '2000'H, serviceKey 1 }, gapIndicators { duration 66, gapInterval 33 } }
TC_BeginReq
51, oSCF
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
Expiration of interval
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 analysedInformation, createdCallSegmentAssociation 1 }
TC_InvokeReq
100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
51, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload}
TC_InvokeReq
100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
52, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
expiration of duration
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
53, oSSF, TRUE
TC_InvokeInd
101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}
TC_InvokeReq
100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
53, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef

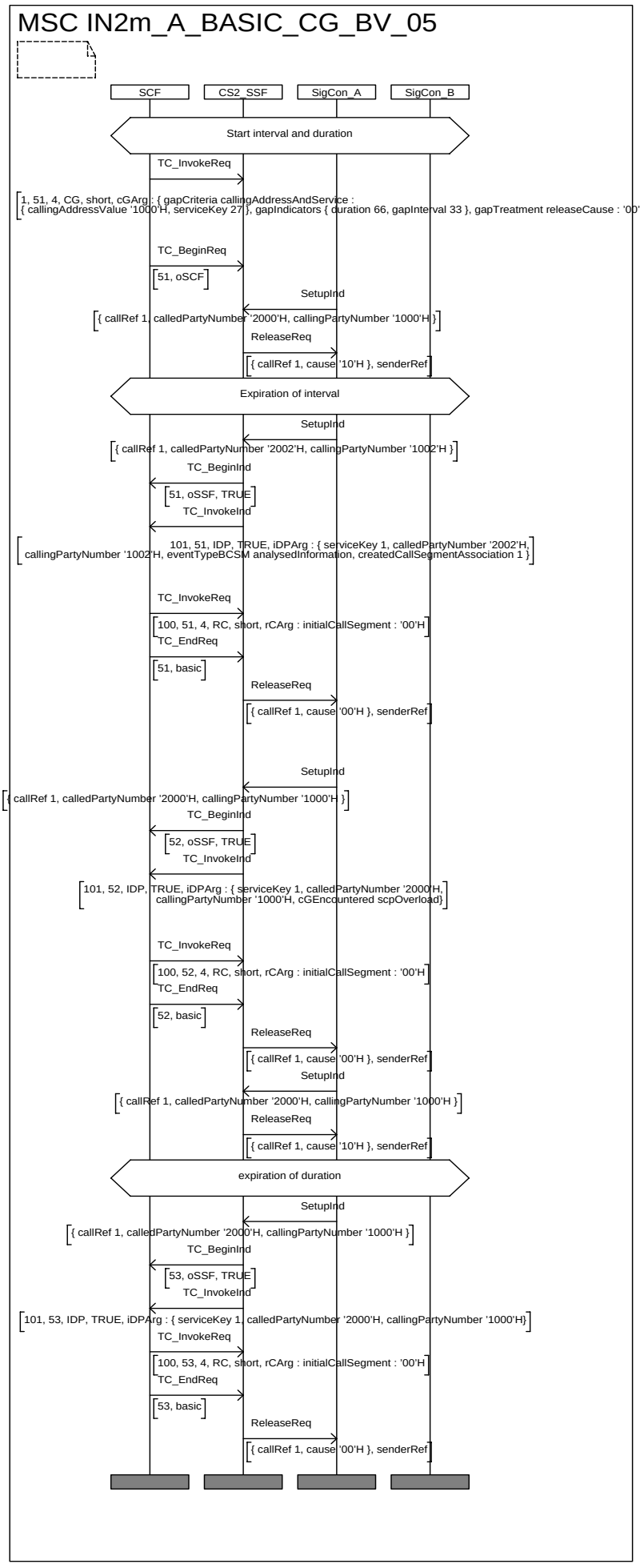
IN2_A_BASIC_CG_BV_03	
Purpose:	Test of **CallGap** procedure and optional parameter callingAddressAndService
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory parameters with: - gapCriteria: callingAddressAndService with any valid 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


MSC IN2m_A_BASIC_CG_BV_03
SCF
CS2_SSF
SigCon_A
SigCon_B
Start interval and duration
TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapCriteria callingAddressAndService : { callingAddressValue '1000'H, serviceKey 27 }, gapIndicators { duration 66, gapInterval 33 } }
TC_BeginReq
51, oSCF
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2002'H, callingPartyNumber '1002'H }
TC_BeginInd
51, oSSF, TRUE
TC_InvokeInd
101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2002'H, callingPartyNumber '1002'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }
TC_InvokeReq
100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
51, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
Expiration of interval
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload }
TC_InvokeReq
100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
52, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
Expiration of duration
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
53, oSSF, TRUE
TC_InvokeInd
101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}
TC_InvokeReq
100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
53, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef

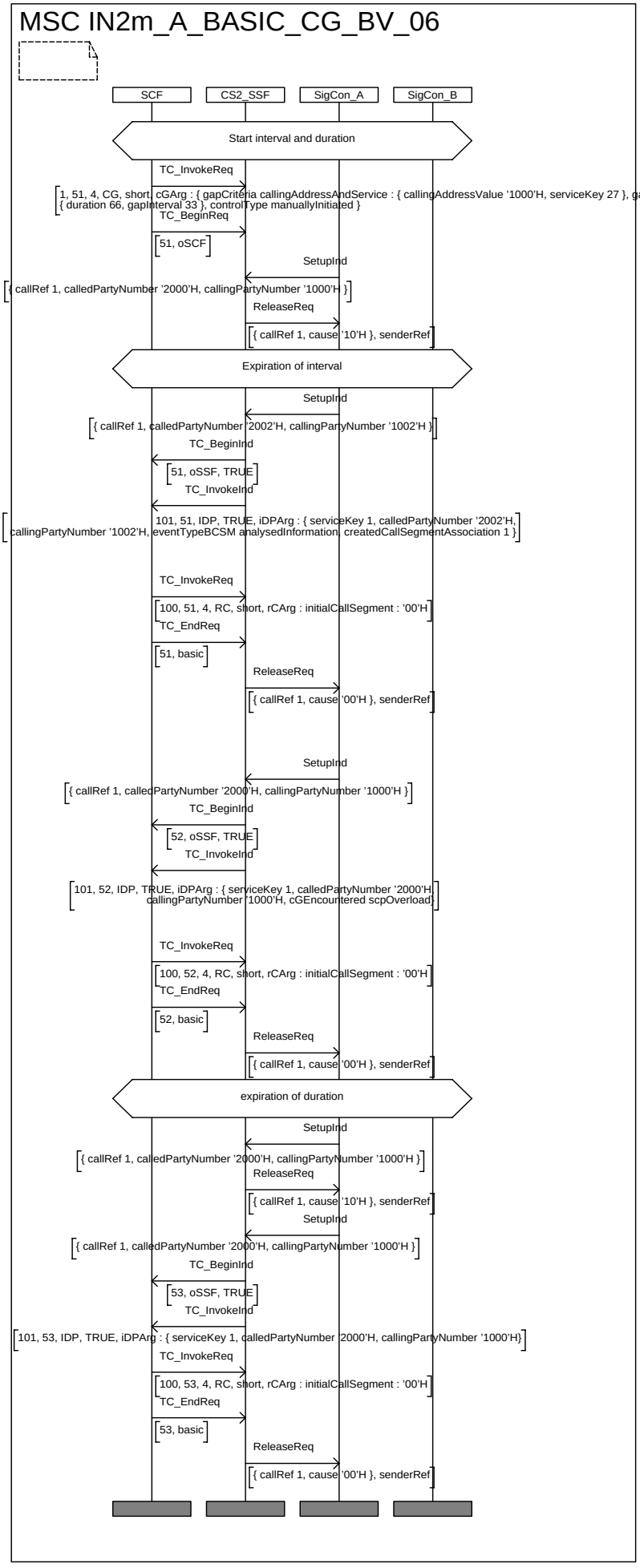
IN2_A_BASIC_CG_BV_04	
Purpose:	Test of **CallGap** procedure and optional parameters callingAddressAndService and gapTreatment via inbandInfo
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory and optional parameters with: - gapCriteria: 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
SCF
CS2_SSF
SigCon_A
SigCon_B
Start interval and duration
TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapCriteria callingAddressAndService : { callingAddressValue '1000'H, serviceKey 27 }, gapIndicators { duration 66, gapInterval 33 }, gapTreatment InformationToSend : InbandInfo : { messageID text : { messageContent "AABB" } } }
TC_BeginReq
51, oSCF
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
Expiration of interval
SetupInd
{ callRef 1, calledPartyNumber '2002'H, callingPartyNumber '1002'H }
TC_BeginInd
51, oSSF, TRUE
TC_InvokeInd
101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2002'H, callingPartyNumber '1002'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }
TC_InvokeReq
100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
51, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload}
TC_InvokeReq
100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
52, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
Expiration of duration
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
53, oSSF, TRUE
TC_InvokeInd
101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}
TC_InvokeReq
100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
53, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef

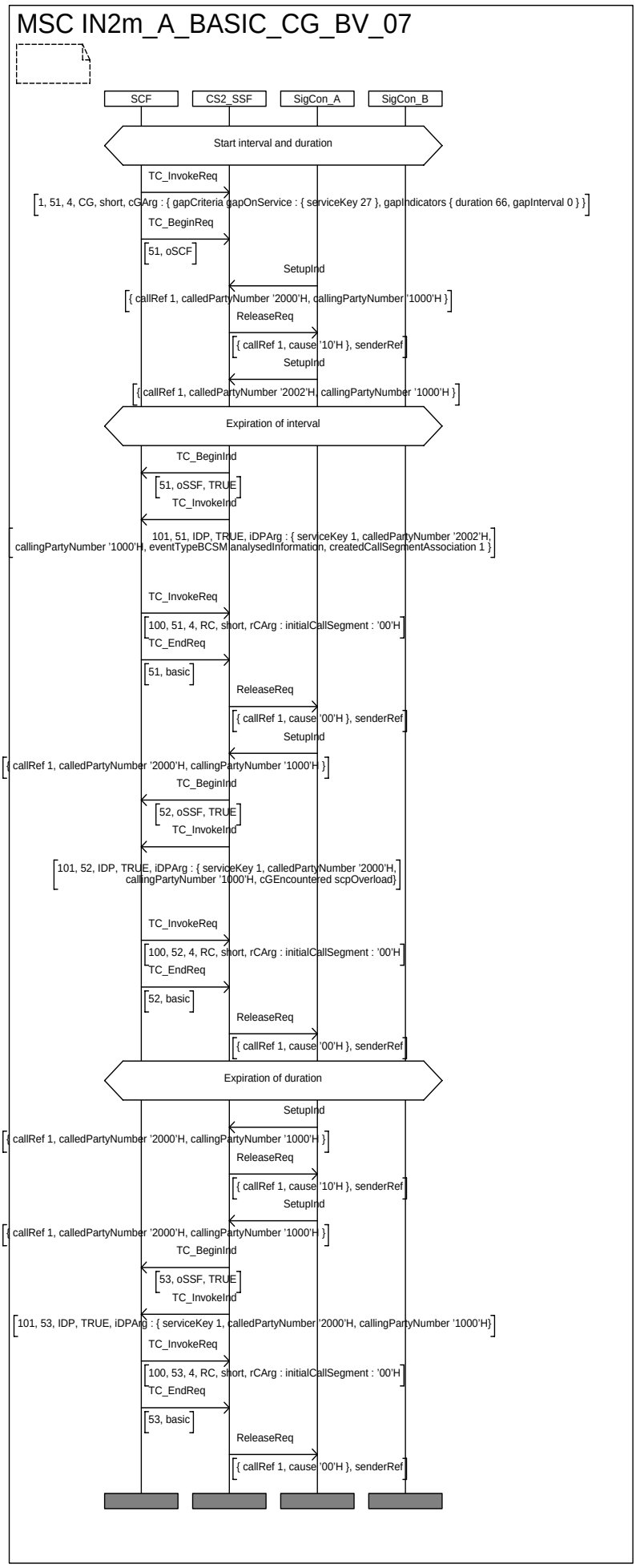
IN2_A_BASIC_CG_BV_05	
Purpose:	Test of **CallGap** procedure and optional parameters callingAddressAndService and gapTreatment with releaseCause
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory and optional parameters with: - gapCriteria: callingAddressAndService - gapIndicators duration being a duration value in seconds gapInterval being an interval value in seconds - gapTreatment releaseCause being any value except default value
Pass criteria	- 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 - 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 cause value 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_05
SCF
CS2_SSF
SigCon_A
SigCon_B
Start interval and duration
TC_InvokeReq
[1, 51, 4, CG, short, cGArg : { gapCriteria callingAddressAndService : { callingAddressValue '1000'H, serviceKey 27 }, gapIndicators { duration 66, gapInterval 33 }, gapTreatment releaseCause : '00'H }]
TC_BeginReq
[51, oSCF]
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
ReleaseReq
[{ callRef 1, cause '10'H }, senderRef]
Expiration of interval
SetupInd
[{ callRef 1, calledPartyNumber '2002'H, callingPartyNumber '1002'H }]
TC_BeginInd
[51, oSSF, TRUE]
TC_InvokeInd
[101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2002'H, callingPartyNumber '1002'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }]
TC_InvokeReq
[100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H]
TC_EndReq
[51, basic]
ReleaseReq
[{ callRef 1, cause '00'H }, senderRef]
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
TC_BeginInd
[52, oSSF, TRUE]
TC_InvokeInd
[101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload}]
TC_InvokeReq
[100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H]
TC_EndReq
[52, basic]
ReleaseReq
[{ callRef 1, cause '00'H }, senderRef]
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
ReleaseReq
[{ callRef 1, cause '10'H }, senderRef]
expiration of duration
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
TC_BeginInd
[53, oSSF, TRUE]
TC_InvokeInd
[101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}]
TC_InvokeReq
[100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H]
TC_EndReq
[53, basic]
ReleaseReq
[{ callRef 1, cause '00'H }, senderRef]

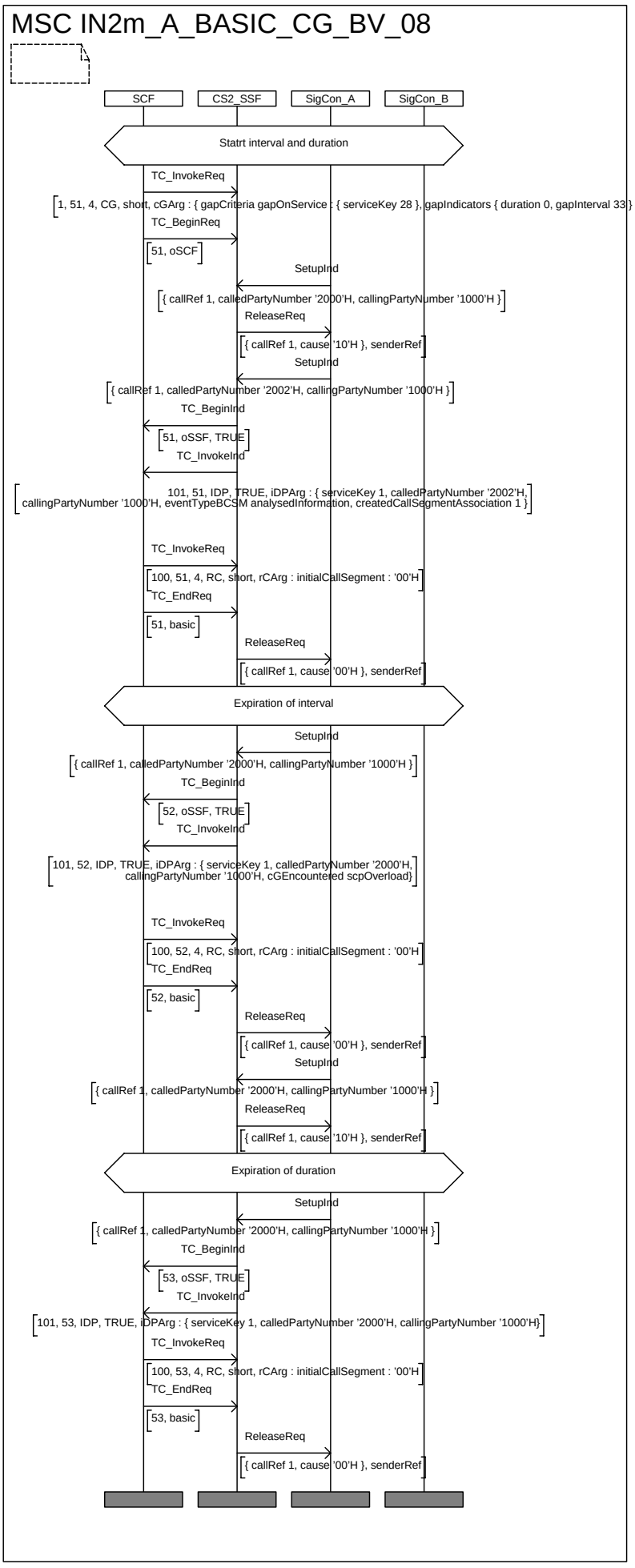
IN2_A_BASIC_CG_BV_06	
Purpose:	Test of **CallGap** procedure and optional parameters callingAddressAndService and controlType
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory and optional parameters with: - gapCriteria: callingAddressAndService - gapIndicators duration being a duration value in seconds gapInterval being an interval value in seconds - controlType being manuallyInitiated
Pass criteria	- Check that SSF releases a call when callgapping is active for the calling address and service key used in SetupInd - 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


MSC IN2m_A_BASIC_CG_BV_06
SCF
CS2_SSF
SigCon_A
SigCon_B
Start interval and duration
TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapCriteria callingAddressAndService : { callingAddressValue '1000'H, serviceKey 27 }, gapIndicators { duration 66, gapInterval 33 }, controlType manuallyInitiated }
TC_BeginReq
51, oSCF
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
Expiration of interval
SetupInd
{ callRef 1, calledPartyNumber '2002'H, callingPartyNumber '1002'H }
TC_BeginInd
51, oSSF, TRUE
TC_InvokeInd
101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2002'H, callingPartyNumber '1002'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }
TC_InvokeReq
100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
51, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload }
TC_InvokeReq
100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
52, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
expiration of duration
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
53, oSSF, TRUE
TC_InvokeInd
101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}
TC_InvokeReq
100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
53, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef

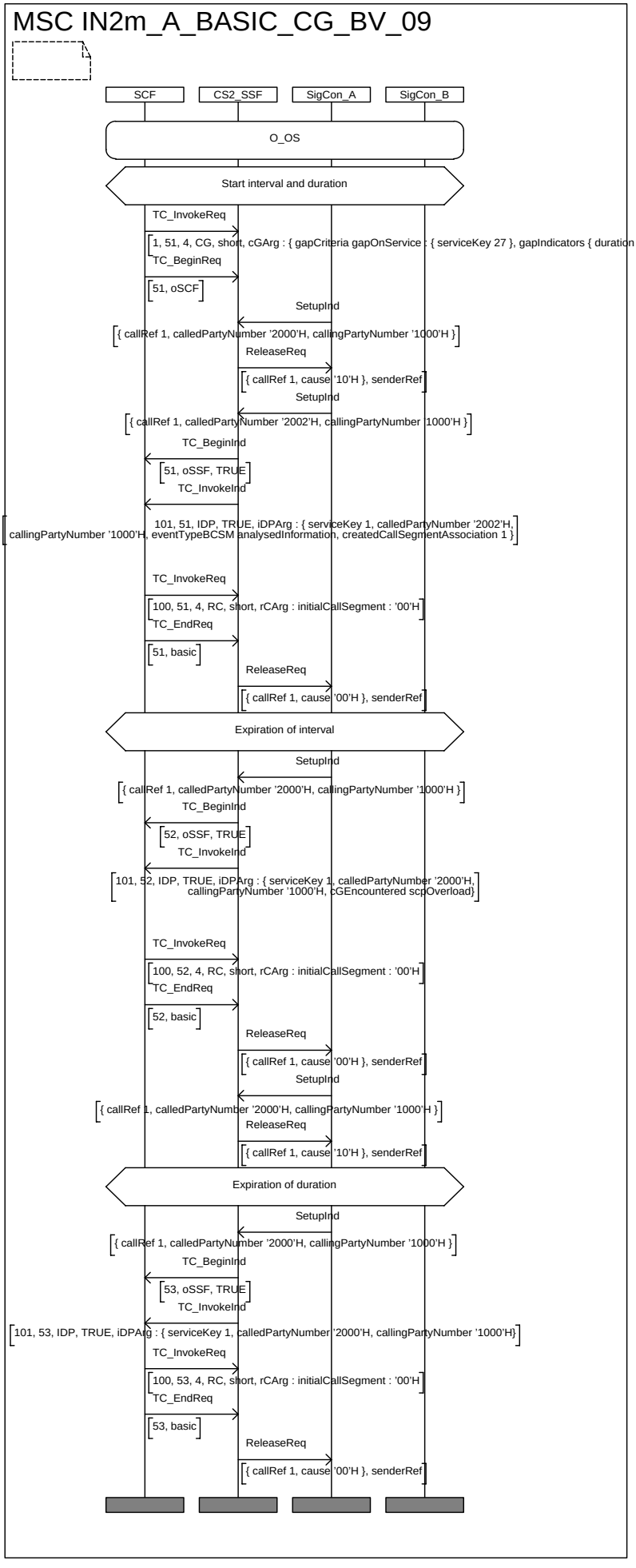
XXXX	IN2_A_BASIC_CG_BV_07
Purpose:	Test of **CallGap** procedure and serviceKey parameter
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory parameters only, with: - gapCriteria: gapOnService with any valid value for serviceKey, - gapIndicators duration being a duration value in seconds gapInterval being an interval value in seconds set to 0
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


MSC IN2m_A_BASIC_CG_BV_07
SCF
CS2_SSF
SigCon_A
SigCon_B
Start interval and duration
TC_InvokeReq
[1, 51, 4, CG, short, cGArg : { gapCriteria gapOnService : { serviceKey 27 }, gapIndicators { duration 66, gapInterval 0 } }]
TC_BeginReq
[51, oSCF]
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
ReleaseReq
[{ callRef 1, cause '10'H }, senderRef]
SetupInd
[{ callRef 1, calledPartyNumber '2002'H, callingPartyNumber '1000'H }]
Expiration of interval
TC_BeginInd
[51, oSSF, TRUE]
TC_InvokeInd
[101, 51, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2002'H, callingPartyNumber '1000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }]
TC_InvokeReq
[100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H]
TC_EndReq
[51, basic]
ReleaseReq
[{ callRef 1, cause '00'H }, senderRef]
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
TC_BeginInd
[52, oSSF, TRUE]
TC_InvokeInd
[101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload}]
TC_InvokeReq
[100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H]
TC_EndReq
[52, basic]
ReleaseReq
[{ callRef 1, cause '00'H }, senderRef]
Expiration of duration
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
ReleaseReq
[{ callRef 1, cause '10'H }, senderRef]
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
TC_BeginInd
[53, oSSF, TRUE]
TC_InvokeInd
[101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}]
TC_InvokeReq
[100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H]
TC_EndReq
[53, basic]
ReleaseReq
[{ callRef 1, cause '00'H }, senderRef]

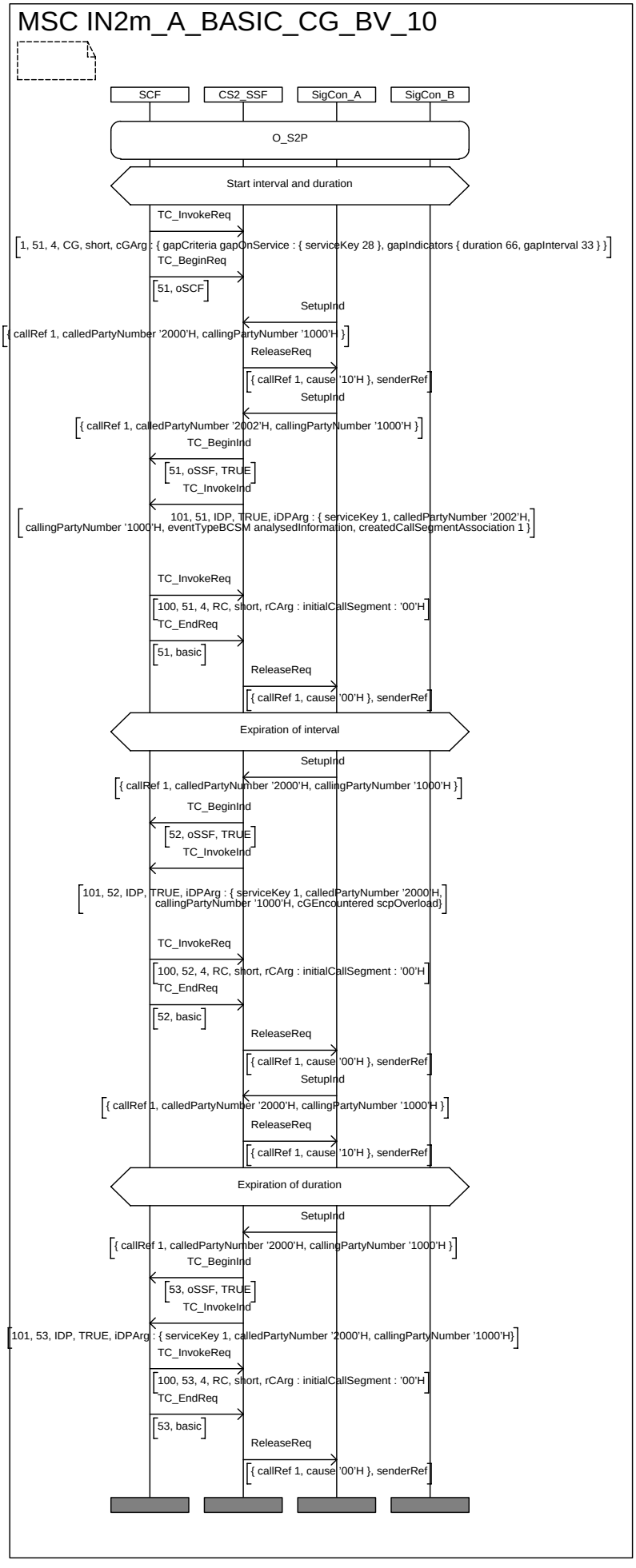
XXXX	IN2_A_BASIC_CG_BV_08
Purpose:	Test of **CallGap** procedure and serviceKey parameter
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory parameters only, with: - gapCriteria: gapOnService with any valid value for serviceKey, - 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


MSC IN2m_A_BASIC_CG_BV_08
SCF
CS2_SSF
SigCon_A
SigCon_B
Statrt interval and duration
TC_InvokeReq
[1, 51, 4, CG, short, cGArg : { gapCriteria gapOnService : { serviceKey 28 }, gapIndicators { duration 0, gapInterval 33} }]
TC_BeginReq
[51, oSCF]
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
ReleaseReq
[{ callRef 1, cause '10'H }, senderRef]
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 analysedInformation, createdCallSegmentAssociation 1 }]
TC_InvokeReq
[100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H]
TC_EndReq
[51, basic]
ReleaseReq
[{ callRef 1, cause '00'H }, senderRef]
Expiration of interval
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
TC_BeginInd
[52, oSSF, TRUE]
TC_InvokeInd
[101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload}]
TC_InvokeReq
[100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H]
TC_EndReq
[52, basic]
ReleaseReq
[{ callRef 1, cause '00'H }, senderRef]
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
ReleaseReq
[{ callRef 1, cause '10'H }, senderRef]
Expiration of duration
SetupInd
[{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
TC_BeginInd
[53, oSSF, TRUE]
TC_InvokeInd
[101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}]
TC_InvokeReq
[100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H]
TC_EndReq
[53, basic]
ReleaseReq
[{ callRef 1, cause '00'H }, senderRef]

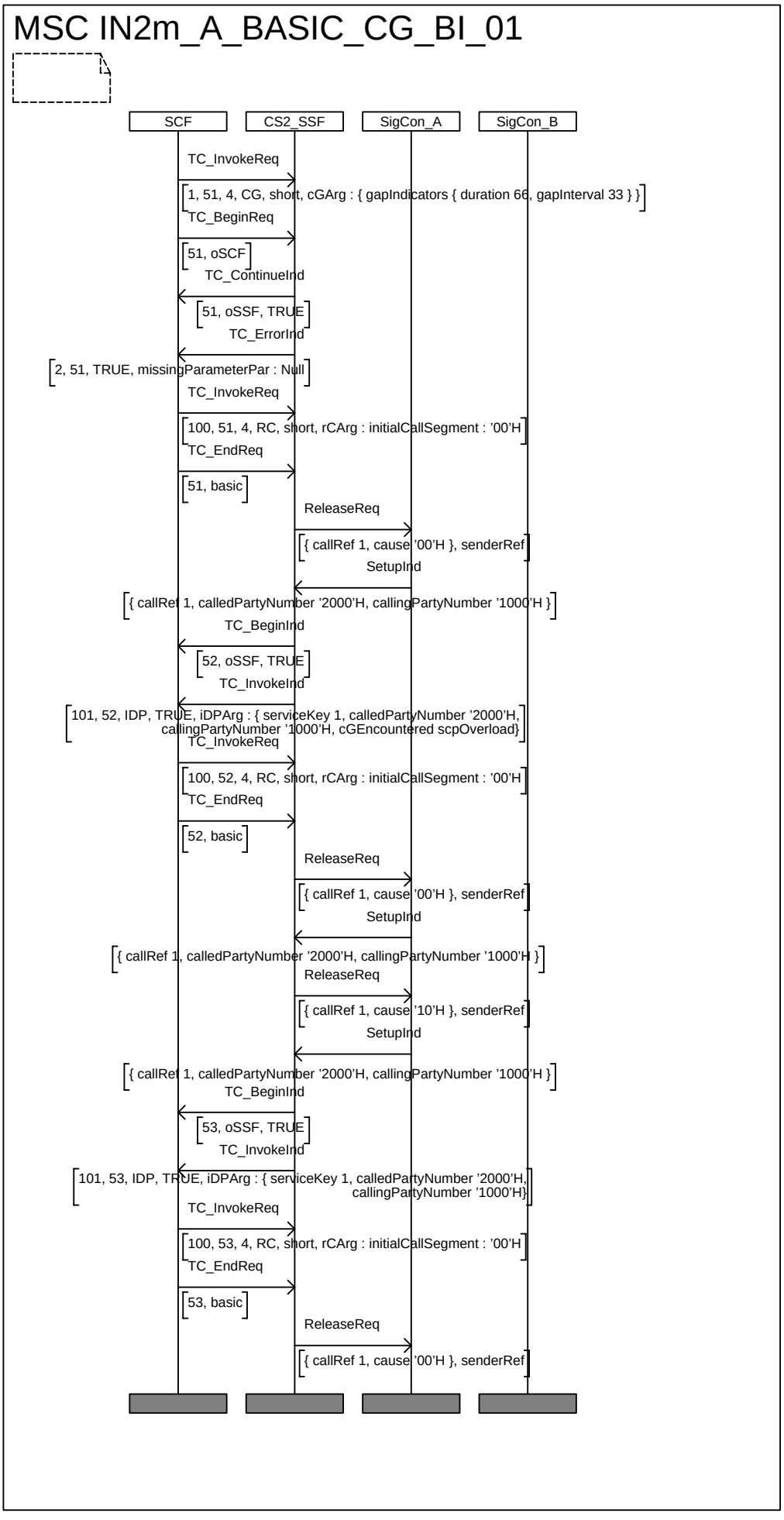
XXXX	IN2_A_BASIC_CG_BV_09
Purpose:	Test of **CallGap** procedure in WaitForInstruction state
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF a **CallGap** invoke containing mandatory parameters only, with: - gapCriteria: gapOnService with any valid value for serviceKey, - 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"
Postamble:	none


MSC IN2m_A_BASIC_CG_BV_09
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
Start interval and duration
TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapCriteria gapOnService : { serviceKey 27 }, gapIndicators { duration 66, gapInterval 33 } }
TC_BeginReq
51, oSCF
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
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 analysedInformation, createdCallSegmentAssociation 1 }
TC_InvokeReq
100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
51, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
Expiration of interval
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, CGEncountered scpOverload}
TC_InvokeReq
100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
52, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
Expiration of duration
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
53, oSSF, TRUE
TC_InvokeInd
101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}
TC_InvokeReq
100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
53, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef

XXXX	IN2_A_BASIC_CG_BV_10
Purpose:	Test of **CallGap** procedure in Monitoring state
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
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"
Postamble:	none

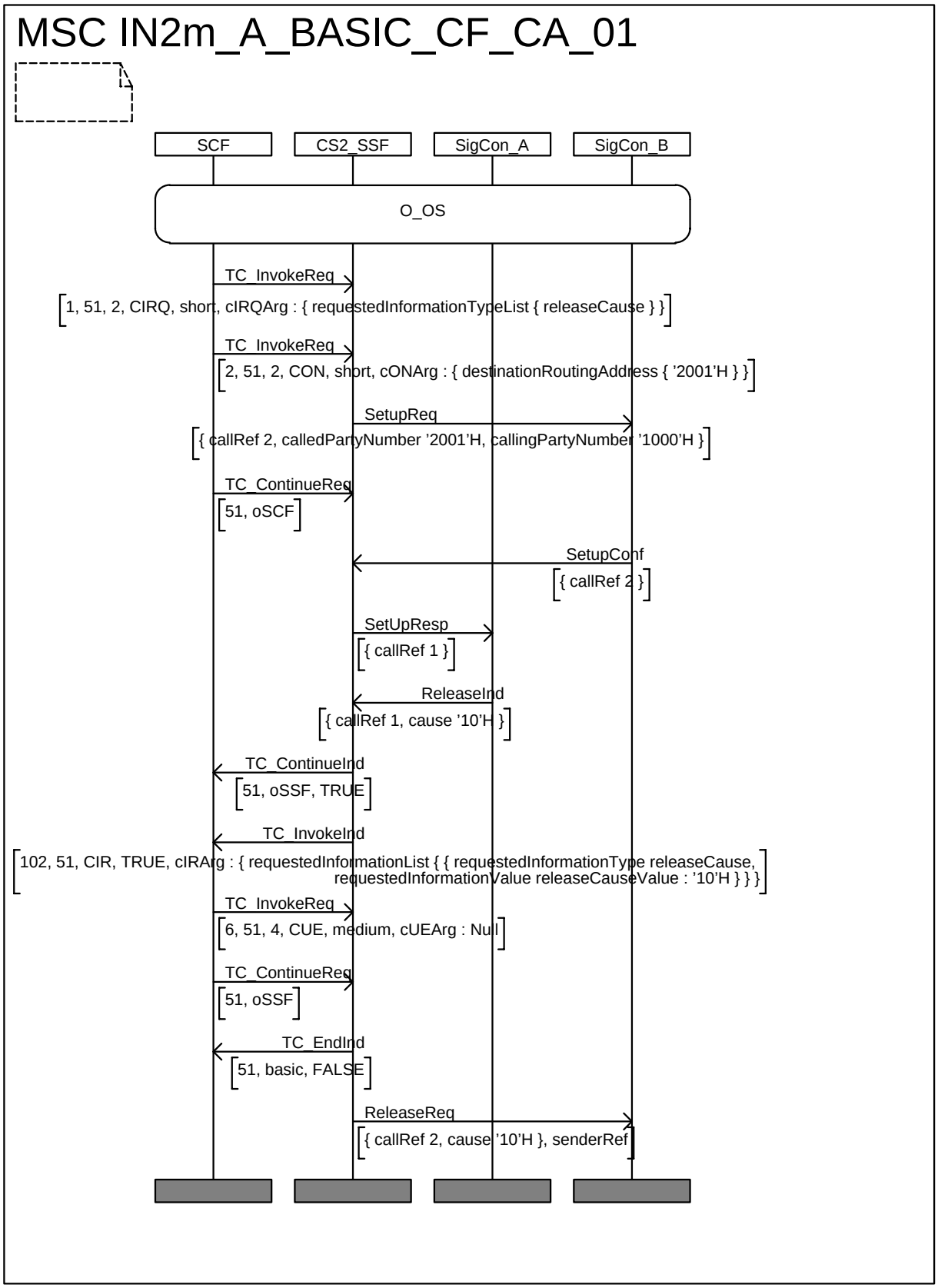

MSC IN2m_A_BASIC_CG_BV_10
SCF
CS2_SSF
SigCon_A
SigCon_B
O_S2P
Start interval and duration
TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapCriteria gapOnService : { serviceKey 28 }, gapIndicators { duration 66, gapInterval 33 } }
TC_BeginReq
51, oSCF
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
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 analysedInformation, createdCallSegmentAssociation 1 }
TC_InvokeReq
100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
51, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
Expiration of interval
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload}
TC_InvokeReq
100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
52, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
Expiration of duration
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
53, oSSF, TRUE
TC_InvokeInd
101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}
TC_InvokeReq
100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
53, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef

IN2_A_BASIC_CG_BI_01	
Purpose:	Test of **CallGap** procedure and missing parameter
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke with missing mandatory parameter gapCriteria
Pass criteria	Check that SSF rejects the invoke
Postamble:	none

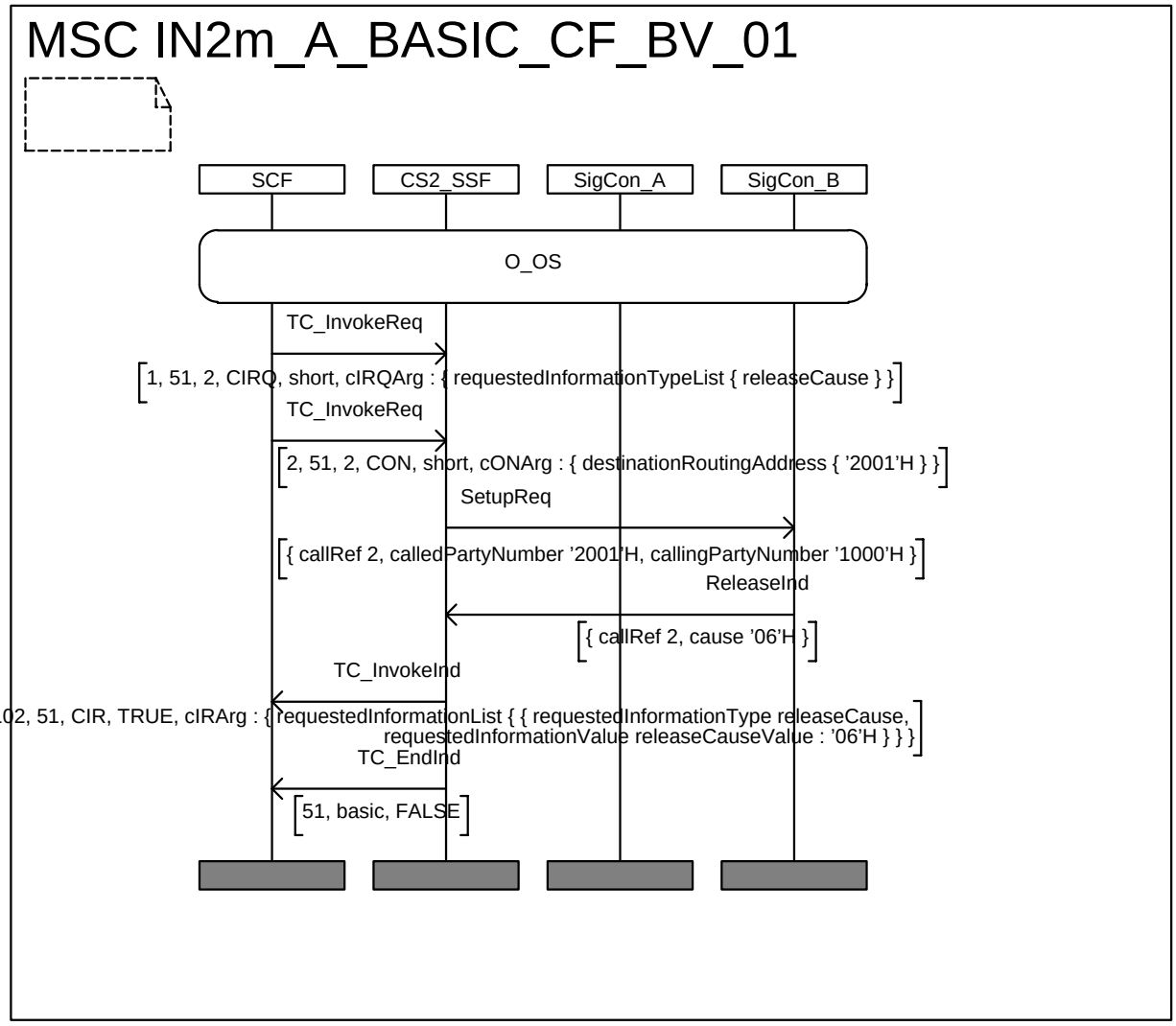

MSC IN2m_A_BASIC_CG_BI_01
SCF
CS2_SSF
SigCon_A
SigCon_B
TC_InvokeReq
1, 51, 4, CG, short, cGArg : { gapIndicators { duration 66, gapInterval 33 } }
TC_BeginReq
51, oSCF
TC_ContinueInd
51, oSSF, TRUE
TC_ErrorInd
2, 51, TRUE, missingParameterPar : Null
TC_InvokeReq
100, 51, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
51, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
52, oSSF, TRUE
TC_InvokeInd
101, 52, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H, cGEncountered scpOverload}
TC_InvokeReq
100, 52, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
52, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
ReleaseReq
{ callRef 1, cause '10'H }, senderRef
SetupInd
{ callRef 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
TC_BeginInd
53, oSSF, TRUE
TC_InvokeInd
101, 53, IDP, TRUE, iDPArg : { serviceKey 1, calledPartyNumber '2000'H, callingPartyNumber '1000'H}
TC_InvokeReq
100, 53, 4, RC, short, rCArg : initialCallSegment : '00'H
TC_EndReq
53, basic
ReleaseReq
{ callRef 1, cause '00'H }, senderRef

6.4.5 CallInformation procedure

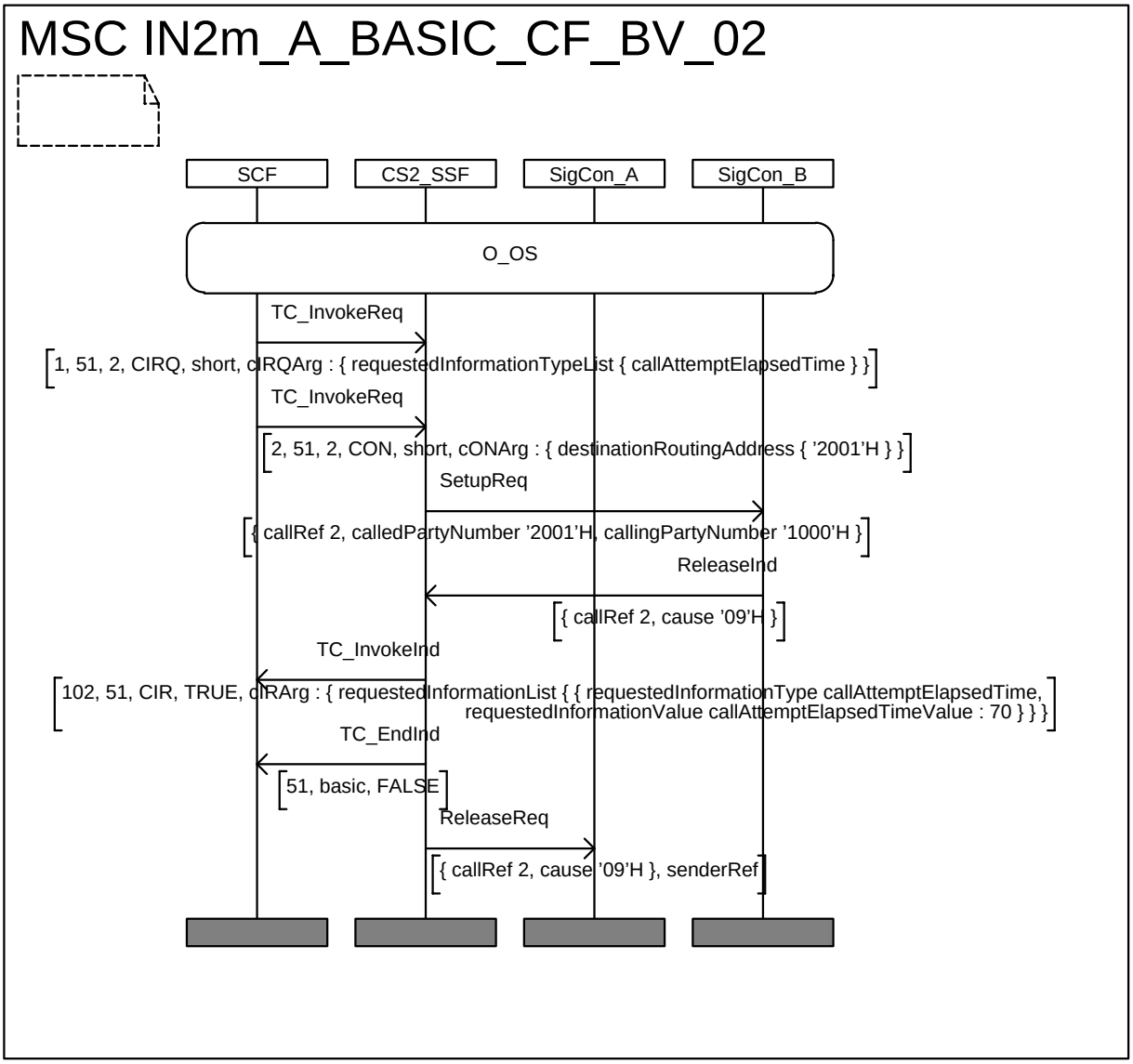
IN2_A_BASIC_CF_CA_01	
Purpose:	Test of **CallInformation** procedure
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF a **CallInformationRequest** invoke, containing mandatory parameters only and indicating a single information type, with at least the parameters: - requestedInformationTypeList including: - requestedInformationType being releaseCause, followed by a **Connect** to establish a Connection with SigConB When the connection is established, SigConA releases the call
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 SSF becomes idle.
Postamble:	none


MSC IN2m_A_BASIC_CF_CA_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
[1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { releaseCause } }]
TC_InvokeReq
[2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }]
SetupReq
[{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }]
TC_ContinueReq
[51, oSCF]
SetupConf
[{ callRef 2 }]
SetUpResp
[{ callRef 1 }]
ReleaseInd
[{ callRef 1, cause '10'H }]
TC_ContinueInd
[51, oSSF, TRUE]
TC_InvokeInd
[102, 51, CIR, TRUE, cIRArg : { requestedInformationList { { requestedInformationType releaseCause, requestedInformationValue releaseCauseValue : '10'H } } }]
TC_InvokeReq
[6, 51, 4, CUE, medium, cUEArg : Null]
TC_ContinueReq
[51, oSSF]
TC_EndInd
[51, basic, FALSE]
ReleaseReq
[{ callRef 2, cause '10'H }, senderRef]

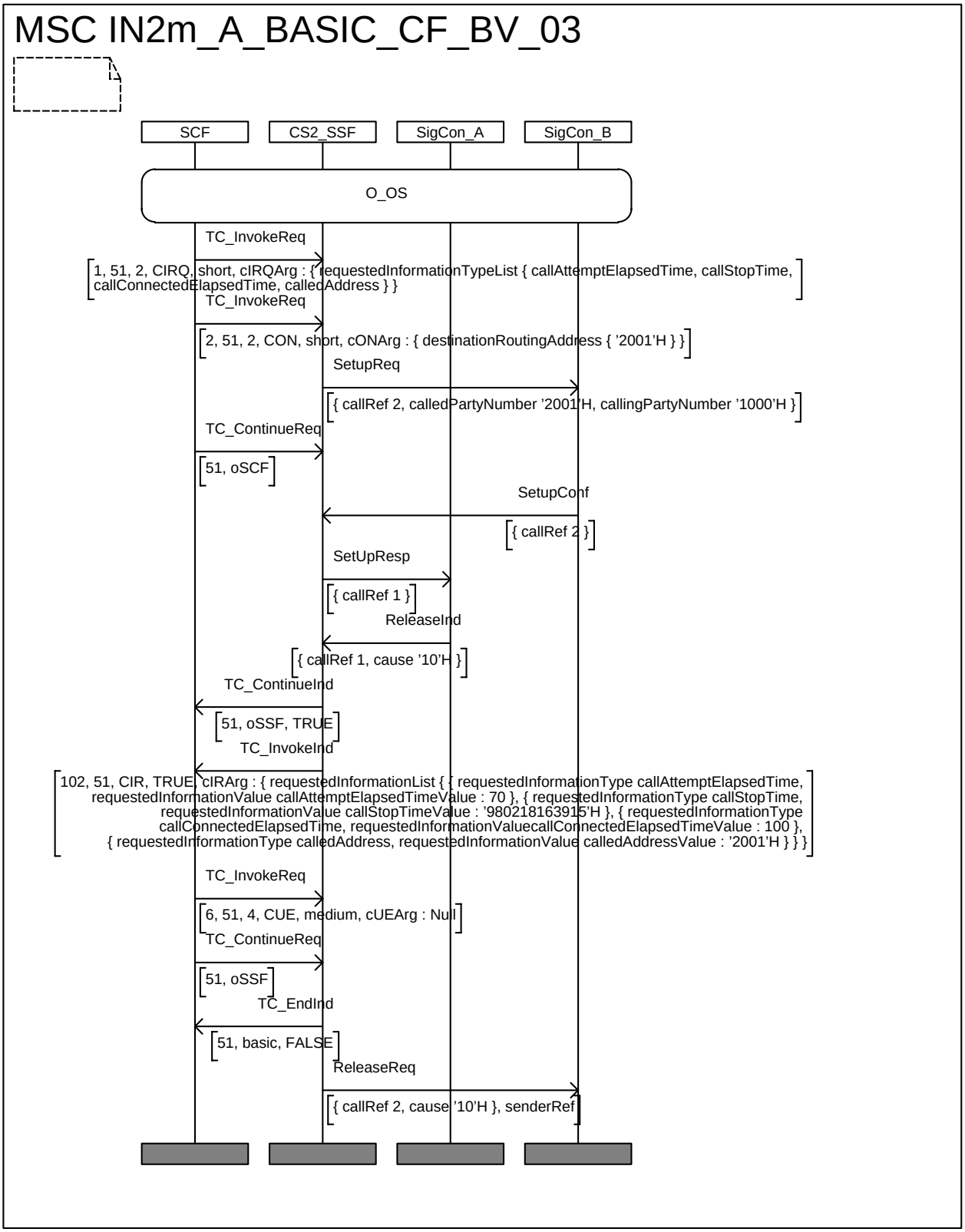
IN2_A_BASIC_CF_BV_01	
Purpose:	Test of **CallInformation** procedure and release parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS In addition, user B is declared busy
Test description	SCF sends to SSF a **CallInformationRequest** invoke, containing mandatory parameters only and indicating a single information type, with at least the parameters: - requestedInformationTypeList including: - requestedInformationType being releaseCause, followed by a Continue to establish a Connection with SigConB But the connection is not established, as B is busy
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
Postamble:	none


MSC IN2m_A_BASIC_CF_BV_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { releaseCause } }
TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
ReleaseInd
{ callRef 2, cause '06'H }
TC_InvokeInd
102, 51, CIR, TRUE, cIRArg : { requestedInformationList { { requestedInformationType releaseCause,
requestedInformationValue releaseCauseValue : '06'H } } }
TC_EndInd
51, basic, FALSE

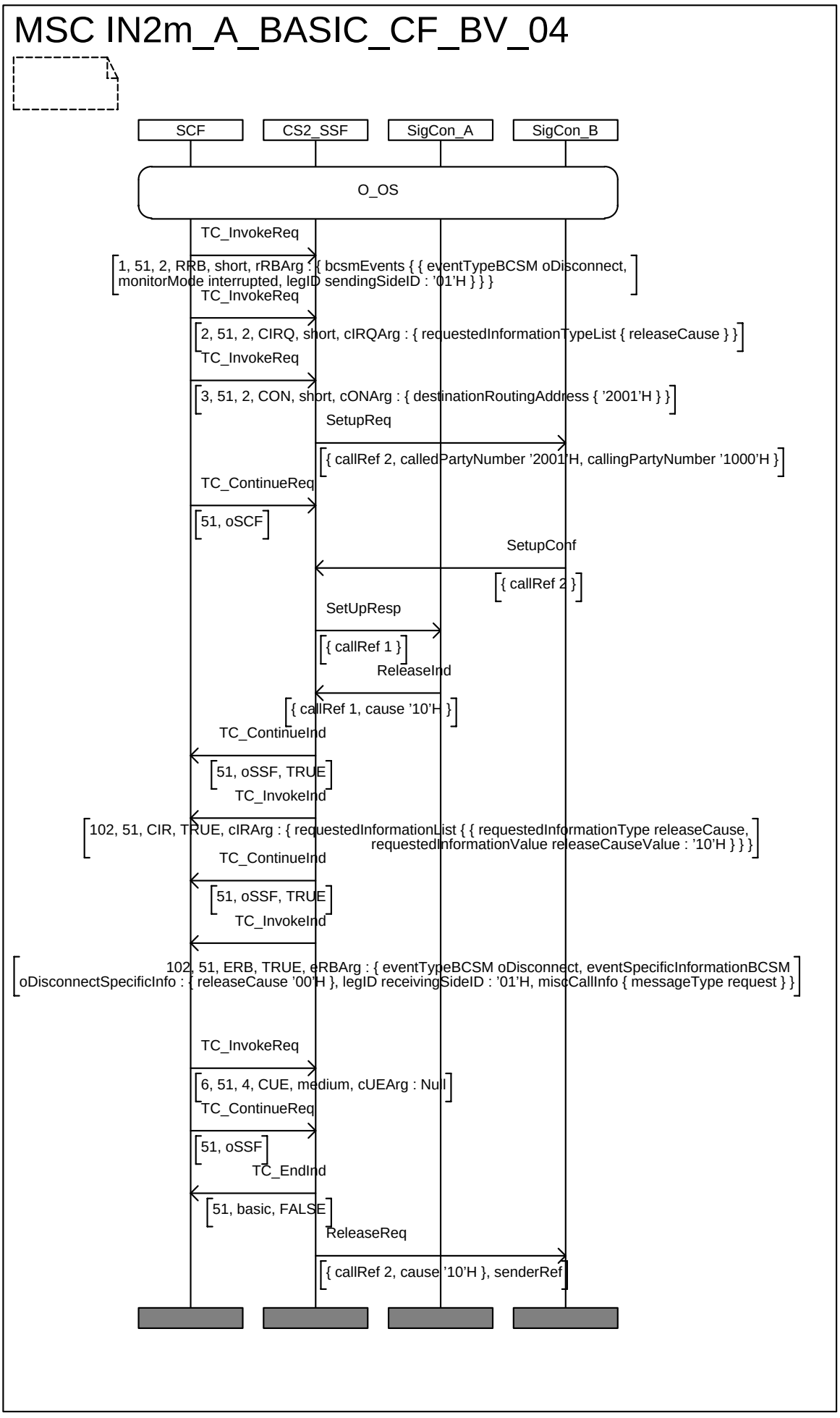
IN2_A_BASIC_CF_BV_02	
Purpose:	Test of **CallInformation** procedure and time parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS In addition, user B is declared not answering within timeout
Test description	SCF sends to SSF a **CallInformationRequest** invoke, containing mandatory parameters 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


MSC IN2m_A_BASIC_CF_BV_02
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { callAttemptElapsedTime } }
TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
ReleaseInd
{ callRef 2, cause '09'H }
TC_InvokeInd
102, 51, CIR, TRUE, cIRArg : { requestedInformationList { { requestedInformationType callAttemptElapsedTime, requestedInformationValue callAttemptElapsedTimeValue : 70 } } }
TC_EndInd
51, basic, FALSE
ReleaseReq
{ callRef 2, cause '09'H }, senderRef

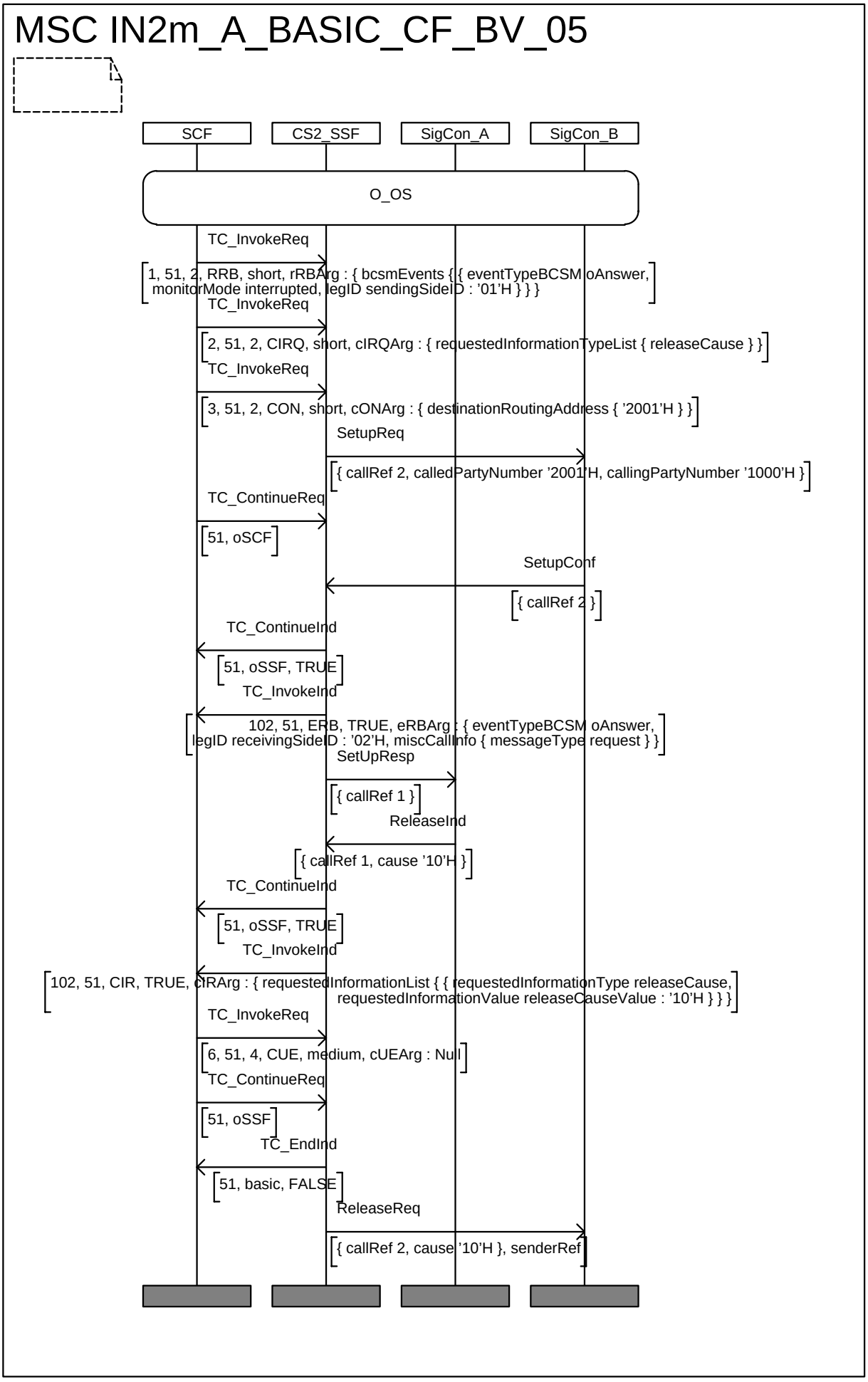
IN2_A_BASIC_CF_BV_03	
Purpose:	Test of **CallInformation** procedure and elapsed time parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF a **CallInformationRequest** invoke, containing mandatory parameters and indicating a multiple information type, with at least the parameters: - requestedInformationTypeList including: - requestedInformationType (callAttemptElapsedTime), also including: - requestedInformationType (callStopTime), also including: - requestedInformationType (callConnectedElapsedTime), and including: - requestedInformationType (calledAddress), followed by a Continue to establish a Connection with SigConB
Pass criteria	- 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: - requestedInformationList including: - requestedInformationType (callAttemptElapsedTime), - requestedInformationValue being callAttemptElapsedTimeValue, also including: - requestedInformationType (callStopTime), - requestedInformationValue being callStopTimeValue, also including: - requestedInformationType (callConnectedElapsedTime), - requestedInformationValue being callConnectedElapsedTimeValue, and including: - requestedInformationType (calledAddress), - requestedInformationValue being calledAddressValue
Postamble:	none


MSC IN2m_A_BASIC_CF_BV_03
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { callAttemptElapsedTime, callStopTime, callConnectedElapsedTime, calledAddress } }
TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
TC_ContinueReq
51, oSCF
SetupConf
{ callRef 2 }
SetUpResp
{ callRef 1 }
ReleaseInd
{ callRef 1, cause '10'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, CIR, TRUE, cIRArg : { requestedInformationList { { requestedInformationType callAttemptElapsedTime, requestedInformationValue callAttemptElapsedTimeValue : 70 }, { requestedInformationType callStopTime, requestedInformationValue callStopTimeValue : '980218163915'H }, { requestedInformationType callConnectedElapsedTime, requestedInformationValuecallConnectedElapsedTimeValue : 100 }, { requestedInformationType calledAddress, requestedInformationValue calledAddressValue : '2001'H } } }
TC_InvokeReq
6, 51, 4, CUE, medium, cUEArg : Null
TC_ContinueReq
51, oSSF
TC_EndInd
51, basic, FALSE
ReleaseReq
{ callRef 2, cause '10'H }, senderRef

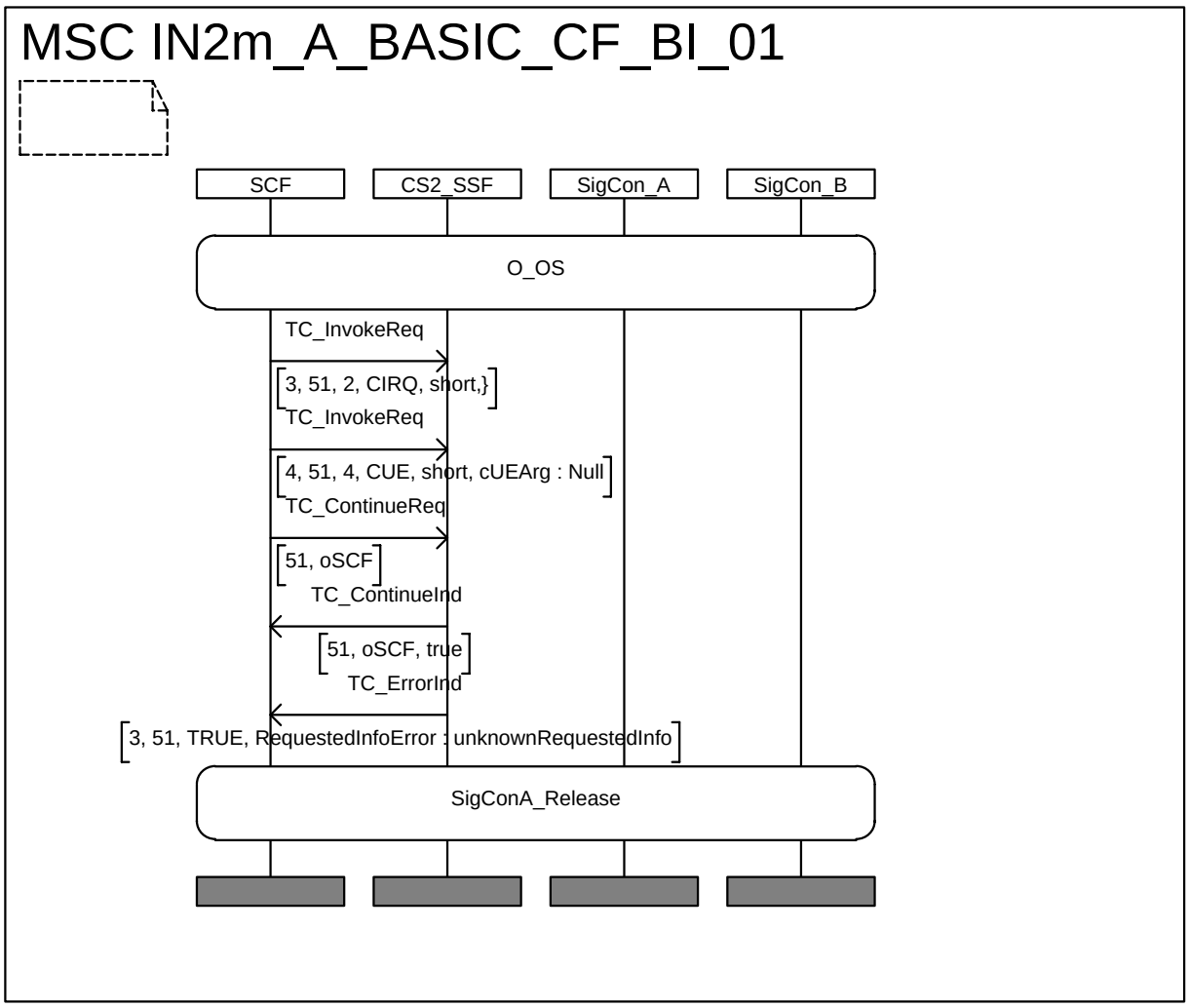
IN2_A_BASIC_CF_BV_04	
Purpose:	Test of **CallInformation** procedure combined with **RequestReportBCSMEvent** procedure.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oDisconnect - monitoringMode=interrupted 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


MSC IN2m_A_BASIC_CF_BV_04
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }
TC_InvokeReq
2, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { releaseCause } }
TC_InvokeReq
3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
TC_ContinueReq
51, oSCF
SetupConf
{ callRef 2 }
SetUpResp
{ callRef 1 }
ReleaseInd
{ callRef 1, cause '10'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, CIR, TRUE, cIRArg : { requestedInformationList { { requestedInformationType releaseCause, requestedInformationValue releaseCauseValue : '10'H } } }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oDisconnect, eventSpecificInformationBCSM oDisconnectSpecificInfo : { releaseCause '00'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }
TC_InvokeReq
6, 51, 4, CUE, medium, cUEArg : Null
TC_ContinueReq
51, oSSF
TC_EndInd
51, basic, FALSE
ReleaseReq
{ callRef 2, cause '10'H }, senderRef

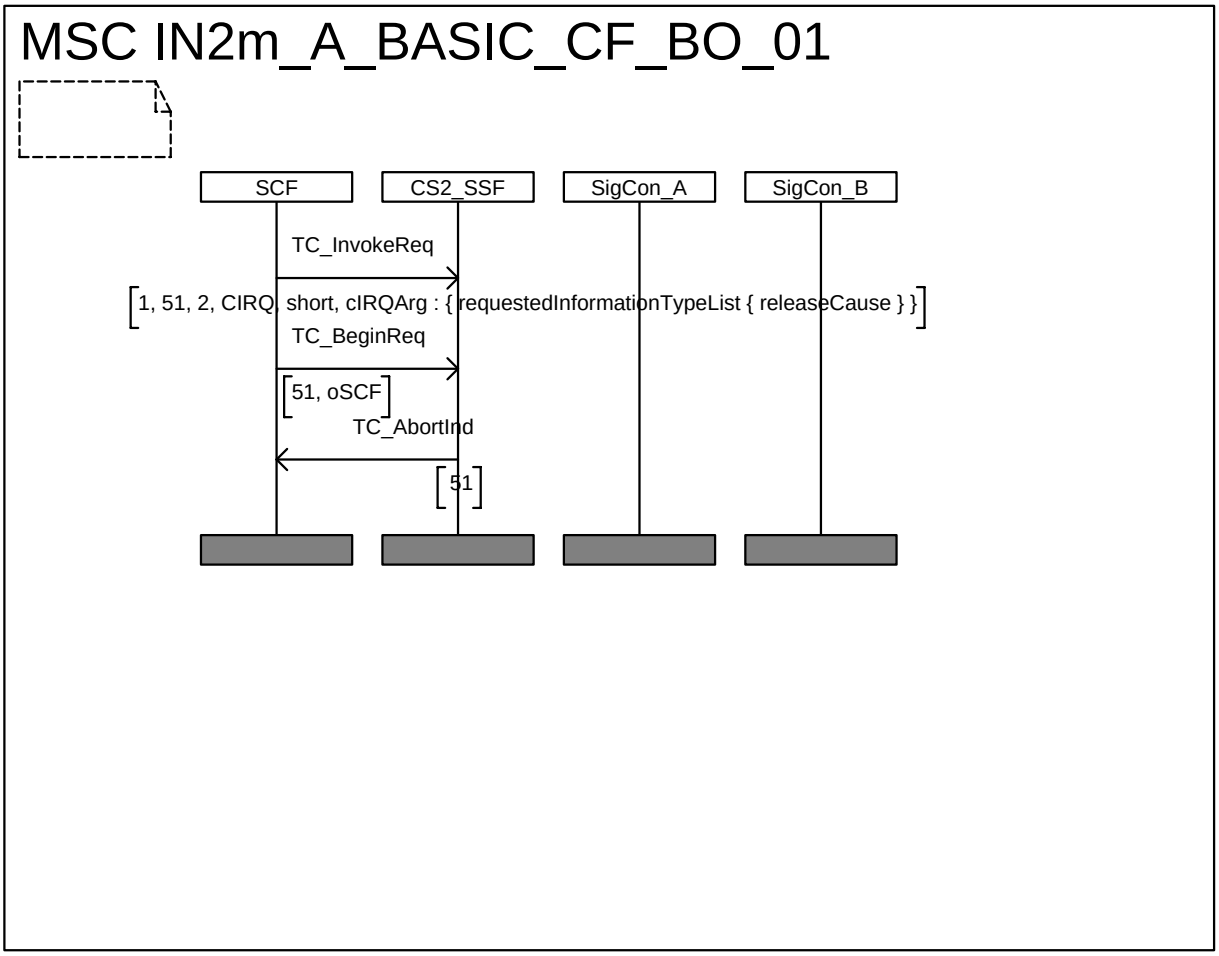
IN2_A_BASIC_CF_BV_05	
Purpose:	Test of **CallInformation** procedure combined with **RequestReportBCSMEvent** procedure.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oAnswer - monitoringMode=interrupted 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


MSC IN2m_A_BASIC_CF_BV_05
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID sendingSideID : '01'H } } }
TC_InvokeReq
2, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { releaseCause } }
TC_InvokeReq
3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
TC_ContinueReq
51, oSCF
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 }
ReleaseInd
{ callRef 1, cause '10'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, CIR, TRUE, cIRArg : { requestedInformationList { { requestedInformationType releaseCause, requestedInformationValue releaseCauseValue : '10'H } } }
TC_InvokeReq
6, 51, 4, CUE, medium, cUEArg : Null
TC_ContinueReq
51, oSSF
TC_EndInd
51, basic, FALSE
ReleaseReq
{ callRef 2, cause '10'H }, senderRef

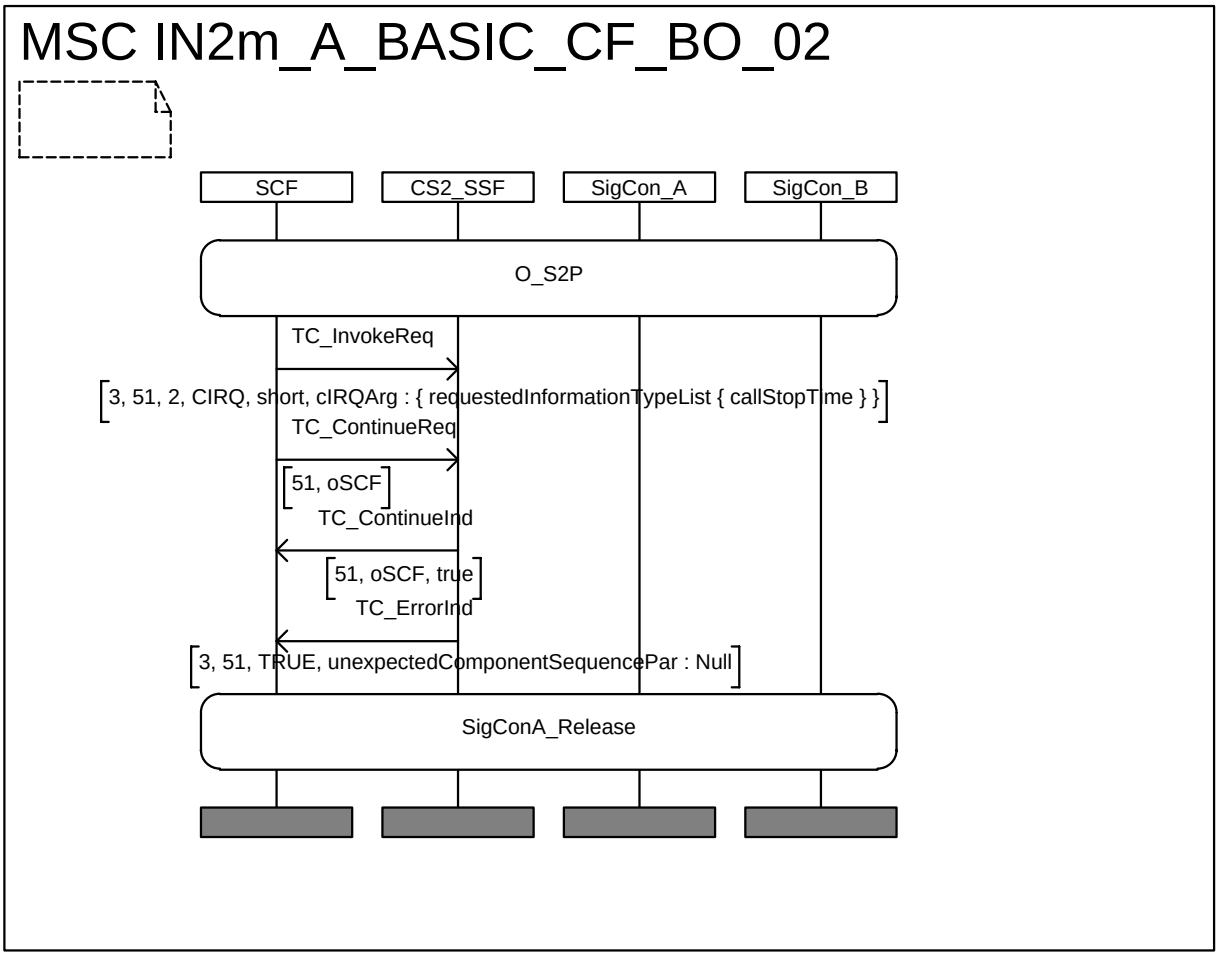
IN2_A_BASIC_CF_BI_01	
Purpose:	Test of **CallInformation** procedure with invalid parameters
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF a **CallInformationRequest** invoke, with - RequestedInformationTypeList, containing an invalid parameter followed by a Continue to establish a Connection with SigConB
Pass criteria	- Check that SSF sends back **CallInformationRequest** error to SCF indicating : requestedInfoError
Postamble:	SigConA_Release



IN2_A_BASIC_CF_BO_01	
Purpose:	Test of **CallInformation** procedure in wrong state (idle)
Requirement ref	
Selection Cond.	
Preamble:	None
Test description	SCF sends **CallInformationRequest** invoke to SSF
Pass criteria	Check that SSF sends to SCF a **TC_ABORT**
Postamble:	None

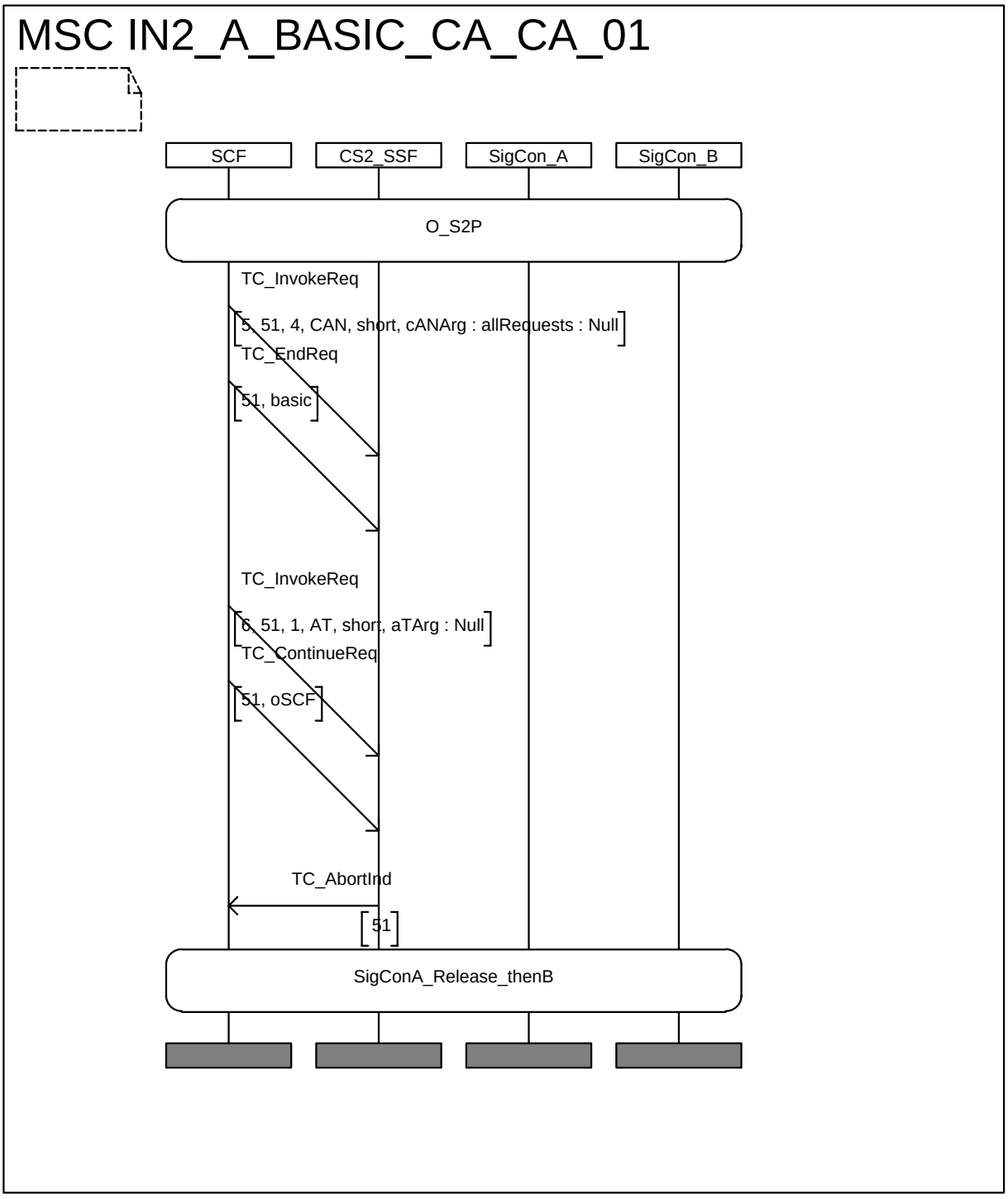


IN2_A_BASIC_CF_BO_02	
Purpose:	Test of **CallInformation** procedure in wrong state (monitoring)
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
Test description	SCF sends **CallInformationRequest** invoke to SSF
Pass criteria	Check that SSF sends to SCF a **CallInformationRequest** error with an indication of UnexpectedComponentSequence
Postamble:	SigConA_Release_thenB



6.4.6 Cancel procedure

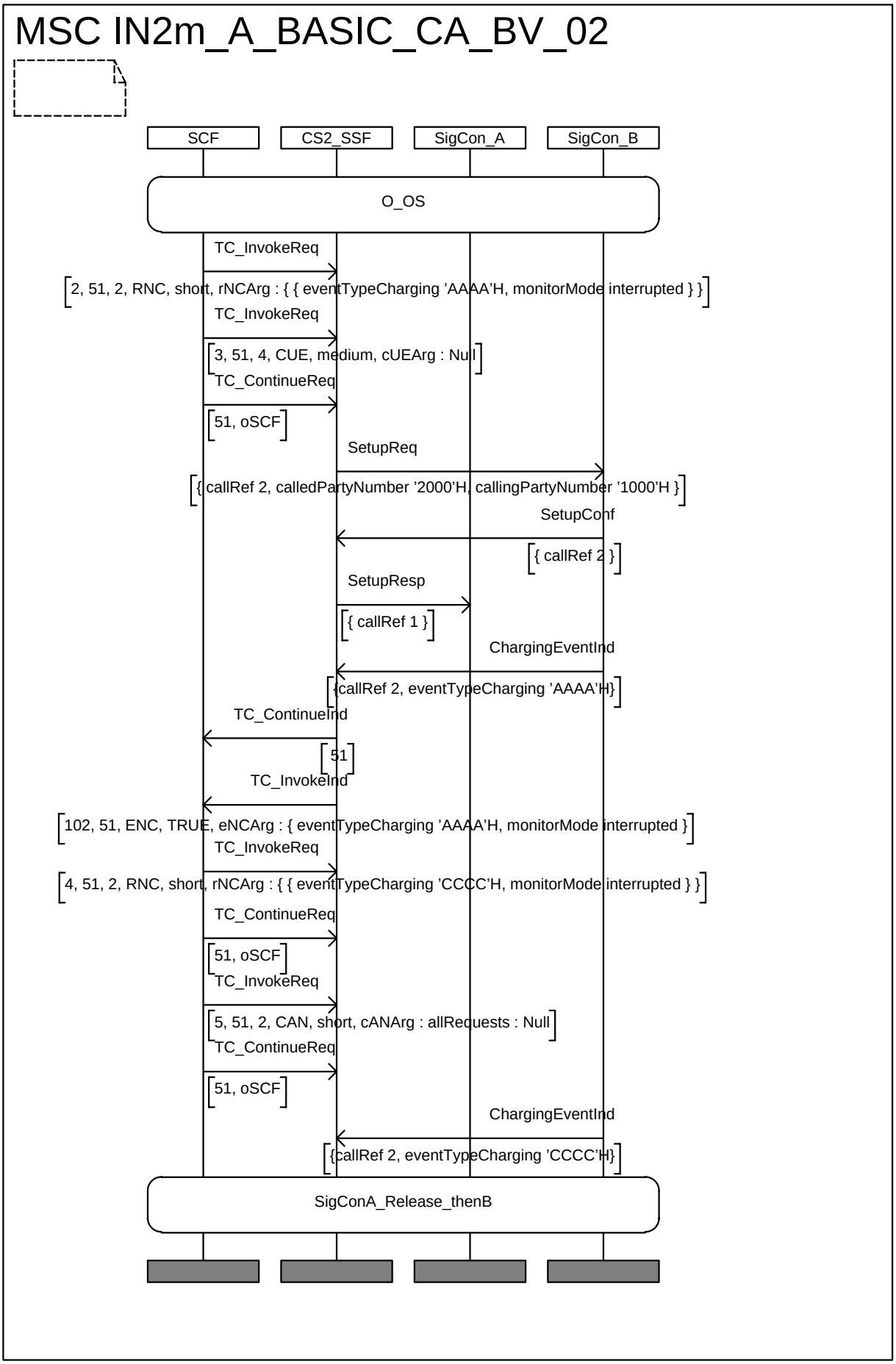
IN2_A_BASIC_CA_CA_01	
Purpose:	Test of **Cancel** base procedure
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
Test description	**Cancel** invoke sent by SCF to SSF, containing allRequests
Pass criteria	- Check that SSF returns to idle state - To ensure that SSF is now in idle state, SCF sends **ActivityTest** invoke to SSF with DialogueId 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_S2P
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

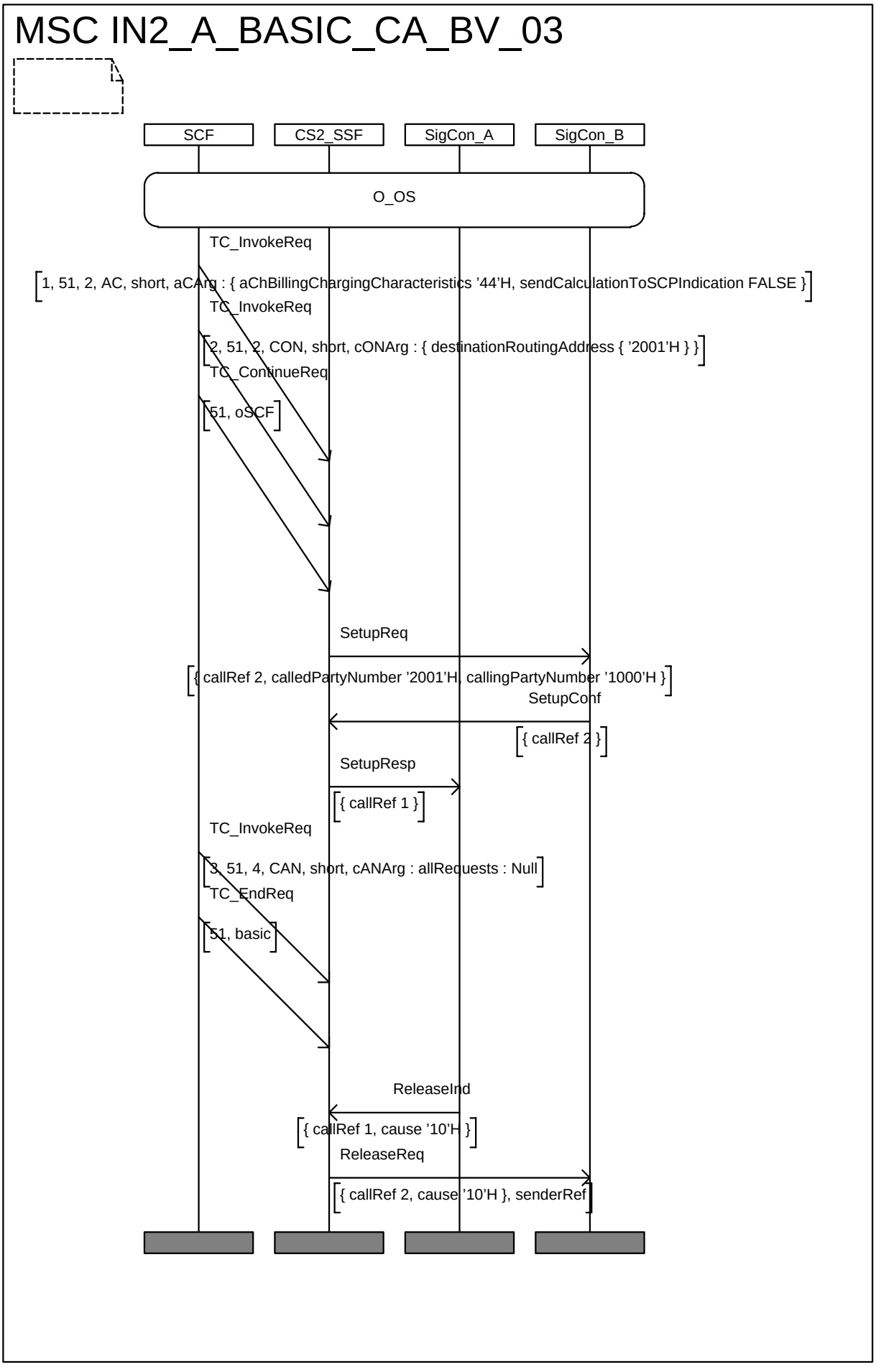
IN2_A_BASIC_CA_BV_01

This test purpose was dropped.

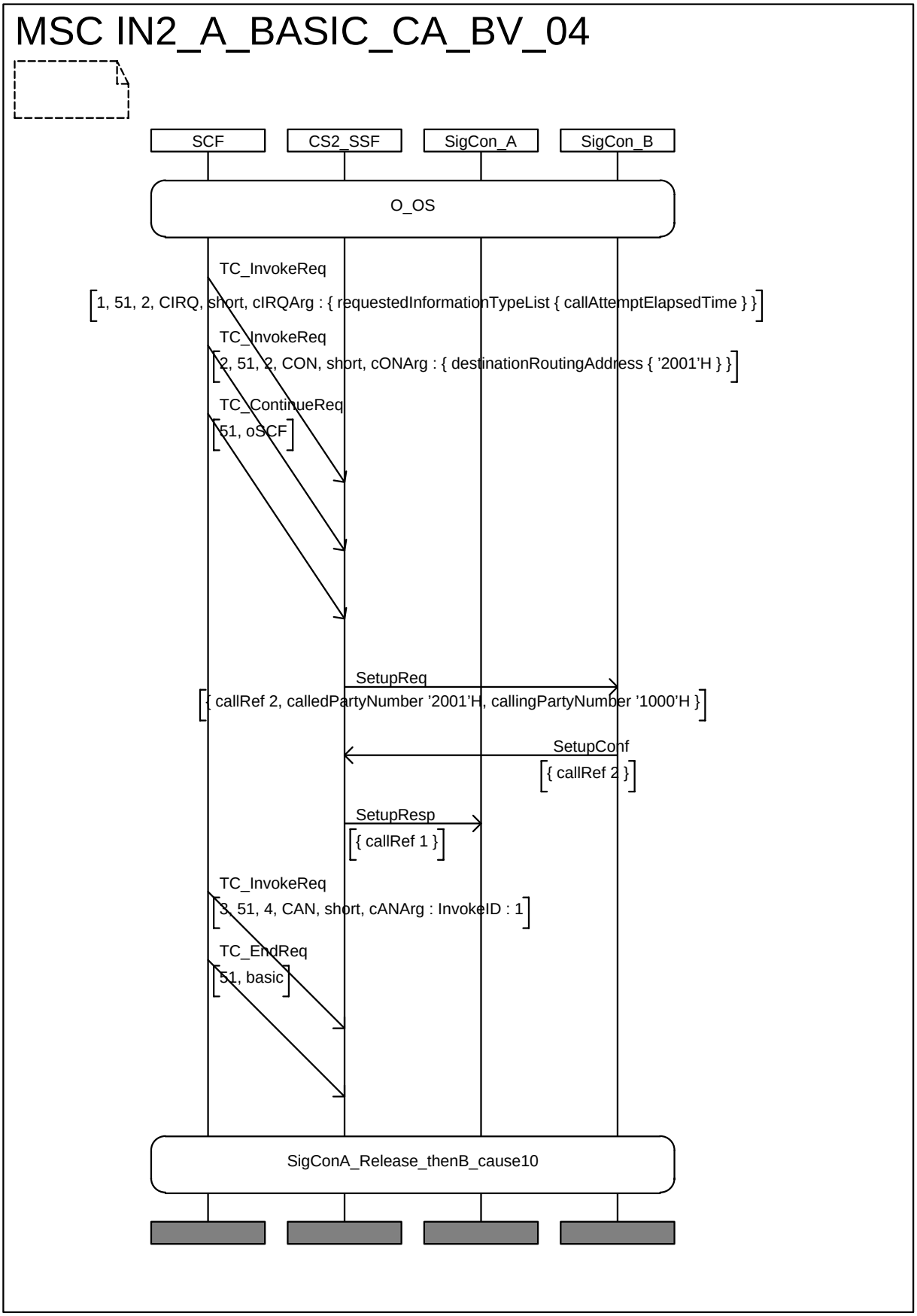
IN2_A_BASIC_CA_BV_02	
Purpose:	Test of **Cancel** procedure on RequestNotificationChargingEvent
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestNotificationChargingEvent** invoke containing mandatory parameters only, with: - eventTypeCharging, - monitorMode (interrupted) After triggering of charging event from SigConA, SSF sends to SCF an **EventNotificationCharging** invoke with the indication of eventTypeCharging SCF sends to SSF Continue invoke then a new **RequestNotificationChargingEvent** invoke containing mandatory parameters only, with: - eventTypeCharging, - monitorMode (interrupted) followed by a **Cancel** invoke containing allRequests
Pass criteria	- Check that SSF cancels the request for an **EventNotificationCharging** and **does** not send it to SCF when the calling party (SigConA) triggers the charging event.
Postamble:	SigConA_Release_thenB


MSC IN2m_A_BASIC_CA_BV_02
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
2, 51, 2, RNC, short, rNCArg : { { eventTypeCharging 'AAAA'H, monitorMode interrupted } }
TC_InvokeReq
3, 51, 4, CUE, medium, cUEArg : Null
TC_ContinueReq
51, oSCF
SetupReq
{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
SetupConf
{ callRef 2 }
SetupResp
{ callRef 1 }
ChargingEventInd
{callRef 2, eventTypeCharging 'AAAA'H}
TC_ContinueInd
51
TC_InvokeInd
102, 51, ENC, TRUE, eNCArg : { eventTypeCharging 'AAAA'H, monitorMode interrupted }
TC_InvokeReq
4, 51, 2, RNC, short, rNCArg : { { eventTypeCharging 'CCCC'H, monitorMode interrupted } }
TC_ContinueReq
51, oSCF
TC_InvokeReq
5, 51, 2, CAN, short, cANArg : allRequests : Null
TC_ContinueReq
51, oSCF
ChargingEventInd
{callRef 2, eventTypeCharging 'CCCC'H}
SigConA_Release_thenB

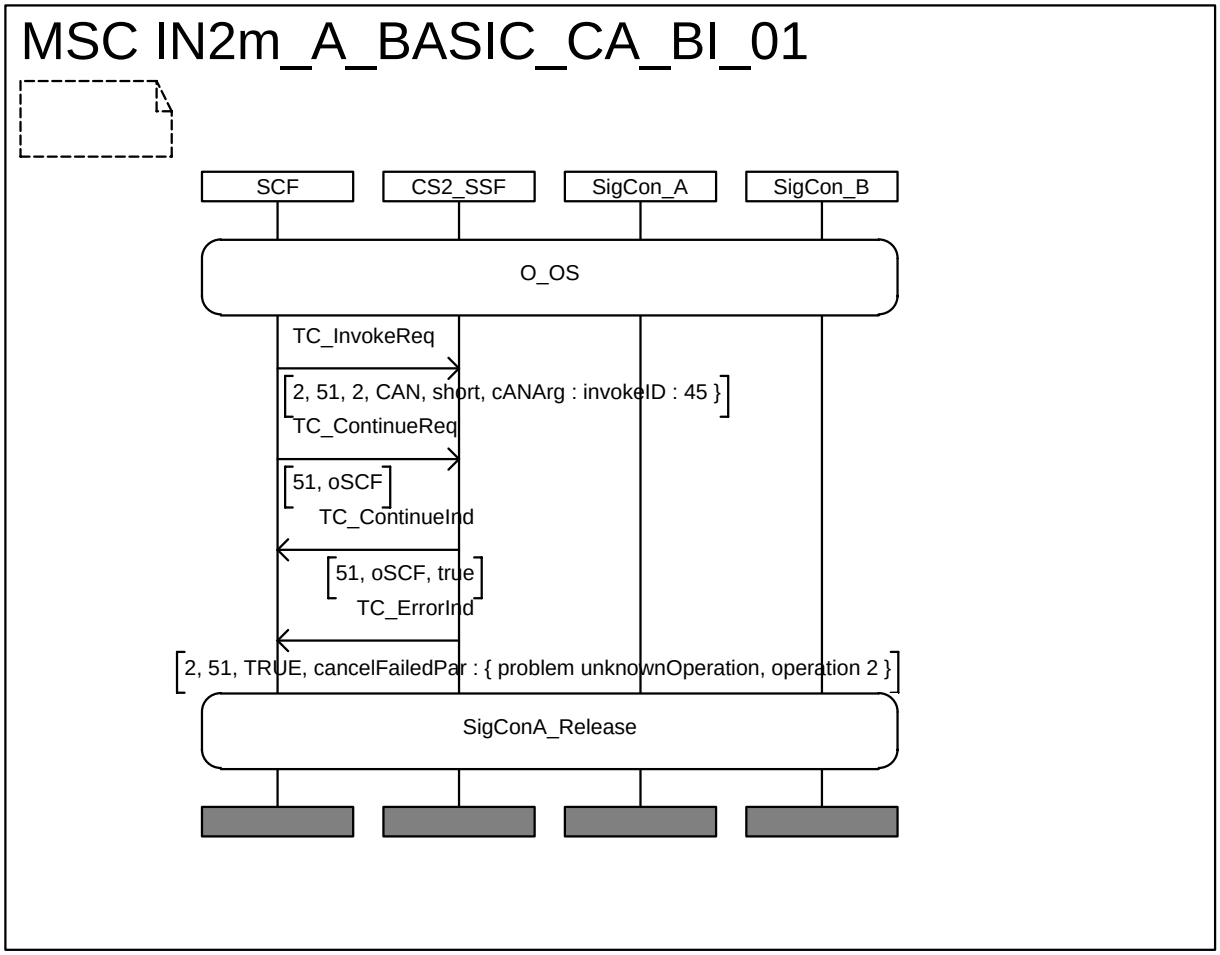
IN2_A_BASIC_CA_BV_03	
Purpose:	Test of **Cancel** procedure on ApplyCharging
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **ApplyCharging** invoke containing mandatory parameter aChBillingChargingCharacteristics followed by a Connect to establish a Connection with SigConB When the connection is established, **Cancel** invoke is sent by SCF to SSF, containing allRequests
Pass criteria	- Check that SSF cancels the request for an **ApplyChargingReport** and **does** not send it to SCF when the calling party (SigConA) releases the call.
Postamble:	none


MSC IN2_A_BASIC_CA_BV_03
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '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 }
SetupConf
{ callRef 2 }
SetupResp
{ callRef 1 }
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

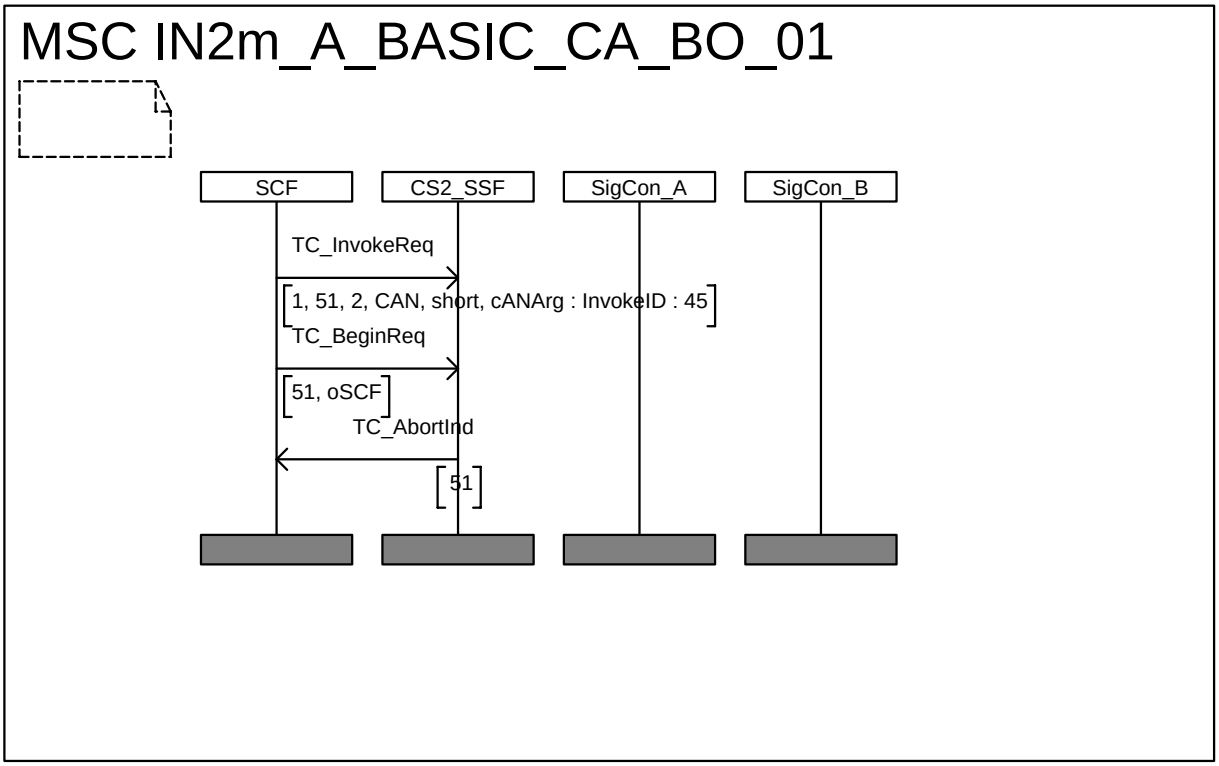
IN2_A_BASIC_CA_BV_04	
Purpose:	Test of **Cancel** procedure on CallInformationRequest
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **CallInformationRequest** invoke containing mandatory parameters requestedInformationTypeList including: re requestedInformationType (callAttemptElapsedTime), followed by a Connect to establish a Connection with SigConB When the connection is established, **Cancel** invoke is sent by SCF to SSF, containing invokeID
Pass criteria	- Check that SSF cancels the request for an **CallInformationReport** and **does** not send it to SCF when the calling party (SigConA) releases the call.
Postamble:	SigConA_Release_thenB_cause10


MSC IN2_A_BASIC_CA_BV_04
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
[1, 51, 2, CIRQ, short, cIRQArg : { requestedInformationTypeList { callAttemptElapsedTime } }]
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 }]
TC_InvokeReq
[3, 51, 4, CAN, short, cANArg : InvokeID : 1]
TC_EndReq
[51, basic]
SigConA_Release_thenB_cause10

IN2_A_BASIC_CA_BI_01	
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 invoke Id
Pass criteria	- Check that SSF sends to SCF **Cancel** with error cancelFailed
Postamble:	SigConA_Release

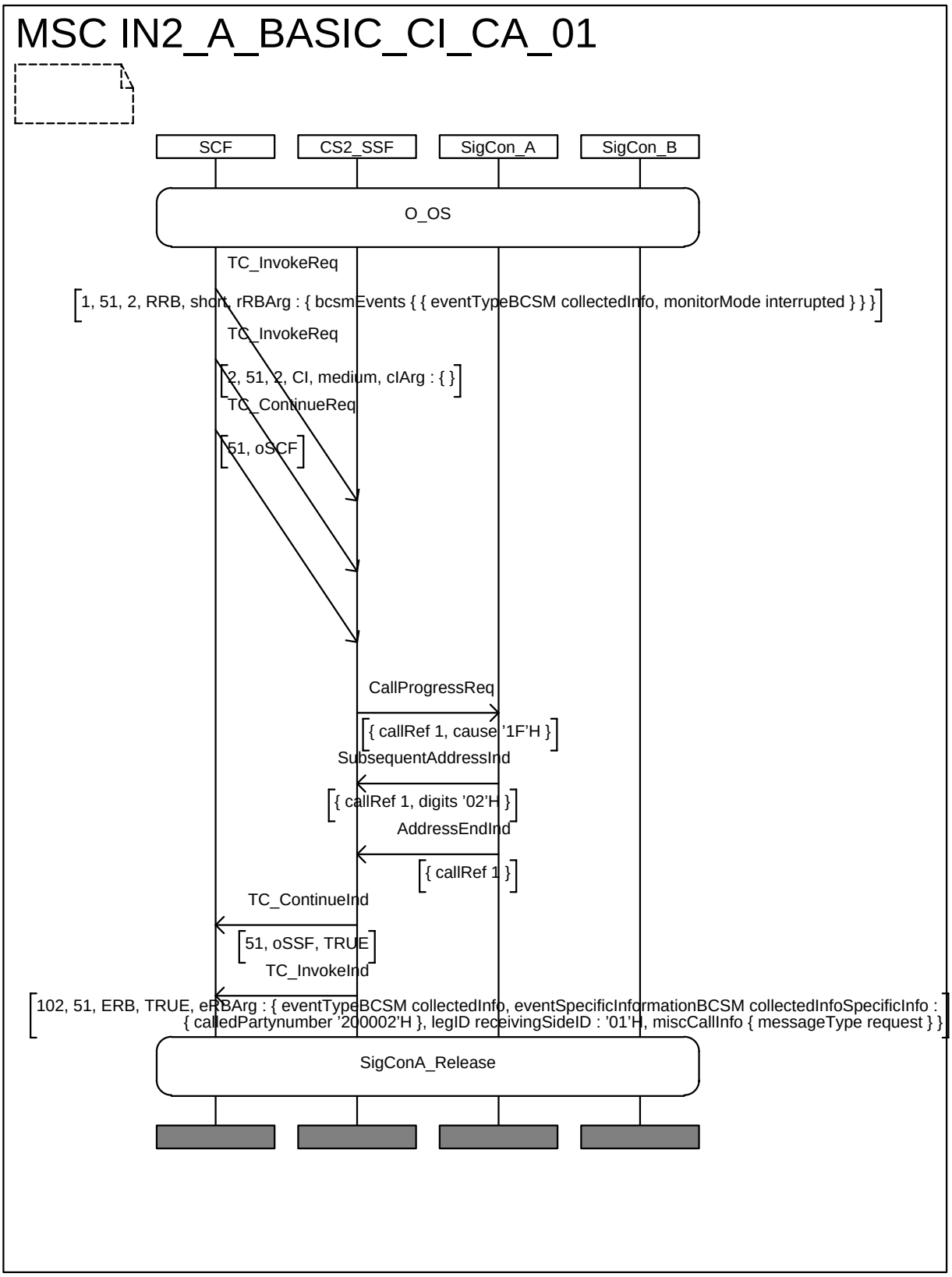


IN2_A_BASIC_CA_BO_01	
Purpose:	Test of **Cancel** procedure in wrong (idle) state
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	**Cancel** invoke sent by SCF to SSF, containing invokeID
Pass criteria	- Check that SSFsends a **TC-ABORT**
Postamble:	none

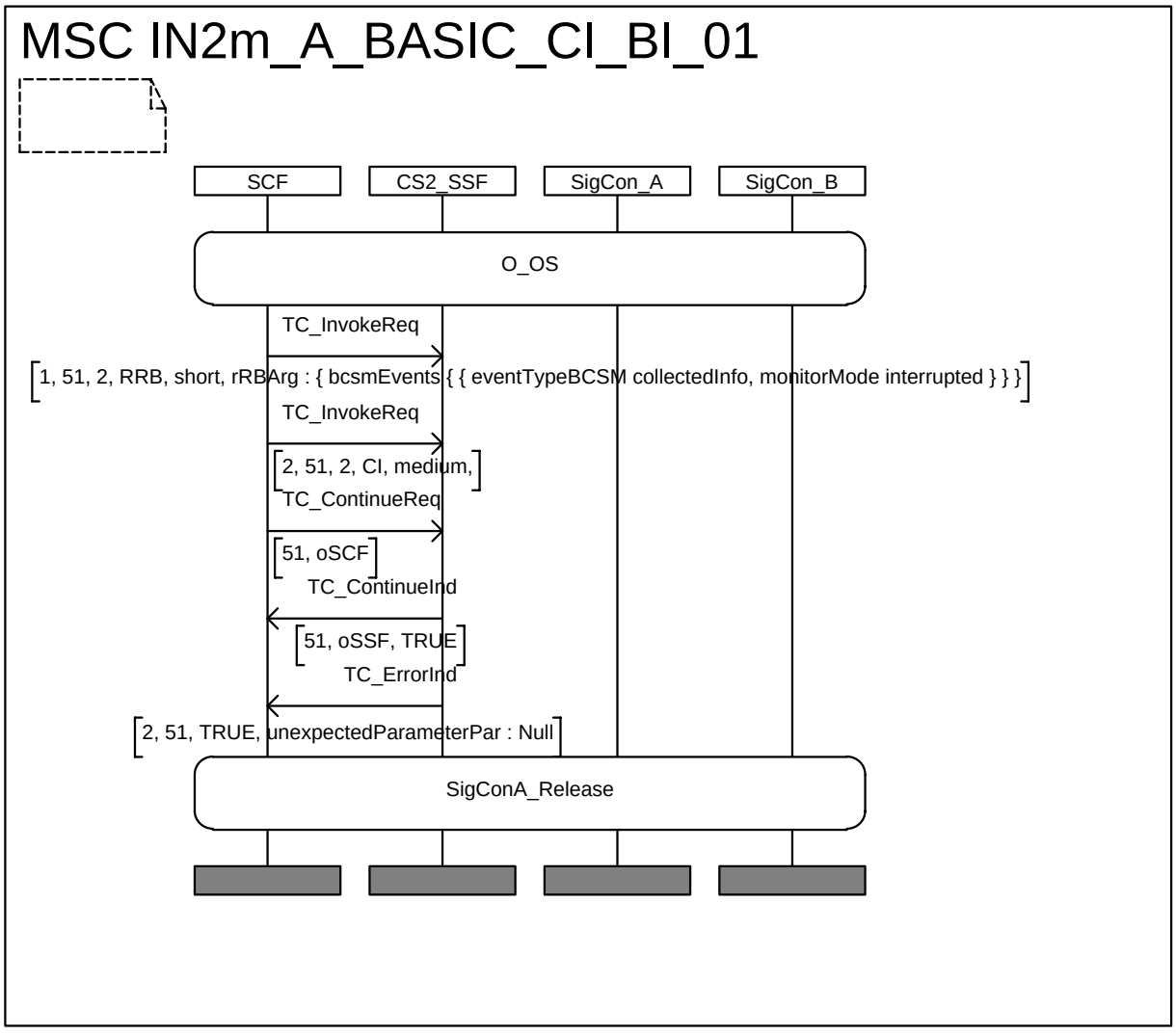


6.4.7 CollectInformation procedure

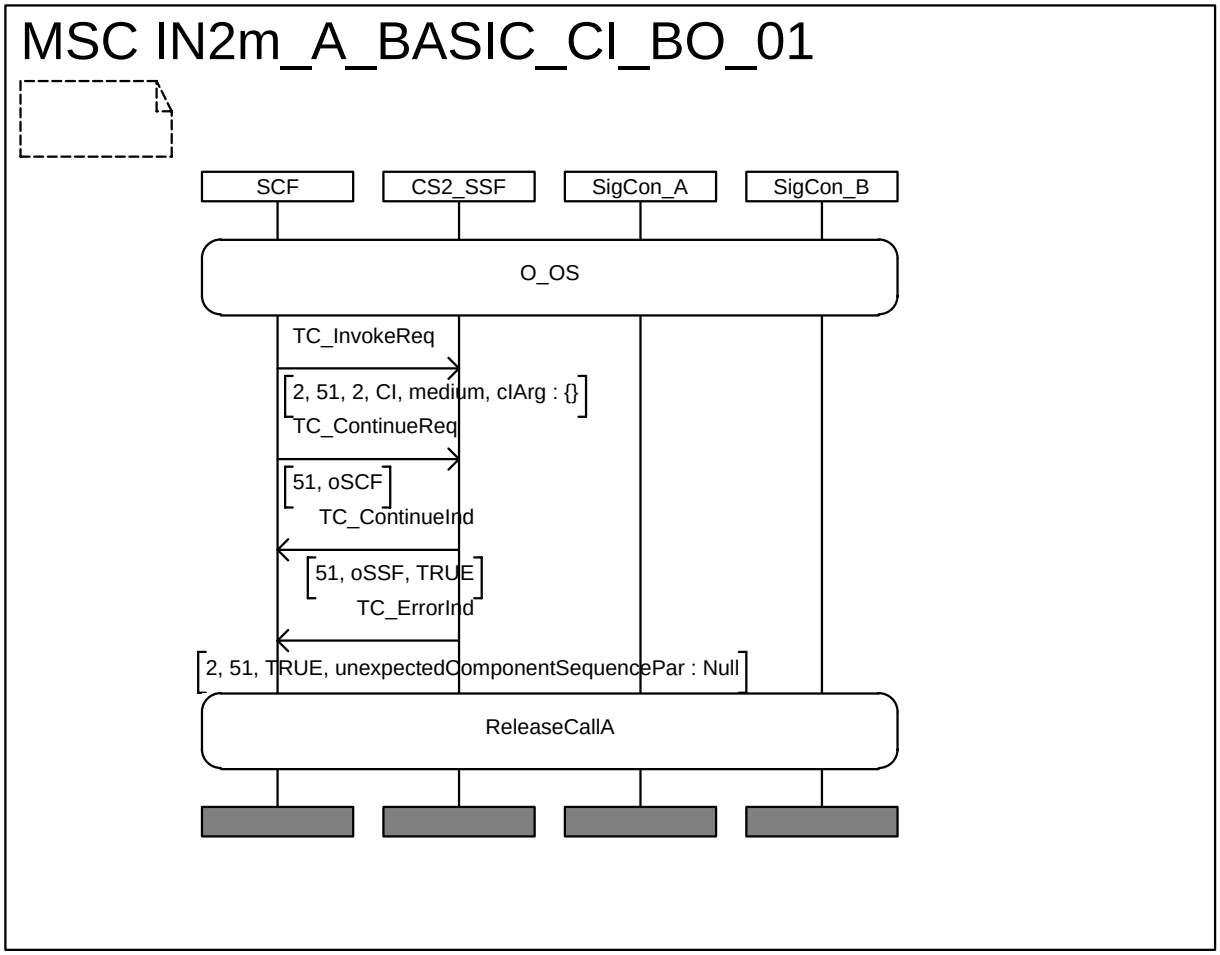
IN2_A_BASIC_CI_CA_01	
Purpose:	Test of **CollectInformation** base procedure
Requirement ref	
Selection Cond.	
Preamble:	O_OS Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=collectedInfo - 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


MSC IN2_A_BASIC_CI_CA_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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

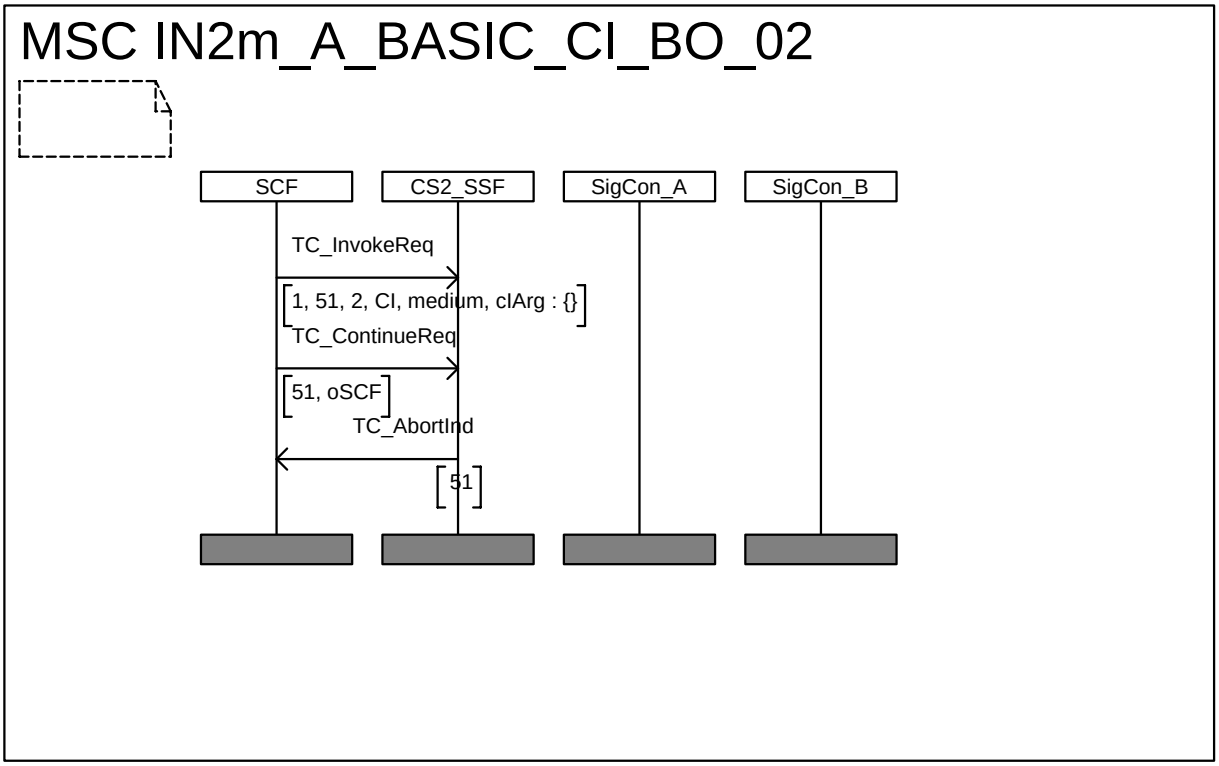
IN2_A_BASIC_CI_BI_01	
Purpose:	Test of **CollectInformation** procedure and unknown CallsegmentId
Requirement ref	
Selection Cond.	CS-2 only
Preamble:	O_OS
Test description	SCF - SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= collectedInfo - monitoringMode=interrupted 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



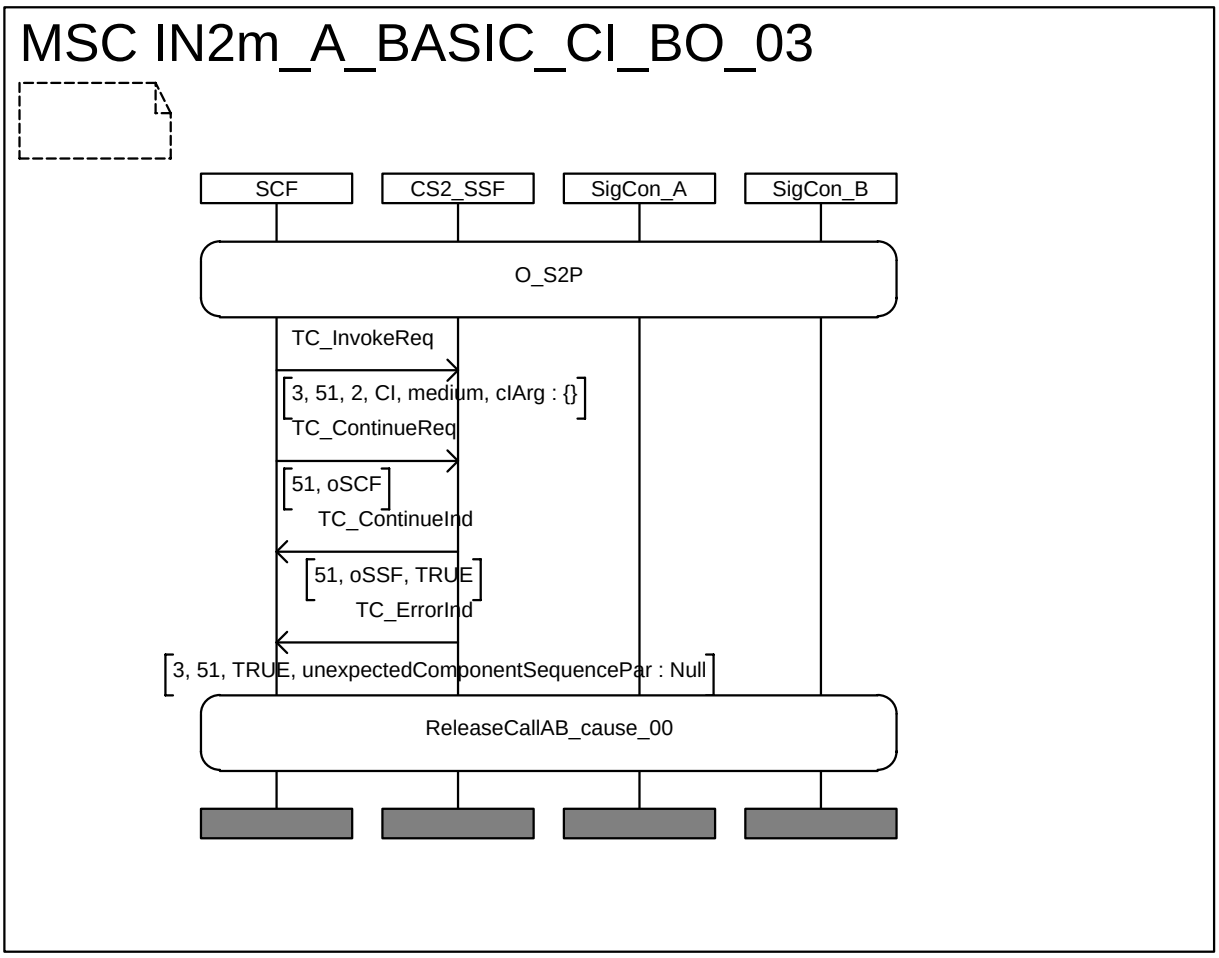
IN2_A_BASIC_CI_BO_01	
Purpose:	Test of **CollectInformation** procedure in wrong sequence
Requirement ref	
Selection Cond.	
Preamble:	O_OS Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF sends **CollectInformation** invoke to SSF without sending before any **RequestReportBCSMEvent** invoke
Pass criteria	Check that SSF sends to SCF a **CollectInformation** error with an indication of UnexpectedComponentSequence
Postamble:	ReleaseCallA



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

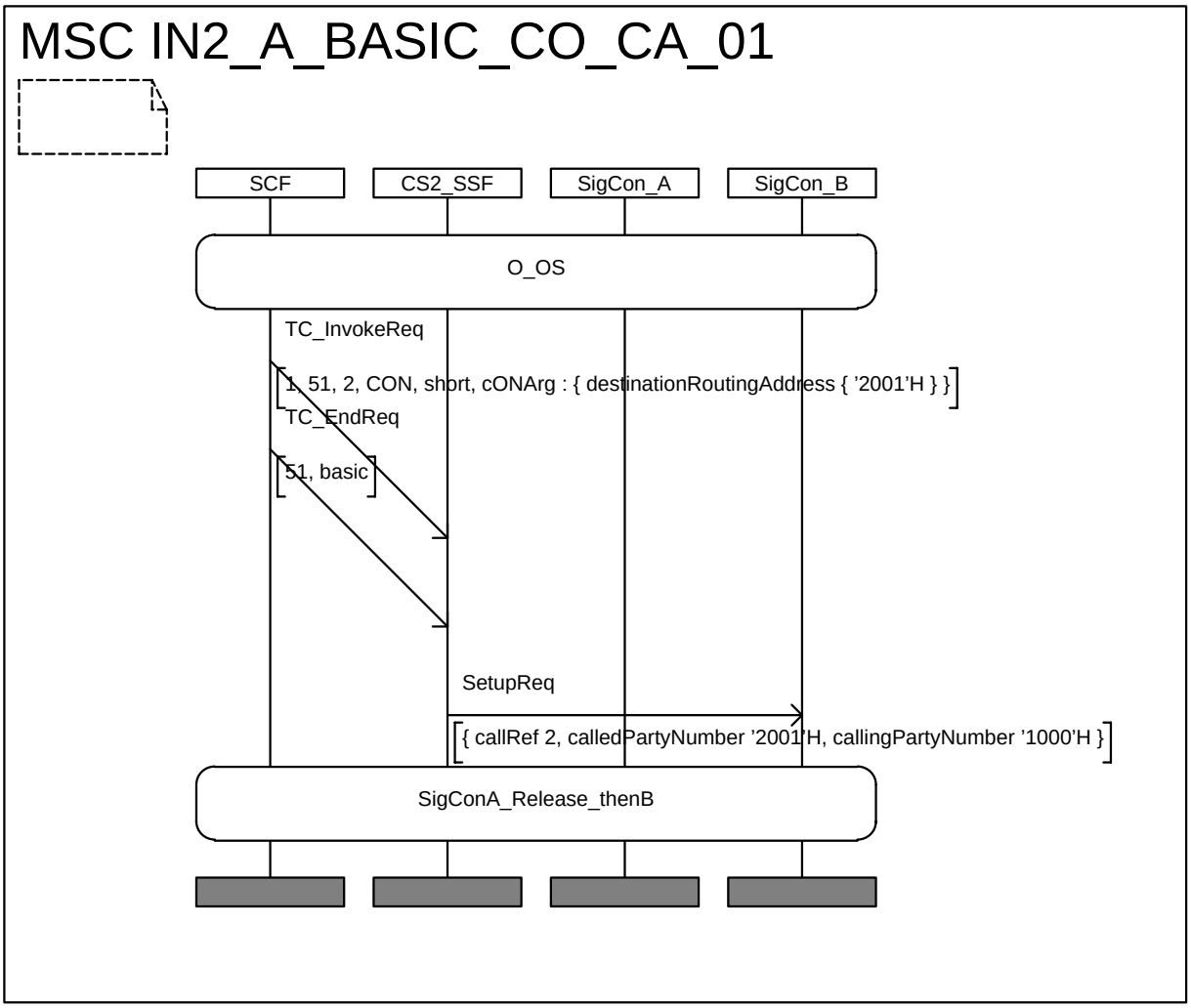


IN2_A_BASIC_CI_BO_03	
Purpose:	Test of **CollectInformation** procedure in wrong state (monitoring state)
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
Test description	SCF sends **CollectInformation** invoke to SSF from Monitoring state
Pass criteria	Check that SSF sends to SCF a **CollectInformation** error with an indication of UnexpectedComponentSequence
Postamble:	ReleaseCallAB_cause_00

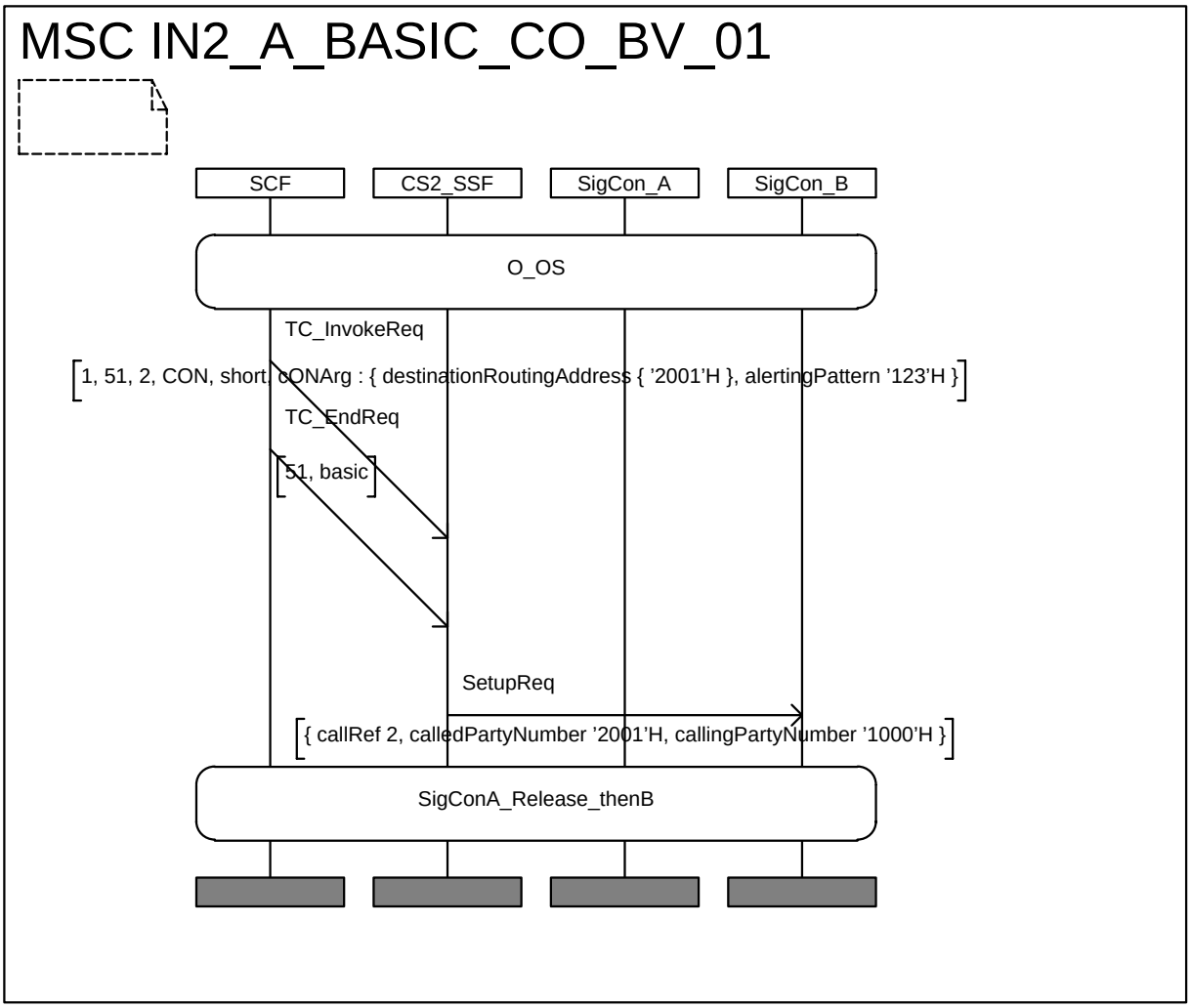


6.4.8 Connect procedure

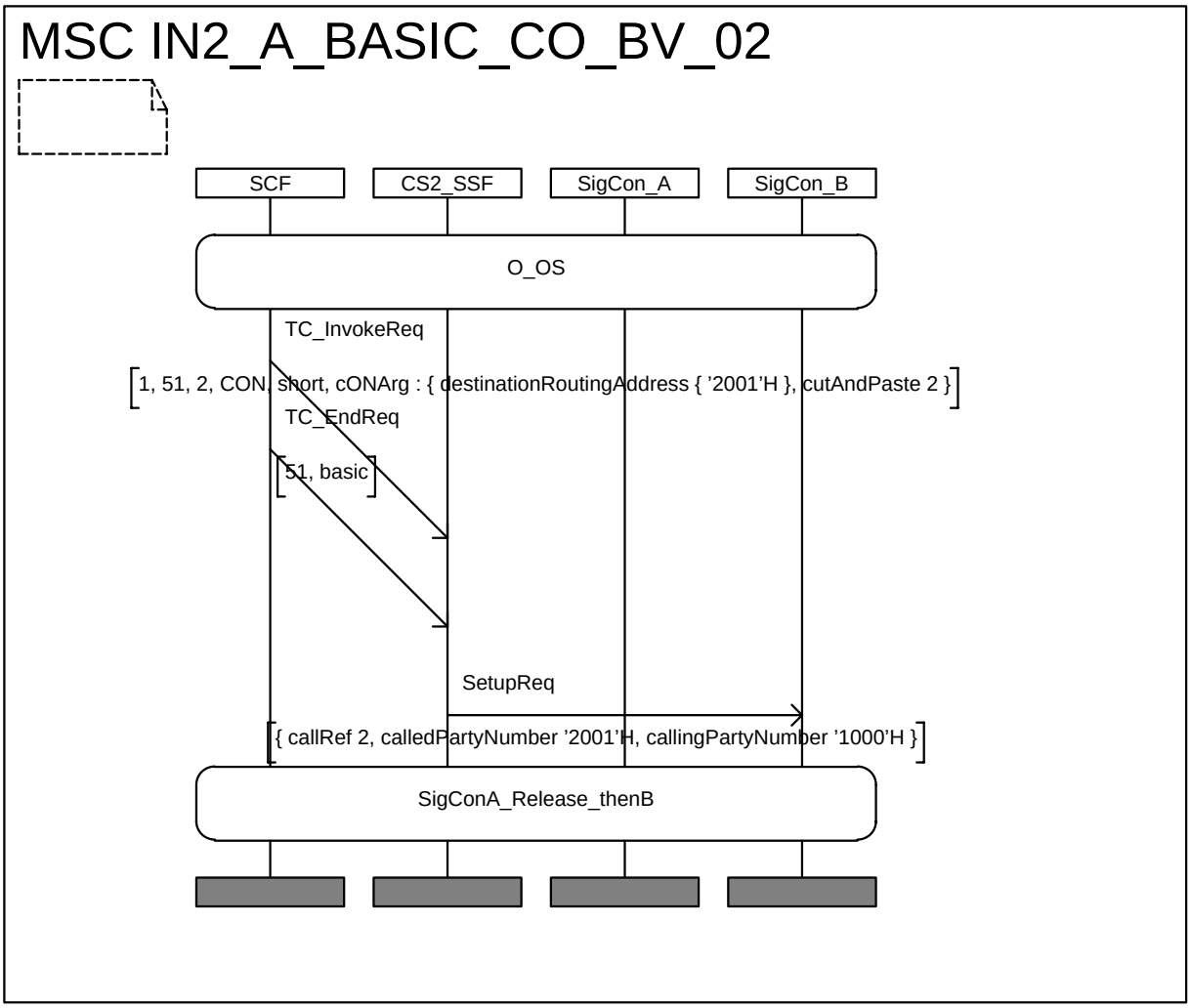
IN2_A_BASIC_CO_CA_01	
Purpose:	Test of **Connect** base procedure
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Connect** invoke with mandatory parameters only, with destinationRoutingAddress SSF sends a SetupRequest to B side
Pass criteria	Check that the relevant parameters are mapped from Connect into the **Setup** request
Postamble:	SigConA_Release_thenB



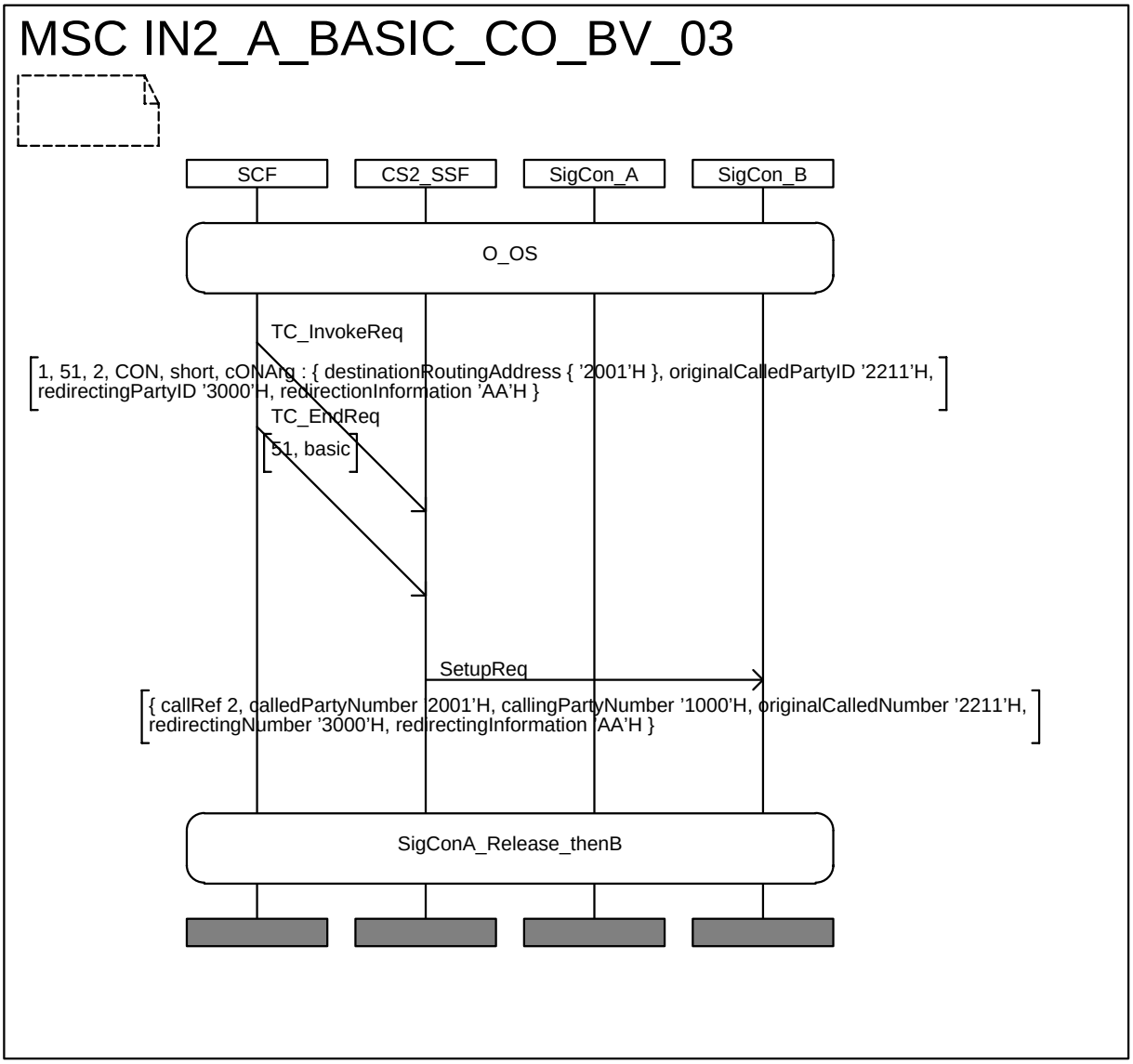
	IN2_A_BASIC_CO_BV_01
Purpose:	Test of **Connect** procedure with alertingPattern parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Connect** invoke with mandatory and optional parameters, with destinationRoutingAddress alertingPattern SSF sends a SetupRequest to B side
Pass criteria	Check that the connect operation is not rejected
Postamble:	SigConA_Release_thenB



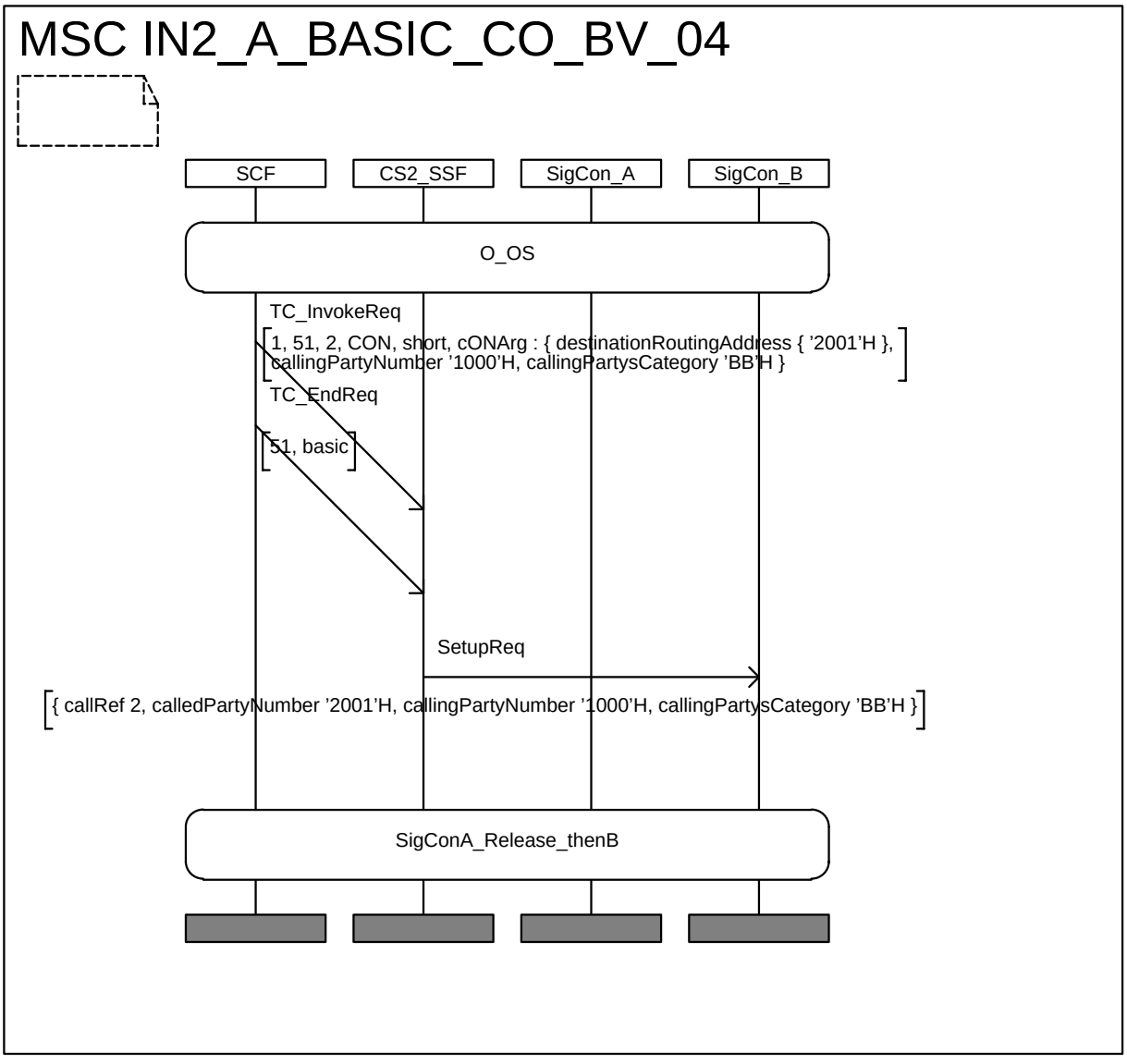
	IN2_A_BASIC_CO_BV_02
Purpose:	Test of **Connect** procedure with cutAndPaste parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Connect** invoke with mandatory and optional parameters, with destinationRoutingAddress cutAndPaste SSF sends a SetupRequest to B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



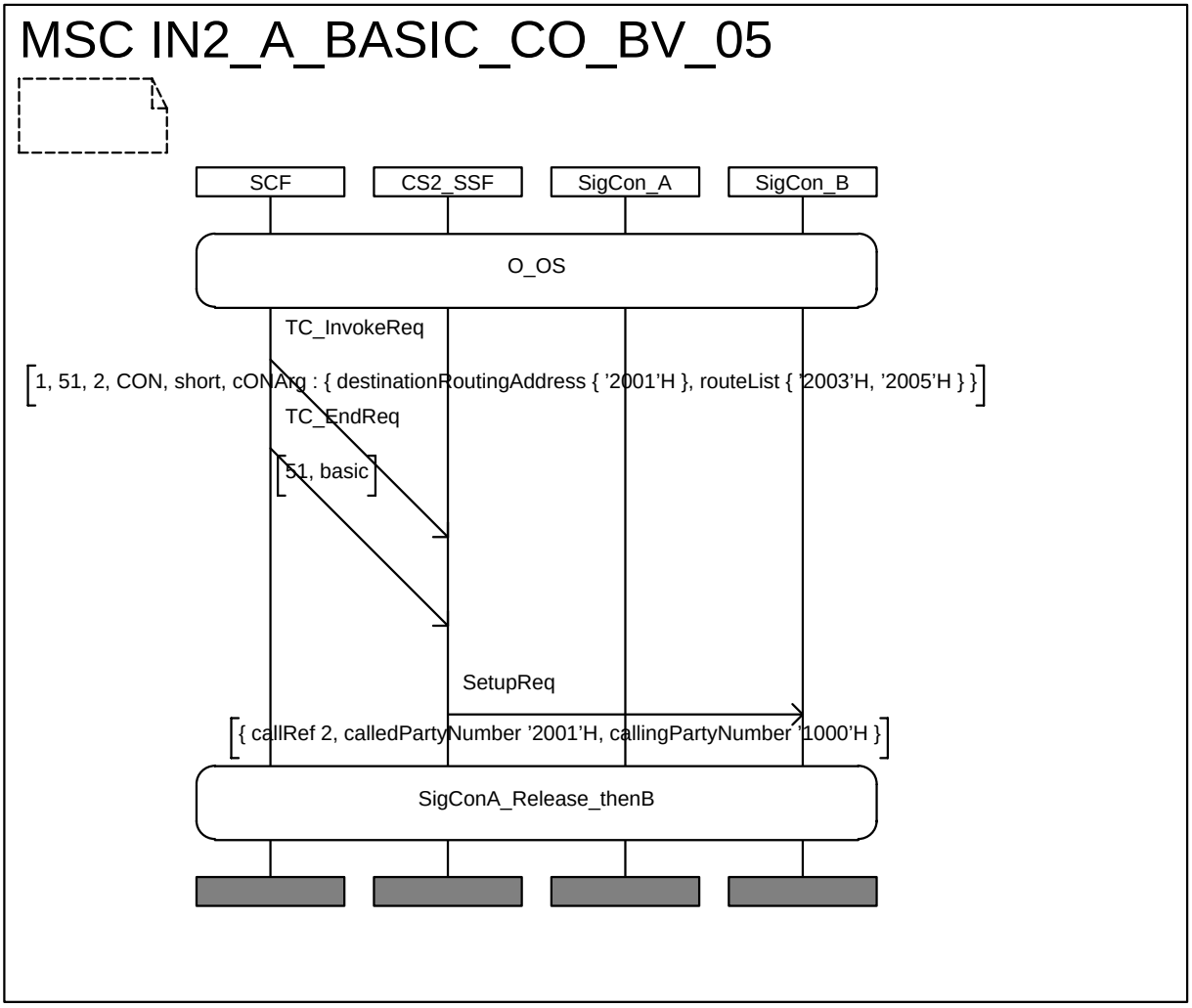
	IN2_A_BASIC_CO_BV_03
Purpose:	Test of **Connect** procedure with supplementary services parameters
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Connect** invoke containing parameters related to supplementary services, with: - destinationRoutingAddress, - originalCalledPartyID, - redirectingPartyID, - redirectionInformation SSF sends a SetupRequest to B side
Pass criteria	Check that the above parameters are mapped from Connect into the Setup request: - destinationRoutingAddress ----------> calledPartyNumber - originalCalledPartyID----------> originalCalledNumber - redirectingPartyID----------> redirectingNumber - redirectionInformation----------> redirectionInformation
Postamble:	SigConA_Release_thenB


MSC IN2_A_BASIC_CO_BV_03
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H }, originalCalledPartyID '2211'H, redirectingPartyID '3000'H, redirectionInformation 'AA'H }
TC_EndReq
51, basic
SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H, originalCalledNumber '2211'H, redirectingNumber '3000'H, redirectingInformation 'AA'H }
SigConA_Release_thenB

IN2_A_BASIC_CO_BV_04	
Purpose:	Test of **Connect** procedure with optional parameters related to the calling party
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF 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 Setup request: - destinationRoutingAddress----------> calledPartyNumber - callingPartyNumber----------> callingPartyNumber - callingPartysCategory----------> callingPartysCategory
Postamble:	SigConA_Release_thenB

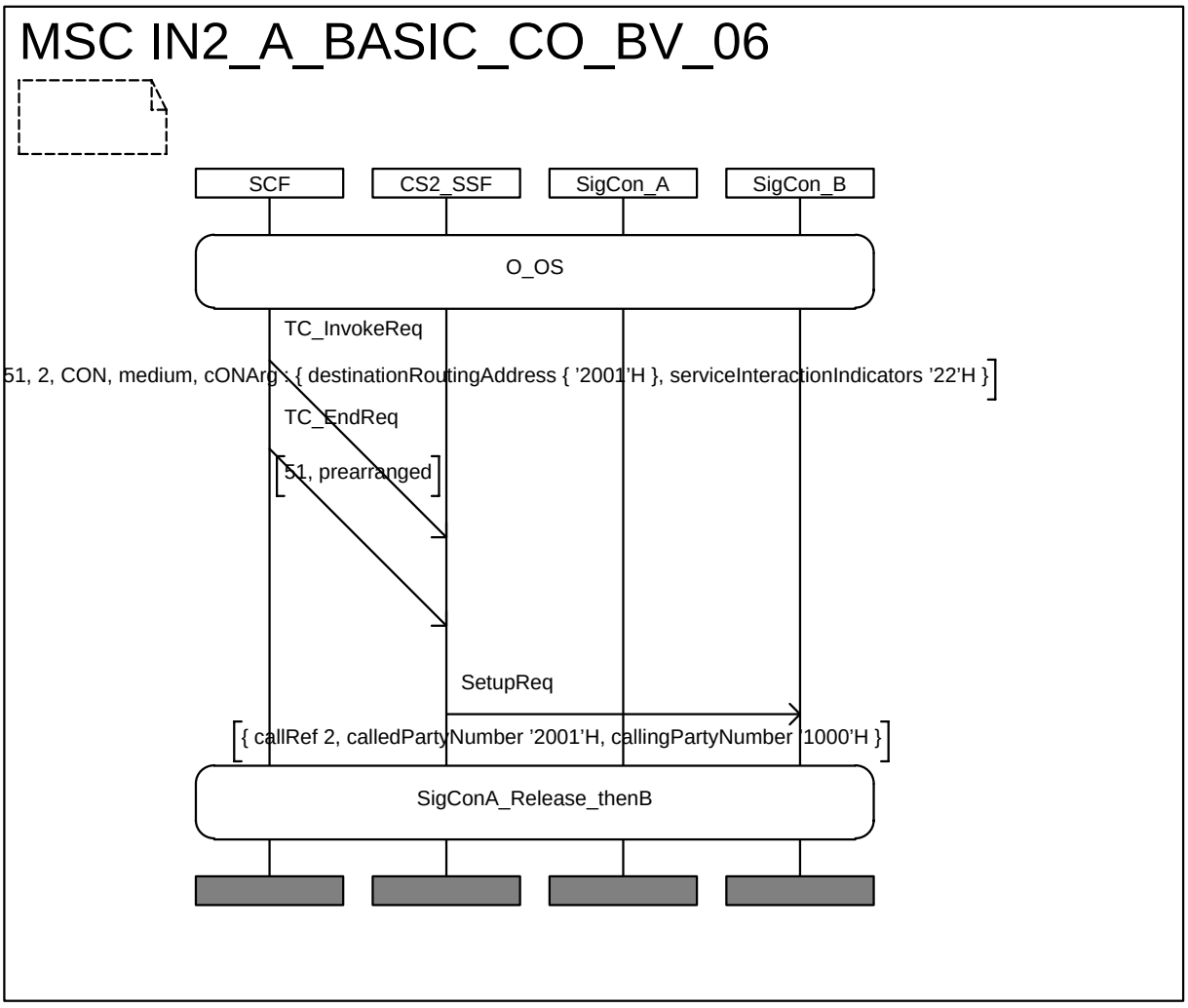


XXXX	IN2_A_BASIC_CO_BV_05
Purpose:	Test of **Connect** procedure with optional parameter routeList
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **Connect** invoke containing mandatory and optional parameters - destinationRoutingAddress, - routeList (with two 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

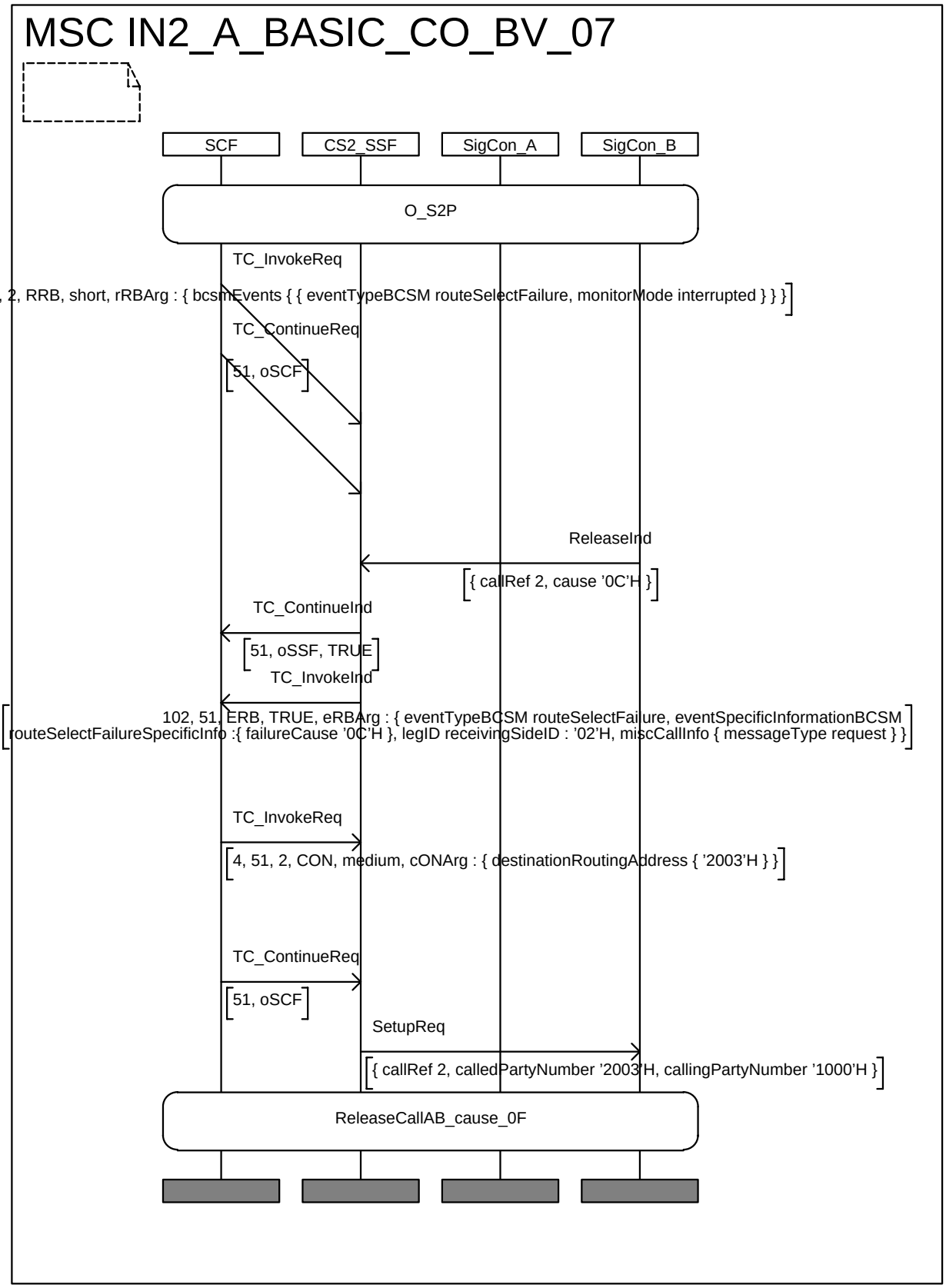


XXXX	IN2_A_BASIC_CO_BV_06
Purpose:	Test of **Connect** procedure with optional parameter serviceInteractionIndicators

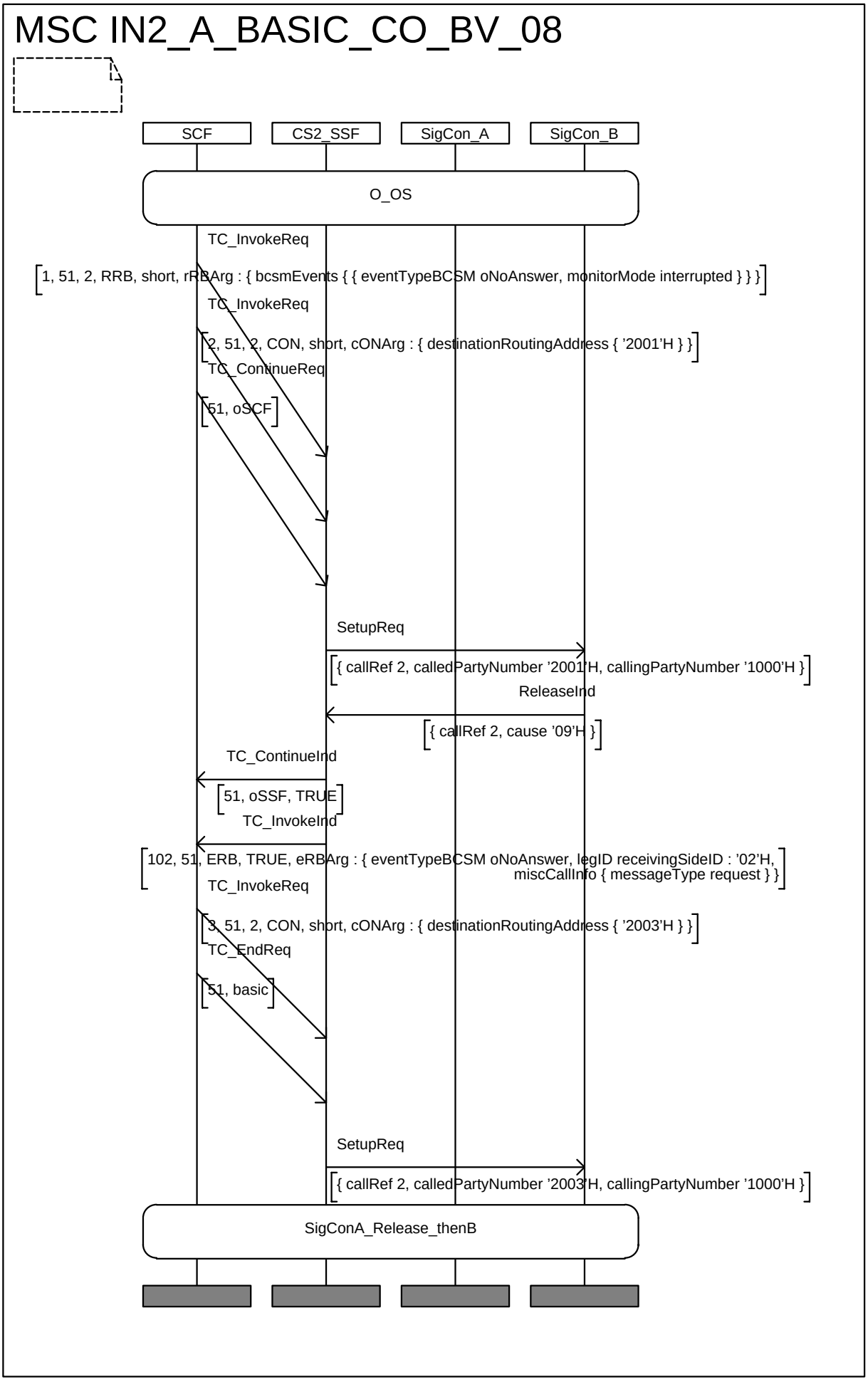
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **Connect** invoke containing mandatory and optional parameters - destinationRoutingAddress, - serviceInteractionIndicators SSF sends a SetupRequest to B side
Pass criteria	Check that the above parameters are mapped from Connect into the Setup request: - destinationRoutingAddress----------> calledPartyNumber, - serviceInteractionIndicators----------> serviceIndicators
Postamble:	SigConA_Release_thenB



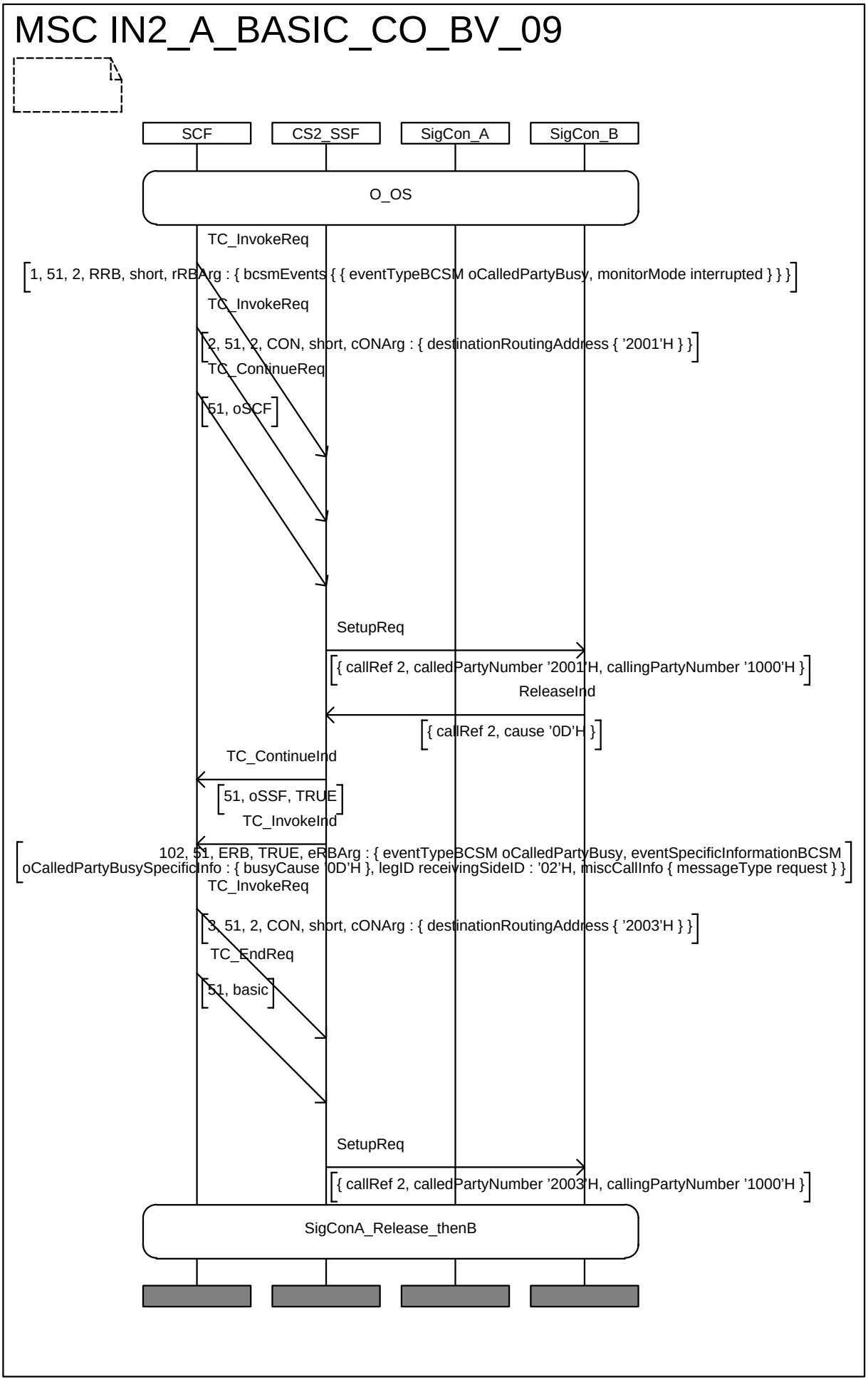
XXXX	IN2_A_BASIC_CO_BV_07
Purpose:	Test of **Connect** procedure received on each allowed DP of O_BCSM
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
Test description	SCF sends a RRB operation to arm routeSelectFailure event SigConB sends a ReleaseInd with release cause being routeSelectFailure SCF 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


MSC IN2_A_BASIC_CO_BV_07
SCF
CS2_SSF
SigCon_A
SigCon_B
O_S2P
TC_InvokeReq
[3, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode interrupted } } }]
TC_ContinueReq
[51, oSCF]
ReleaseInd
[{ callRef 2, cause '0C'H }]
TC_ContinueInd
[51, oSSF, TRUE]
TC_InvokeInd
[102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM routeSelectFailure, eventSpecificInformationBCSM routeSelectFailureSpecificInfo :{ failureCause '0C'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }]
TC_InvokeReq
[4, 51, 2, CON, medium, cONArg : { destinationRoutingAddress { '2003'H } }]
TC_ContinueReq
[51, oSCF]
SetupReq
[{ callRef 2, calledPartyNumber '2003'H, callingPartyNumber '1000'H }]
ReleaseCallAB_cause_0F

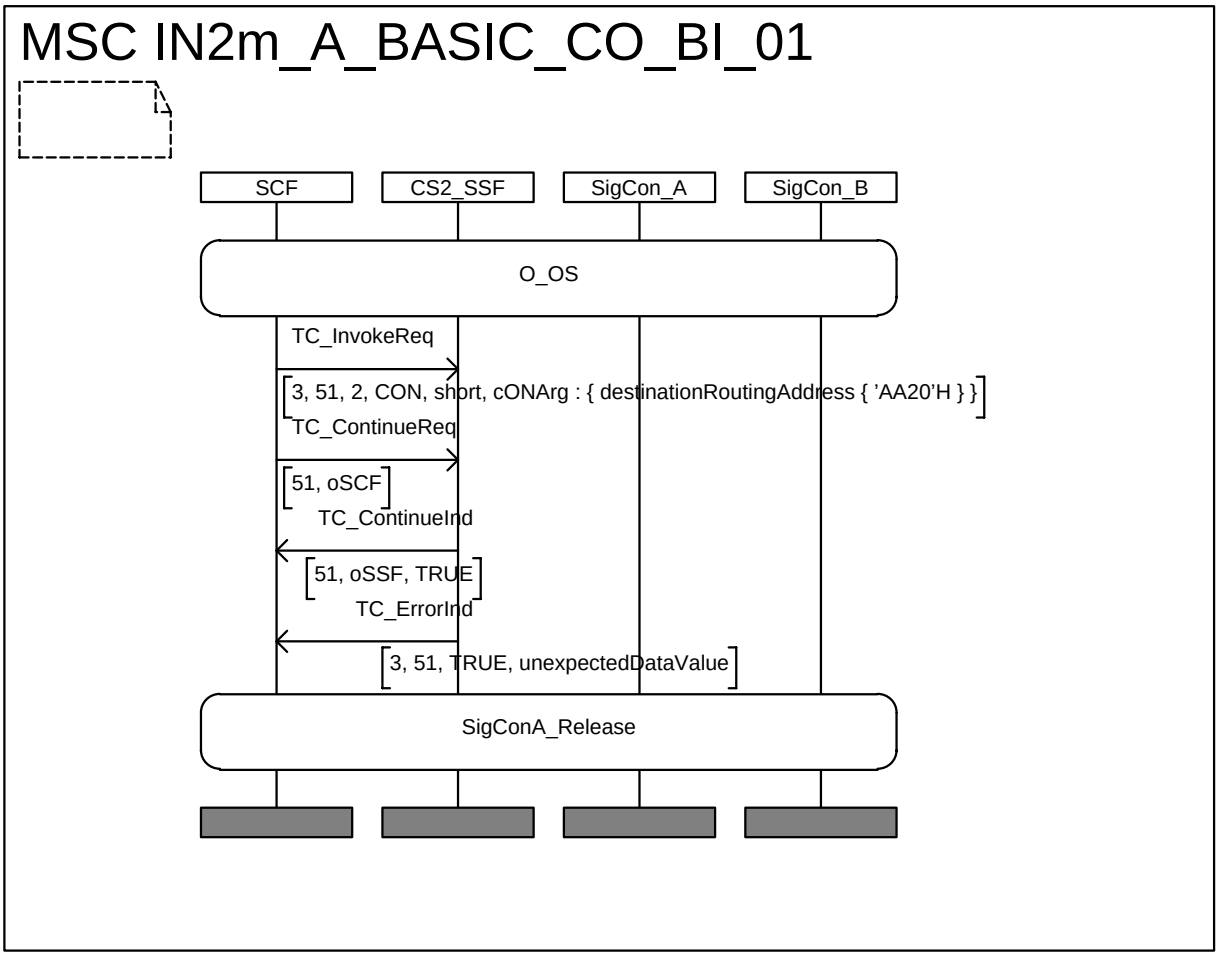
XXXX	IN2_A_BASIC_CO_BV_08
Purpose:	Test of **Connect** procedure in response to oNoAnswer DP.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends a RRB operation to arm oNoAnswer event SCF sends a Connect operation with mandatory parameters SigConB sends a ReleaseInd with release cause being bPtyNoAnswer Scs SCF sends to SSF **Connect** invoke containing mandatory parameter - destinationRoutingAddress,
Pass criteria	Check that SSF sends a SetupReq to B side
Postamble:	SigConA_Release_thenB


MSC IN2_A_BASIC_CO_BV_08
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 } }
TC_InvokeReq
3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2003'H } }
TC_EndReq
51, basic
SetupReq
{ callRef 2, calledPartyNumber '2003'H, callingPartyNumber '1000'H }
SigConA_Release_thenB

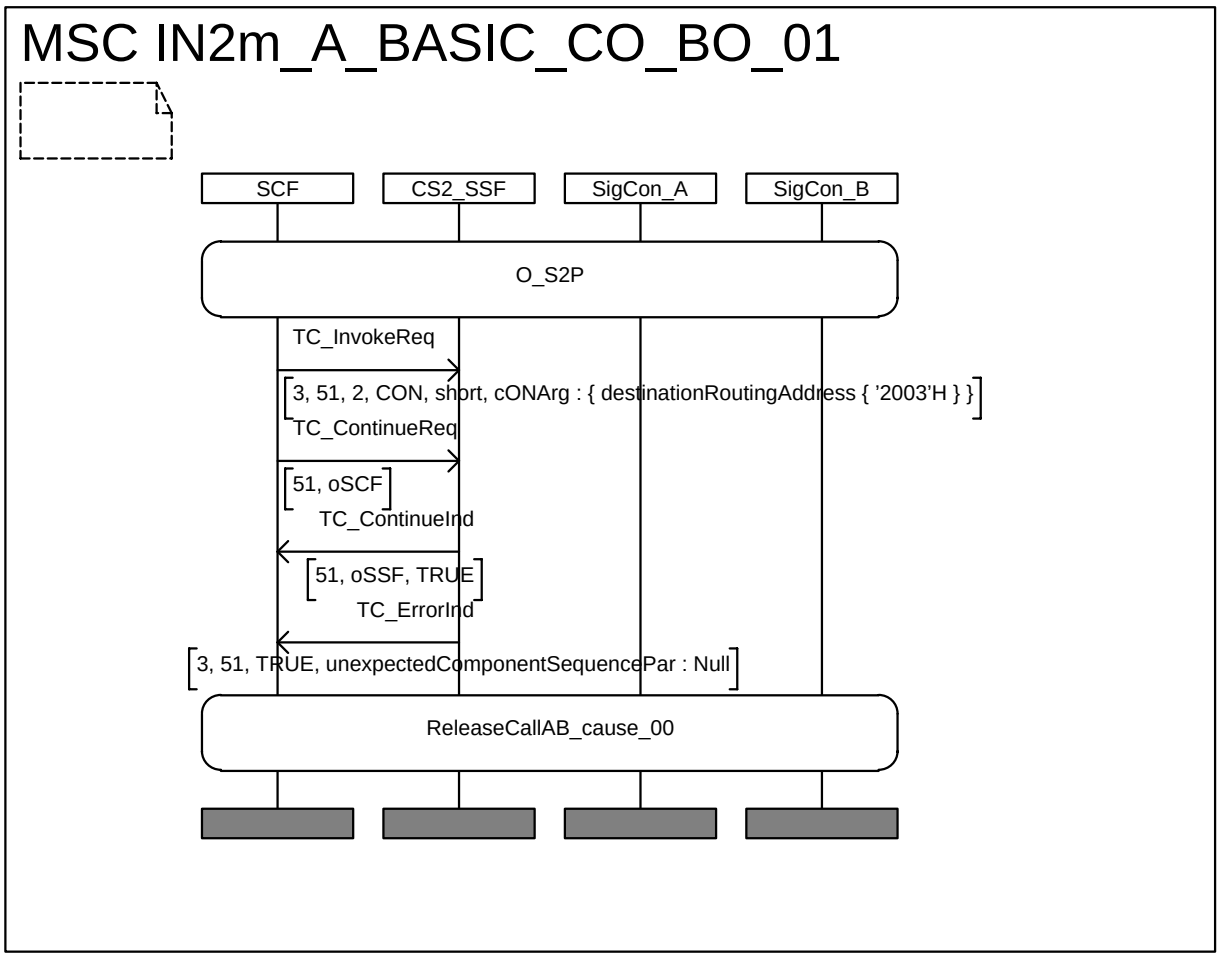
XXXX	IN2_A_BASIC_CO_BV_09
Purpose:	Test of **Connect** procedure in response to oCalledPartyBusy DP.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends a RRB operation to arm oCalledPartyBusy event SC SCF sends a Connect operation with mandatory parameters SigConB sends a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


MSC IN2_A_BASIC_CO_BV_09
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oCalledPartyBusy, eventSpecificInformationBCSM oCalledPartyBusySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }
TC_InvokeReq
3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2003'H } }
TC_EndReq
51, basic
SetupReq
{ callRef 2, calledPartyNumber '2003'H, callingPartyNumber '1000'H }
SigConA_Release_thenB

IN2_A_BASIC_CO_BI_01	
Purpose:	Test of **Connect** procedure with invalid parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF 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

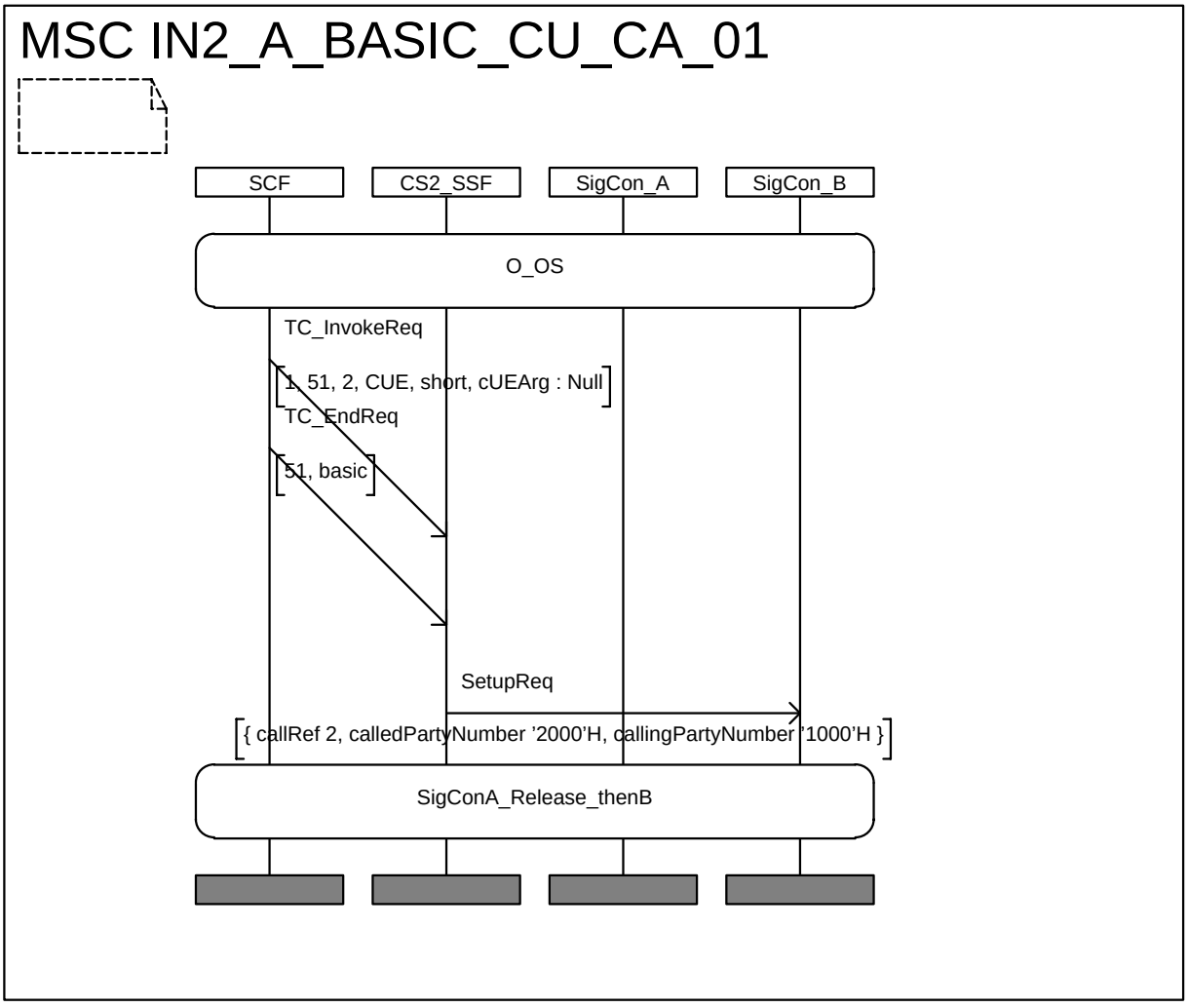


IN2_A_BASIC_CO_BO_01	
Purpose:	Test of **Connect** procedure initiated from wrong state
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
Test description	SCF sends to SSF **Connect** invoke containing mandatory parameters, while in monitoring state
Pass criteria	Check that SSF sends back **Connect** error, with error parameter UnexpectedComponentSequence
Postamble:	ReleaseCallAB_cause_00

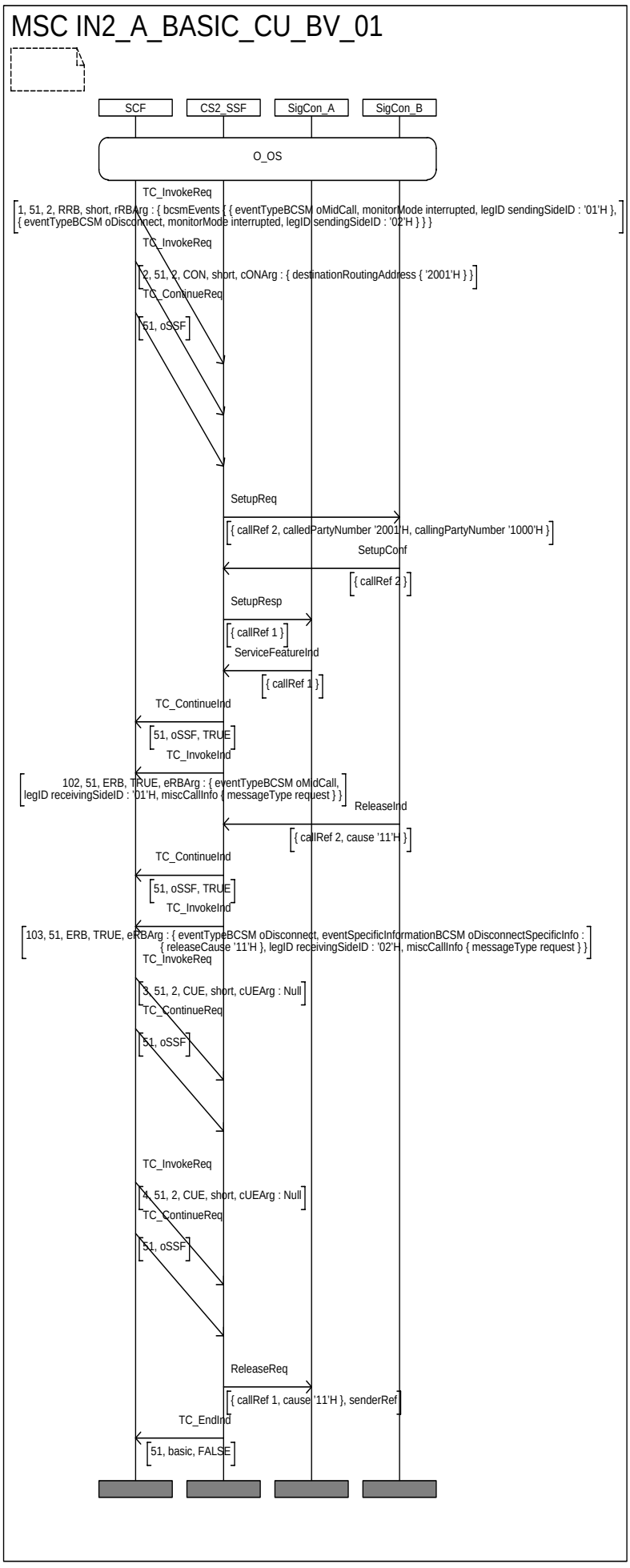


6.4.9 Continue procedure

IN2_A_BASIC_CU_CA_01	
Purpose:	Test of **Continue** procedure
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Continue** invoke
Pass criteria	Check that SSF continues call processing, i.e. SetupReq is detected at SigConB
Postamble:	SigConA_Release_thenB



IN2_A_BASIC_CU_BV_01	
Purpose:	Test of **Continue** procedure when 2 outstanding EDP-Rs are reported
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	1. SCF sends to SSF **RequestReportBCSMEvent** invoke with parameters (oMidCall, interrupted, leg1) and (oDisconnect, interrupted, leg2) 2. SCF sends to SSF **Connect** invoke with a called party number to establish a call to the B-party. 3. After SetupReq is detected at SigConB, the call is answered by SetupConf at SigConB 4. A party issues **ServiceFeatureInd** at SigConA 5. SSF sends to SCF **EventReportBCSMEvent** invoke with parameter (oMidCall, leg1) 6. B party issues **ReleaseInd** at SigConB 7. SSF sends to SCF **EventReportBCSMEvent** invoke with parameter (oDisconnect, leg2) 8. SCF sends to SSF **Continue** invoke (note that the release should not be continued) 9. SCF sends to SSF a 2nd **Continue** invoke
Pass criteria	Check that SSF sends **ReleaseReq** to SigConA
Postamble:	none

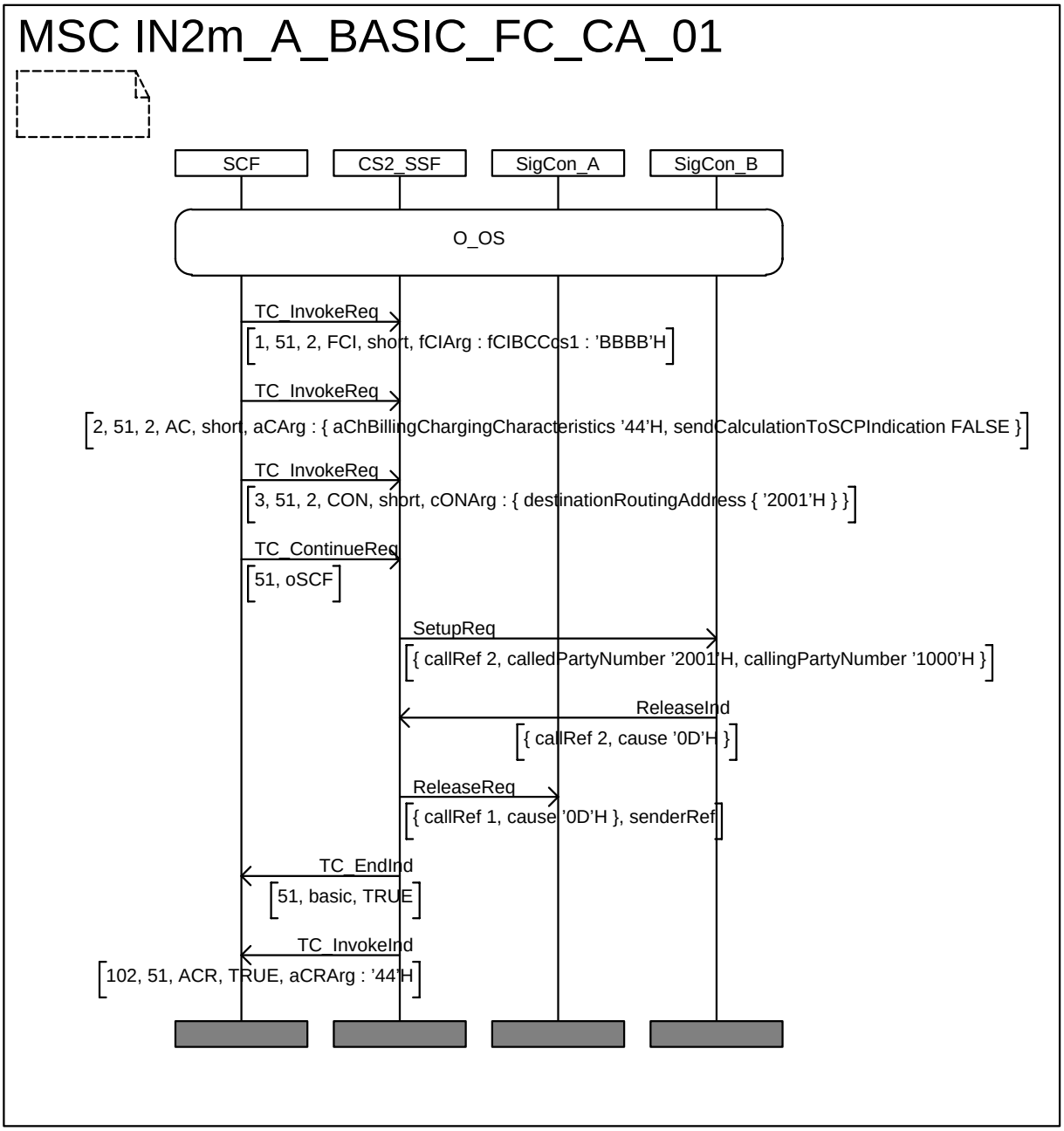

MSC IN2_A_BASIC_CU_BV_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oMidCall, monitorMode interrupted, legID sendingSideID : '01'H },
{ eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '02'H } } }
TC_InvokeReq
2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
TC_ContinueReq
51, oSSF
SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
SetupConf
{ callRef 2 }
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 } }
ReleaseInd
{ callRef 2, cause '11'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
103, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oDisconnect, eventSpecificInformationBCSM oDisconnectSpecificInfo :
{ releaseCause '11'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }
TC_InvokeReq
3, 51, 2, CUE, short, cUEArg : Null
TC_ContinueReq
51, oSSF
TC_InvokeReq
4, 51, 2, CUE, short, cUEArg : Null
TC_ContinueReq
51, oSSF
ReleaseReq
{ callRef 1, cause '11'H }, senderRef
TC_EndInd
51, basic, FALSE

6.4.10 FurnishChargingInformation procedure

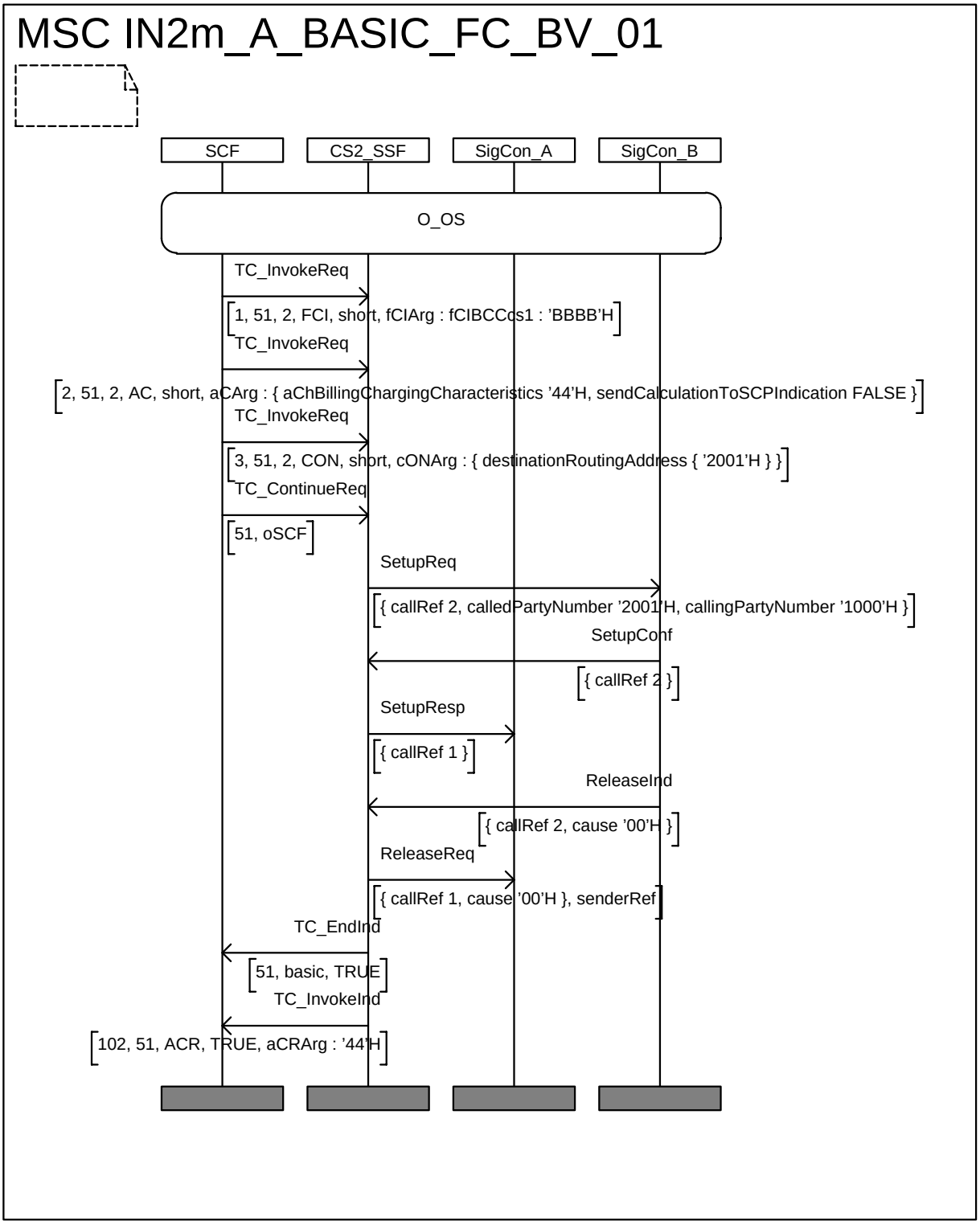
Charging related aspects in IN are network operator specific. Therefore, it is not possible to define useful test purposes for charging procedures using a network operator independant approach. TP specification has to be done by network operators, using INAP procedures themselves. TP could be specified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

Following some examples:

IN2_A_BASIC_FC_CA_01	
Purpose:	Test of **FurnishChargingInformation** procedure
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF 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

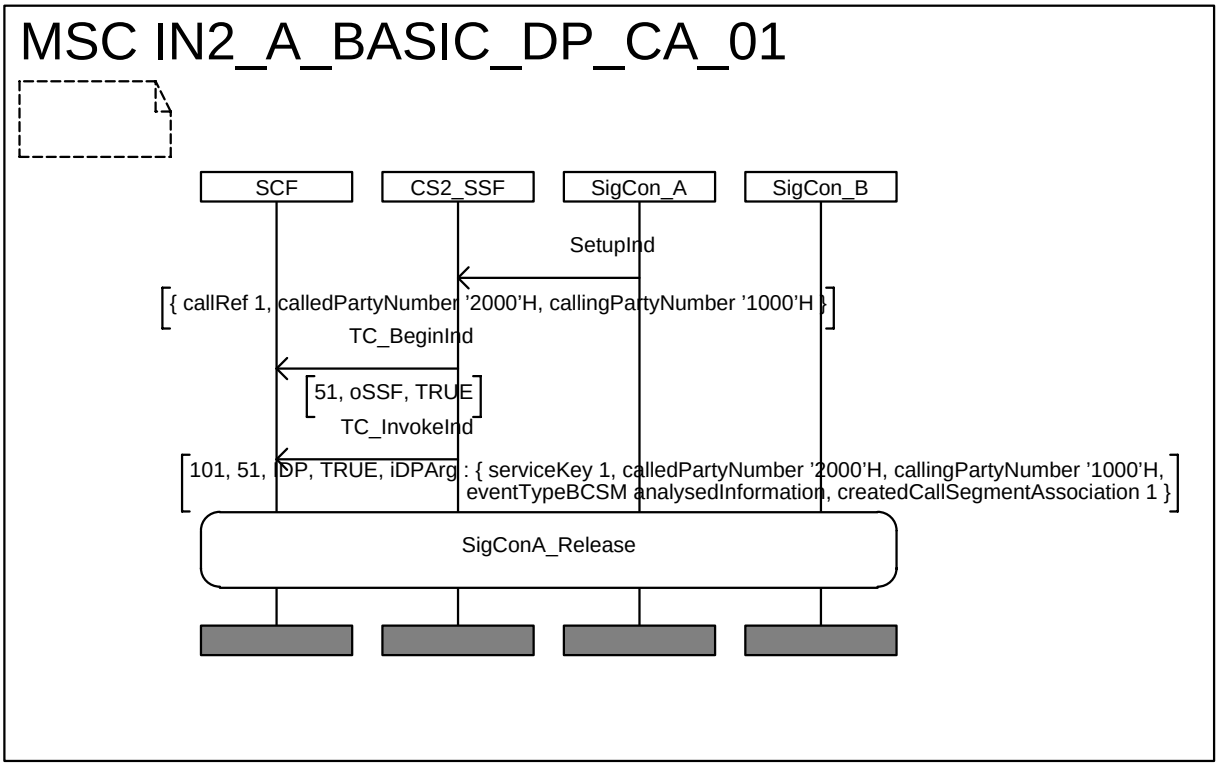

MSC IN2m_A_BASIC_FC_CA_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, FCI, short, fCIArg : fCIBCCcs1 : 'BBBB'H
TC_InvokeReq
2, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '44'H, sendCalculationToSCPIndication FALSE }
TC_InvokeReq
3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
TC_ContinueReq
51, oSCF
SetupReq
{ 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 : '44'H

IN2_A_BASIC_FC_BV_01	
Purpose:	Test of **FurnishChargingInformation** procedure
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF 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

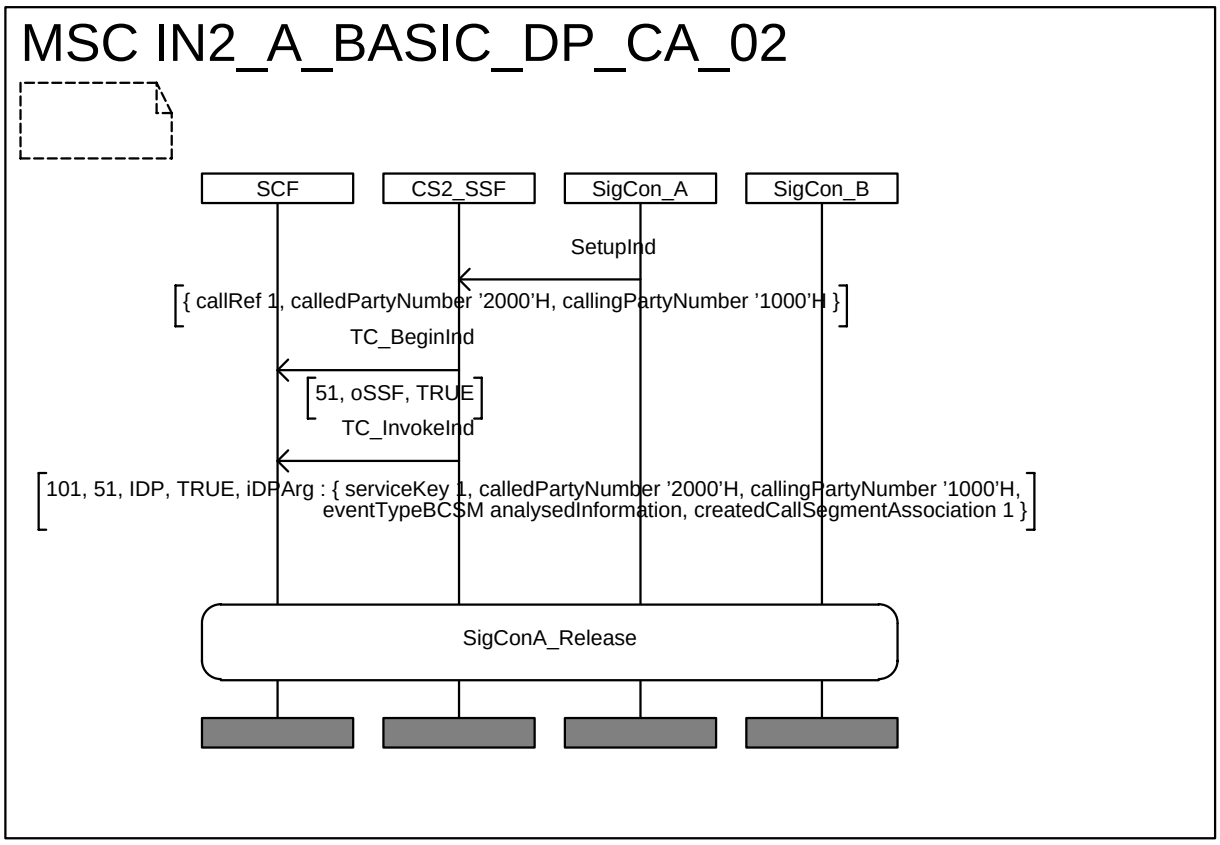

MSC IN2m_A_BASIC_FC_BV_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, FCI, short, fCIArg : fCIBCCcs1 : 'BBBB'H
TC_InvokeReq
2, 51, 2, AC, short, aCArg : { aChBillingChargingCharacteristics '44'H, sendCalculationToSCPIndication FALSE }
TC_InvokeReq
3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }
TC_ContinueReq
51, oSCF
SetupReq
{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }
SetupConf
{ callRef 2 }
SetupResp
{ callRef 1 }
ReleaseInd
{ callRef 2, cause '00'H }
ReleaseReq
{ callRef 1, cause '00'H }, senderRef
TC_EndInd
51, basic, TRUE
TC_InvokeInd
102, 51, ACR, TRUE, aCRArg : '44'H

6.4.11 InitialDP procedure

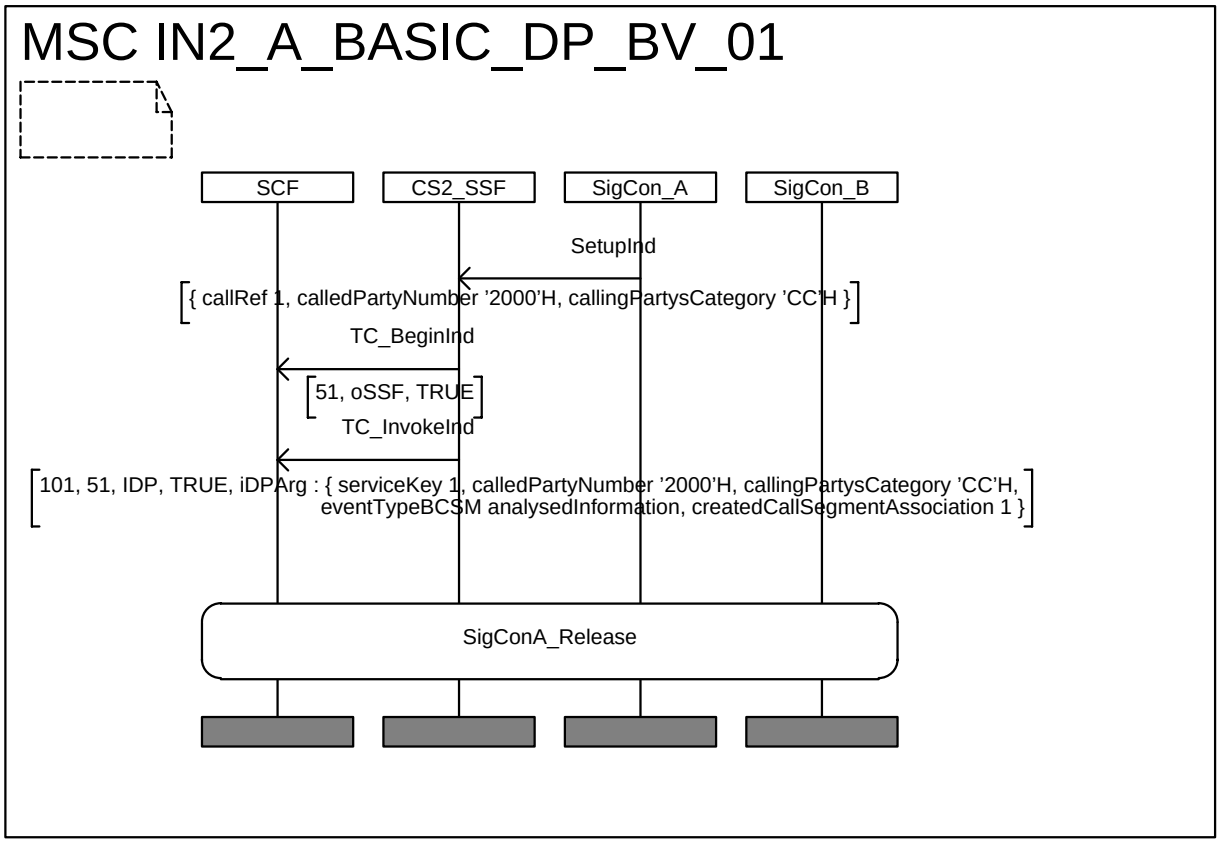
IN2_A_BASIC_DP_CA_01	
Purpose:	Test of **InitialDP** procedure and its parameter calledPartyNumber
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SigConA sends to SSF a **SetupInd** containing at least the parameter: - calledPartyNumber
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to the called party: - calledPartyNumber
Postamble:	SigConA_Release



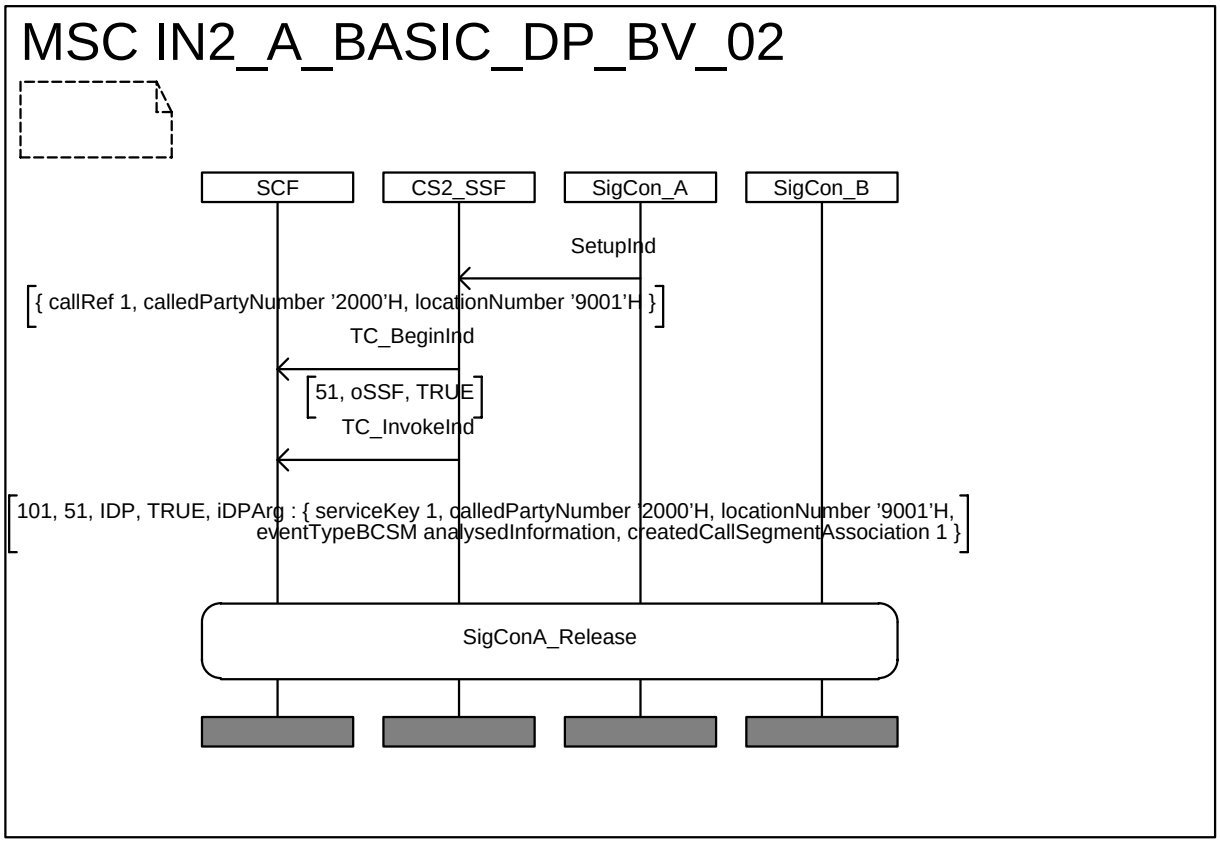
IN2_A_BASIC_DP_CA_02	
Purpose:	Test of **InitialDP** procedure and its parameter callingPartyNumber
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SigConA sends to SSF a **SetupInd** containing at least the parameter: - callingPartyNumber
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to the calling party: - callingPartyNumber
Postamble:	SigConA_Release



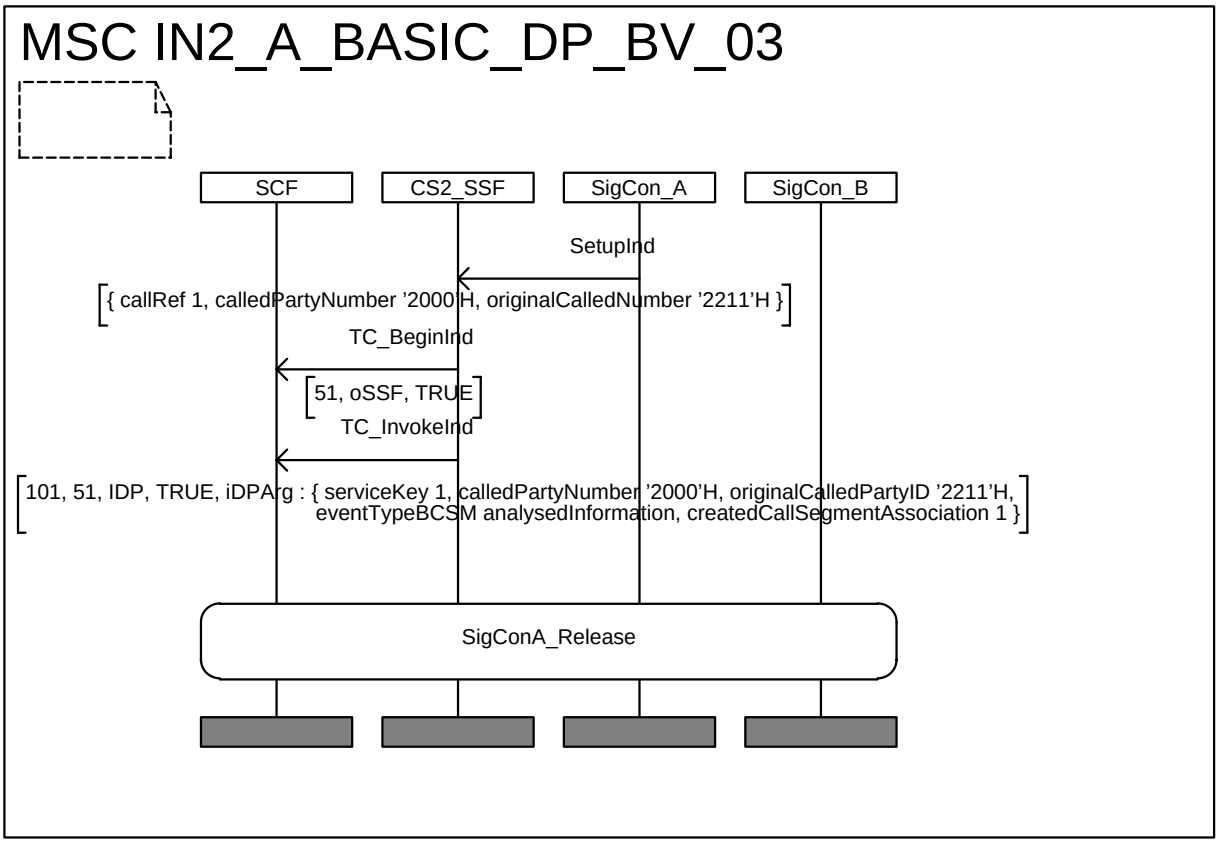
IN2_A_BASIC_DP_BV_01	
Purpose:	Test of **InitialDP** procedure and its parameter callingPartysCategory
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SigConA sends to SSF a **SetupInd** containing at least the parameter: - callingPartysCategory
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to the calling party category: - callingPartysCategory
Postamble:	SigConA_Release



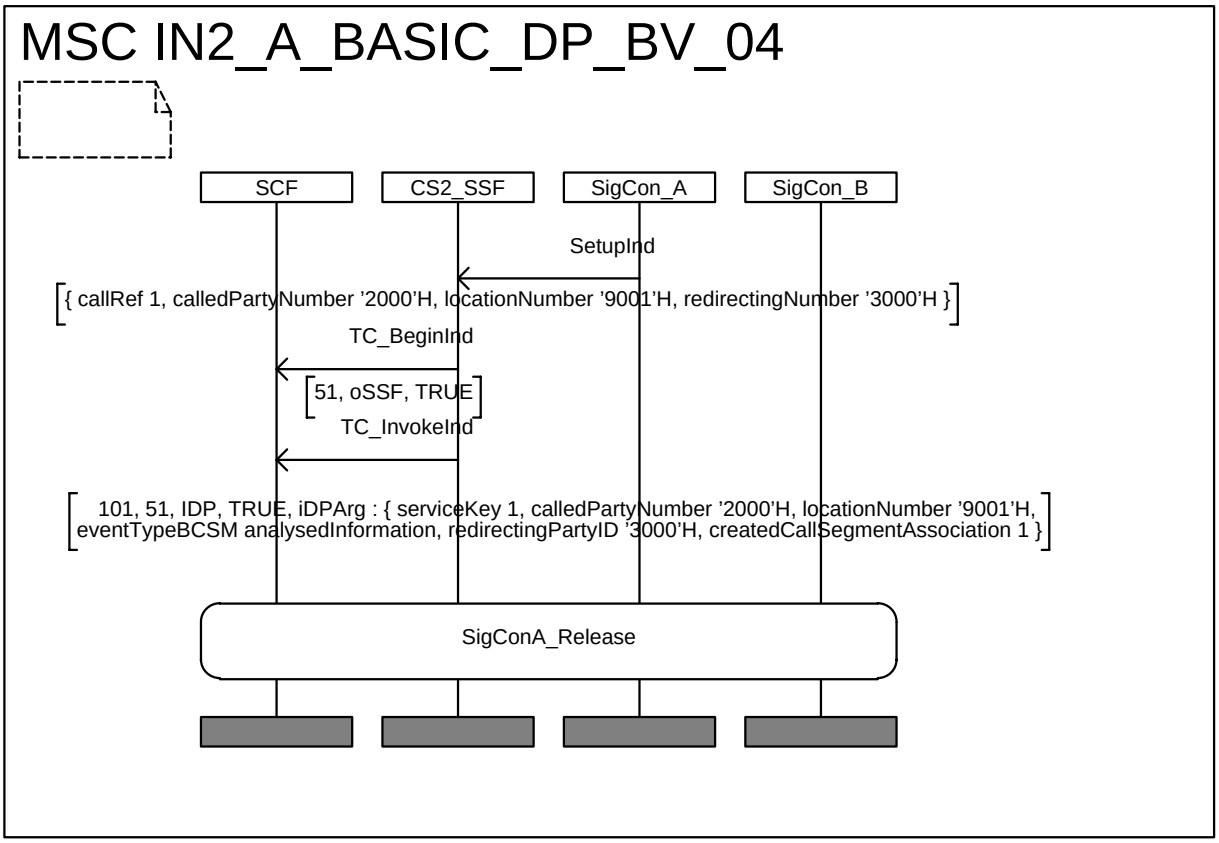
IN2_A_BASIC_DP_BV_02	
Purpose:	Test of **InitialDP** procedure and its parameter locationNumber
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SigConA sends to SSF a **SetupInd** containing at least the parameter: - locationNumber
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to the location information: - locationNumber
Postamble:	SigConA_Release



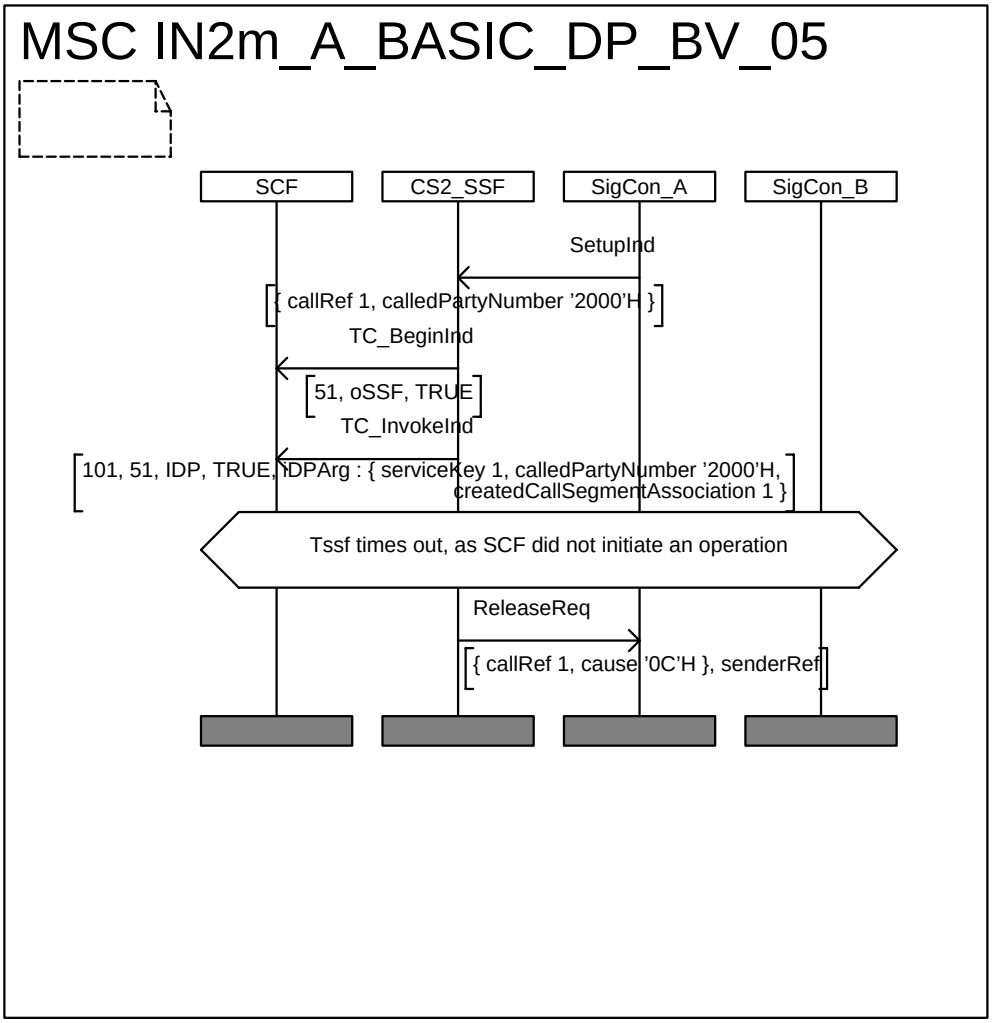
IN2_A_BASIC_DP_BV_03	
Purpose:	Test of **InitialDP** procedure and its parameter originalCalledPartyID
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	Signalling Control A (SigConA) sends to SSF a **SetupInd** containing at least the parameter: - originalCalledPartyID
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to the original called party number: - originalCalledPartyID
Postamble:	SigConA_Release



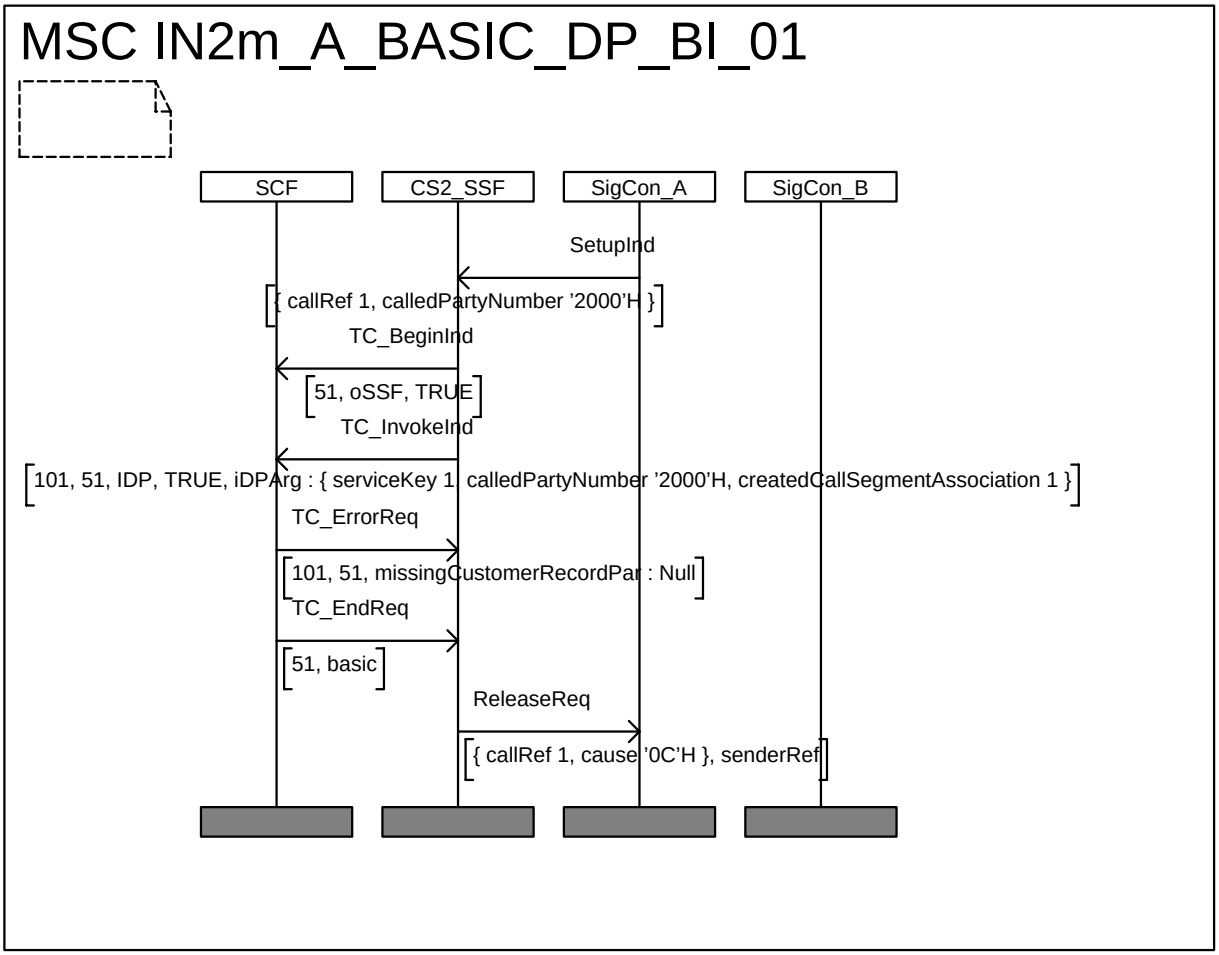
IN2_A_BASIC_DP_BV_04	
Purpose:	Test of **InitialDP** procedure and its parameter redirectingPartyID
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SigConA sends to SSF a **SetupInd** containing at least the parameter: - redirectingPartyID
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to redirecting party number: - redirectingPartyID
Postamble:	SigConA_Release



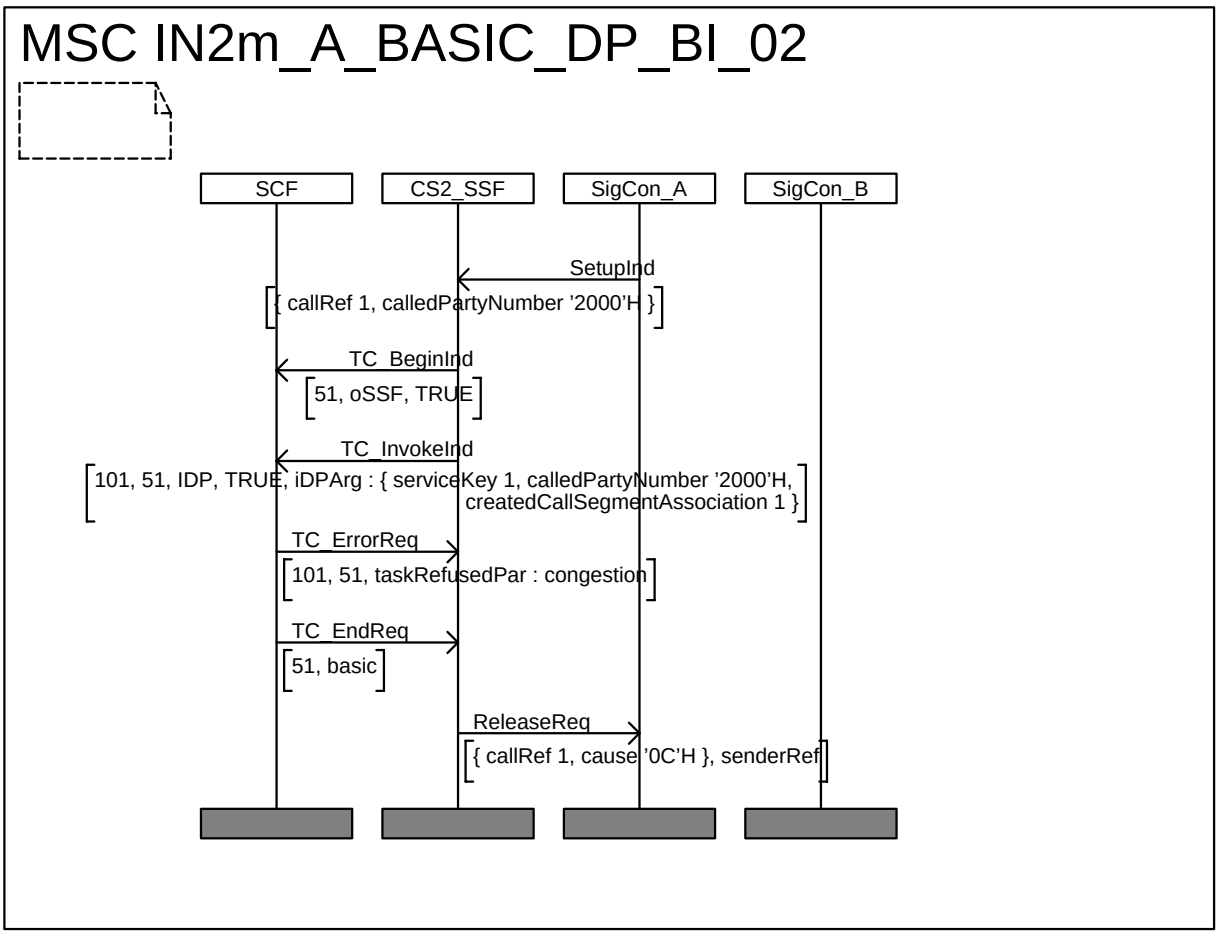
XXXX	IN2_A_BASIC_DP_BV_05
Purpose:	Test of **InitialDP** procedure with timer expiration
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	- SigConA sends a **SetupInd** containing at least the mandatory parameters - SSF sends to SCF an **InitialDP** invoke - SCF does not send to SSF an operation, so timer Tssf expires
Pass criteria	SSF sends release to SigConA to free all resources involved
Postamble:	



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

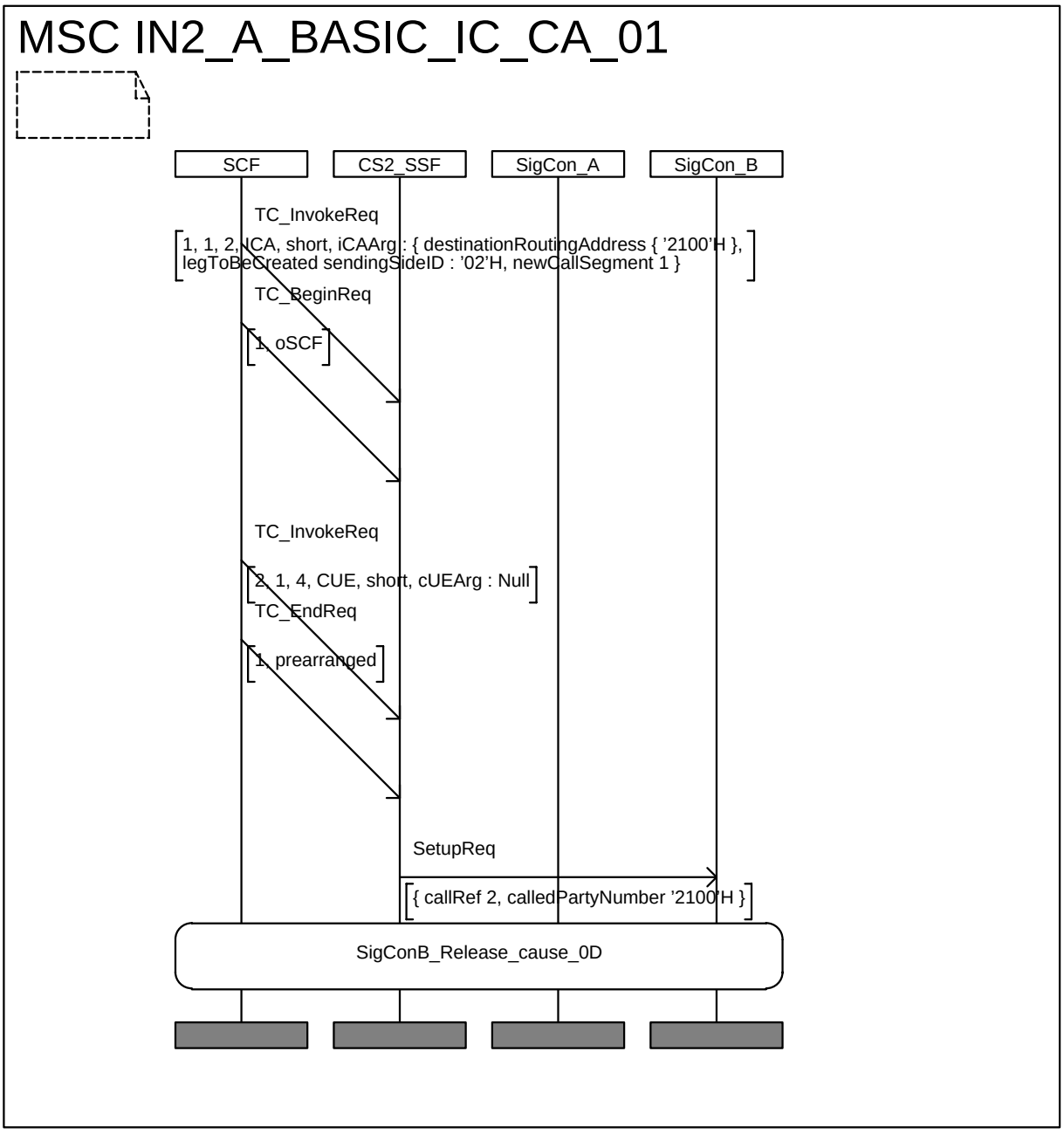


IN2_A_BASIC_DP_BI_02	
Purpose:	Test of **InitialDP** procedure with error
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	- SigConA sends a **SetupInd** containing at least the mandatory parameters - SSF sends to SCF an **InitialDP** invoke - SCF sends to SSF an **InitialDP** error containing parameter: taskRefused
Pass criteria	SSF goes to the idle state.
Postamble:	none

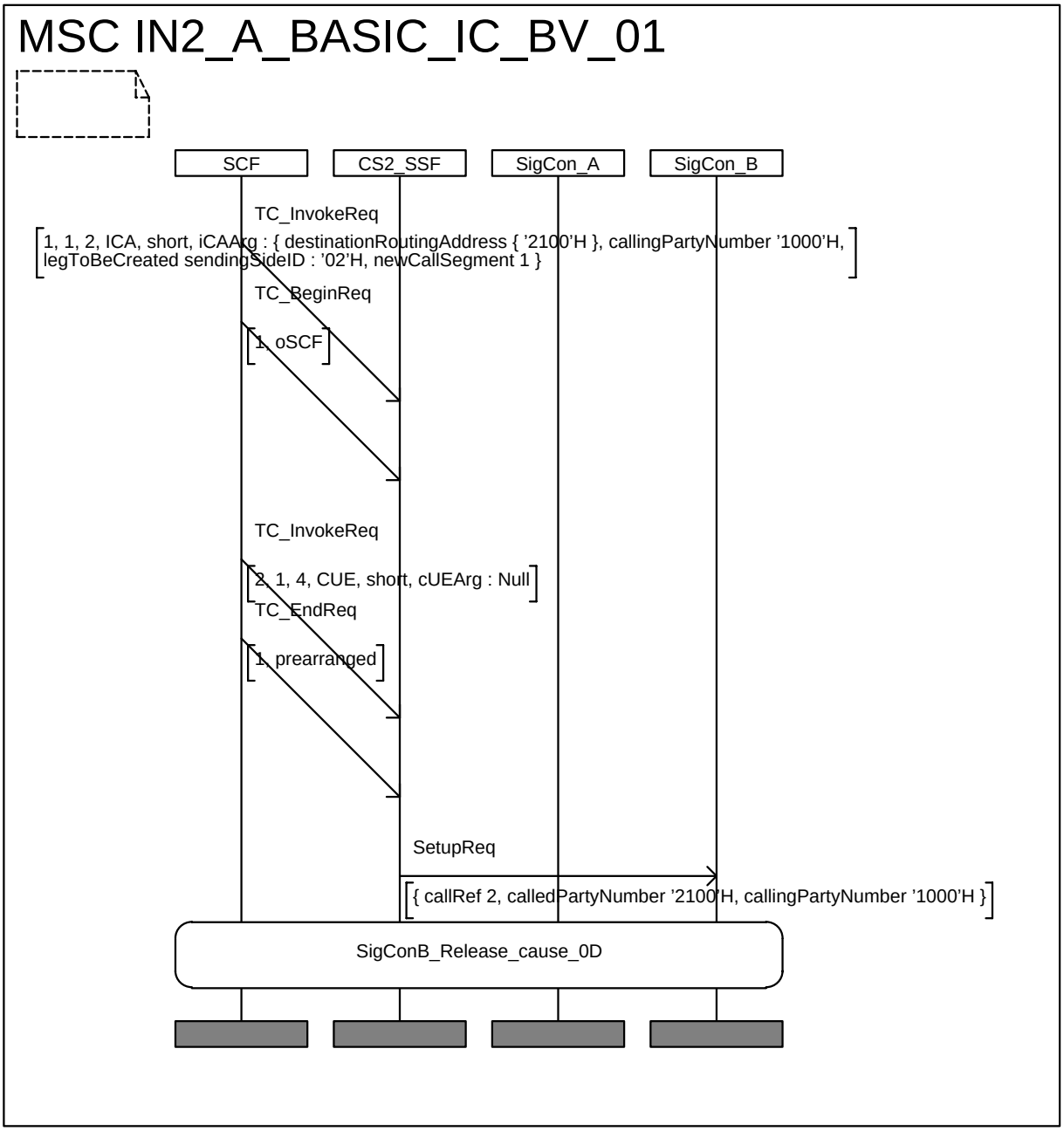


6.4.12 InitiateCallAttempt procedure

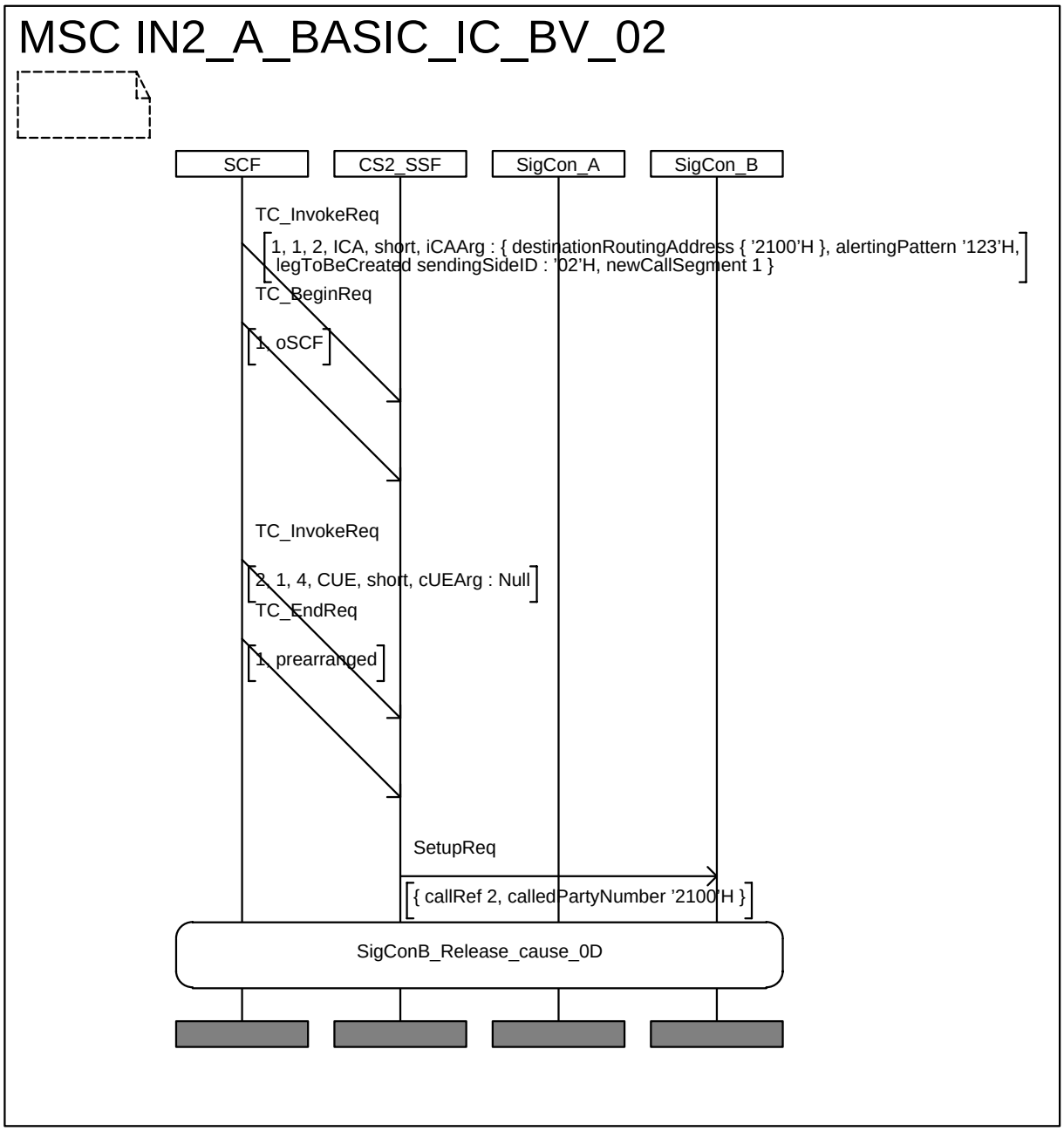
IN2_A_BASIC_IC_CA_01	
Purpose:	Test of **InitiateCallAttempt** base procedure
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF an **InitiateCallAttempt** with mandatory parameters: destinationRoutingAddress followed by a **Continue** invoke
Pass criteria	Check that SSF sends a SetupReq to the proper SigCon according to the **InitiateCallAttempt**
Postamble:	SigConB_Release



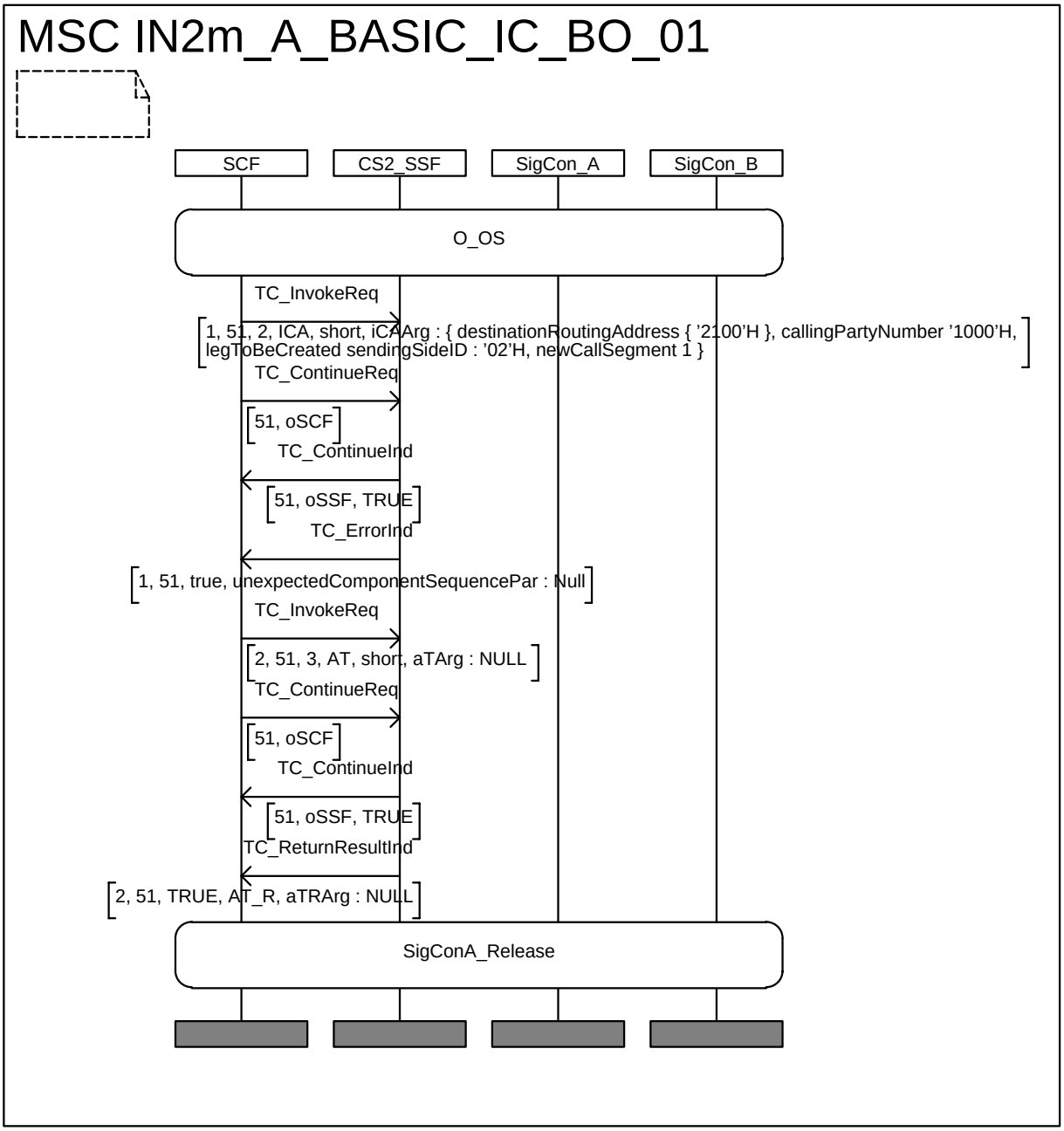
IN2_A_BASIC_IC_BV_01	
Purpose:	Test of **InitiateCallAttempt** procedure with parameter
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF an **InitiateCallAttempt** with mandatory and optional parameters destinationRoutingAddress callingPartyNumber followed by a **Continue** invoke
Pass criteria	Check that SSF sends a SetupReq to the proper SigCon according to the **InitiateCallAttempt**
Postamble:	SigConB_Release


MSC IN2_A_BASIC_IC_BV_01
SCF
CS2_SSF
SigCon_A
SigCon_B
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

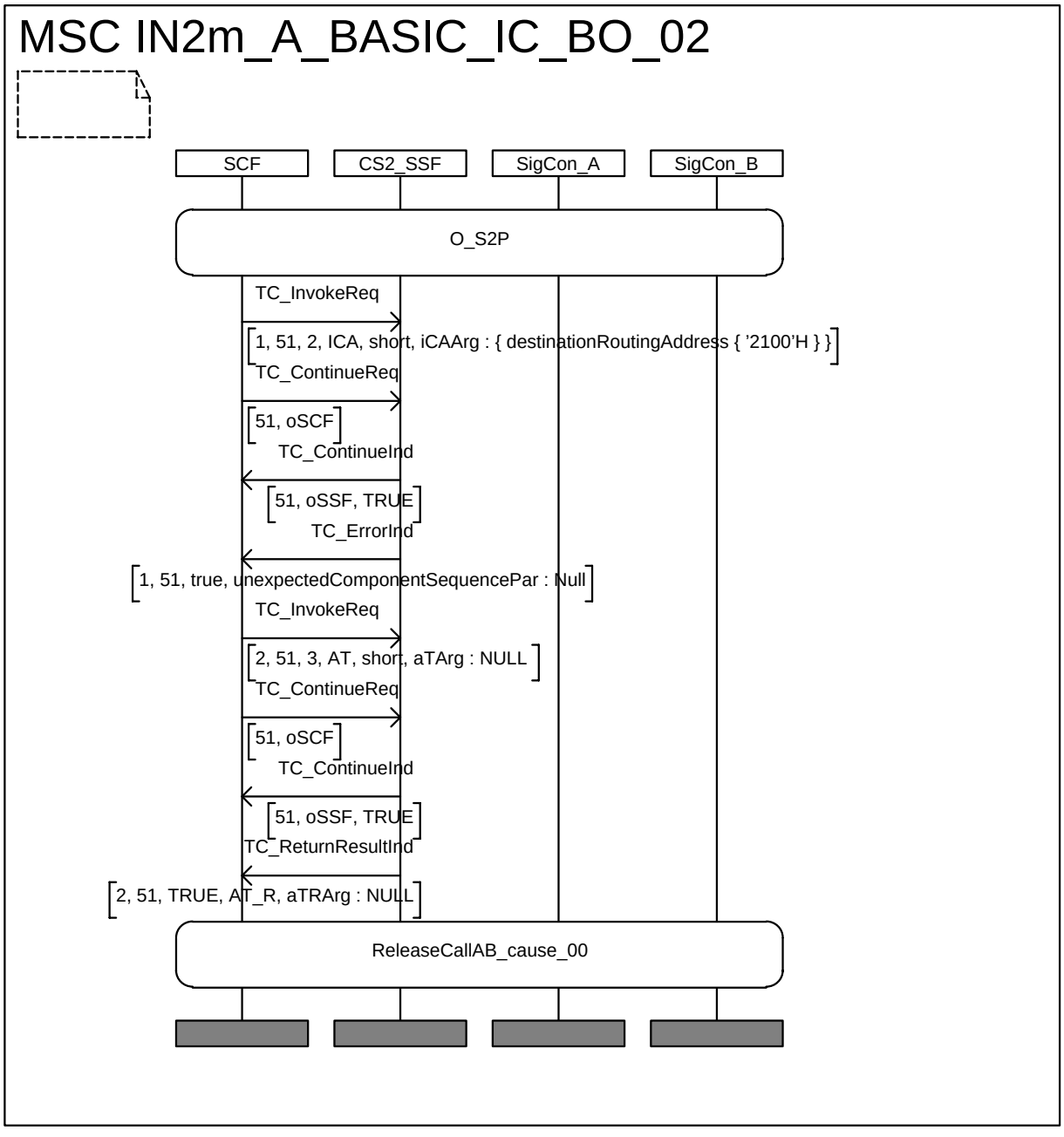
IN2_A_BASIC_IC_BV_02	
Purpose:	Test of **InitiateCallAttempt** procedure with parameter
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF an **InitiateCallAttempt** with mandatory and optional parameters destinationRoutingAddress alertingPattern followed by a **Continue** 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



IN2_A_BASIC_IC_BO_01	
Purpose:	Test of **InitiateCallAttempt** procedure in wrong state (WaitForInstructions)
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF an **InitiateCallAttempt** in a wrong state (WFI)
Pass criteria	Check that SSF returns **InitiateCallAttempt** error with parameter UnexpectedComponentSequence and remains in the same state. To check SSF stays in WaitingForInstructions 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:	SigConA_Release

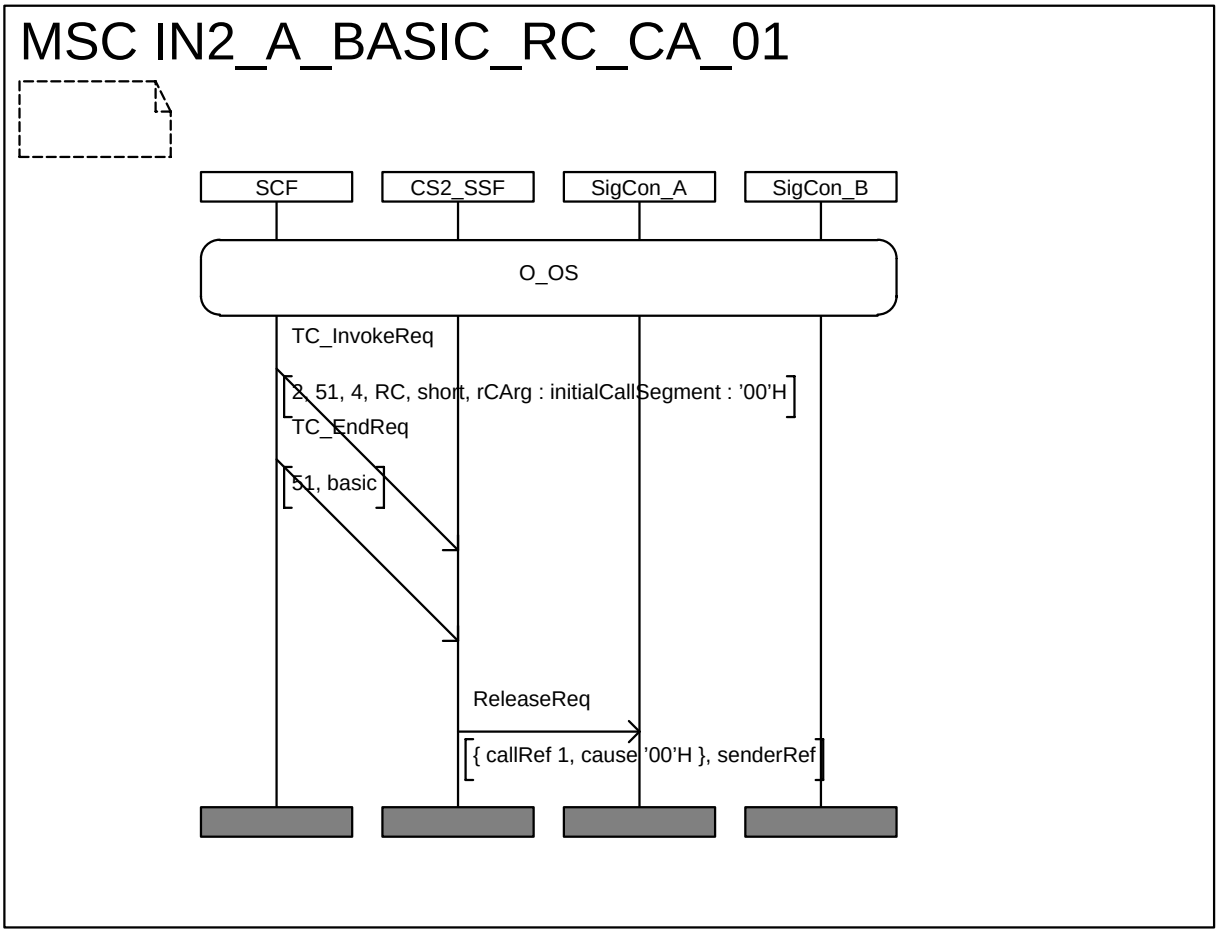

MSC IN2m_A_BASIC_IC_BO_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, ICA, short, iCAArg : { destinationRoutingAddress { '2100'H }, callingPartyNumber '1000'H, legToBeCreated sendingSideID : '02'H, newCallSegment 1 }
TC_ContinueReq
51, oSCF
TC_ContinueInd
51, oSSF, TRUE
TC_ErrorInd
1, 51, 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
SigConA_Release

XXXX	IN2_A_BASIC_IC_BO_02
Purpose:	Test of **InitiateCallAttempt** procedure in wrong state (monitoring)
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
Test description	SCF sends to SSF an **InitiateCallAttempt** in a wrong state (Monitoring)
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

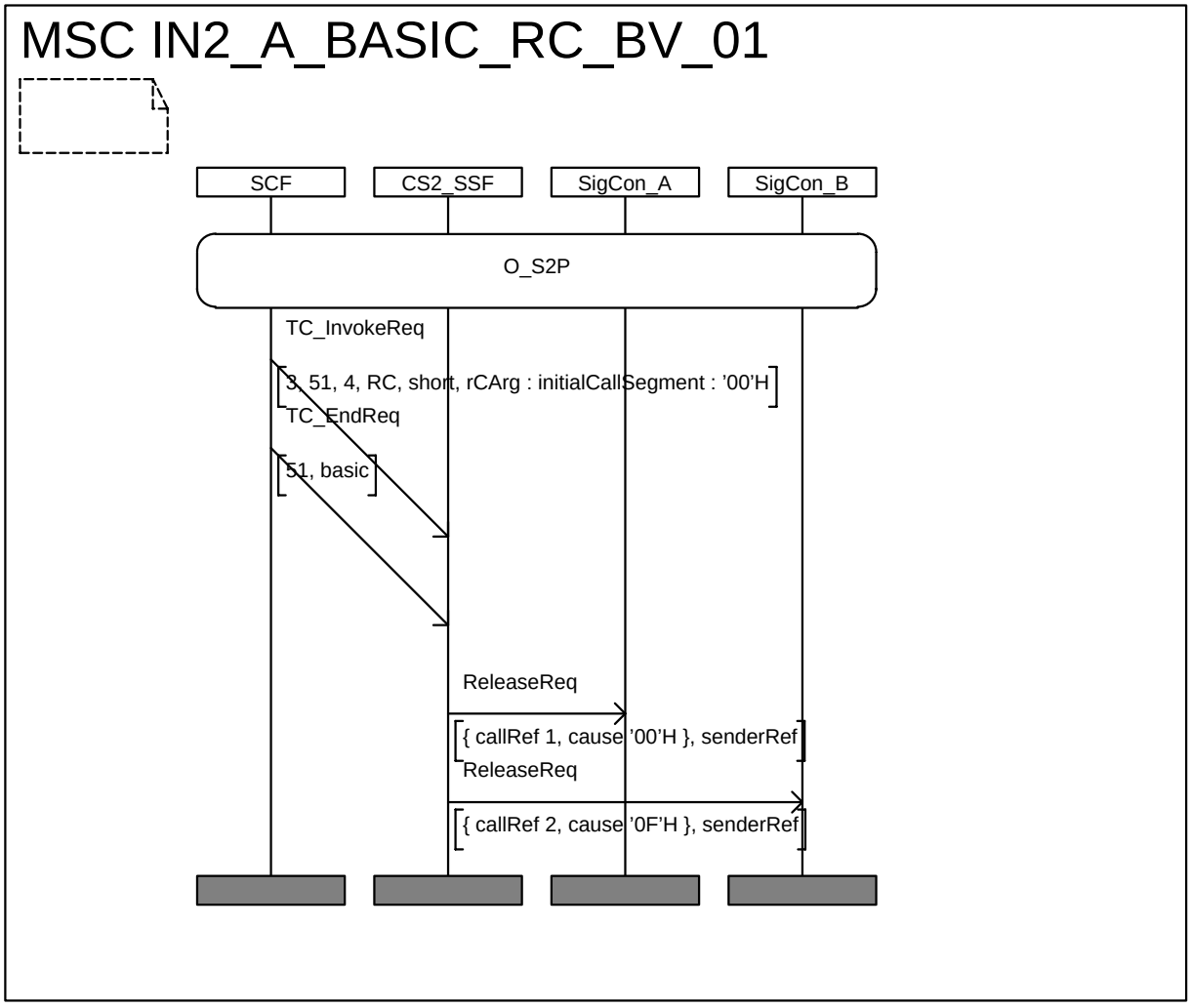


6.4.13 ReleaseCall procedure

IN2_A_BASIC_RC_CA_01	
Purpose:	Test of **ReleaseCall** base procedure
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **ReleaseCall** invoke, with: - initialCallSegment (cause)
Pass criteria	Check that SSF releases the call (ReleaseReq received by SigConA)
Postamble:	none



IN2_A_BASIC_RC_BV_01	
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



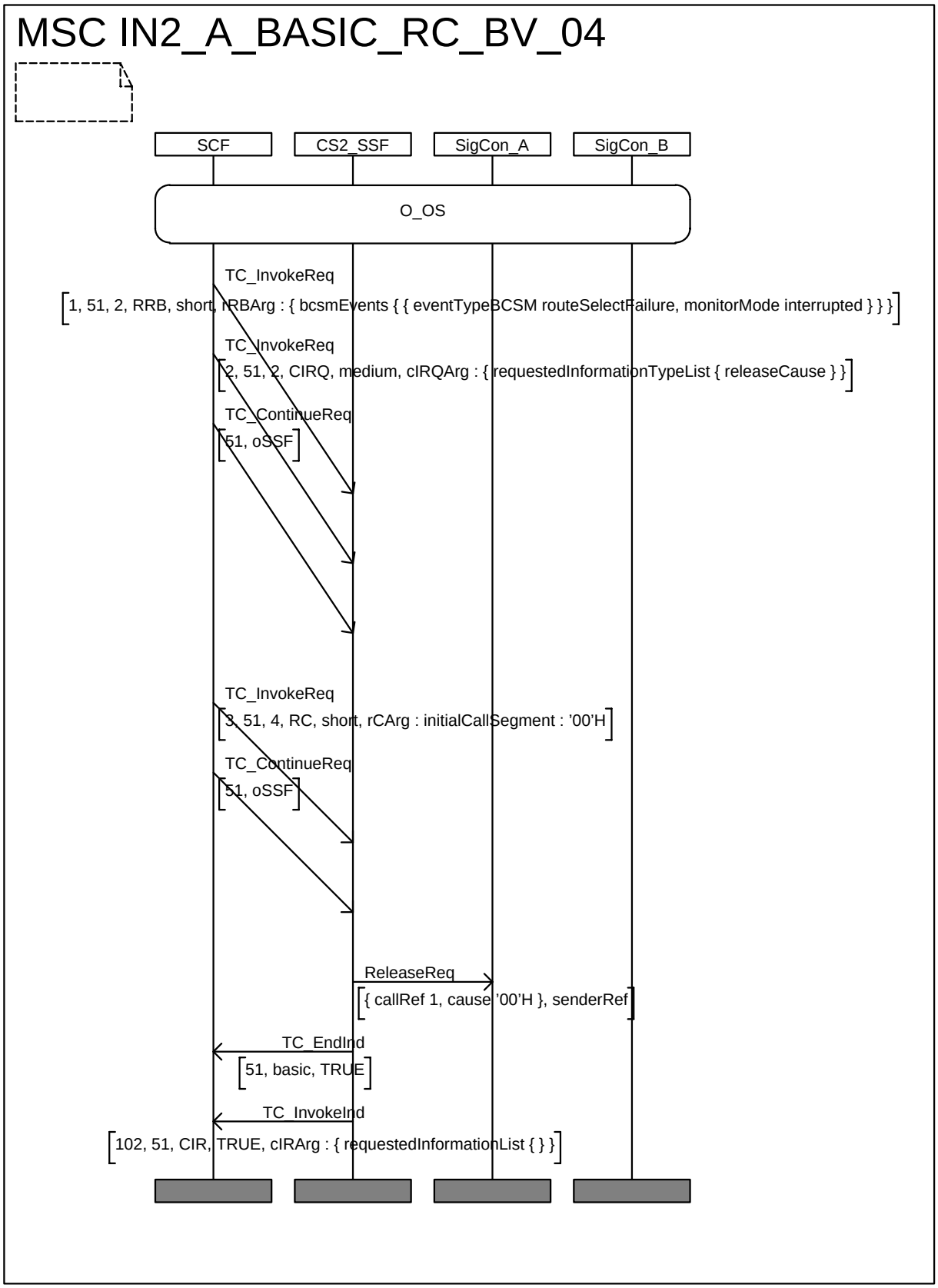
IN2_A_BASIC_RC_BV_02

This TP was dropped.

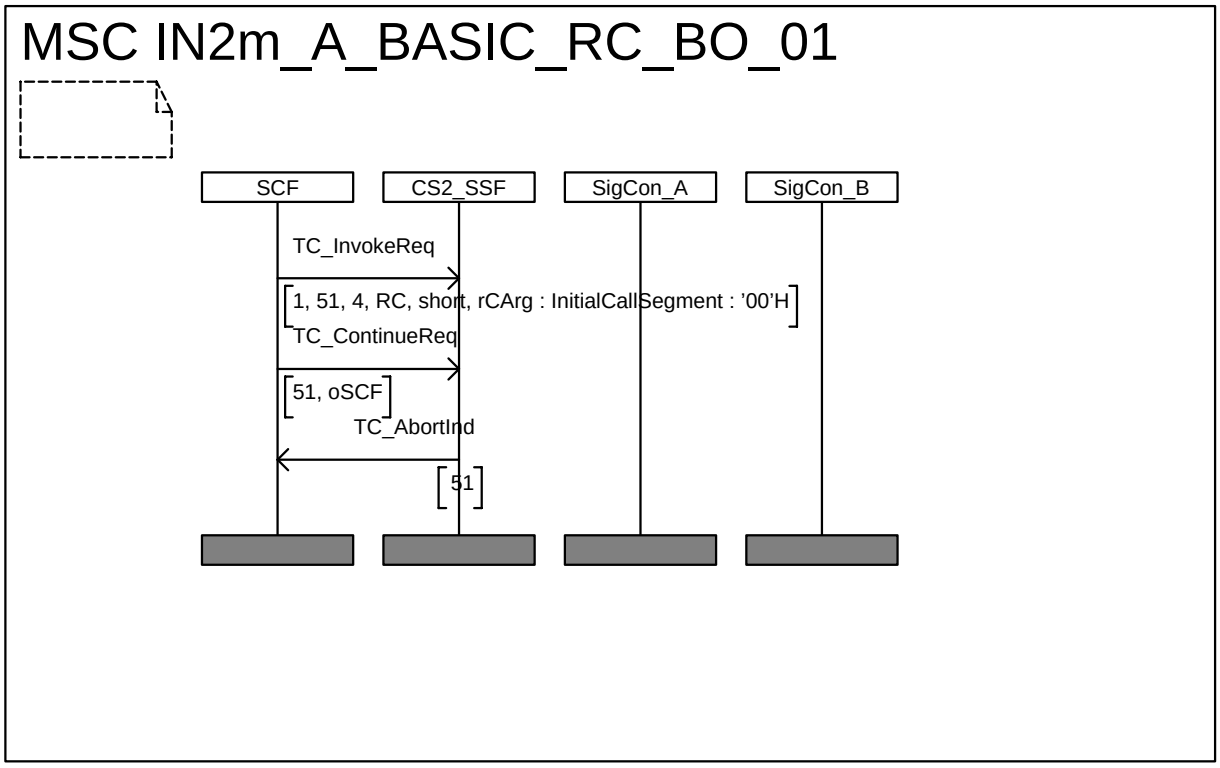
IN2_A_BASIC_RC_BV_03

This TP was dropped.

IN2_A_BASIC_RC_BV_04	
Purpose:	Test of **ReleaseCall** procedure in combination with **CallInformation** and **RequestReportBCSMEvent** procedures
Requirement ref	
Selection Cond.	
Preamble:	O_OS
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** EventReportBCSM is sent Check that SigConA receives a ReleaseReq
Postamble:	none

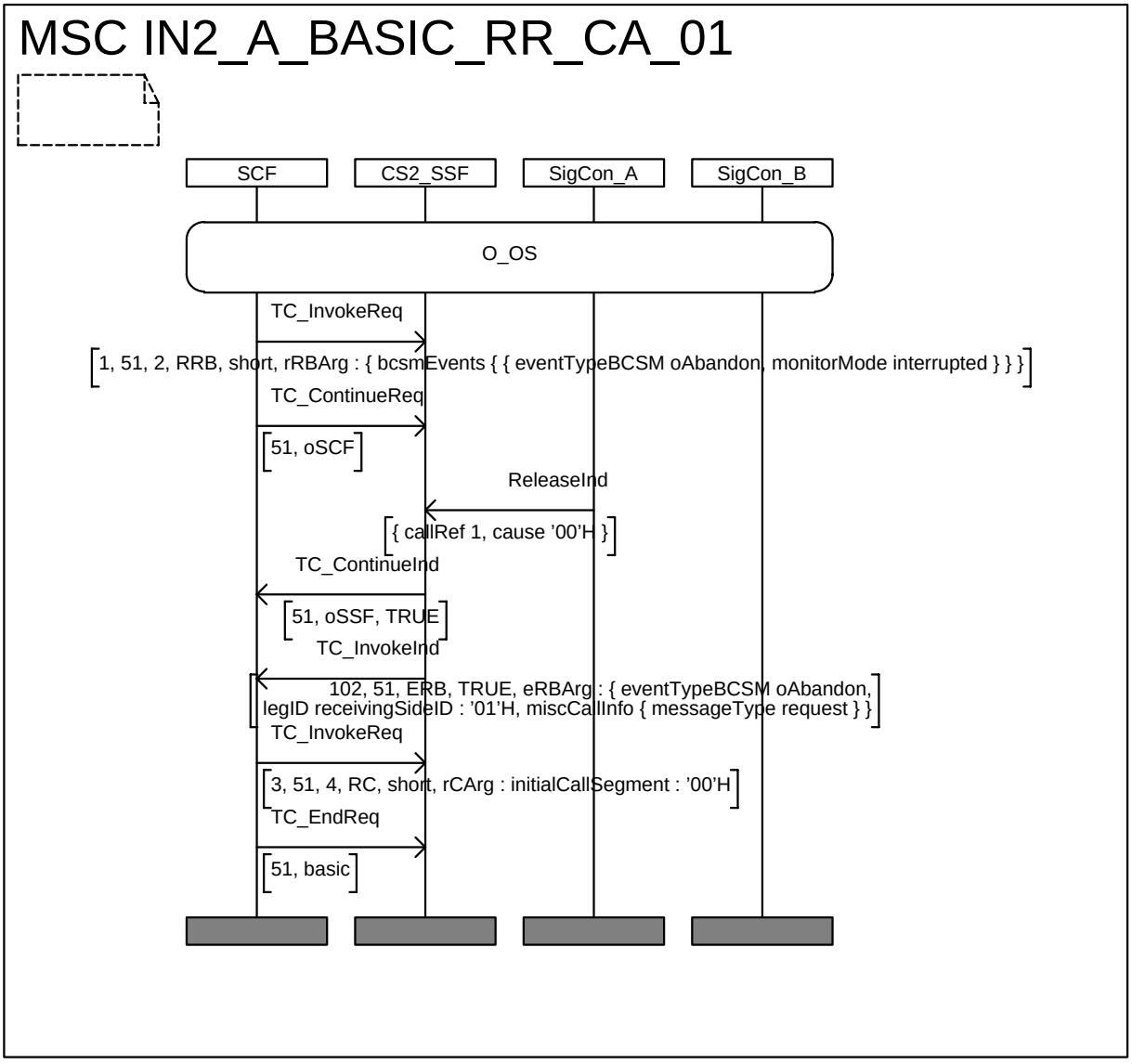

MSC IN2_A_BASIC_RC_BV_04
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
[1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode interrupted } } }]
TC_InvokeReq
[2, 51, 2, CIRQ, medium, cIRQArg : { requestedInformationTypeList { releaseCause } }]
TC_ContinueReq
[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 { } }]

IN2_A_BASIC_RC_BO_01	
Purpose:	Test of **ReleaseCall** procedure form wrong state (idle)
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **ReleaseCall** invoke, with: - initialCallSegment (cause)
Pass criteria	Check that SSF rejects the invoke
Postamble:	none

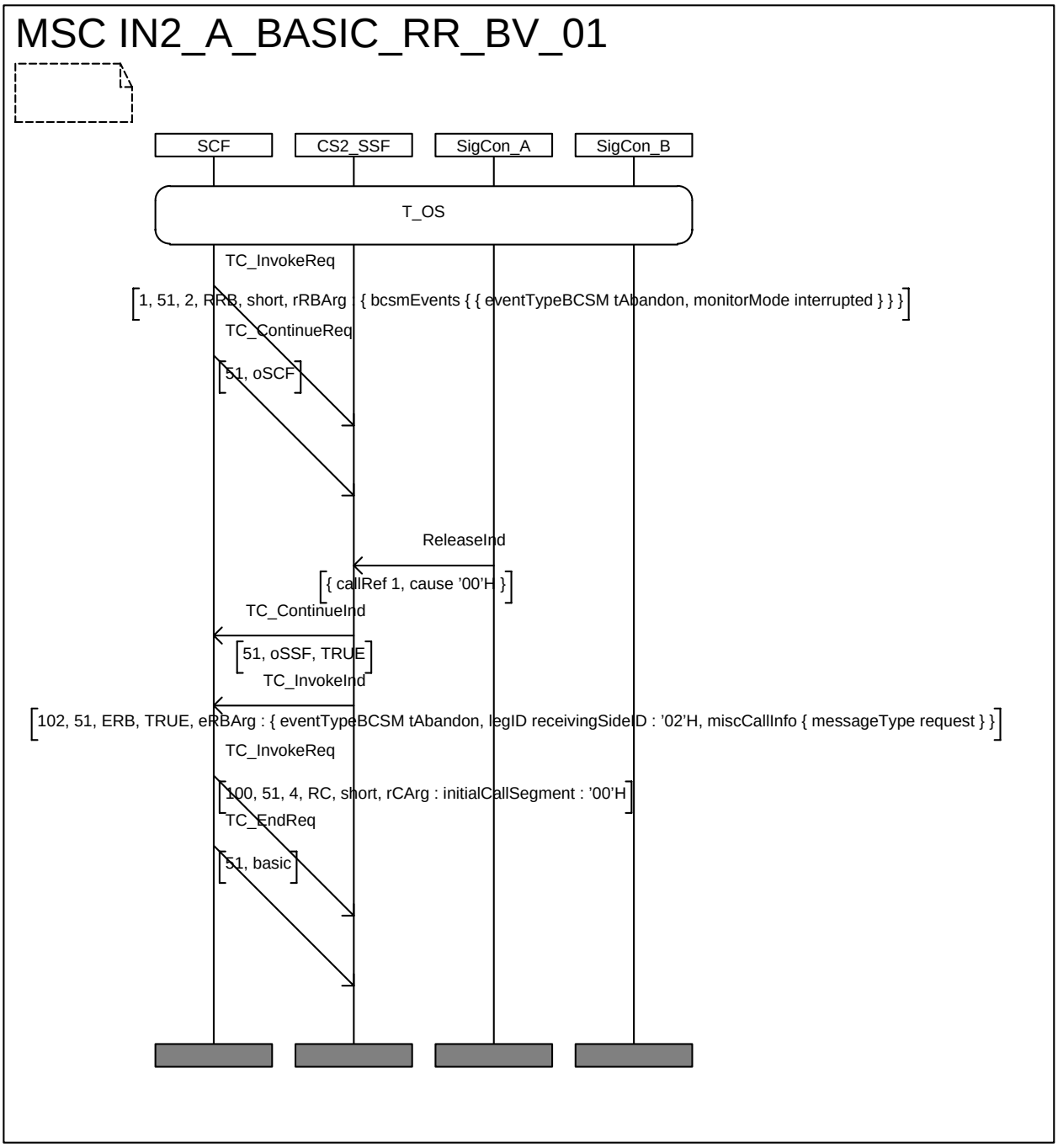


6.4.14 RequestReportBCSMEvent procedure

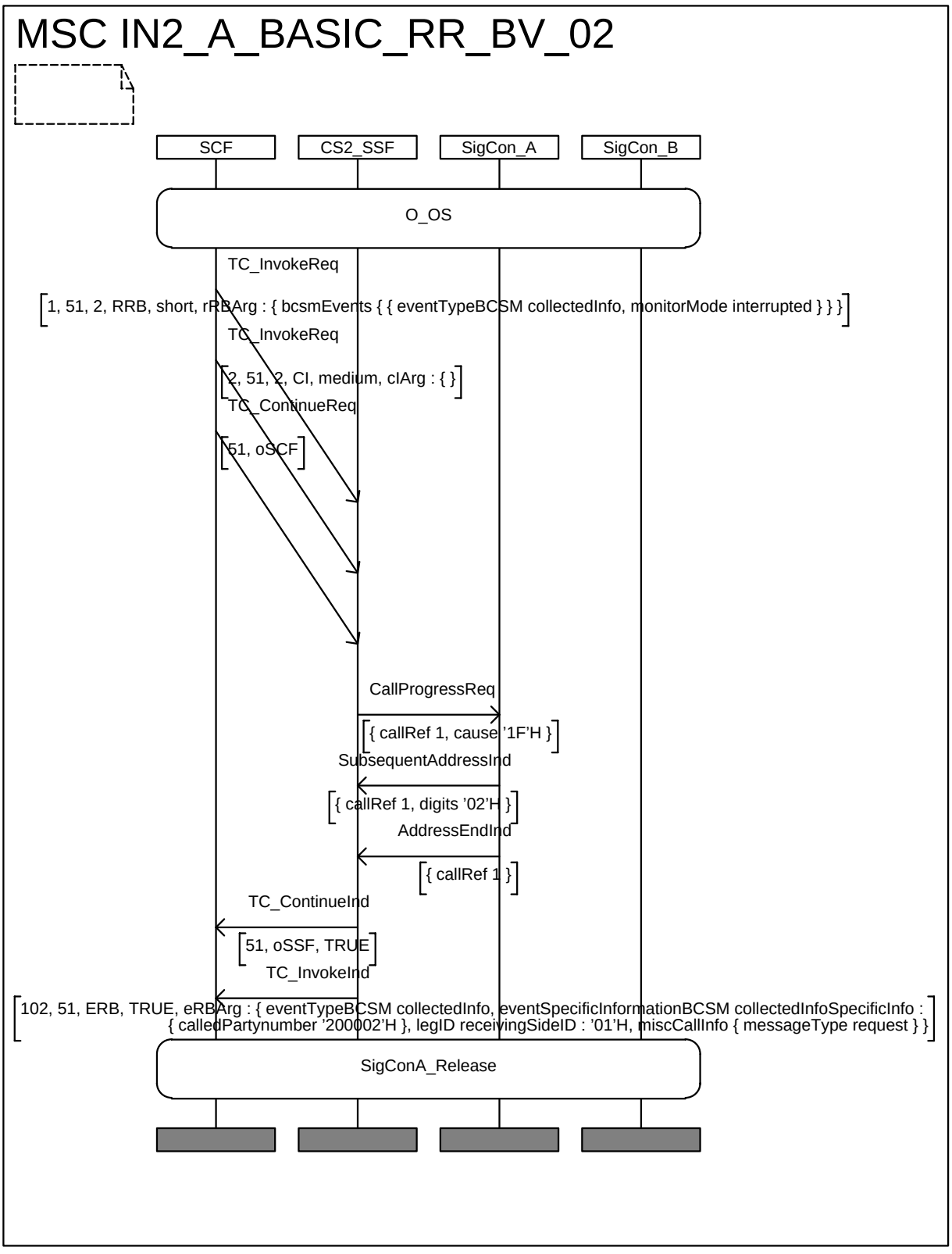
IN2_A_BASIC_RR_CA_01	
Purpose:	Test of **RequestReportBCSMEvent** base procedure
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oAbandon - monitoringMode=interrupted then the calling party 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=oAbandon**
Postamble:	none



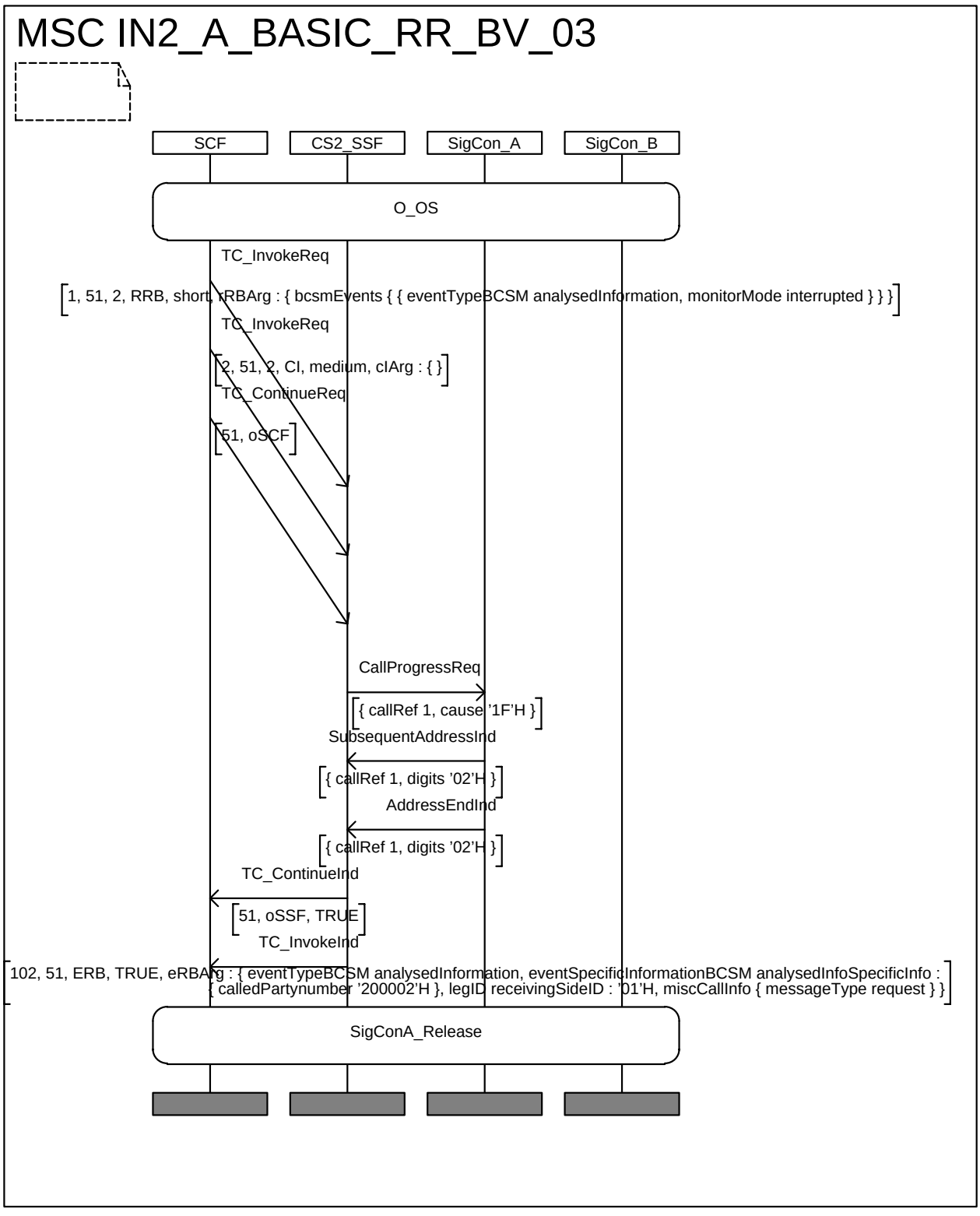
IN2_A_BASIC_RR_BV_01	
Purpose:	Test of **RequestReportBCSMEvent** procedure and tAbandon indication
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tAbandon - monitoringMode=interrupted then 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
SCF
CS2_SSF
SigCon_A
SigCon_B
T_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM tAbandon, monitorMode interrupted } } }
TC_ContinueReq
51, oSCF
ReleaseInd
{ callRef 1, cause '00'H }
TC_ContinueInd
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

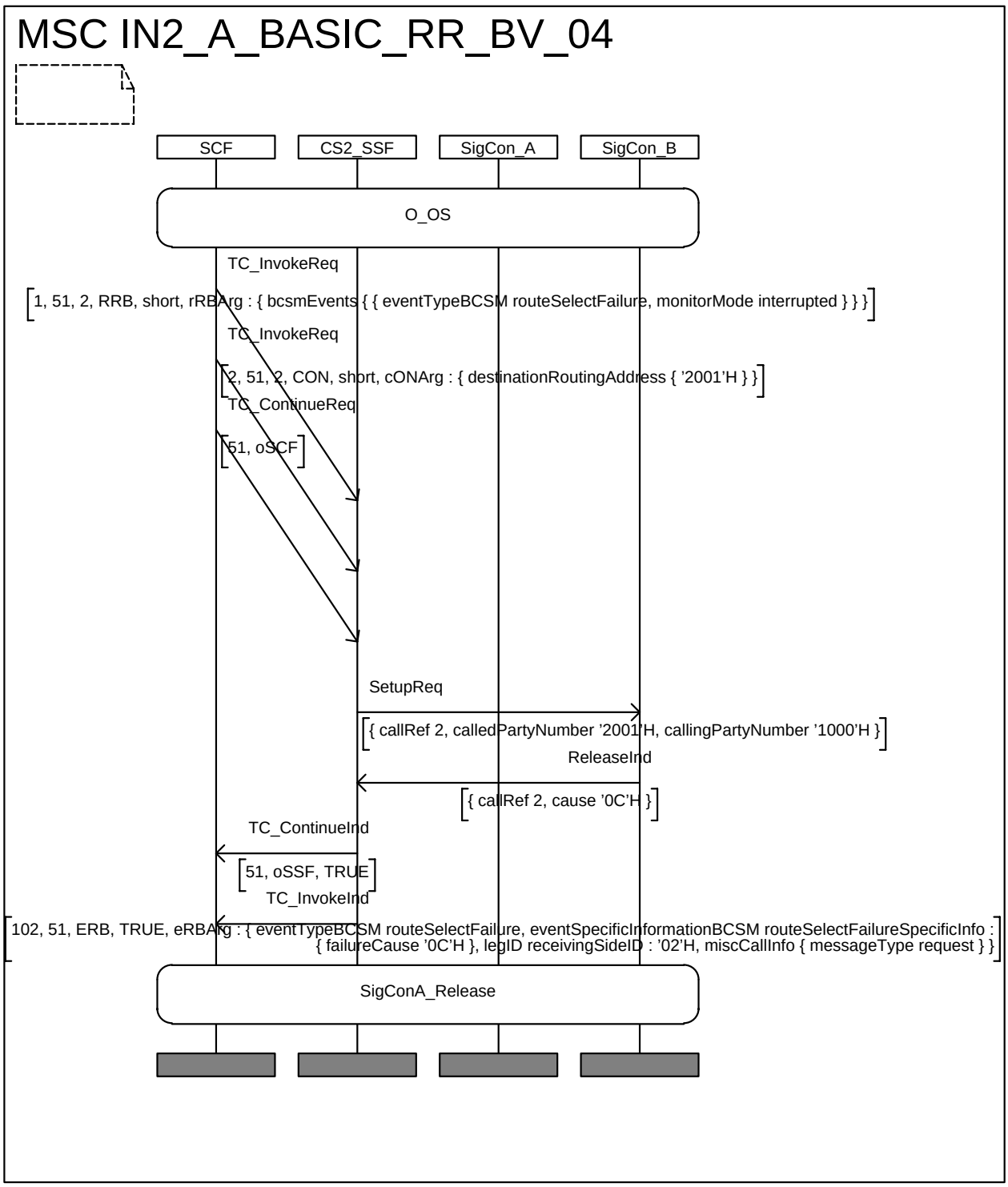
IN2_A_BASIC_RR_BV_02 (see also IN2_A_BASIC_CI_CA_01)	
Purpose:	Test of **RequestReportBCSMEvent** procedure and collectedInfo indication
Requirement ref	
Selection Cond.	
Preamble:	O_OS Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF SCF sends to SSF **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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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

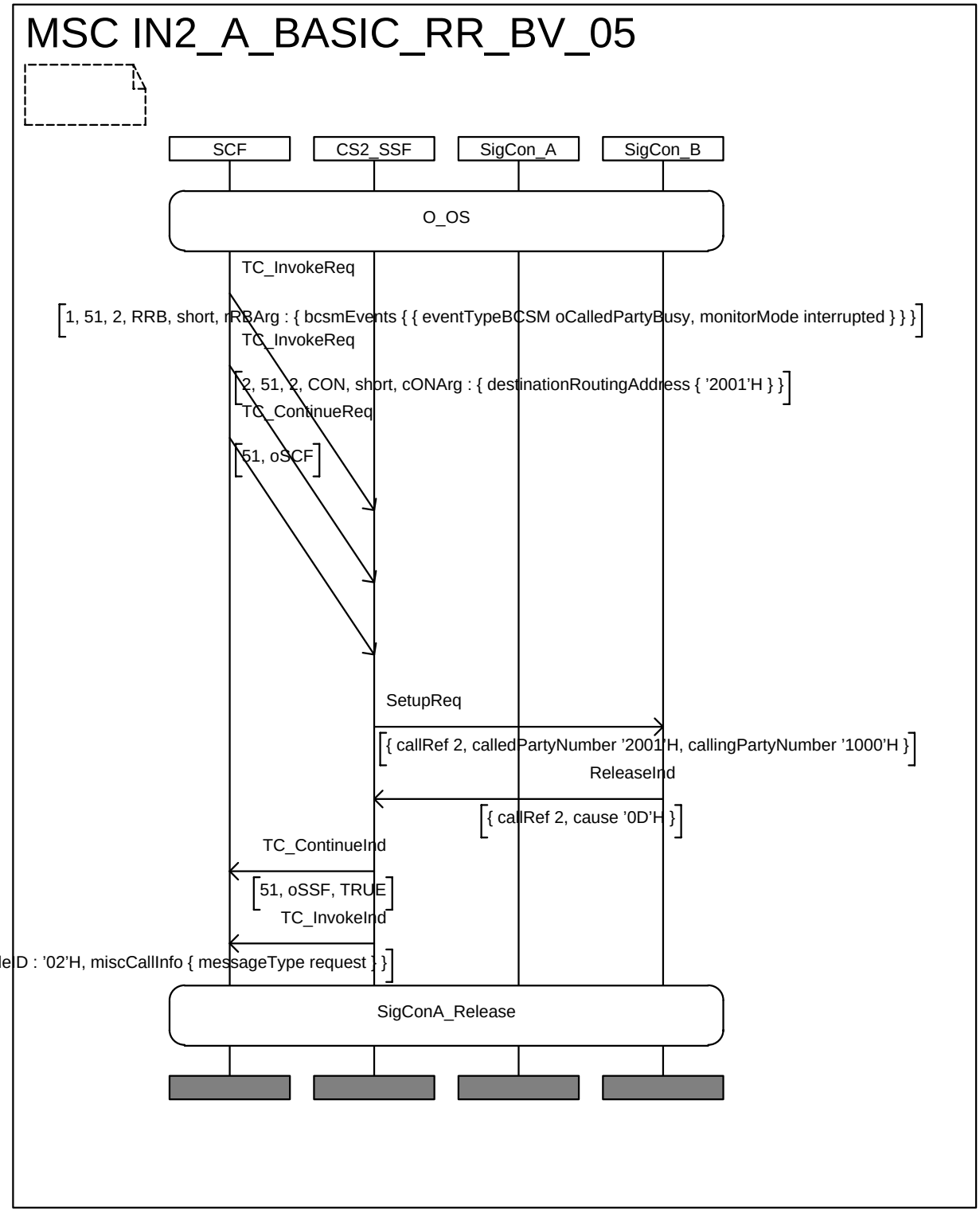
IN2_A_BASIC_RR_BV_03	
Purpose:	Test of **RequestReportBCSMEvent** procedure and analysedInfo indication
Requirement ref	
Selection Cond.	
Preamble:	O_OS Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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

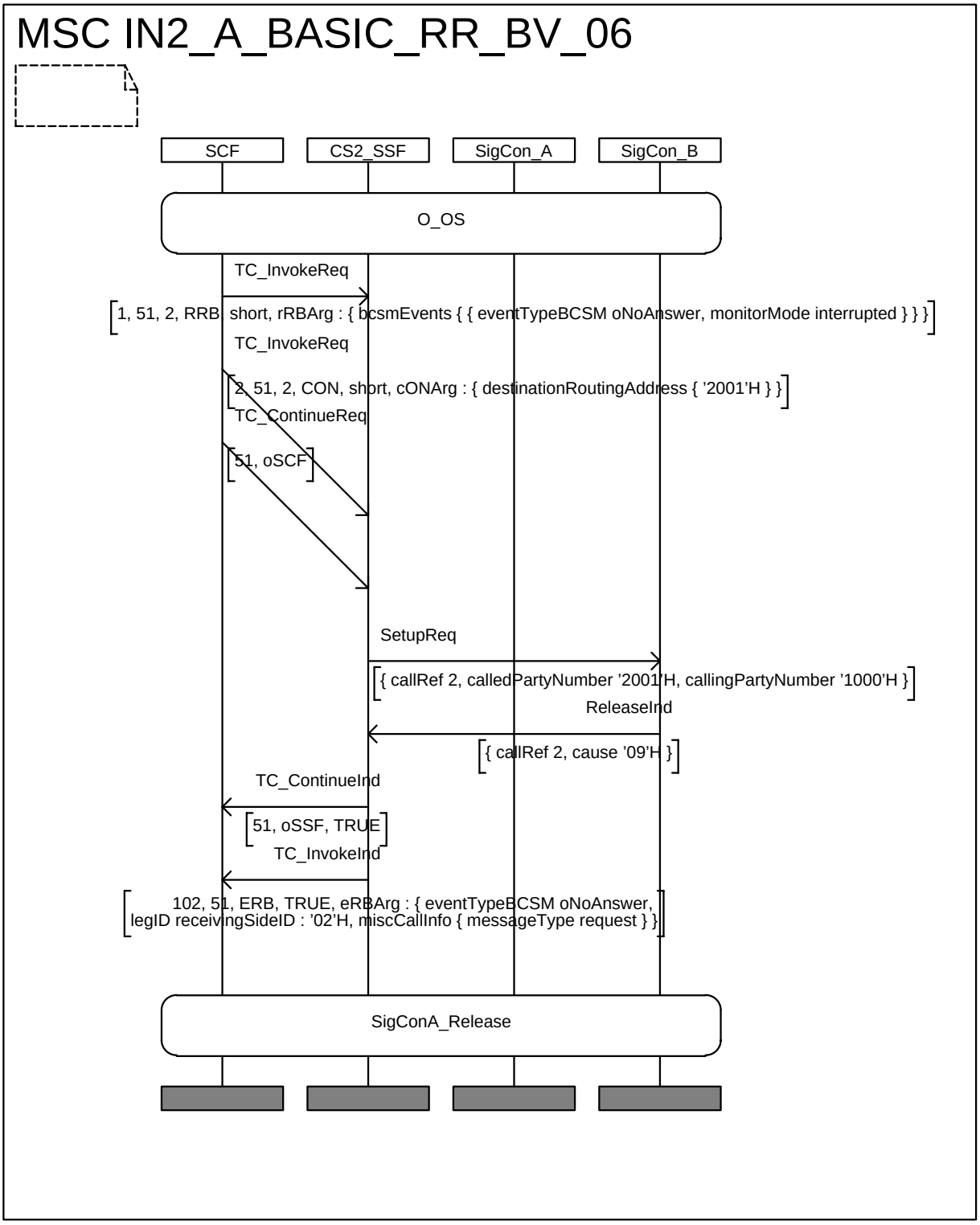
IN2_A_BASIC_RR_BV_04	
Purpose:	Test of **RequestReportBCSMEvent** procedure and routeSelectFailure indication
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=routeSelectFailure - monitoringMode=interrupted 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, 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 '0C'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, ERB, TRUE, eRBARg : { eventTypeBCSM routeSelectFailure, eventSpecificInformationBCSM routeSelectFailureSpecificInfo : { failureCause '0C'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }
SigConA_Release

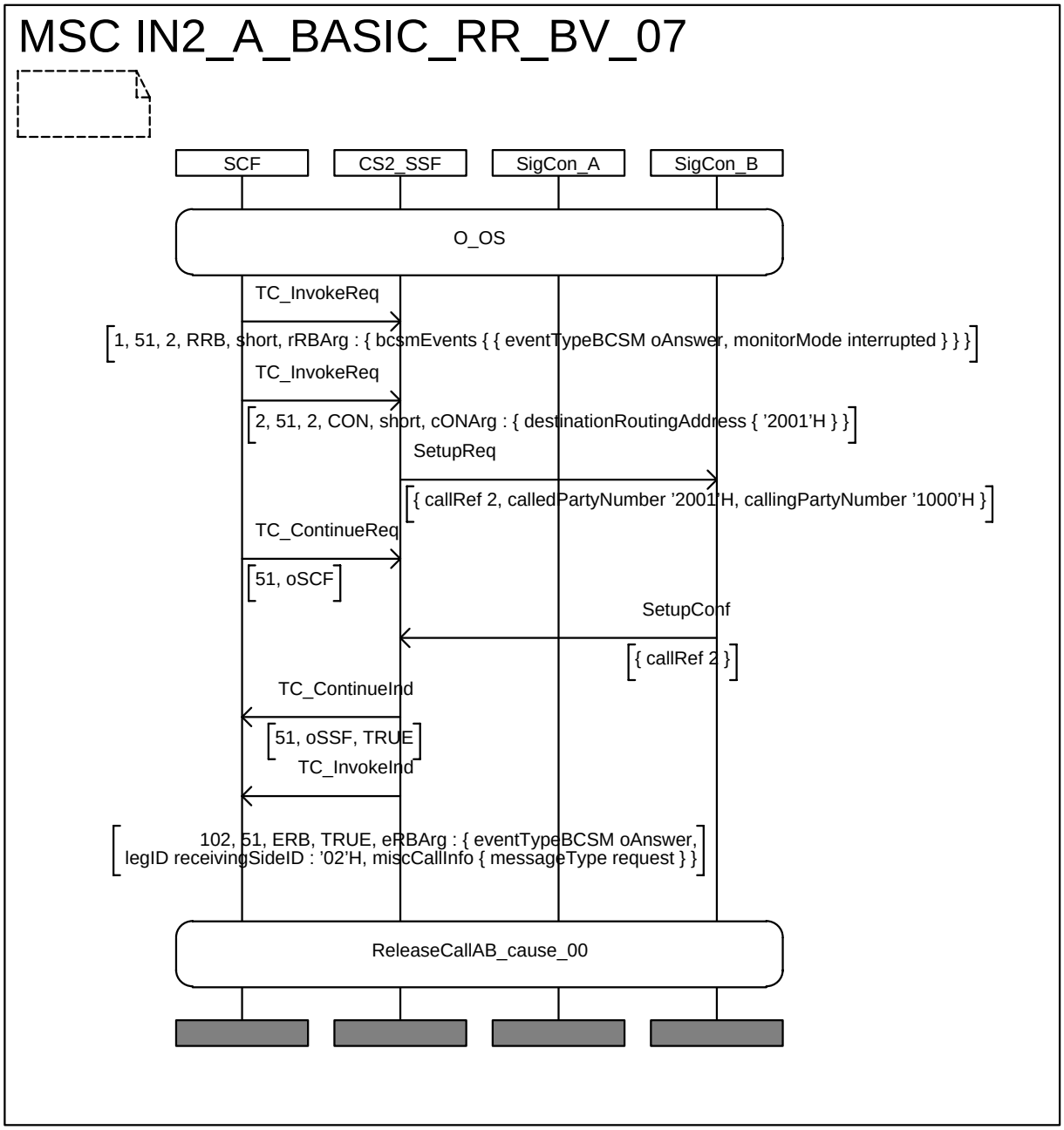
IN2_A_BASIC_RR_BV_05	
Purpose:	Test of **RequestReportBCSMEvent** procedure and oCalledPartyBusy indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS In addition, user B is declared busy
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oCalledPartyBusy - monitoringMode=interrupted followed by a **Connect** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call **(ReleaseInd)** with bPtyBusy_UDUB
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** oCalledPartyBusy
Postamble:	SigConA_Release


MSC IN2_A_BASIC_RR_BV_05
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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
D receivingSideID : '02'H, miscCallInfo { messageType request } }
SigConA_Release

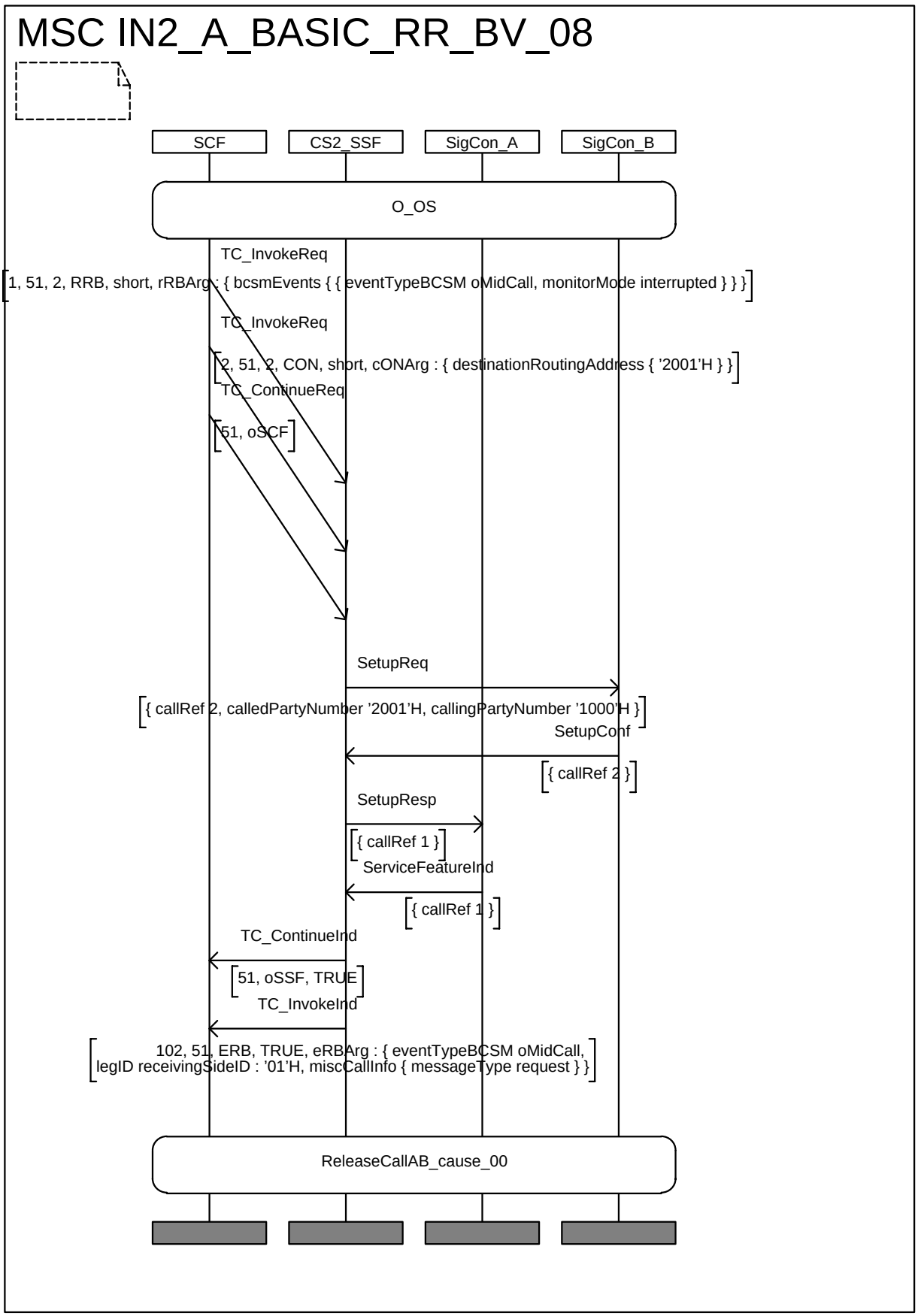
IN2_A_BASIC_RR_BV_06	
Purpose:	Test of **RequestReportBCSMEvent** procedure and oNoAnswer indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oNoAnswer - monitoringMode=interrupted followed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


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

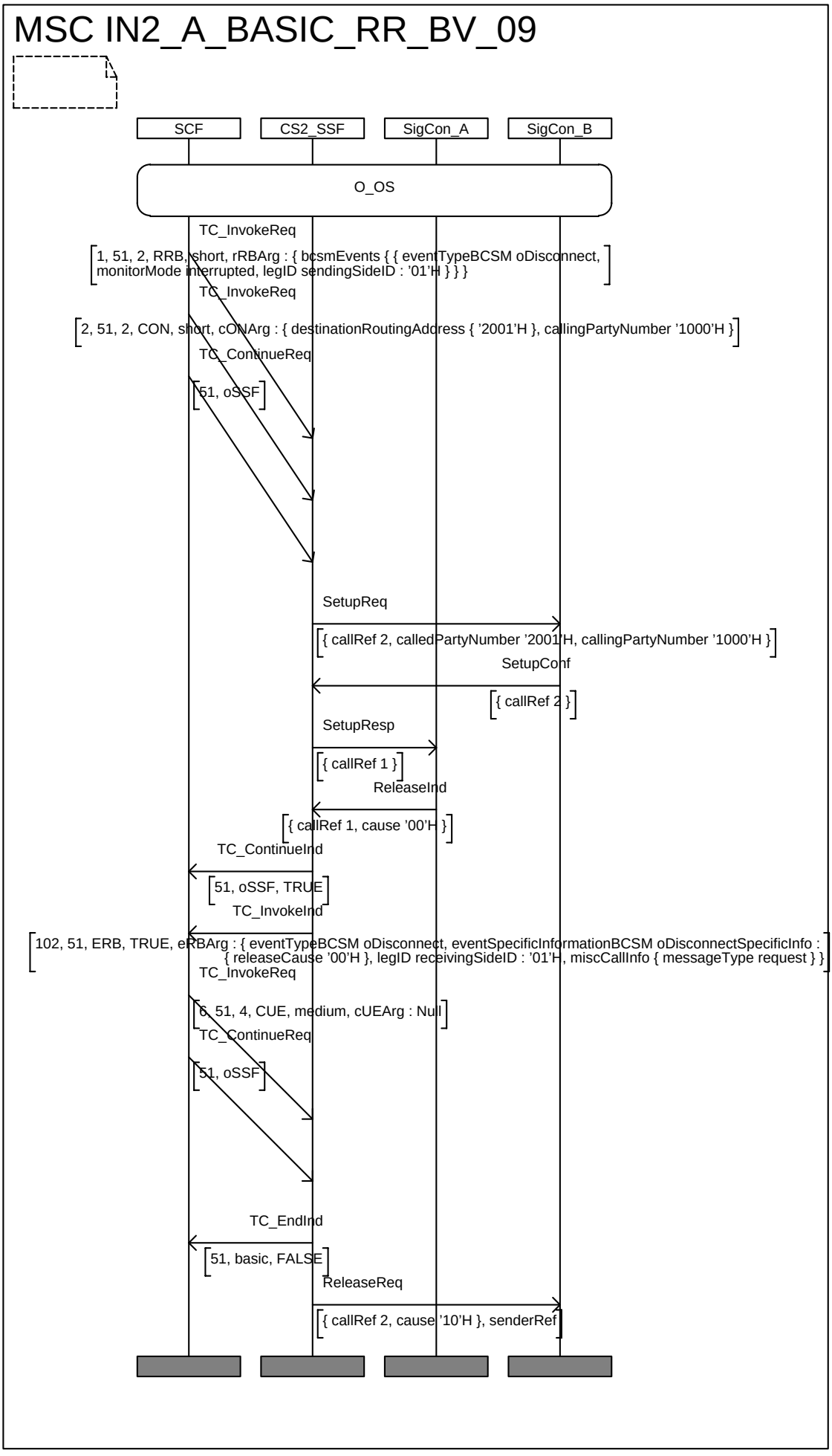
IN2_A_BASIC_RR_BV_07	
Purpose:	Test of **RequestReportBCSMEvent** procedure and oAnswer indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oAnswer - monitoringMode=interrupted followed 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



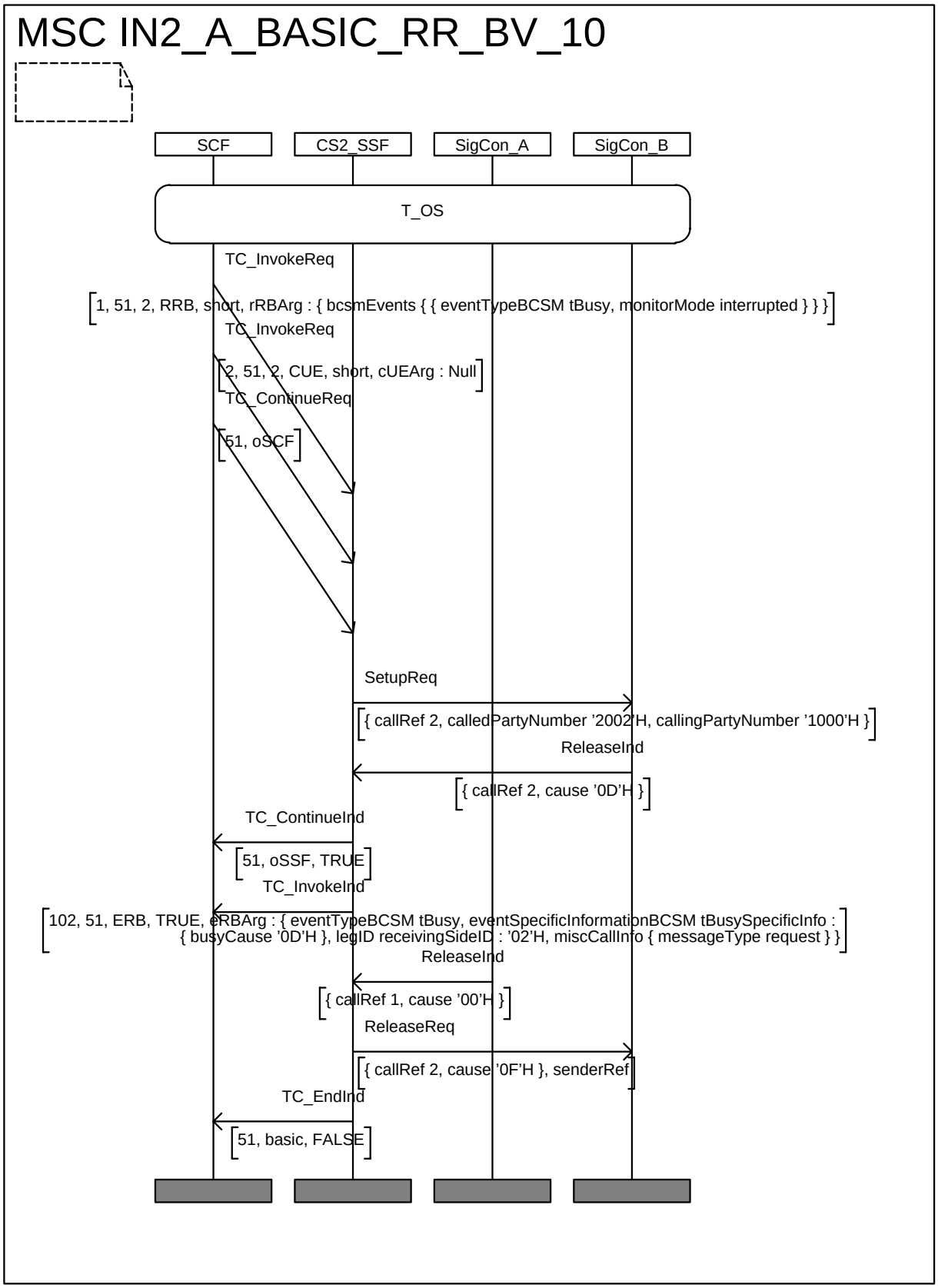
IN2_A_BASIC_RR_BV_08	
Purpose:	Test of **RequestReportBCSMEvent** procedure and oMidCall indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oMidCall - monitoringMode=interrupted followed 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oMidCall, 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 }
SetupConf
{ callRef 2 }
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 } }
ReleaseCallAB_cause_00

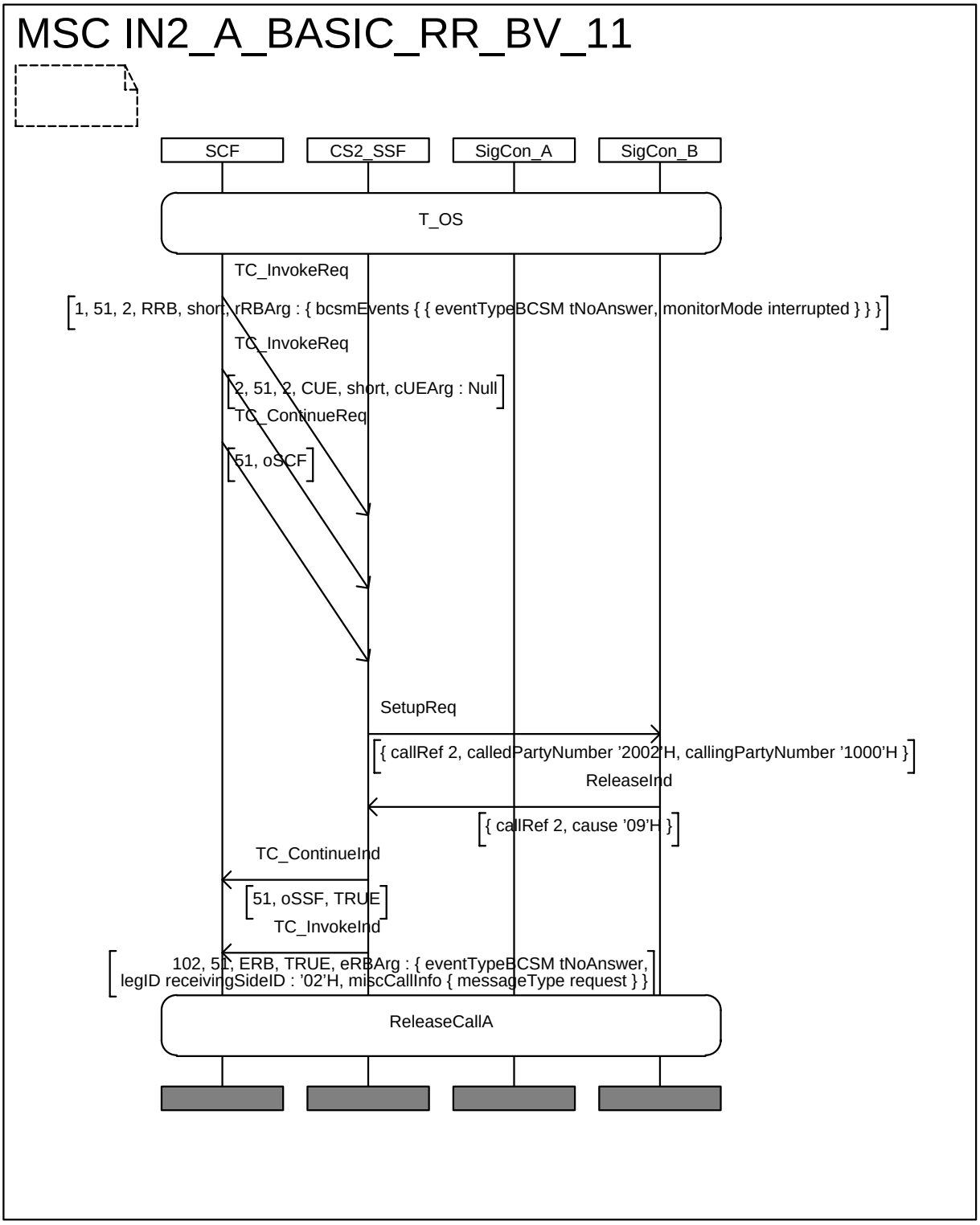
IN2_A_BASIC_RR_BV_09	
Purpose:	Test of **RequestReportBCSMEvent** procedure and oDisconnect indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oDisconnect - monitoringMode=interrupted 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
Postamble:	none


MSC IN2_A_BASIC_RR_BV_09
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
[1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }]
TC_InvokeReq
[2, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H }, callingPartyNumber '1000'H }]
TC_ContinueReq
[51, oSSF]
SetupReq
[{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }]
SetupConf
[{ callRef 2 }]
SetupResp
[{ callRef 1 }]
ReleaseInd
[{ callRef 1, cause '00'H }]
TC_ContinueInd
[51, oSSF, TRUE]
TC_InvokeInd
[102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oDisconnect, eventSpecificInformationBCSM oDisconnectSpecificInfo : { releaseCause '00'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }]
TC_InvokeReq
[6, 51, 4, CUE, medium, cUEArg : Null]
TC_ContinueReq
[51, oSSF]
TC_EndInd
[51, basic, FALSE]
ReleaseReq
[{ callRef 2, cause '10'H }, senderRef]

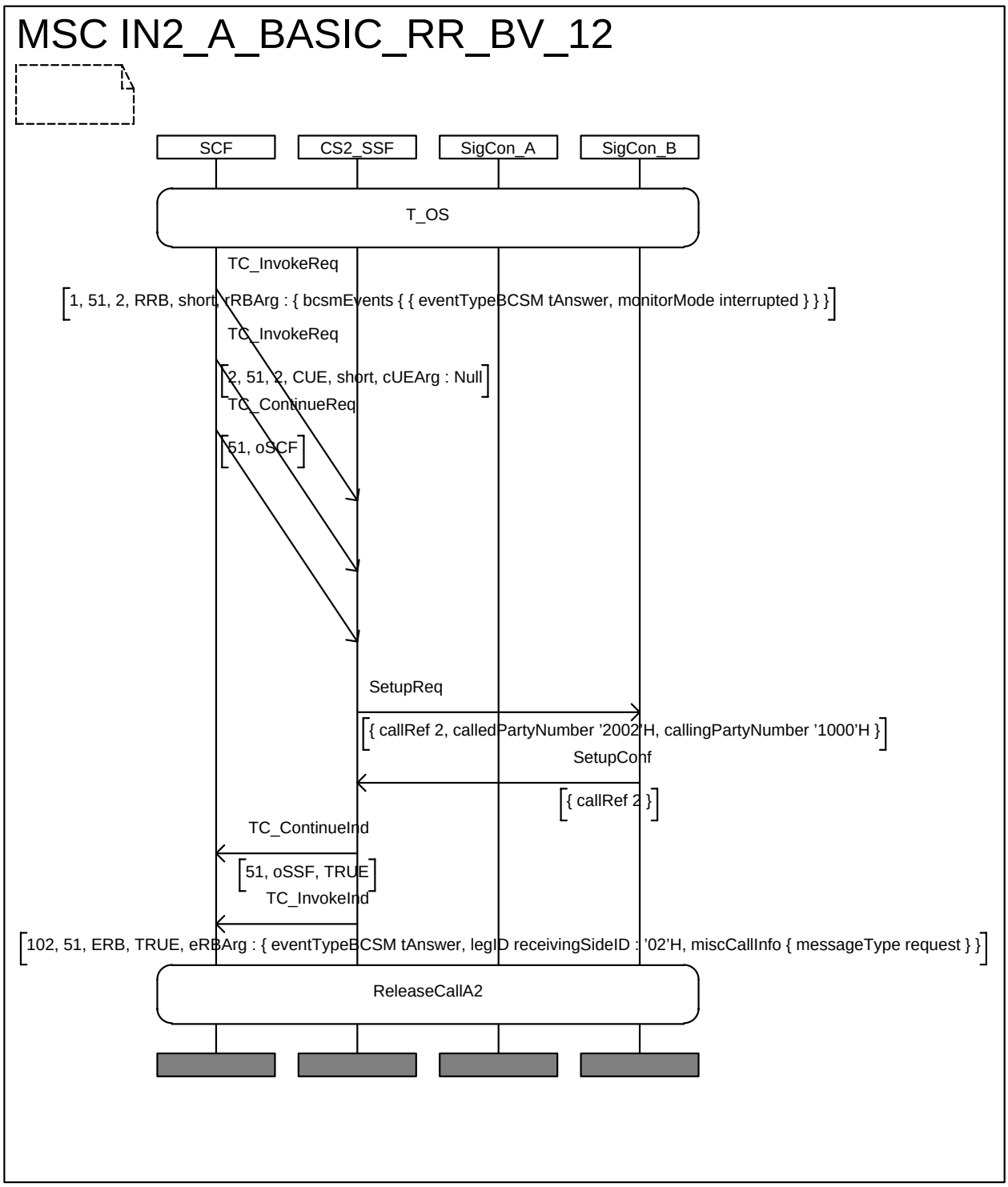
IN2_A_BASIC_RR_BV_10	
Purpose:	Test of **RequestReportBCSMEvent** procedure and tBusy indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS In addition, user B is declared busy
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tBusy - monitoringMode=interrupted followed by a **Continue** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call (**ReleaseInd** sent) with bPtyBusy_UDUB
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** tBusy
Postamble:	none


MSC IN2_A_BASIC_RR_BV_10
SCF
CS2_SSF
SigCon_A
SigCon_B
T_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { 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

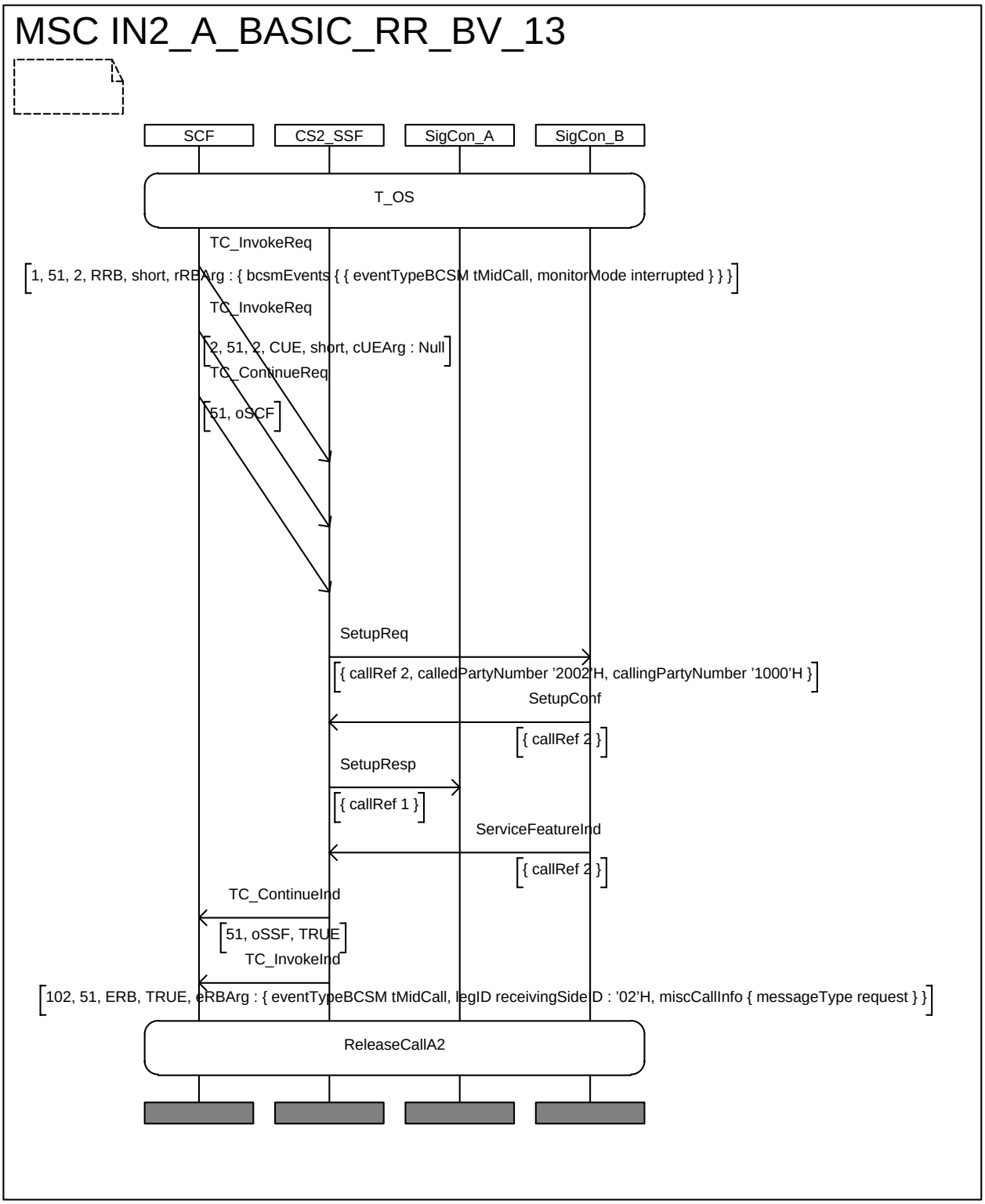
IN2_A_BASIC_RR_BV_11	
Purpose:	Test of **RequestReportBCSMEvent** procedure and tNoAnswer indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tNoAnswer - monitoringMode=interrupted followed by a **Continue** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call (**ReleaseInd** sent) because user B does not answer
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** tNoAnswer
Postamble:	ReleaseCallA


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_ContinueReq
51, oSCF
SetupReq
{ callRef 2, calledPartyNumber '2002'H, callingPartyNumber '1000'H }
ReleaseInd
{ callRef 2, cause '09'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tNoAnswer,
legID receivingSideID : '02'H, miscCallInfo { messageType request } }
ReleaseCallA

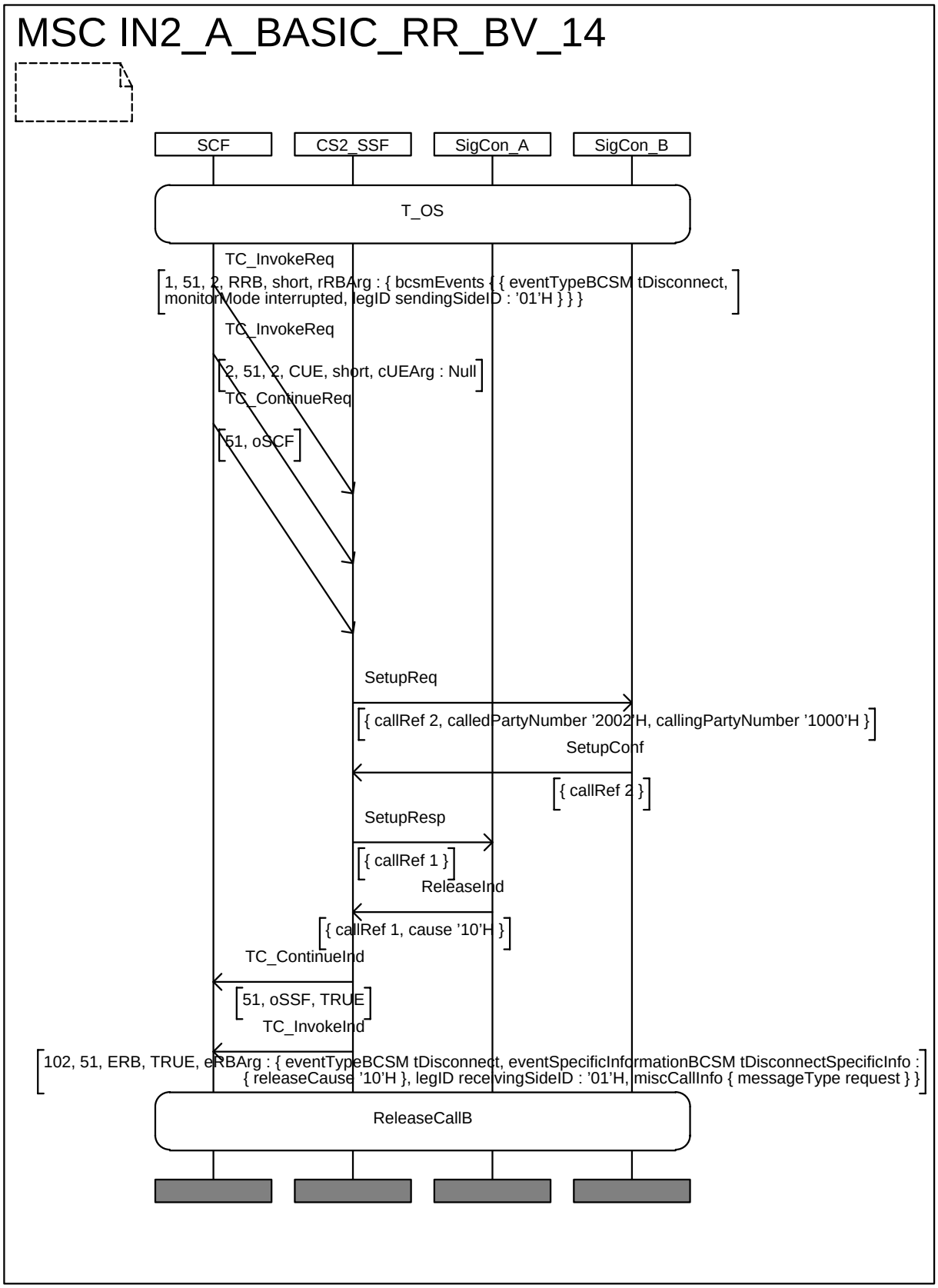
IN2_A_BASIC_RR_BV_12	
Purpose:	Test of **RequestReportBCSMEvent** procedure and tAnswer indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tAnswer - monitoringMode=interrupted followed 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:	ReleaseCallA


MSC IN2_A_BASIC_RR_BV_12
SCF
CS2_SSF
SigCon_A
SigCon_B
T_OS
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, 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 } }]
ReleaseCallA2

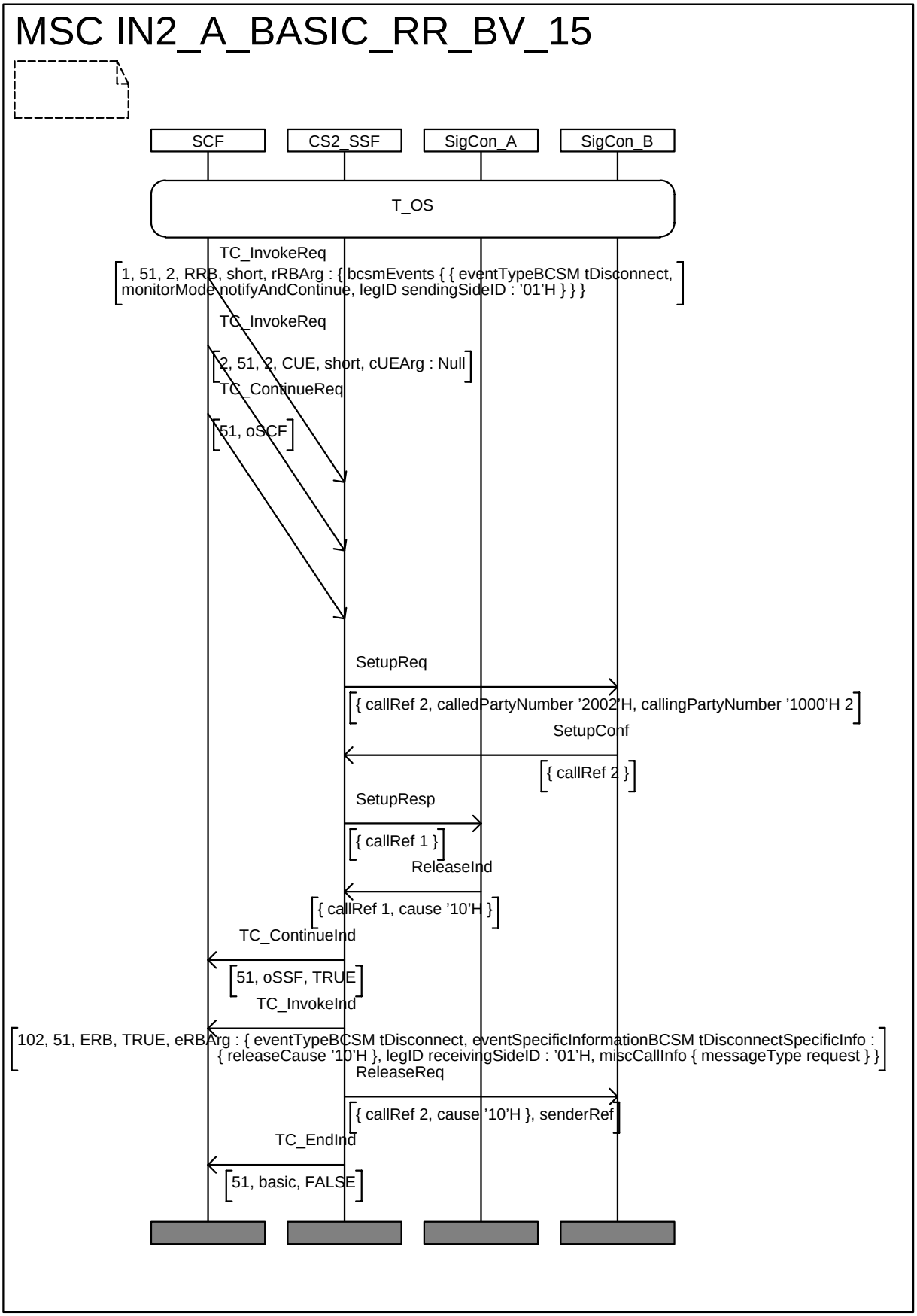
IN2_A_BASIC_RR_BV_13	
Purpose:	Test of **RequestReportBCSMEvent** procedure and tMidCall indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= tMidCall - monitoringMode=interrupted followed 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
SCF
CS2_SSF
SigCon_A
SigCon_B
T_OS
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
[{ callRef 2 }]
SetupResp
[{ callRef 1 }]
ServiceFeatureInd
[{ callRef 2 }]
TC_ContinueInd
[51, oSSF, TRUE]
TC_InvokeInd
[102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tMidCall, legID receivingSideID : '02'H, miscCallInfo { messageType request } }]
ReleaseCallA2

IN2_A_BASIC_RR_BV_14	
Purpose:	Test of **RequestReportBCSMEvent** procedure and tDisconnect indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF 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
SCF
CS2_SSF
SigCon_A
SigCon_B
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
{ callRef 2, calledPartyNumber '2002'H, callingPartyNumber '1000'H }
SetupConf
{ callRef 2 }
SetupResp
{ callRef 1 }
ReleaseInd
{ callRef 1, cause '10'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tDisconnect, eventSpecificInformationBCSM tDisconnectSpecificInfo : { releaseCause '10'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }
ReleaseCallB

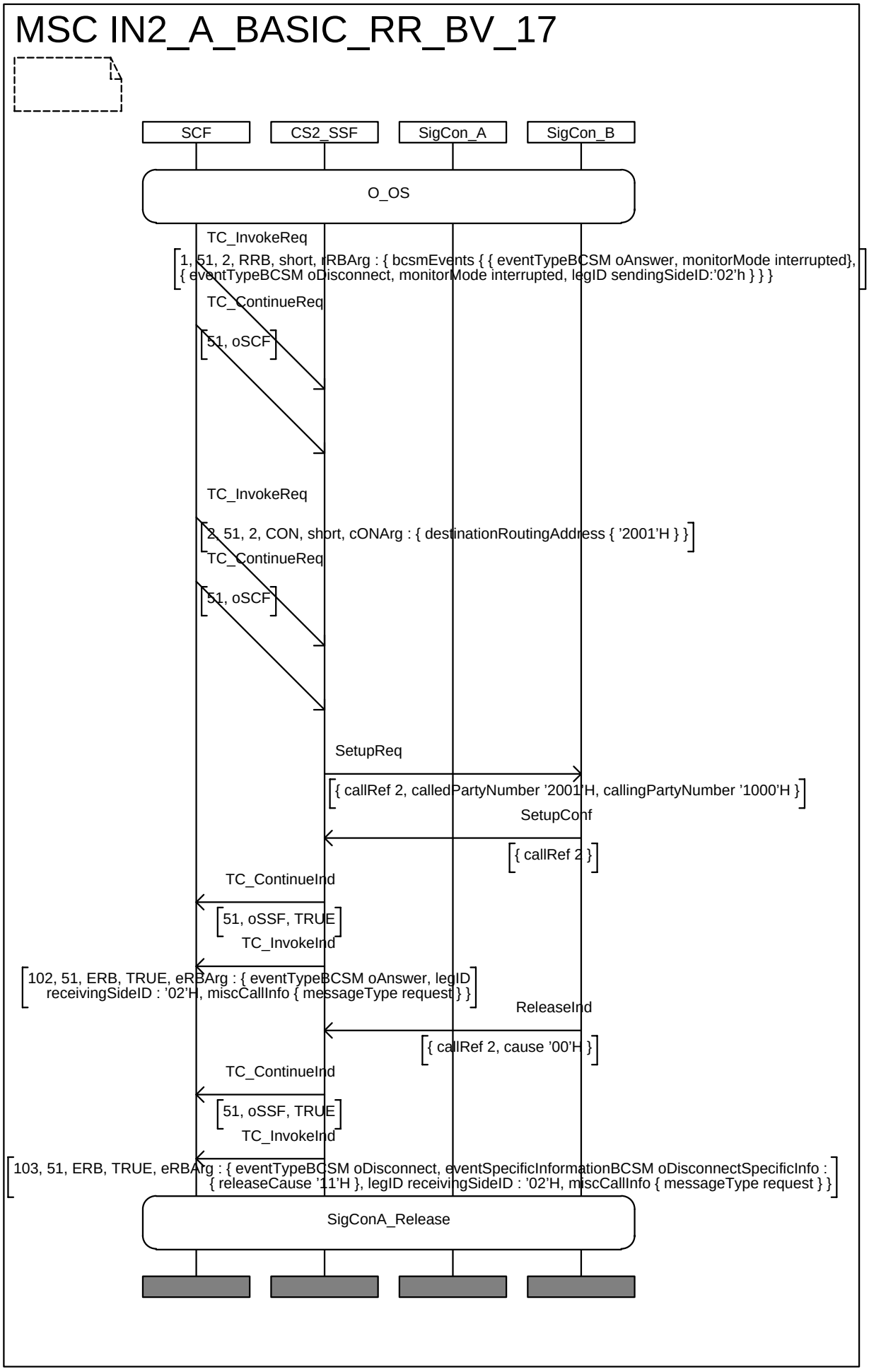
IN2_A_BASIC_RR_BV_15	
Purpose:	Test of **RequestReportBCSMEvent** procedure and tDisconnect indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= tDisconnect - monitoringMode=notifyAndContinue followed by a **Continue** invoke The IUT establishes the call, sends a SetUpReq to B side Then 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 - Check that SigConB is receiving a ReleaseReq to continue clearing the call
Postamble:	none


MSC IN2_A_BASIC_RR_BV_15
SCF
CS2_SSF
SigCon_A
SigCon_B
T_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM tDisconnect, monitorMode notifyAndContinue, legID sendingSideID : '01'H } } }
TC_InvokeReq
2, 51, 2, CUE, short, cUEArg : Null
TC_ContinueReq
51, oSCF
SetupReq
{ callRef 2, calledPartyNumber '2002'H, callingPartyNumber '1000'H 2
SetupConf
{ callRef 2 }
SetupResp
{ callRef 1 }
ReleaseInd
{ callRef 1, cause '10'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tDisconnect, eventSpecificInformationBCSM tDisconnectSpecificInfo : { releaseCause '10'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }
ReleaseReq
{ callRef 2, cause '10'H }, senderRef
TC_EndInd
51, basic, FALSE

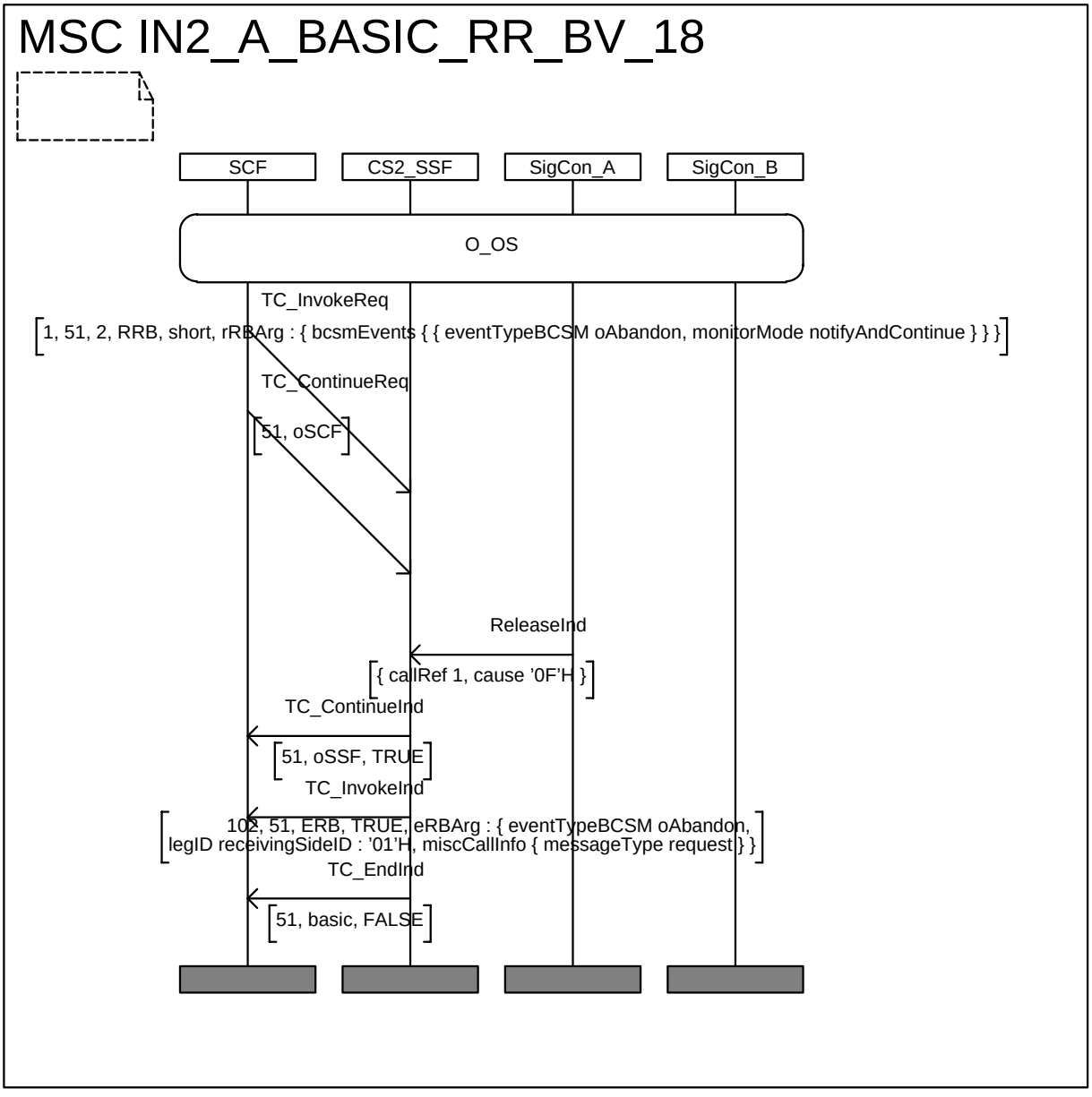
IN2_A_BASIC_RR_BV_16

This TP has been deleted.

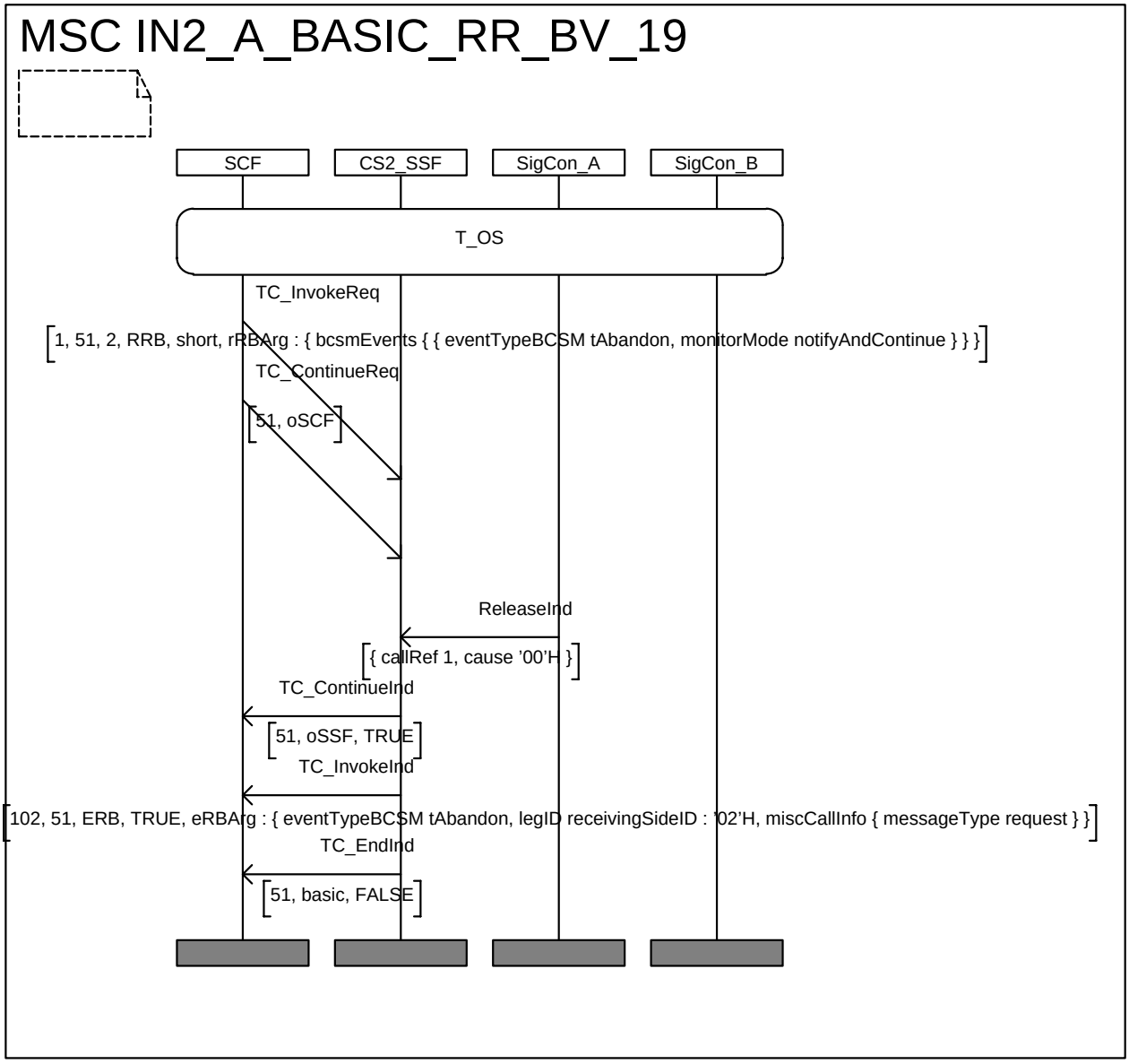
IN2_A_BASIC_RR_BV_17	
Purpose:	Test of **RequestReportBCSMEvent** procedure and oTriggers
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF - SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oAnswer - monitoringMode=interrupted - eventTypeBCSM= oDisconnect - monitoringMode=interrupted followed by Continue invoke - SSF calls SigConB (SetupReq answered with SetupConf)
Pass criteria	- Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** oAnswer - When SigConB is releasing the call (ReleaseInd sent), check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** oDisconnect
Postamble:	SigConA_Release


MSC IN2_A_BASIC_RR_BV_17
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted},
{ eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID:'02'h } } }
TC_ContinueReq
51, oSCF
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 } }
ReleaseInd
{ callRef 2, cause '00'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
103, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oDisconnect, eventSpecificInformationBCSM oDisconnectSpecificInfo :
{ releaseCause '11'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }
SigConA_Release

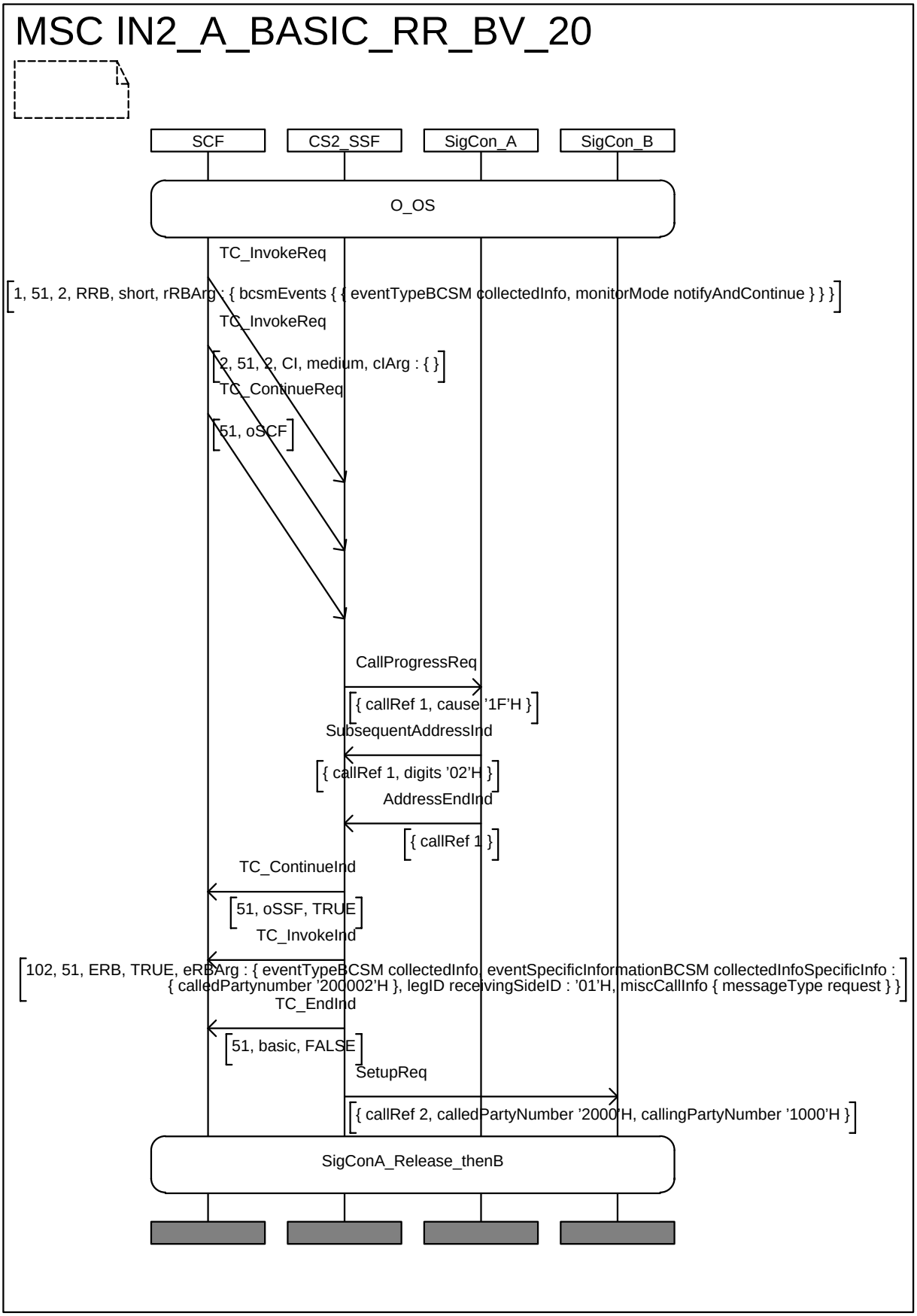
XXXX	IN2_A_BASIC_RR_BV_18
Purpose:	Test of **RequestReportBCSMEvent** procedure and oAbandon
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oAbandon - monitoringMode=notifyAndContinue then 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=oAbandon**
Postamble:	none



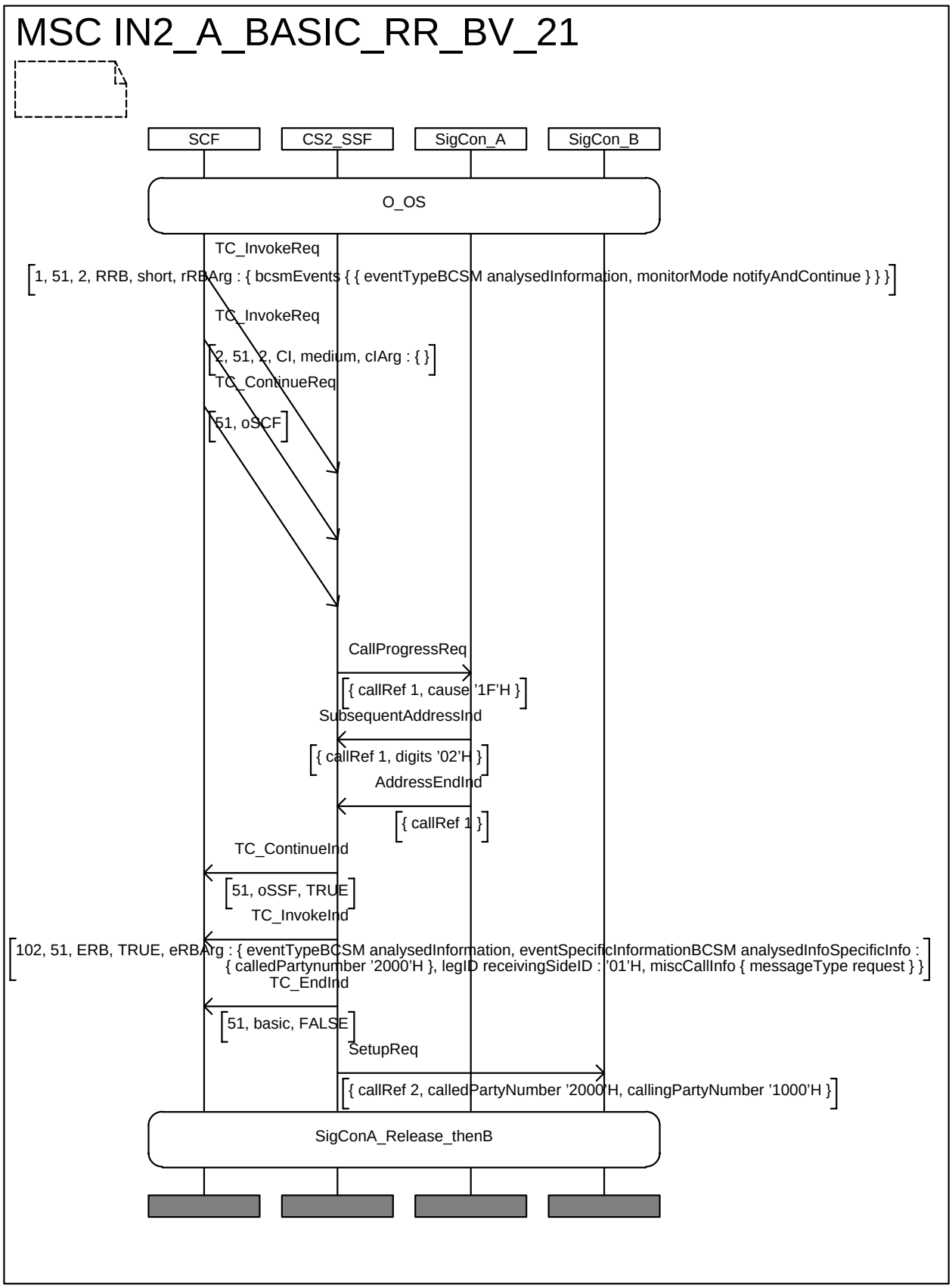
XXXX	IN2_A_BASIC_RR_BV_19
Purpose:	Test of **RequestReportBCSMEvent** procedure and tAbandon
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tAbandon - monitoringMode=notifyAndContinue then 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



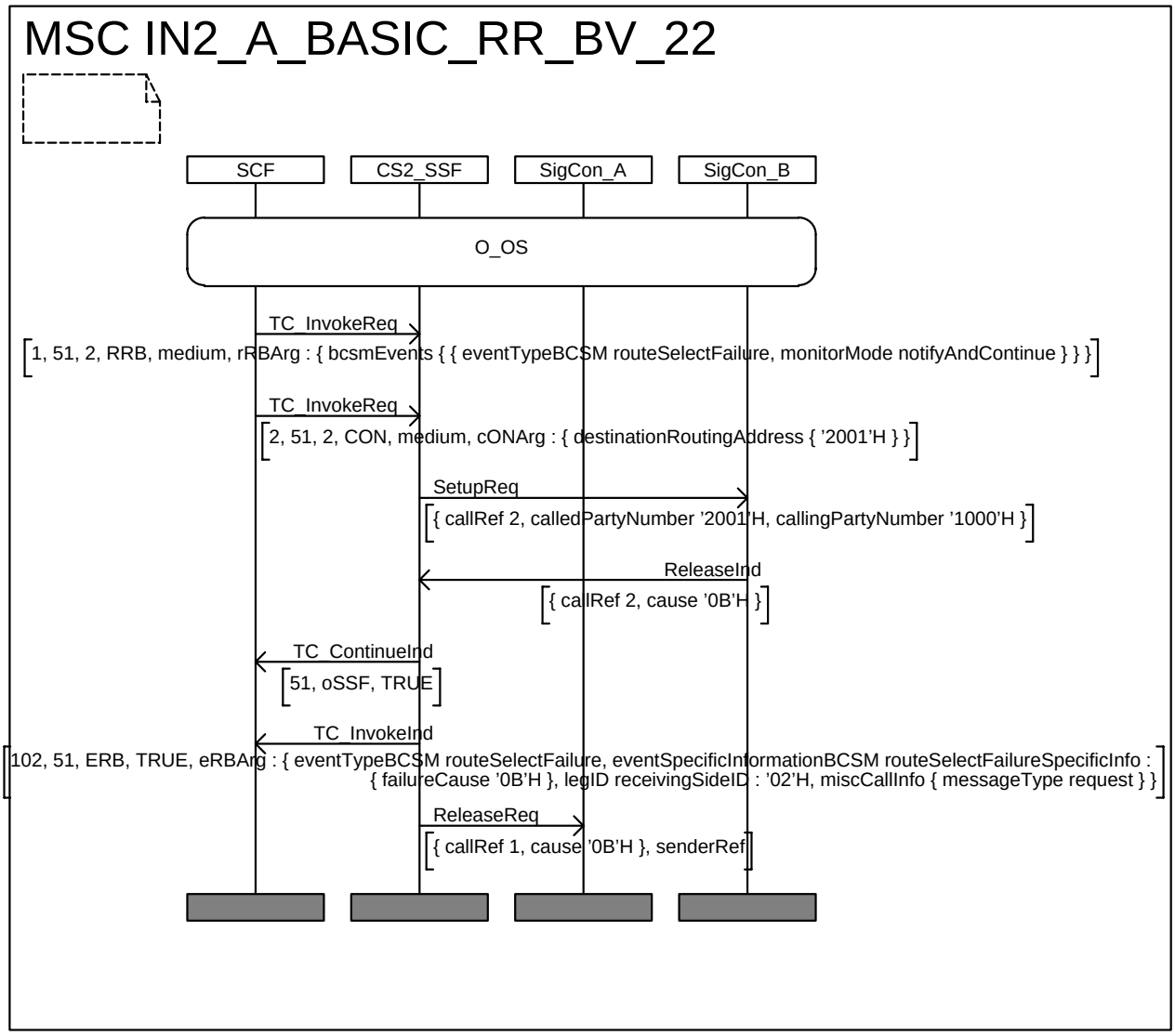
XXXX	IN2_A_BASIC_RR_BV_20
Purpose:	Test of **RequestReportBCSMEvent** procedure and collectedInfo indication
Requirement ref	
Selection Cond.	
Preamble:	O_OS Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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
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 } }
TC_EndInd
51, basic, FALSE
SetupReq
{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
SigConA_Release_thenB

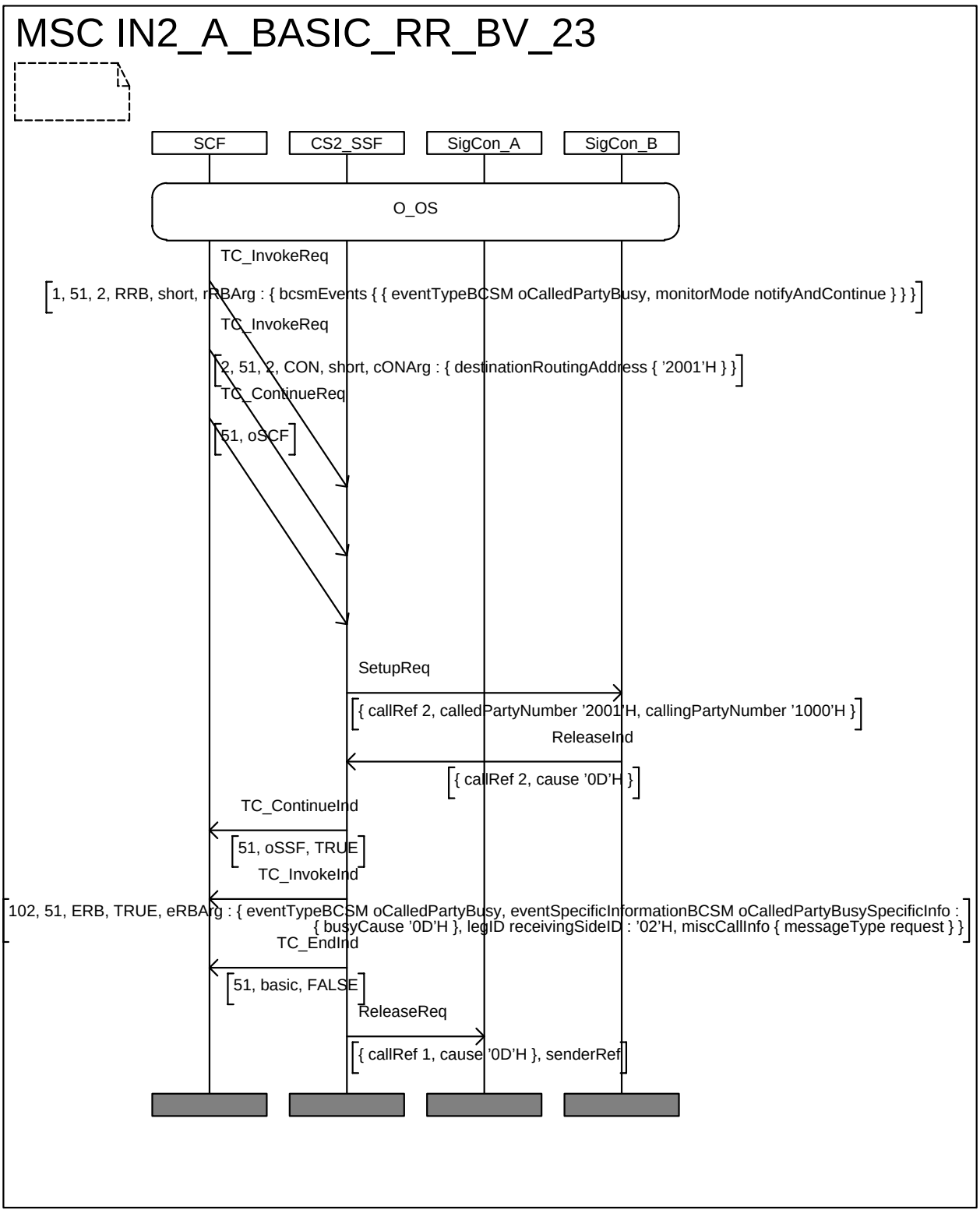
XXXX	IN2_A_BASIC_RR_BV_21
Purpose:	Test of **RequestReportBCSMEvent** procedure and analysedInfo indication
Requirement ref	
Selection Cond.	
Preamble:	O_OS Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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 '1F'H }]
SubsequentAddressInd
[{ callRef 1, digits '02'H }]
AddressEndInd
[{ callRef 1 }]
TC_ContinueInd
[51, oSSF, TRUE]
TC_InvokeInd
[102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM analysedInformation, eventSpecificInformationBCSM analysedInfoSpecificInfo :
{ calledPartynumber '2000'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }]
TC_EndInd
[51, basic, FALSE]
SetupReq
[{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
SigConA_Release_thenB

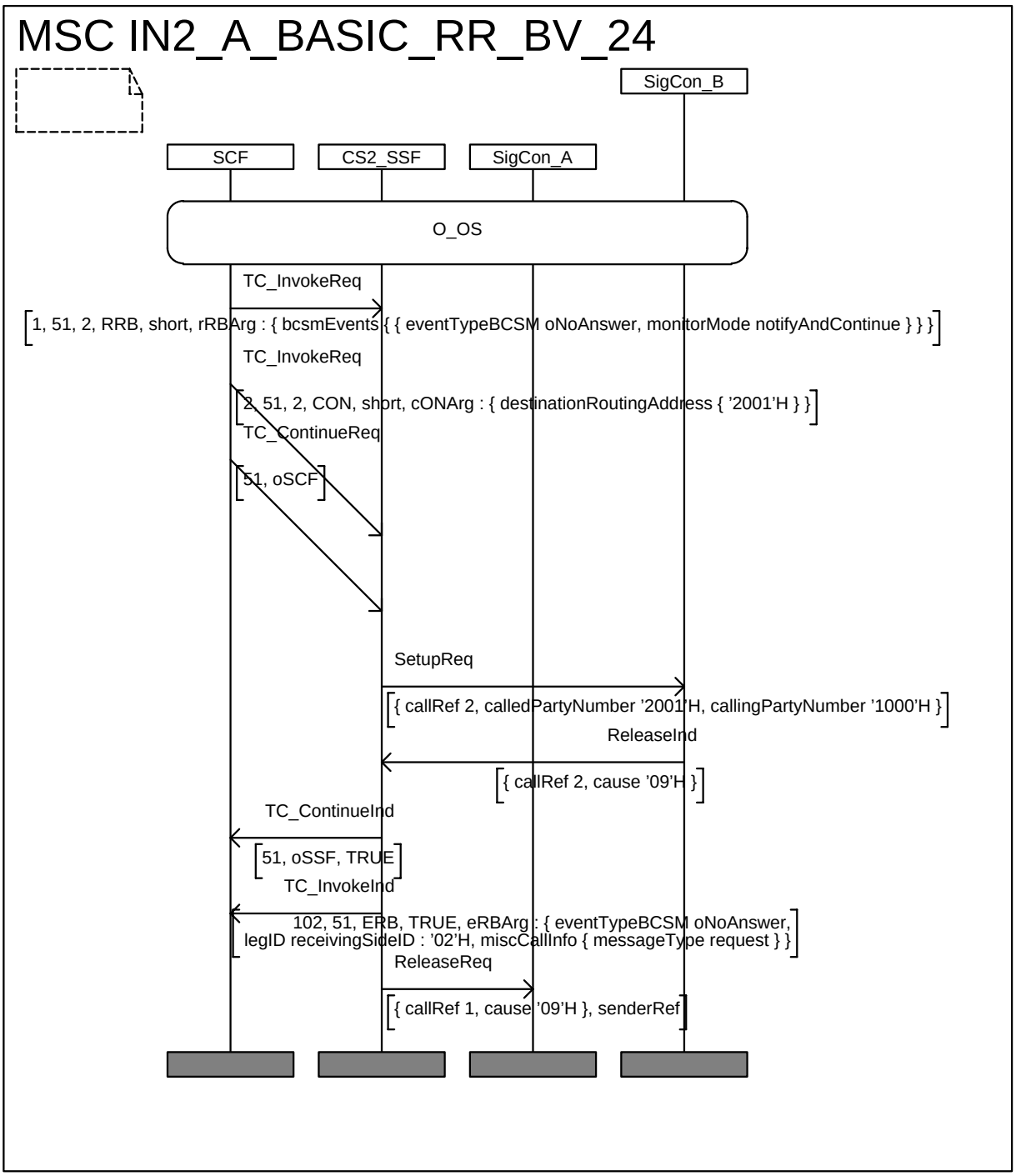
XXXX	IN2_A_BASIC_RR_BV_22
Purpose:	Test of **RequestReportBCSMEvent** procedure and routeSelectFailure indication
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=routeSelectFailure - monitoringMode= 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



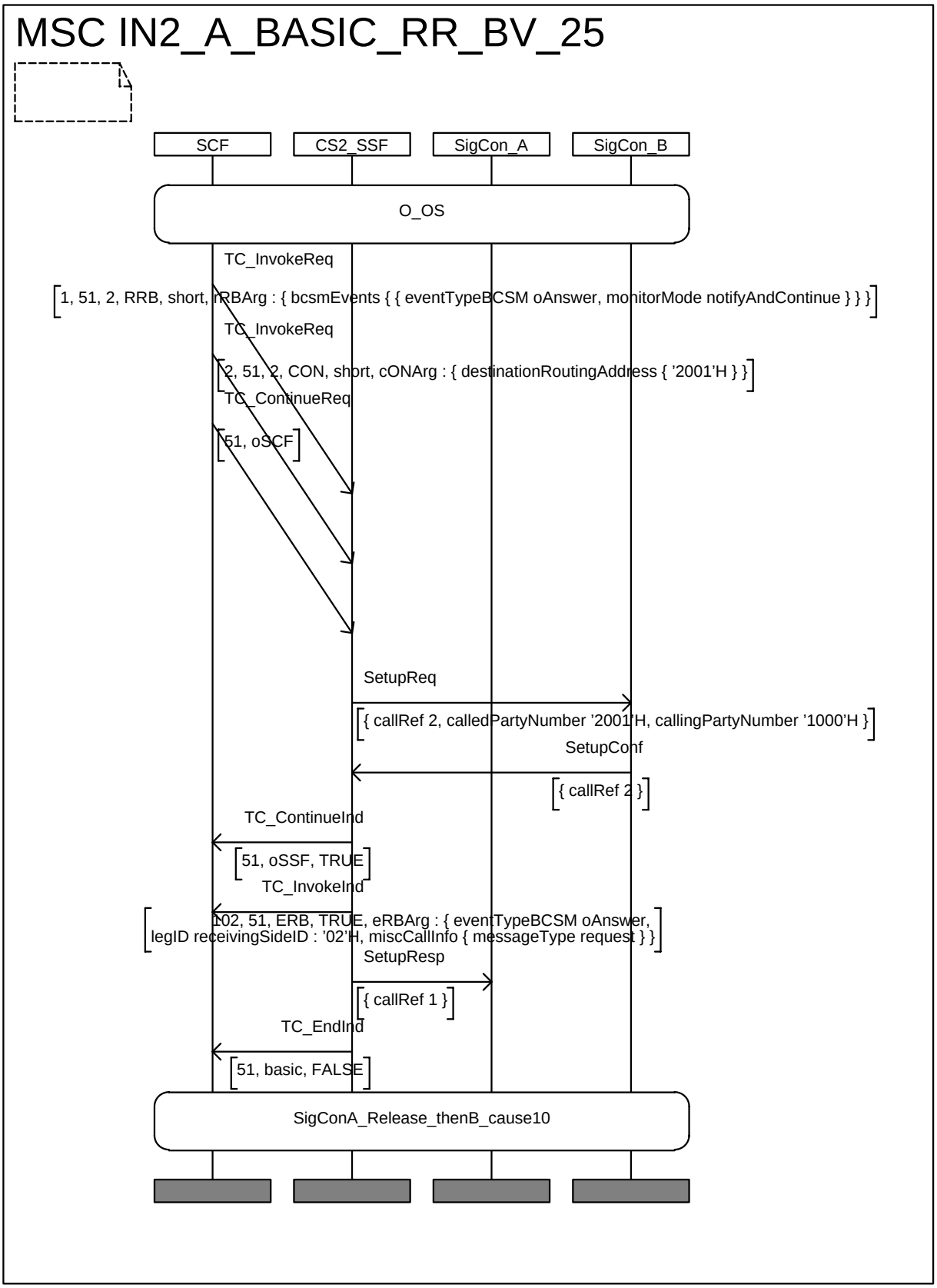
XXXX	IN2_A_BASIC_RR_BV_23
Purpose:	Test of **RequestReportBCSMEvent** procedure and oCalledPartyBusy indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS In addition, user B is declared busy
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oCalledPartyBusy - 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
[1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oCalledPartyBusy, 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 }]
ReleaseInd
[{ callRef 2, cause '0D'H }]
TC_ContinueInd
[51, oSSF, TRUE]
TC_InvokeInd
[102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oCalledPartyBusy, eventSpecificInformationBCSM oCalledPartyBusySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }]
TC_EndInd
[51, basic, FALSE]
ReleaseReq
[{ callRef 1, cause '0D'H }, senderRef]

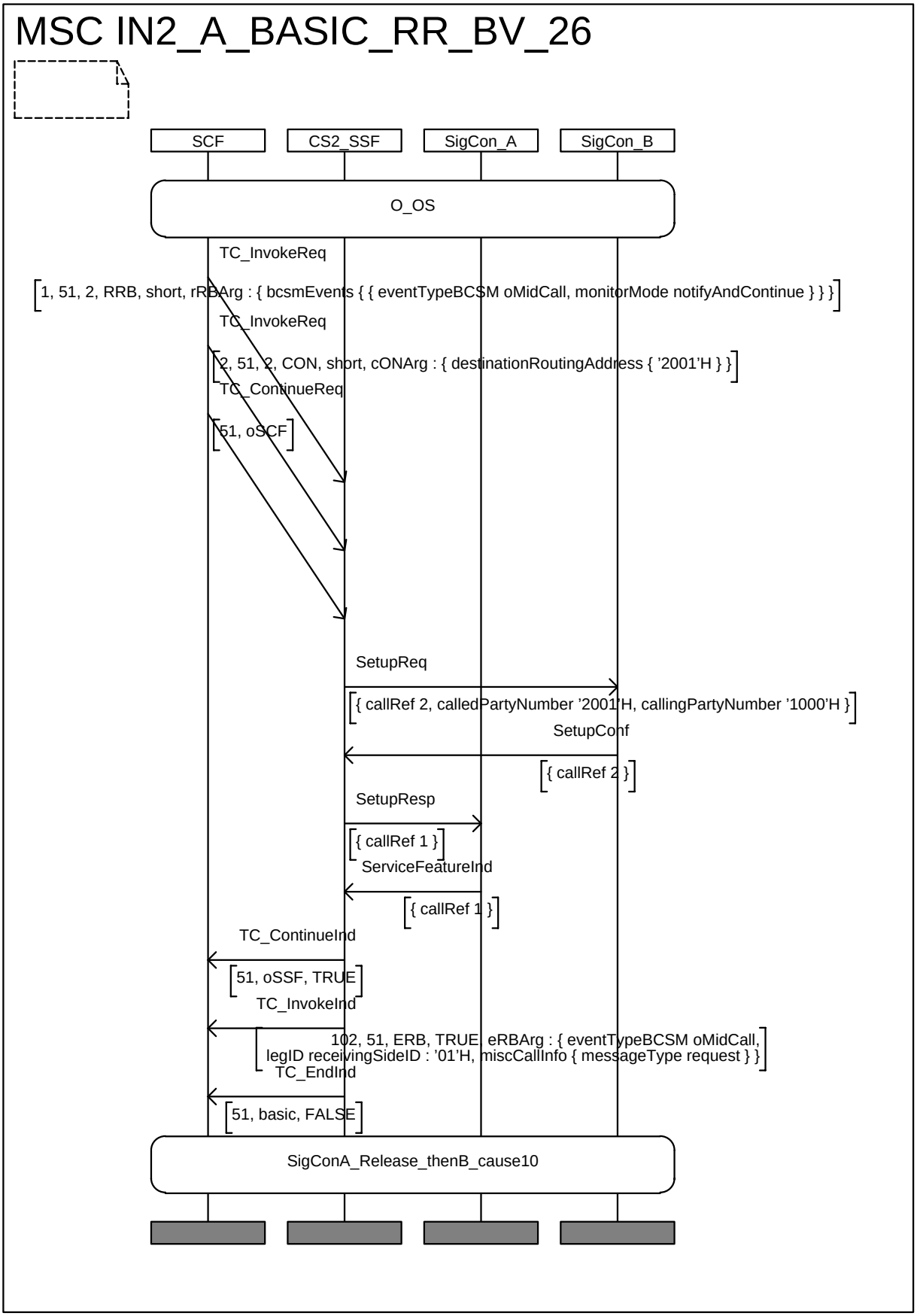
XXXX	IN2_A_BASIC_RR_BV_24
Purpose:	Test of **RequestReportBCSMEvent** procedure and oNoAnswer indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oNoAnswer - monitoringMode= notifyAndContinue 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
SigCon_B
SCF
CS2_SSF
SigCon_A
O_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oNoAnswer, 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 }
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 } }
ReleaseReq
{ callRef 1, cause '09'H }, senderRef

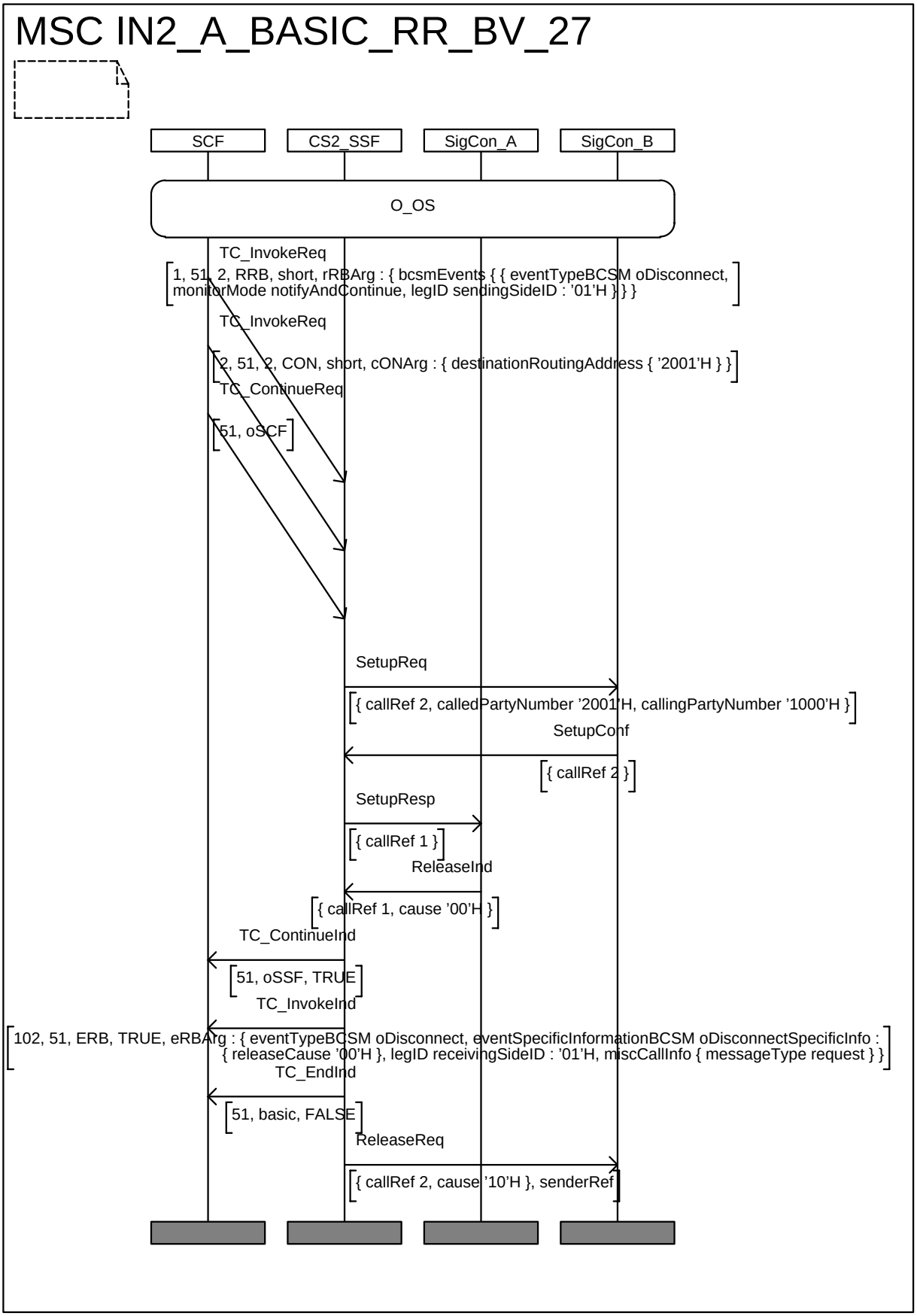
XXXX	IN2_A_BASIC_RR_BV_25
Purpose:	Test of **RequestReportBCSMEvent** procedure and oAnswer indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oAnswer - monitoringMode= notifyAndContinue 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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

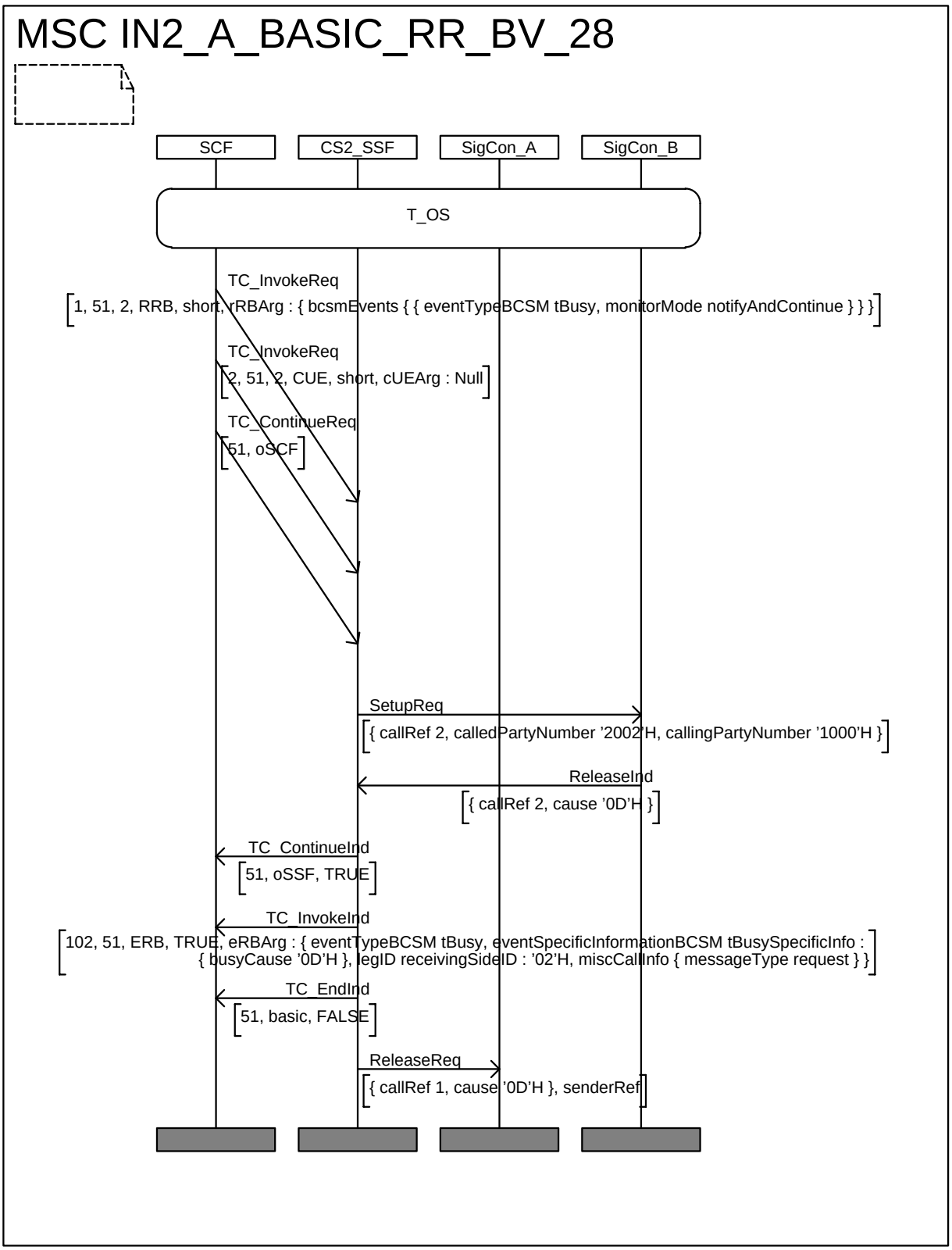
XXXX	IN2_A_BASIC_RR_BV_26
Purpose:	Test of **RequestReportBCSMEvent** procedure and oMidCall indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oMidCall - monitoringMode= notifyAndContinue 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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 }
SetupConf
{ callRef 2 }
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

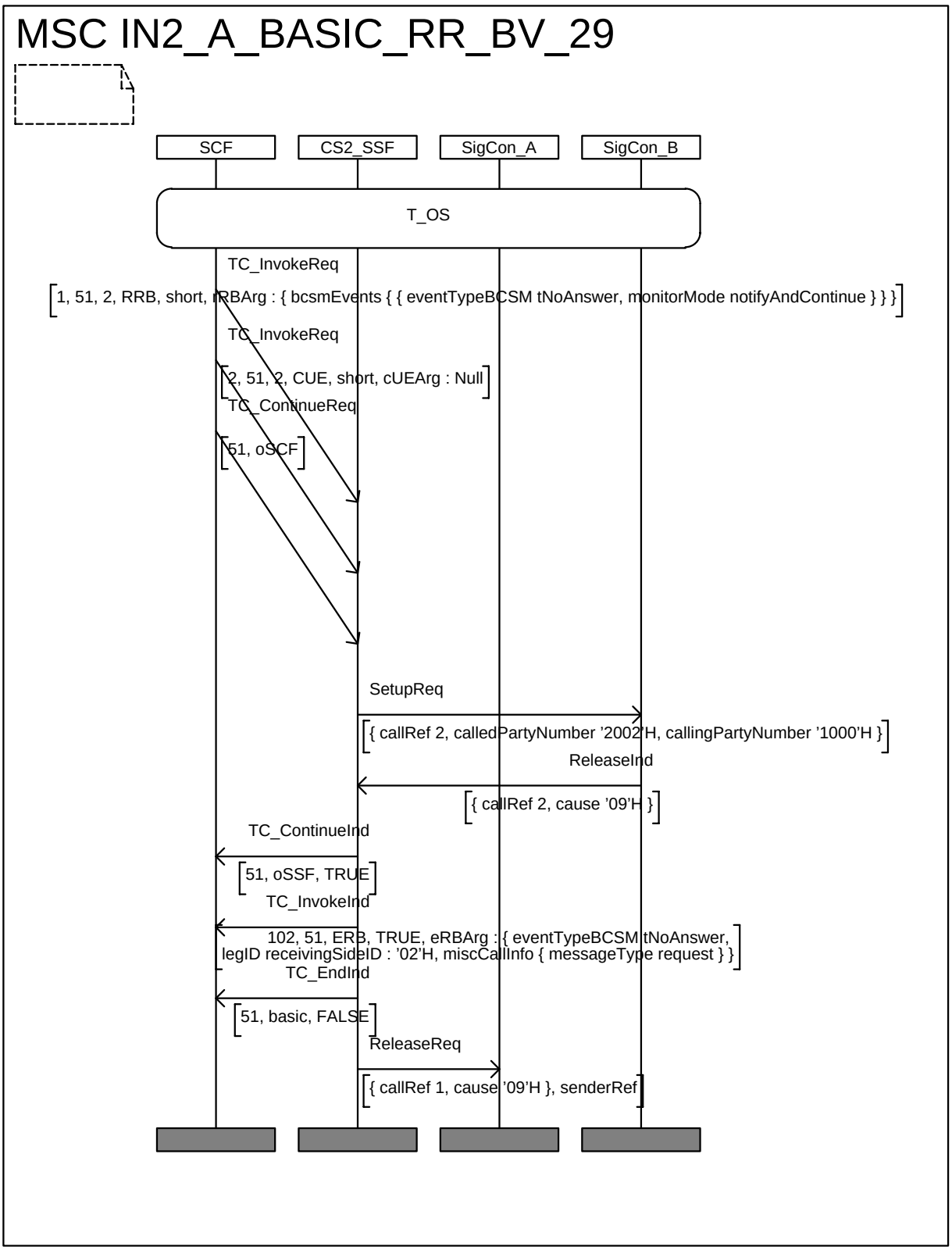
XXXX	IN2_A_BASIC_RR_BV_27
Purpose:	Test of **RequestReportBCSMEvent** procedure and oDisconnect indication.
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oDisconnect - monitoringMode= notifyAndContinue - legID=sendingSideID : "01"H 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 sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** oDisconnect
Postamble:	none


MSC IN2_A_BASIC_RR_BV_27
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode notifyAndContinue, legID 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
{ callRef 2 }
SetupResp
{ callRef 1 }
ReleaseInd
{ callRef 1, cause '00'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM oDisconnect, eventSpecificInformationBCSM oDisconnectSpecificInfo : { releaseCause '00'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }
TC_EndInd
51, basic, FALSE
ReleaseReq
{ callRef 2, cause '10'H }, senderRef

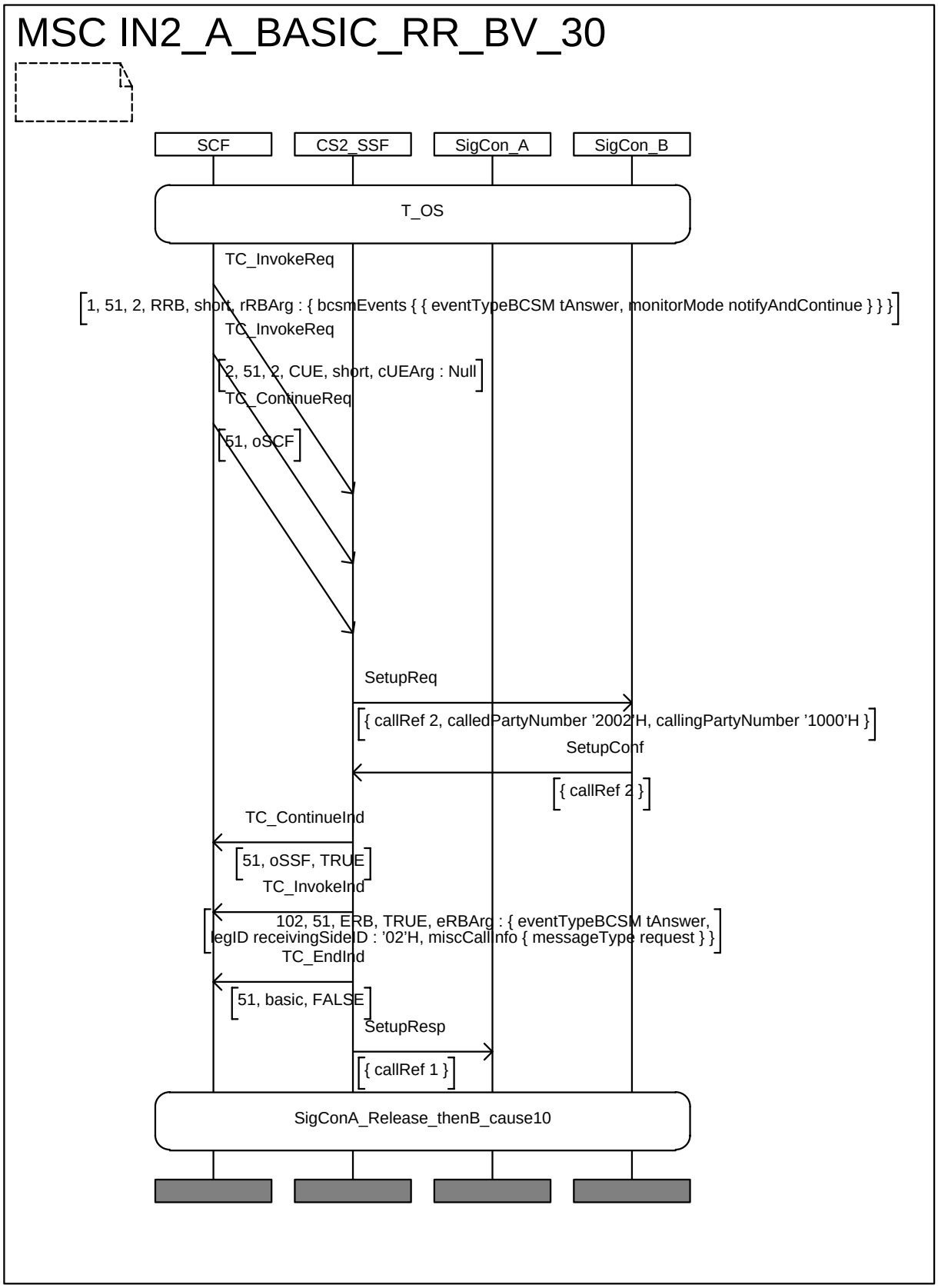
XXXX	IN2_A_BASIC_RR_BV_28
Purpose:	Test of **RequestReportBCSMEvent** procedure and tBusy indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS In addition, user B is declared busy
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tBusy - 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
Postamble:	none


MSC IN2_A_BASIC_RR_BV_28
SCF
CS2_SSF
SigCon_A
SigCon_B
T_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM tBusy, monitorMode notifyAndContinue } } }
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 } }
TC_EndInd
51, basic, FALSE
ReleaseReq
{ callRef 1, cause '0D'H }, senderRef

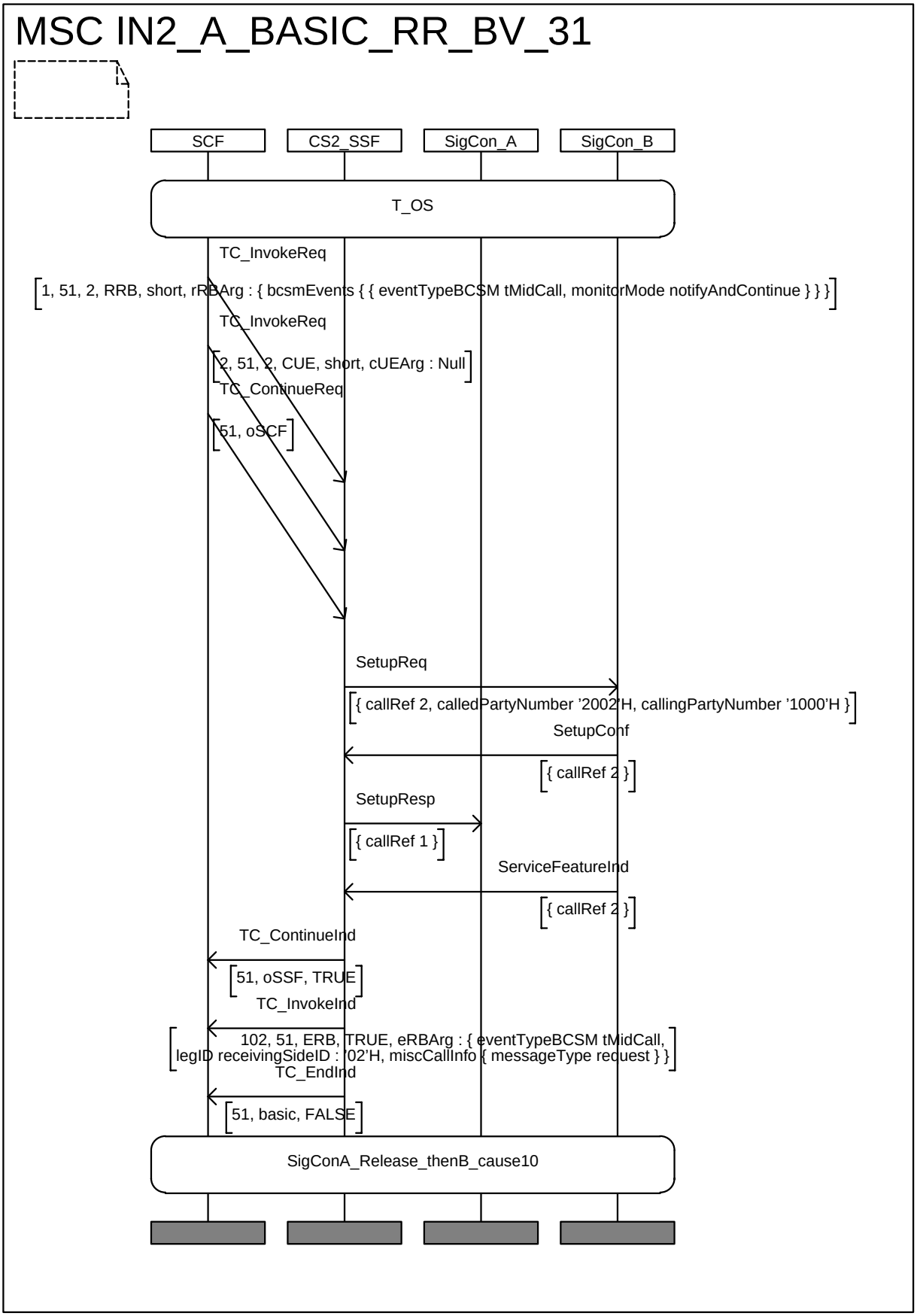
XXXX	IN2_A_BASIC_RR_BV_29
Purpose:	Test of **RequestReportBCSMEvent** procedure and tNoAnswer indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tNoAnswer - monitoringMode= notifyAndContinue followed by a **Continue** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call (**ReleaseInd** sent) because user B does not answer
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** tNoAnswer
Postamble:	none


MSC IN2_A_BASIC_RR_BV_29
SCF
CS2_SSF
SigCon_A
SigCon_B
T_OS
TC_InvokeReq
1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM tNoAnswer, monitorMode notifyAndContinue } } }
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 '09'H }
TC_ContinueInd
51, oSSF, TRUE
TC_InvokeInd
102, 51, ERB, TRUE, eRBArg : { eventTypeBCSM tNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }
TC_EndInd
51, basic, FALSE
ReleaseReq
{ callRef 1, cause '09'H }, senderRef

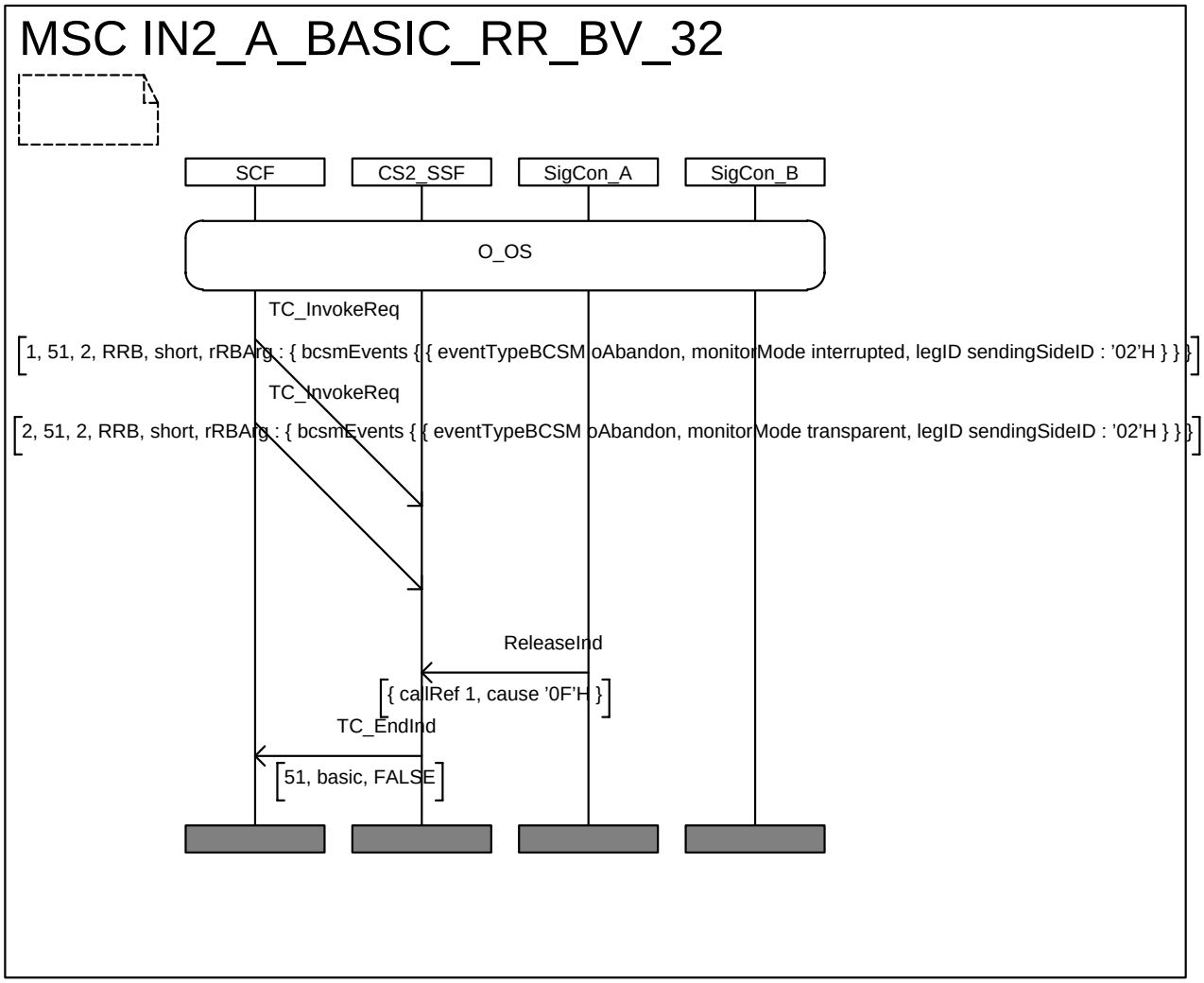
XXXX	IN2_A_BASIC_RR_BV_30
Purpose:	Test of **RequestReportBCSMEvent** procedure and tAnswer indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tAnswer - monitoringMode= notifyAndContinue 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
SCF
CS2_SSF
SigCon_A
SigCon_B
T_OS
TC_InvokeReq
[1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { 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

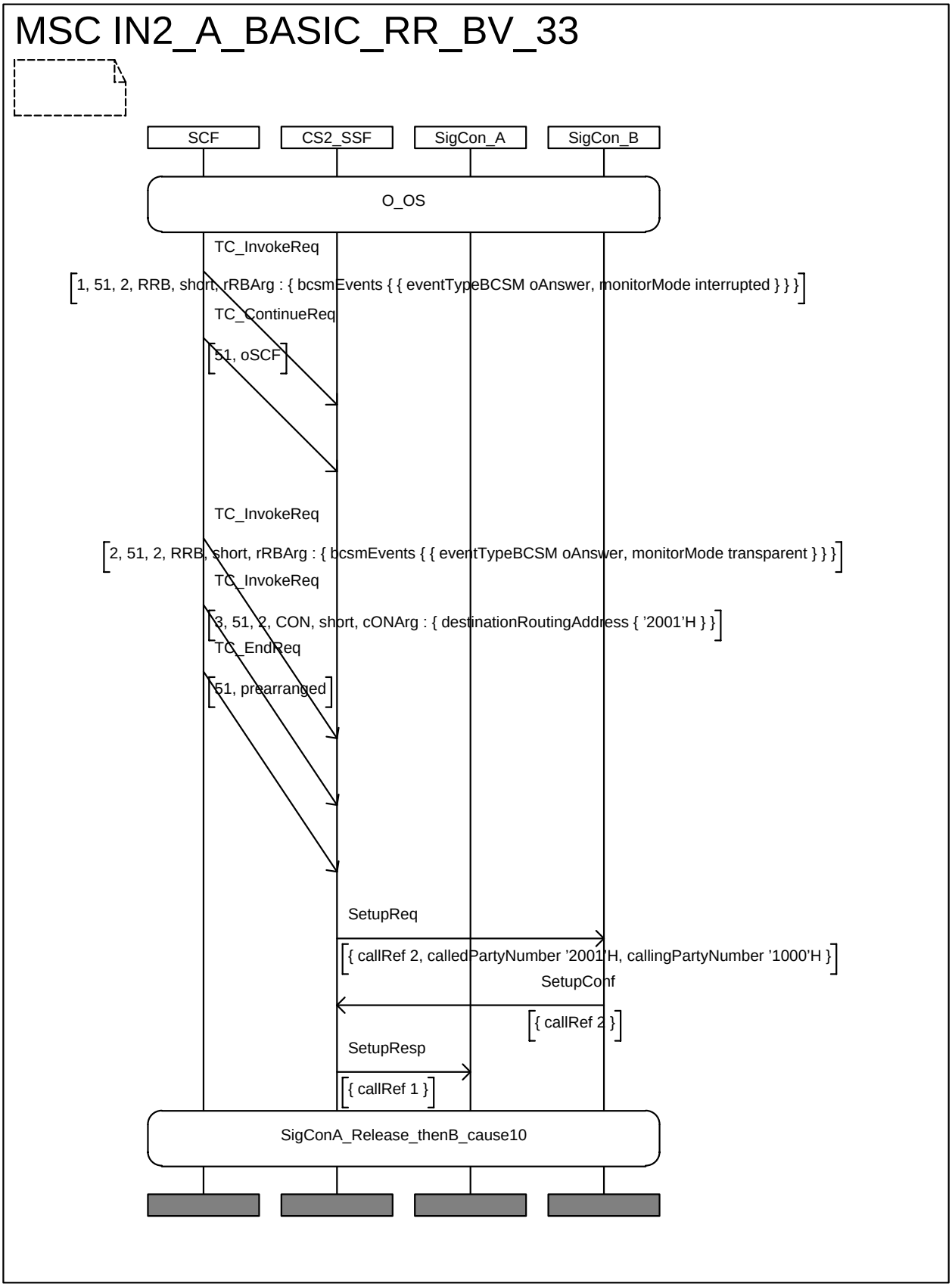
XXXX	IN2_A_BASIC_RR_BV_31
Purpose:	Test of **RequestReportBCSMEvent** procedure and tMidCall indication.
Requirement ref	
Selection Cond.	
Preamble:	T_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= tMidCall - monitoringMode= notifyAndContinue 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
SCF
CS2_SSF
SigCon_A
SigCon_B
T_OS
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,
legID receivingSideID : '02'H, miscCallInfo { messageType request } }
TC_EndInd
51, basic, FALSE
SigConA_Release_thenB_cause10

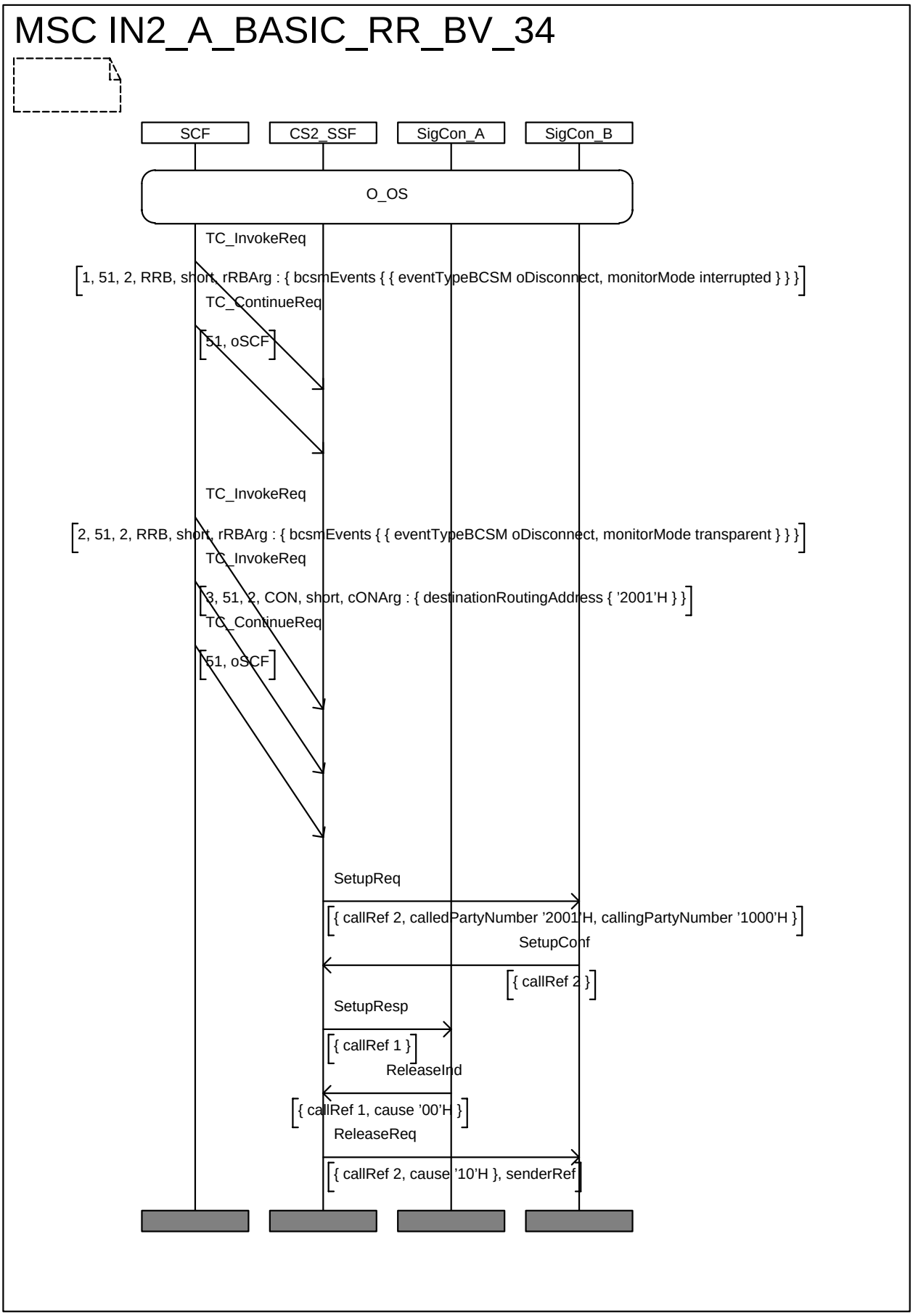
XXXX	IN2_A_BASIC_RR_BV_32
Purpose:	Test of **RequestReportBCSMEvent** procedure and oAbandon - transparent
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - - eventTypeBCSM=oAbandon - monitoringMode=interrupted 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



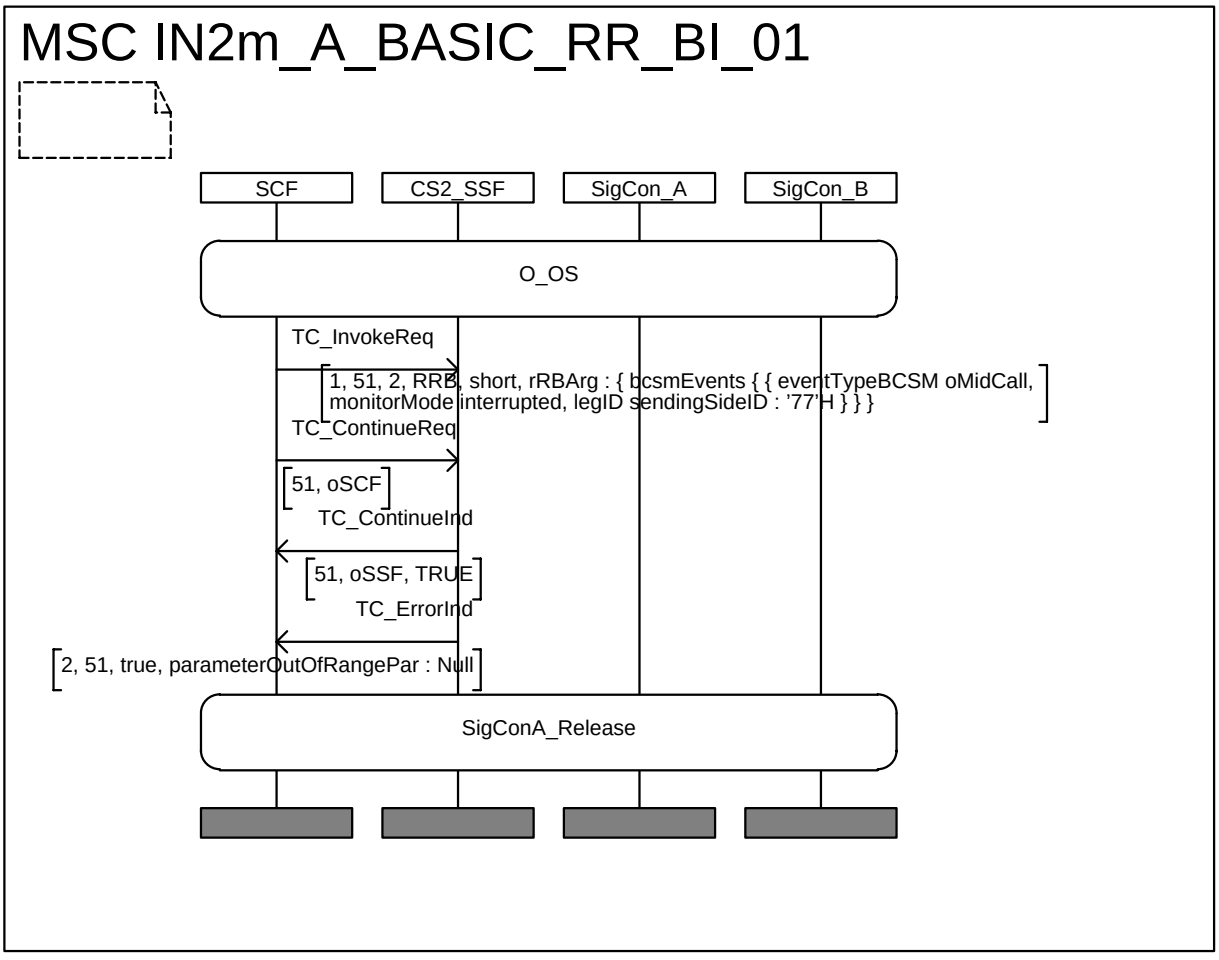
XXXX	IN2_A_BASIC_RR_BV_33
Purpose:	Test of **RequestReportBCSMEvent** procedure and oAnswer - transparent
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - - eventTypeBCSM=oAnswer - monitoringMode=interrupted 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


MSC IN2_A_BASIC_RR_BV_33
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
[1, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted } } }]
TC_ContinueReq
[51, oSCF]
TC_InvokeReq
[2, 51, 2, RRB, short, rRBArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode transparent } } }]
TC_InvokeReq
[3, 51, 2, CON, short, cONArg : { destinationRoutingAddress { '2001'H } }]
TC_EndReq
[51, prearranged]
SetupReq
[{ callRef 2, calledPartyNumber '2001'H, callingPartyNumber '1000'H }]
SetupConf
[{ callRef 2 }]
SetupResp
[{ callRef 1 }]
SigConA_Release_thenB_cause10

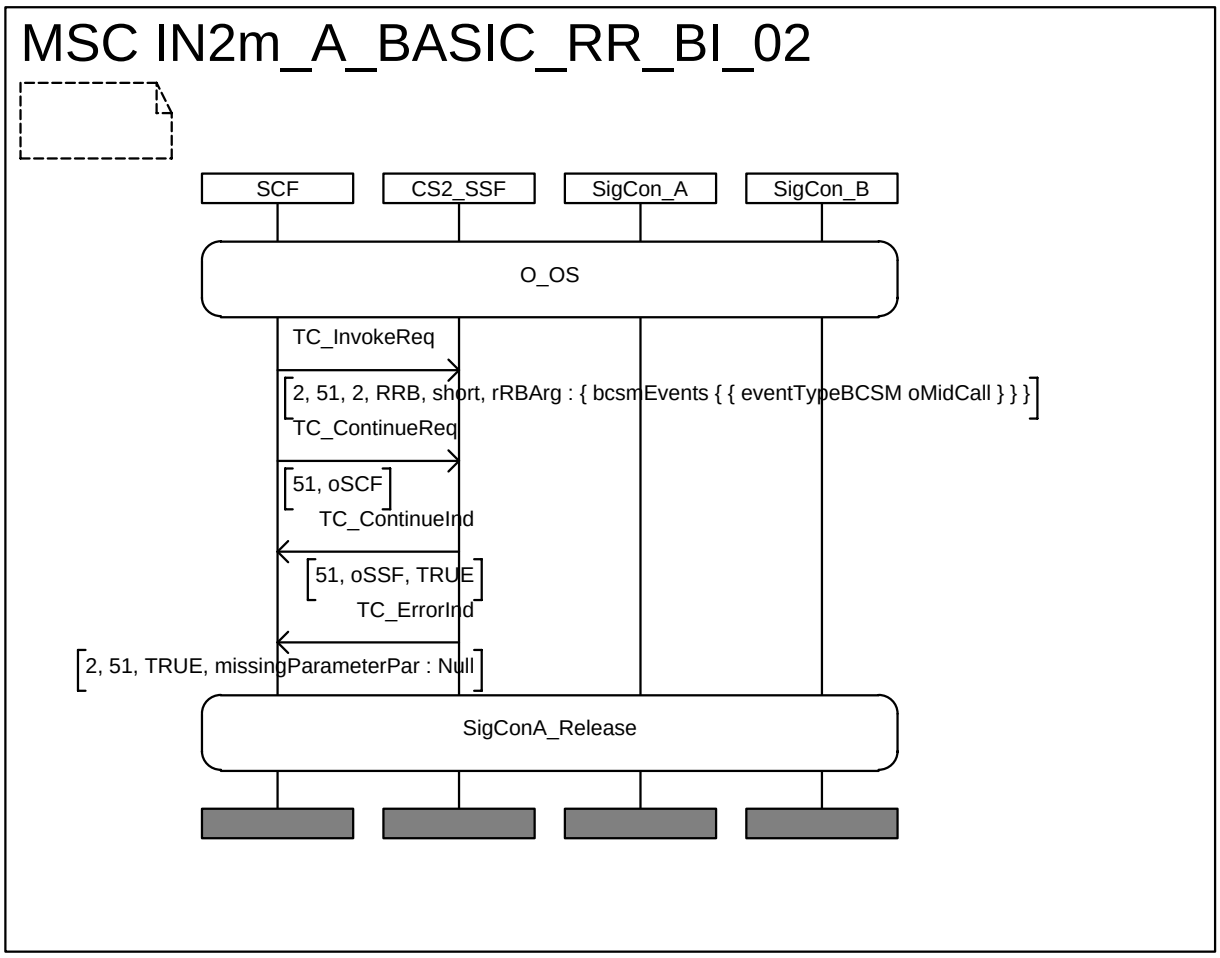
XXXX	IN2_A_BASIC_RR_BV_34
Purpose:	Test of **RequestReportBCSMEvent** procedure and oDisconnect indication - transparent
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oDisconnect - monitoringMode= 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
[3, 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]

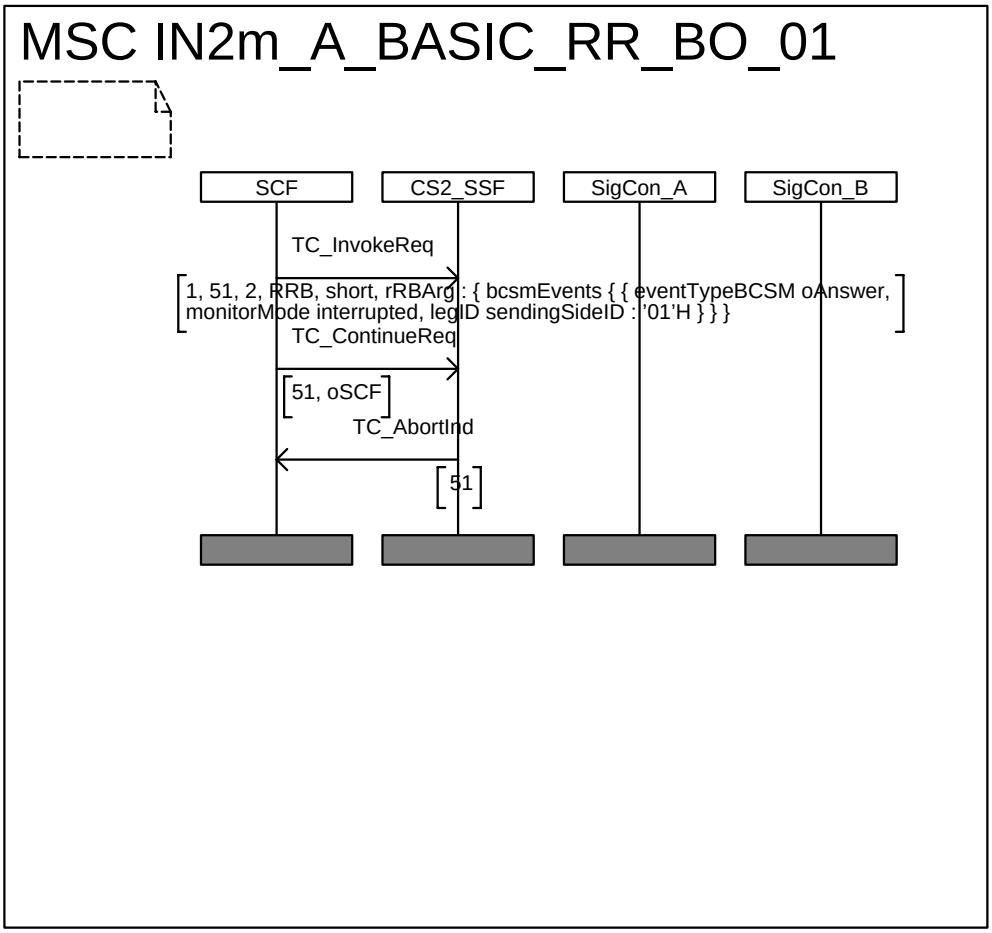
IN2_A_BASIC_RR_BI_01	
Purpose:	Test of **RequestReportBCSMEvent** procedure and out of range parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF - SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oMidCall - monitoringMode=interrupted - legID=invalid value
Pass criteria	- Check that SSF sends to SCF a **RequestReportBCSMEvent** error with the indication of **out of range parameter** - When call Setup is established, check that SSF is not sending to SCF any **EventReportBCSM**
Postamble:	SigConA_Release



IN2_A_BASIC_RR_BI_02	
Purpose:	Test of **RequestReportBCSMEvent** procedure and missing parameter
Requirement ref	
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



IN2_A_BASIC_RR_BO_01	
Purpose:	Test of **RequestReportBCSMEvent** procedure in wrong state
Requirement ref	
Selection Cond.	
Preamble:	none
Test description	SCF - SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oAnswer - monitoringMode=interrupted
Pass criteria	- Check that SSF sends a **TC-ABORT**
Postamble:	none

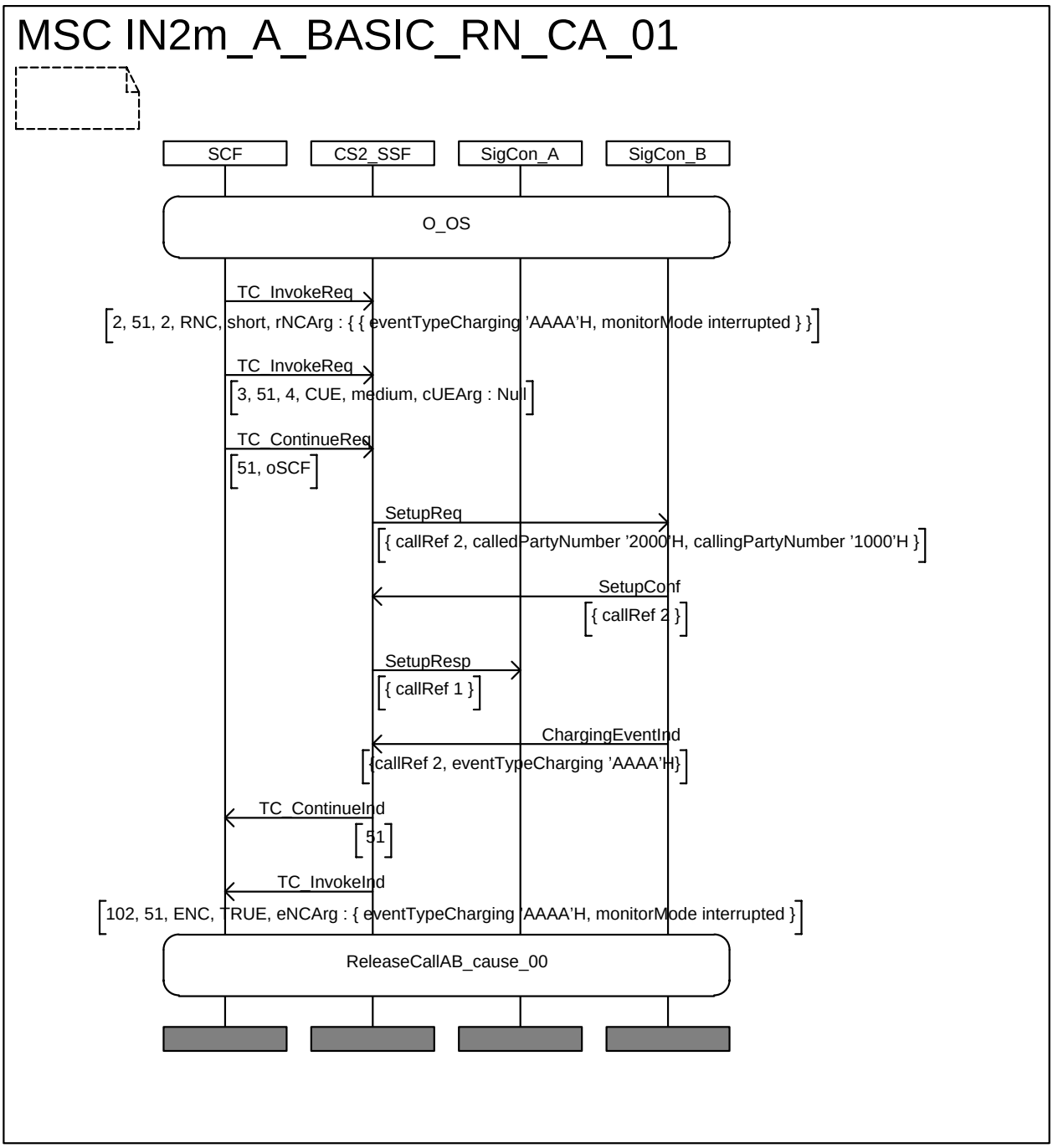


6.4.15 SendChargingInformation procedure

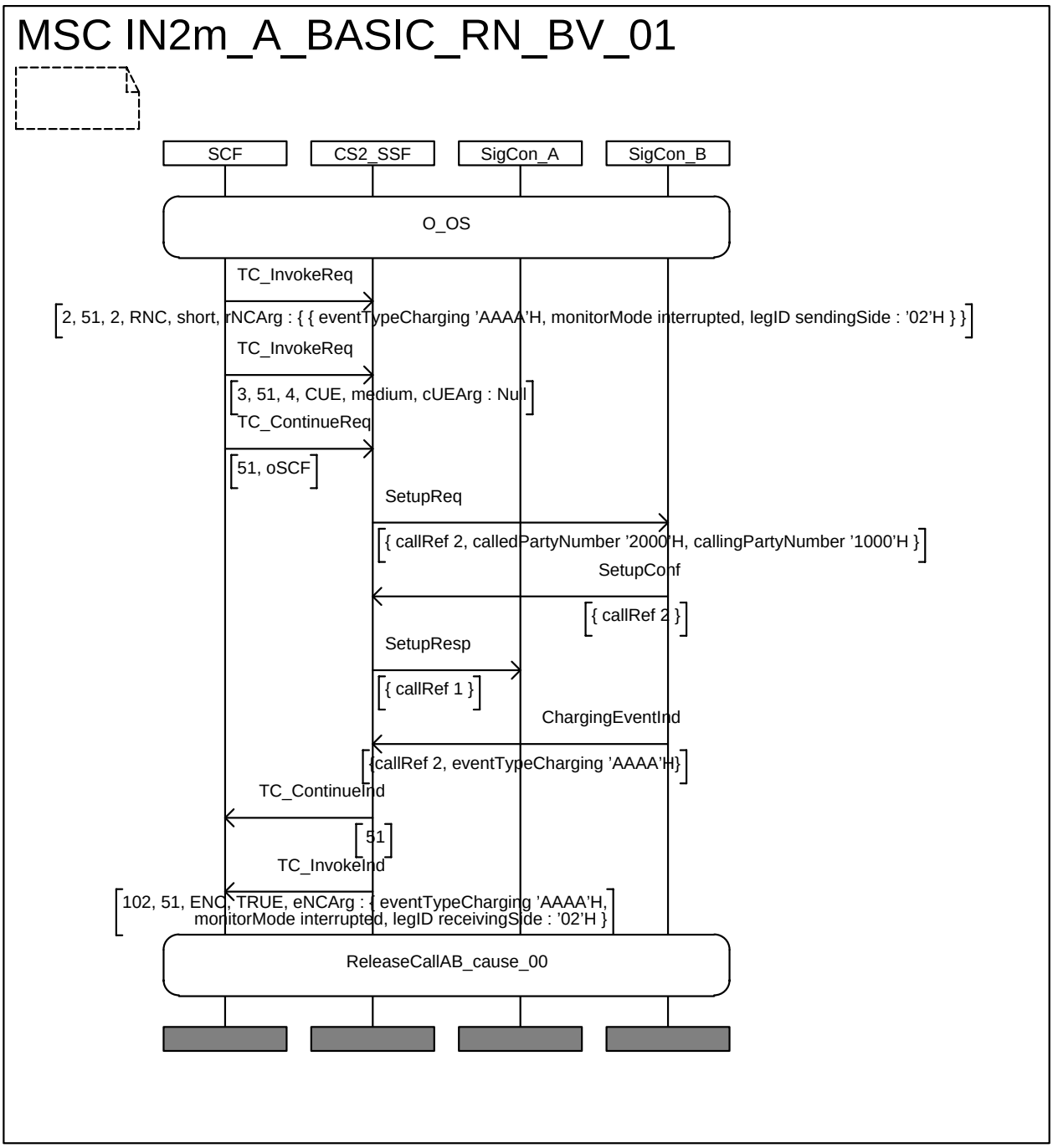
Charging related aspects in IN are network operator specific. Therefore, it is not possible to define useful test purposes for charging procedures using a network operator independant approach. TP specification has to be done by network operators, using INAP procedures themselves. SendChargingInformation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

6.4.16 RequestNotificationChargingEvent procedure

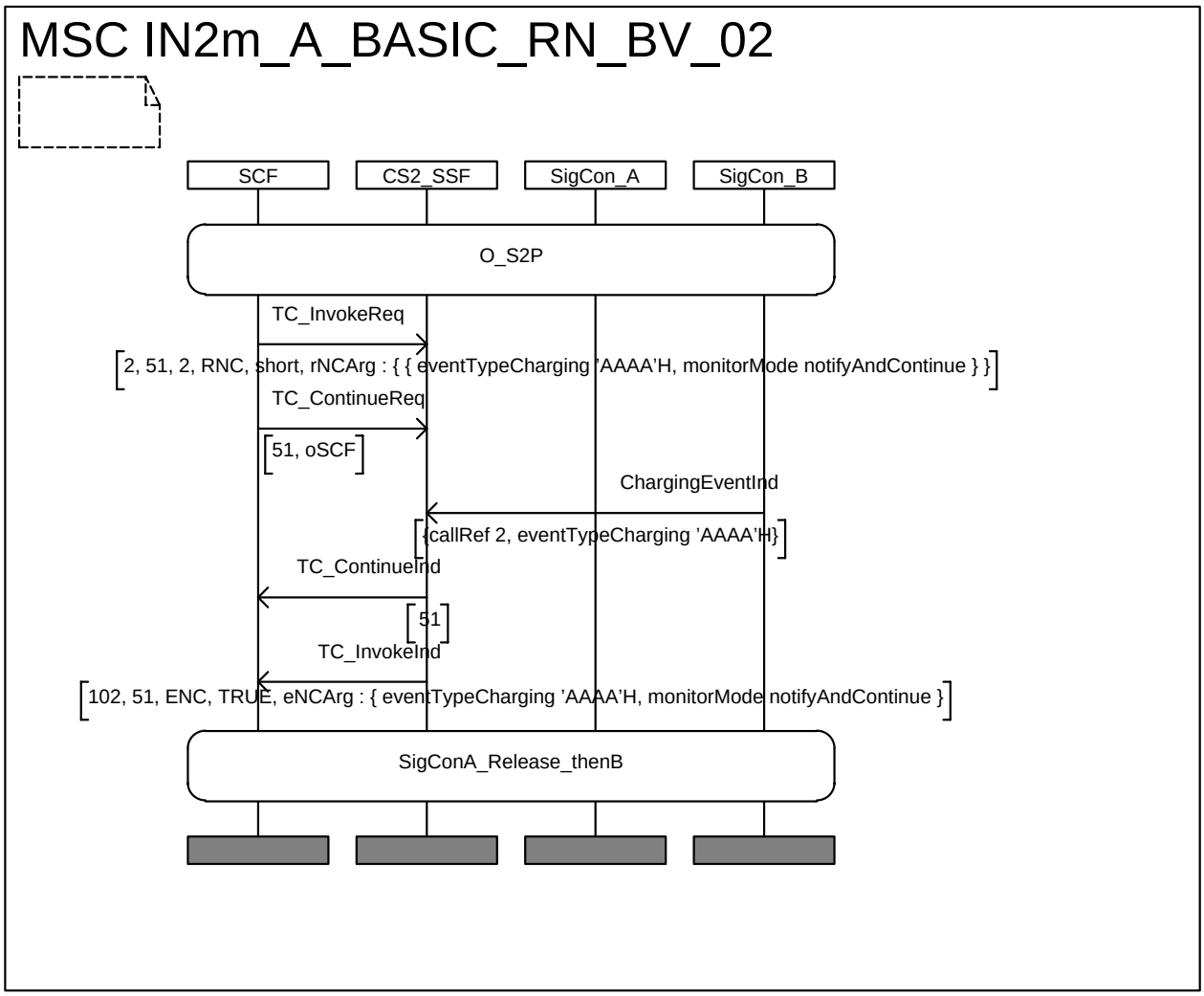
IN2_A_BASIC_RN_CA_01	
Purpose:	Test of **RequestNotificationChargingEvent** base procedure
Requirement ref	
Preamble:	O_OS
Selection Cond.	
Test description	SCF sends to SSF **RequestNotificationChargingEvent** invoke containing mandatory parameters only, with: - ChargingEvent eventTypeCharging, monitorMode (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


MSC IN2m_A_BASIC_RN_CA_01
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
TC_InvokeReq
[2, 51, 2, RNC, short, rNCArg : { { eventTypeCharging 'AAAA'H, monitorMode interrupted } }]
TC_InvokeReq
[3, 51, 4, CUE, medium, cUEArg : Null]
TC_ContinueReq
[51, oSCF]
SetupReq
[{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }]
SetupConf
[{ callRef 2 }]
SetupResp
[{ callRef 1 }]
ChargingEventInd
[{callRef 2, eventTypeCharging 'AAAA'H}]
TC_ContinueInd
[51]
TC_InvokeInd
[102, 51, ENC, TRUE, eNCArg : { eventTypeCharging 'AAAA'H, monitorMode interrupted }]
ReleaseCallAB_cause_00

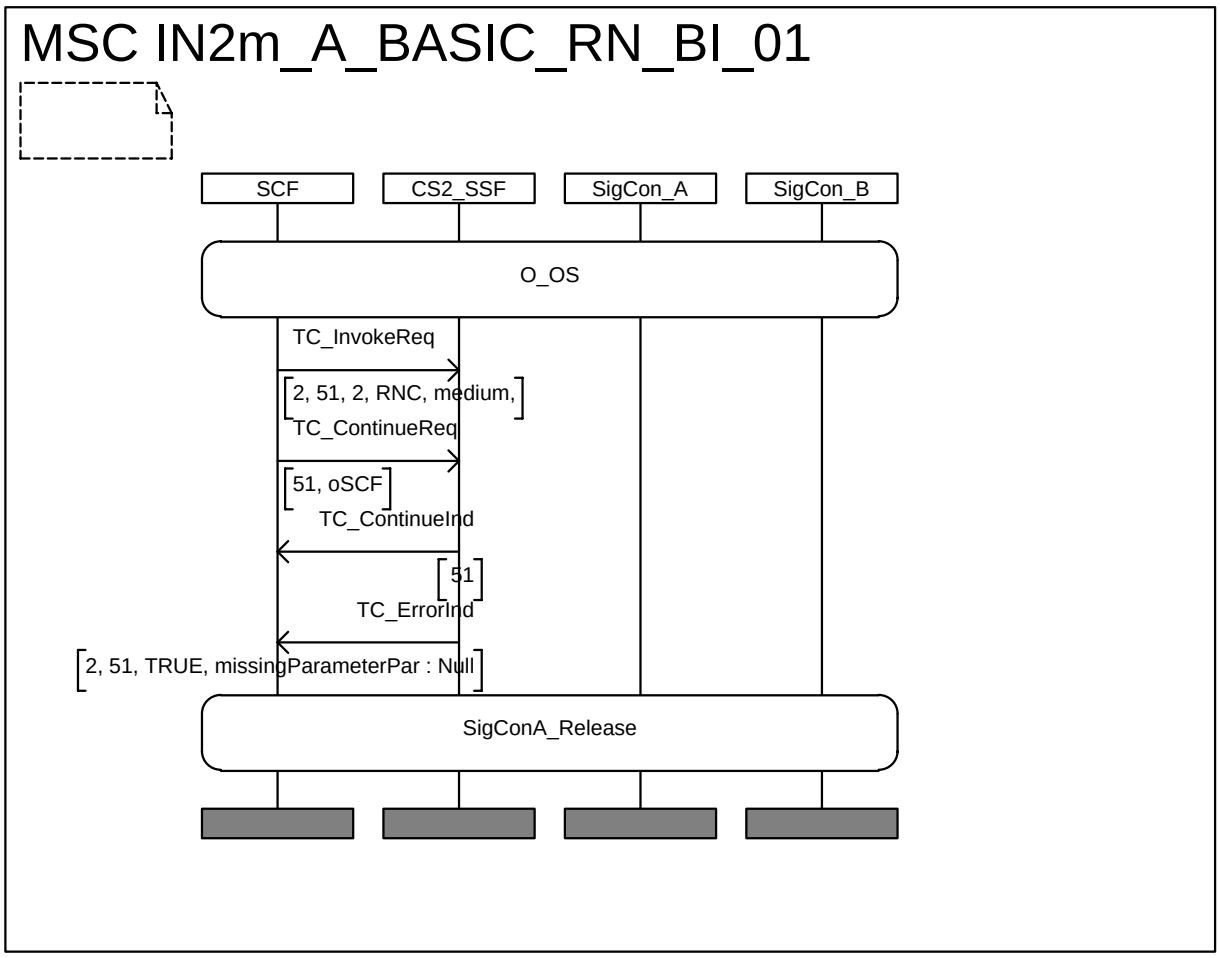
IN2_A_BASIC_RN_BV_01	
Purpose:	Test of **RequestNotificationChargingEvent** procedure with legID parameter
Requirement ref	
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestNotificationChargingEvent** invoke containing mandatory parameters only, with: - ChargingEvent eventTypeCharging, 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
SCF
CS2_SSF
SigCon_A
SigCon_B
O_OS
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
TC_ContinueReq
51, oSCF
SetupReq
{ callRef 2, calledPartyNumber '2000'H, callingPartyNumber '1000'H }
SetupConf
{ callRef 2 }
SetupResp
{ callRef 1 }
ChargingEventInd
{callRef 2, eventTypeCharging 'AAAA'H}
TC_ContinueInd
51
TC_InvokeInd
102, 51, ENC, TRUE, eNCArg : { eventTypeCharging 'AAAA'H,
monitorMode interrupted, legID receivingSide : '02'H }
ReleaseCallAB_cause_00

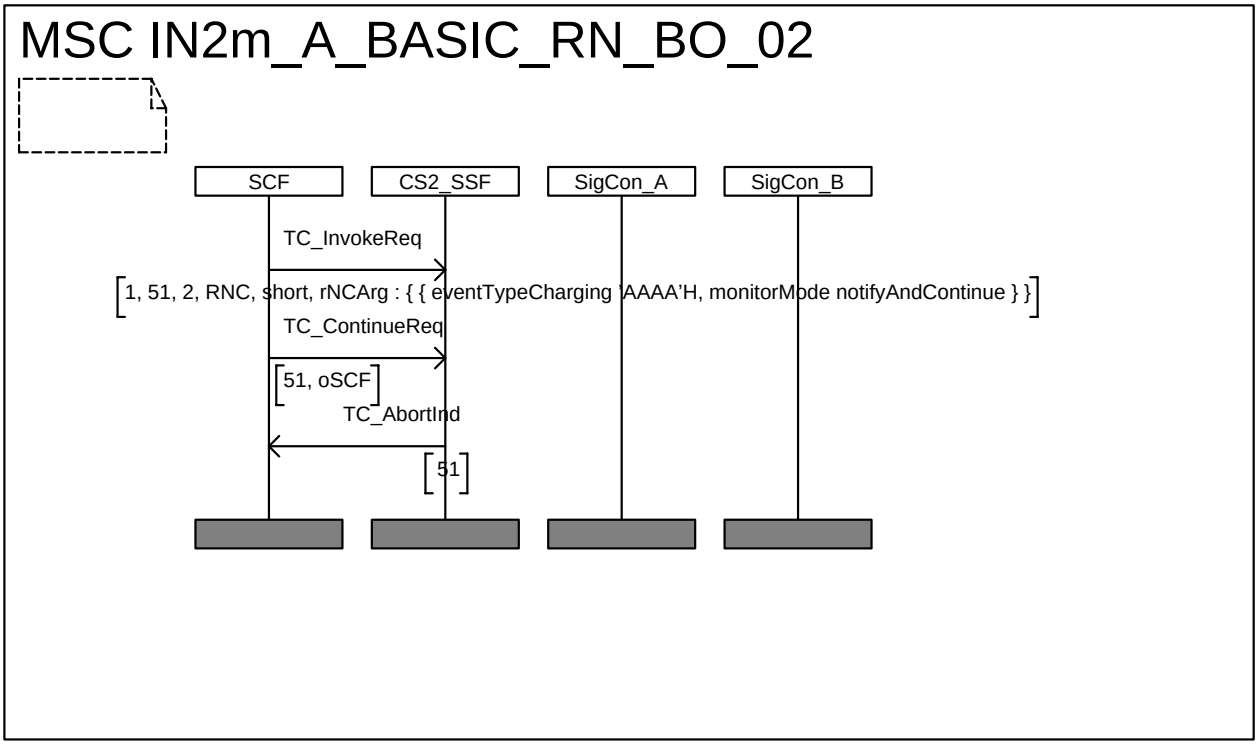
IN2_A_BASIC_RN_BV_02	
Purpose:	Test of **RequestNotificationChargingEvent** procedure with legID parameter
Requirement ref	
Selection Cond.	
Preamble:	O_S2P
Test description	SCF sends to SSF **RequestNotificationChargingEvent** invoke containing mandatory parameters only, with: - ChargingEvent eventTypeCharging, 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



IN2_A_BASIC_RN_BI_01	
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



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



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

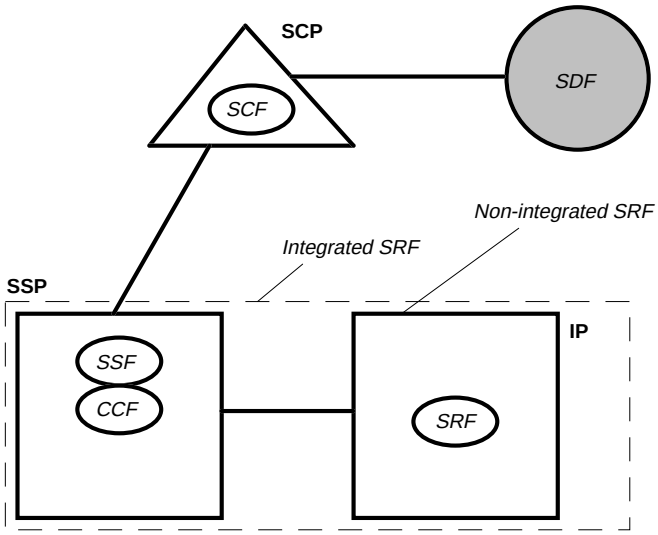
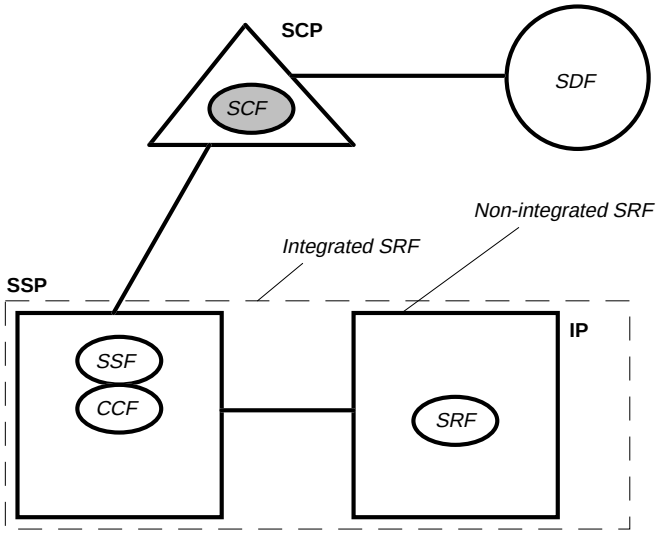


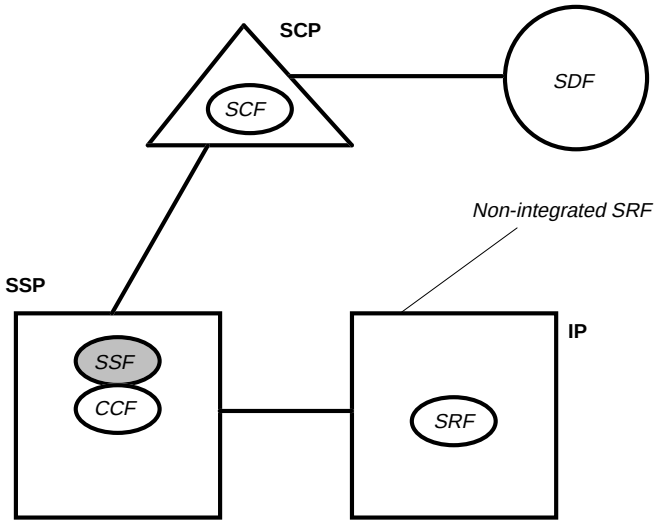
Figure A.1: Configuration 1_1: IUT= SDF

SCP with single SSP



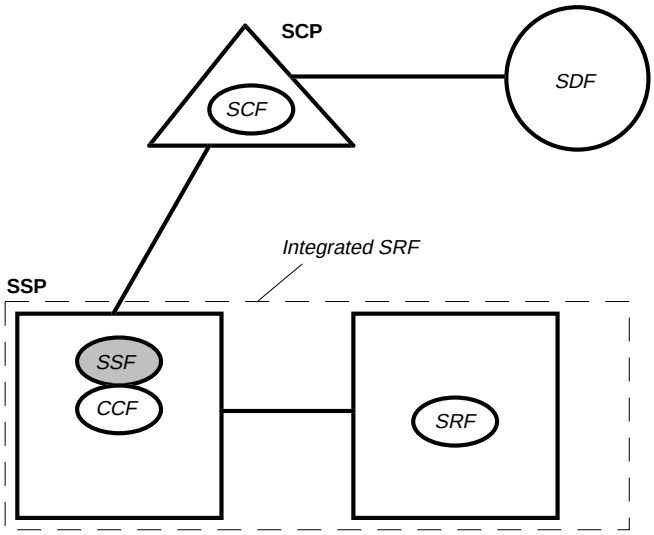
SCP with single SSP

Figure A.2: Configuration 1_2: IUT= SCF



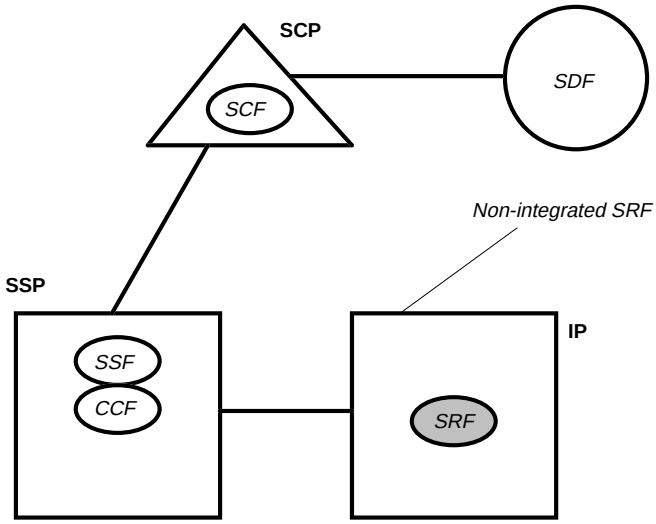
SCP with single SSP

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



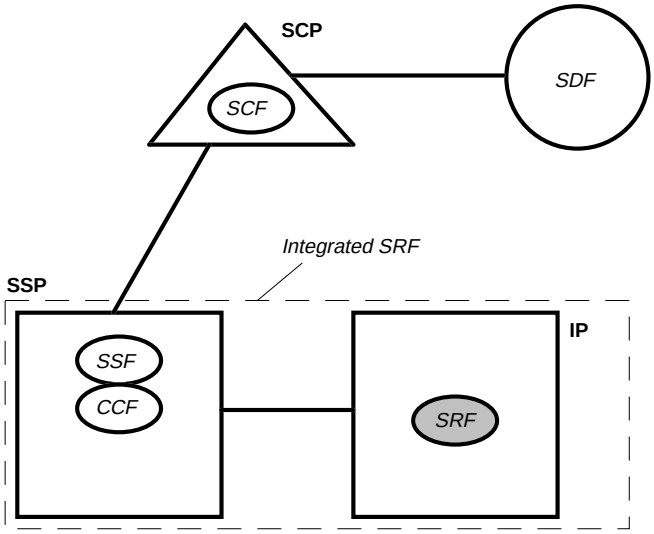
SCP with single SSP

Figure A.4: Configuration 1_4: IUT= SSF (integrated with SRF)



SCP with single SSP

Figure A.5: Configuration 1_5: IUT= SRF (non integrated with SSF)



SCP with single SSP

Figure A.6: Configuration 1_6: IUT= SRF (integrated with SSF)

Functional Configuration 2:

Example for direct path SCP-IP

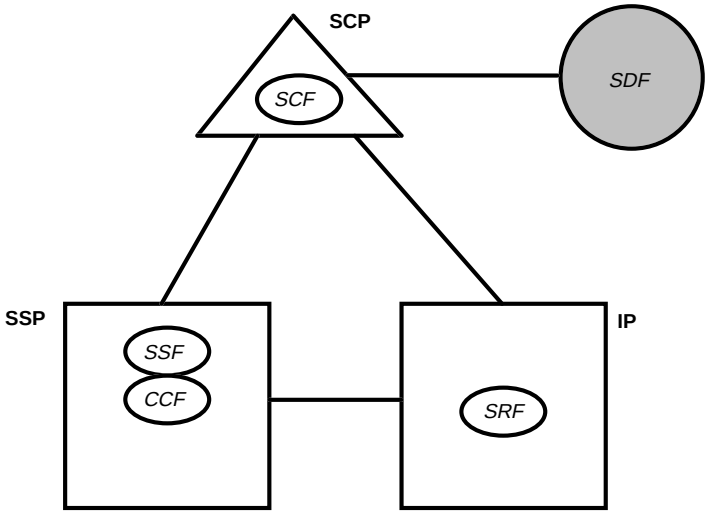


Figure A.7: Configuration 2_1: IUT = SDF

Direct path SCP - IP

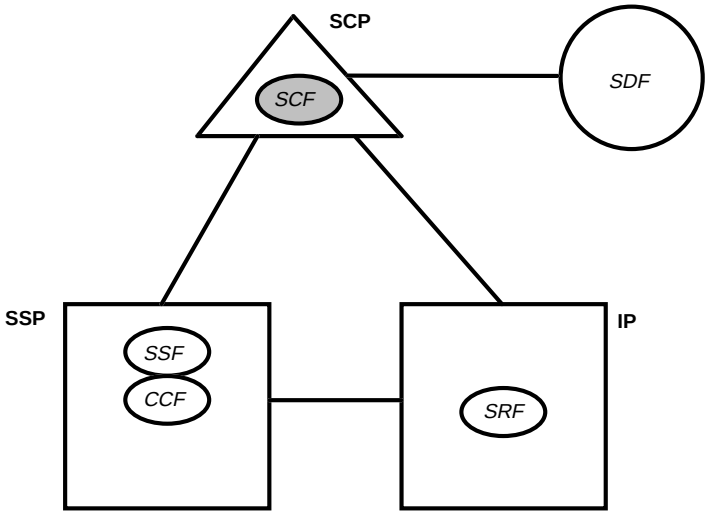


Figure A.8: Configuration 2_2: IUT = SCF

Direct path SCP - IP

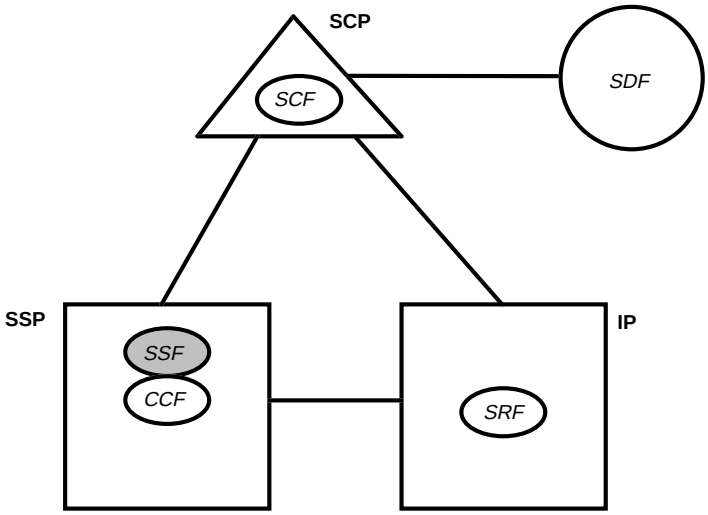


Figure A.9: Configuration 2_3: IUT = SSF

Direct path SCP - IP

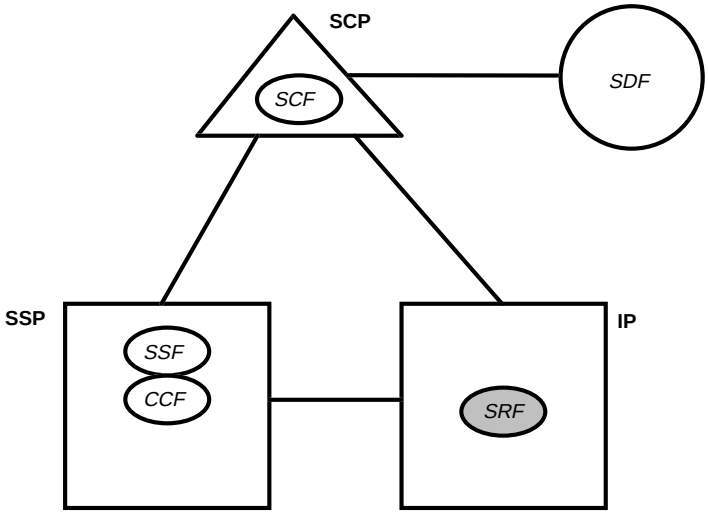


Figure A.10: Configuration 2_4: IUT = SRF

Direct path SCP - IP

Functional Configuration 3:

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

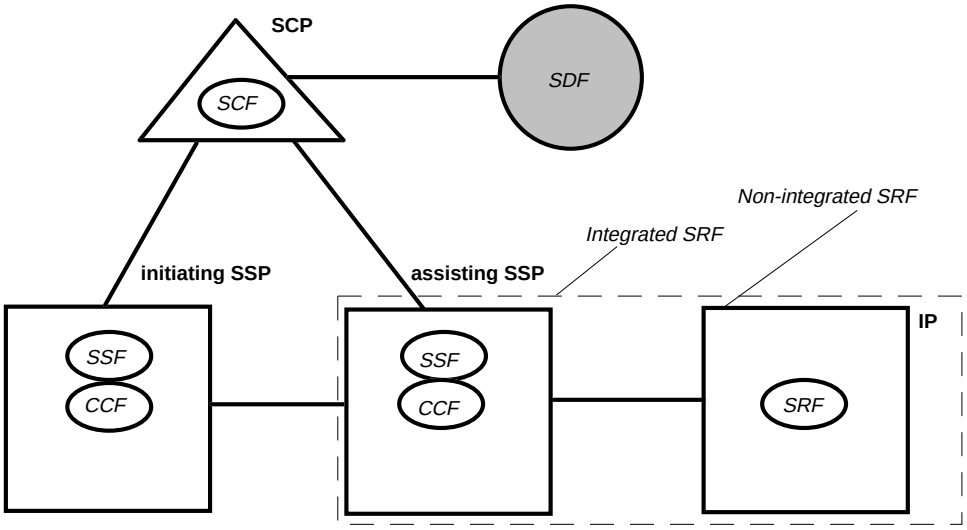


Figure A.11: Configuration 3_1: IUT= SDF

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

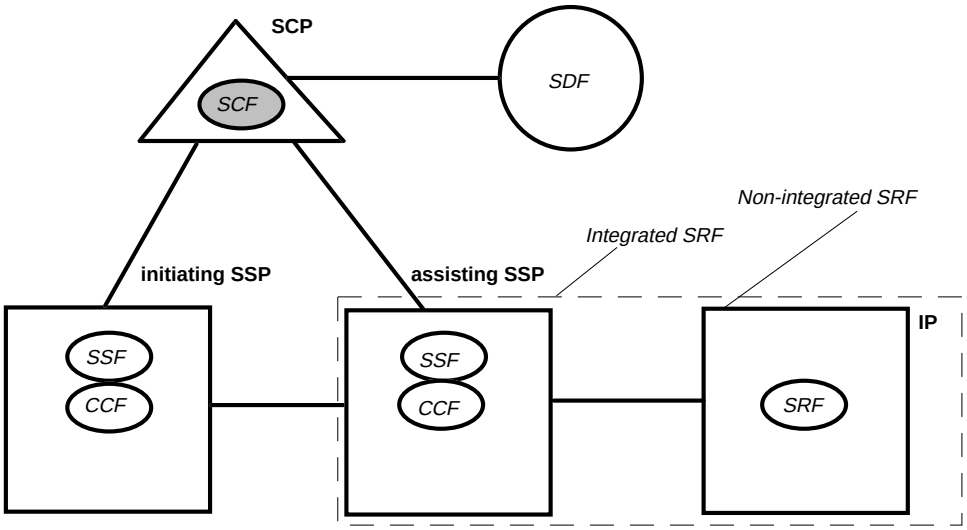


Figure A.12: Configuration 3_2: IUT= SCF

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

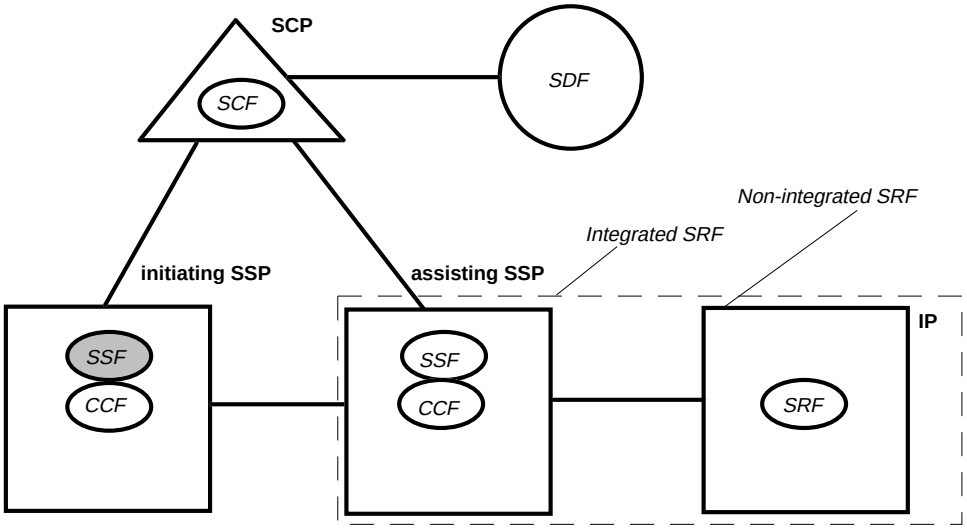


Figure A.13: Configuration 3_3: IUT= SSF of initiating SSP

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

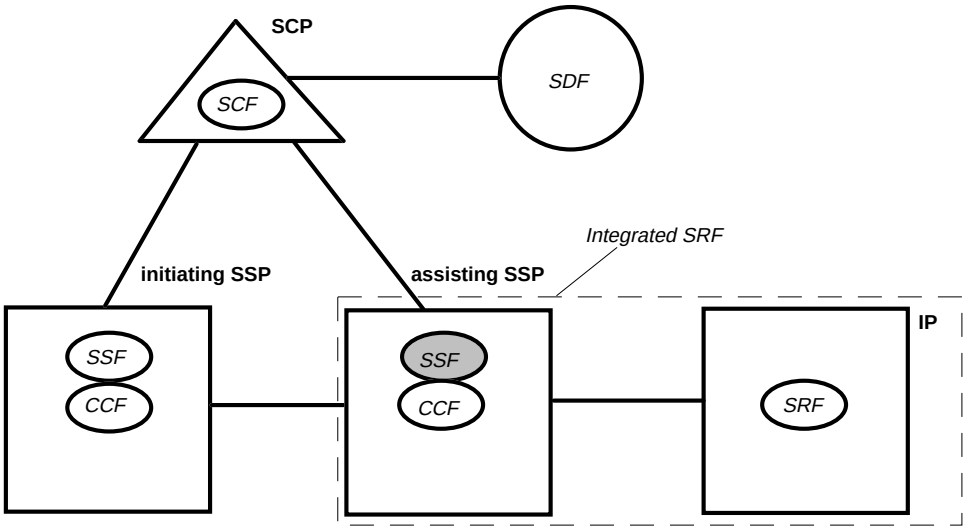


Figure A.14: Configuration 3_4: IUT= SSF of assisting SSP(integrated SRF)

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

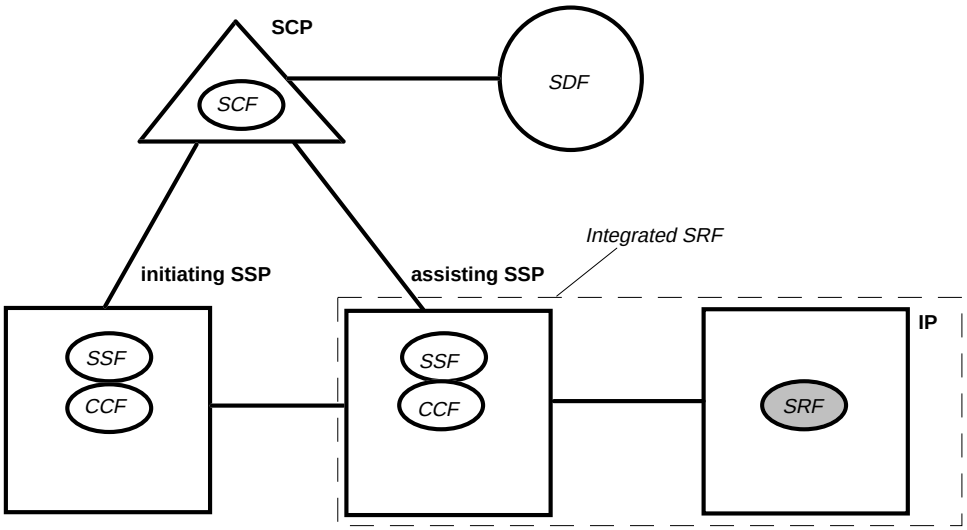


Figure A.15: Configuration 3_5: IUT= SRF (integrated with assisting SSF)

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

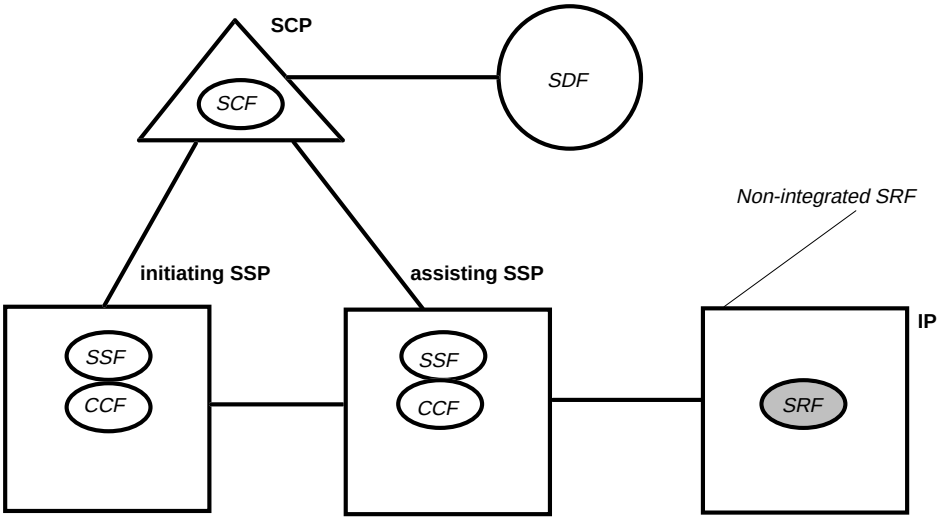


Figure A.16: Configuration 3_6: IUT= SRF (non integrated with assisting SSF)

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

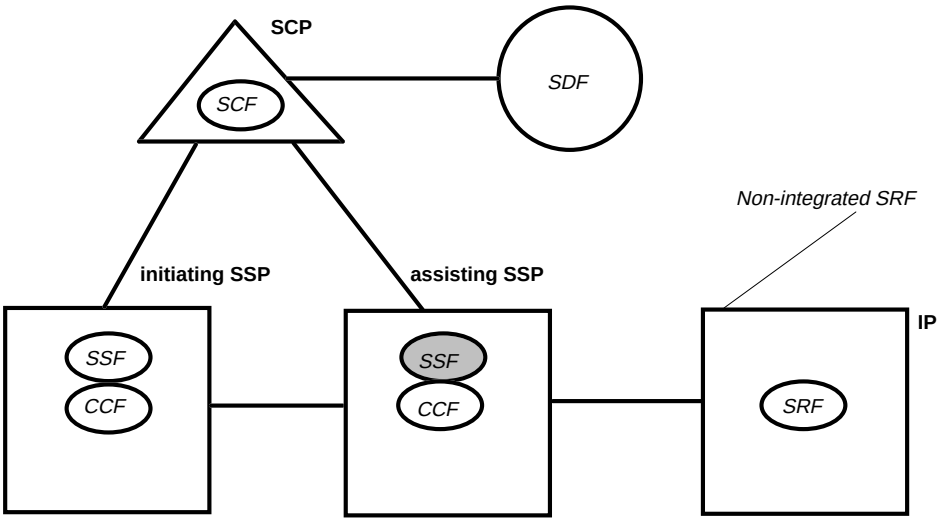


Figure A.17: Configuration 3_7: IUT= SSF of assisting SSP (non integrated SRF)

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

Functional Configuration 4:

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

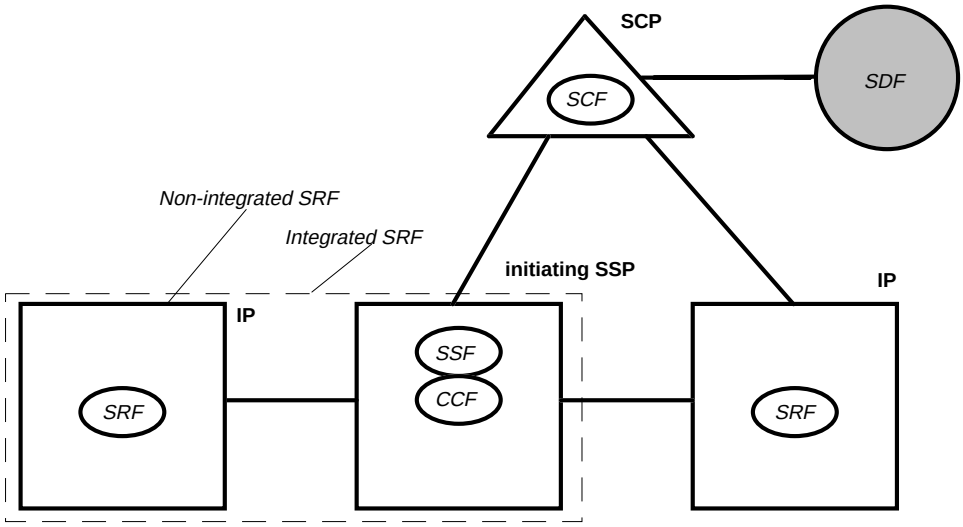


Figure A.18: Configuration 4_1: IUT= SDF

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

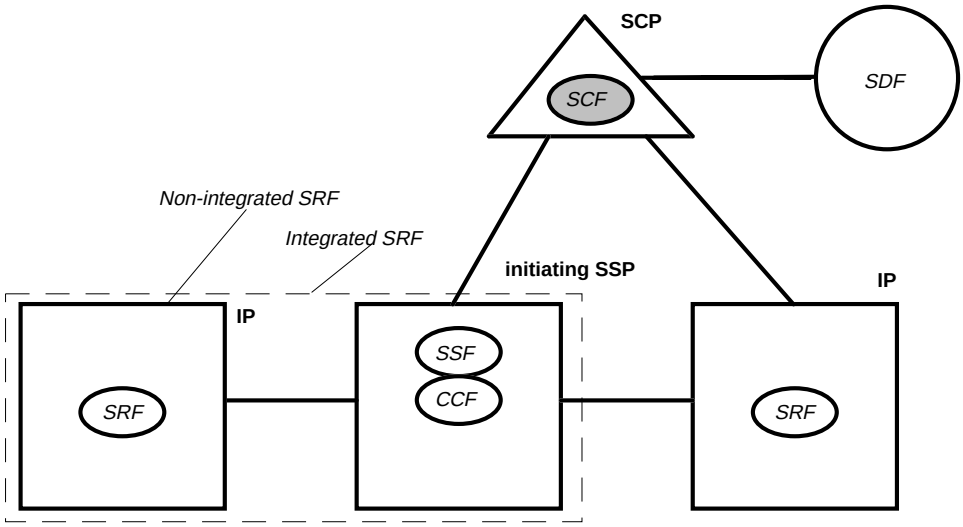


Figure A.19: Configuration 4_2: IUT= SCF

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

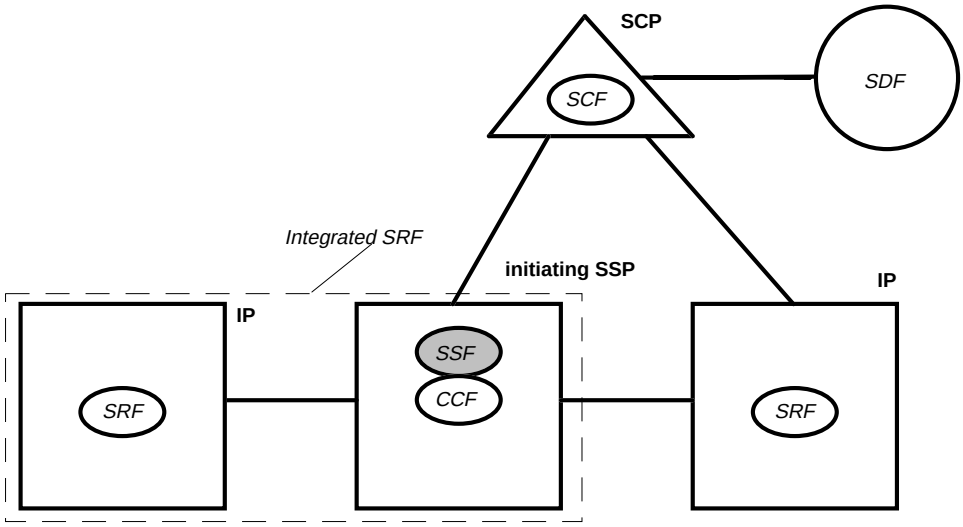


Figure A.20: Configuration 4_3: IUT= SSF of initiating SSP (integrated SRF)

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

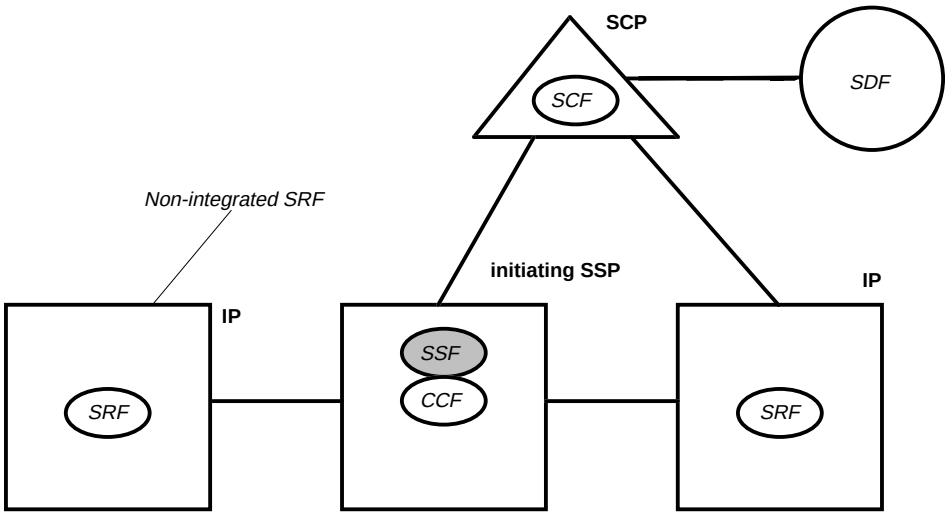


Figure A.21: Configuration 4_4: IUT= SSF of initiating SSP (non integrated SRF)

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

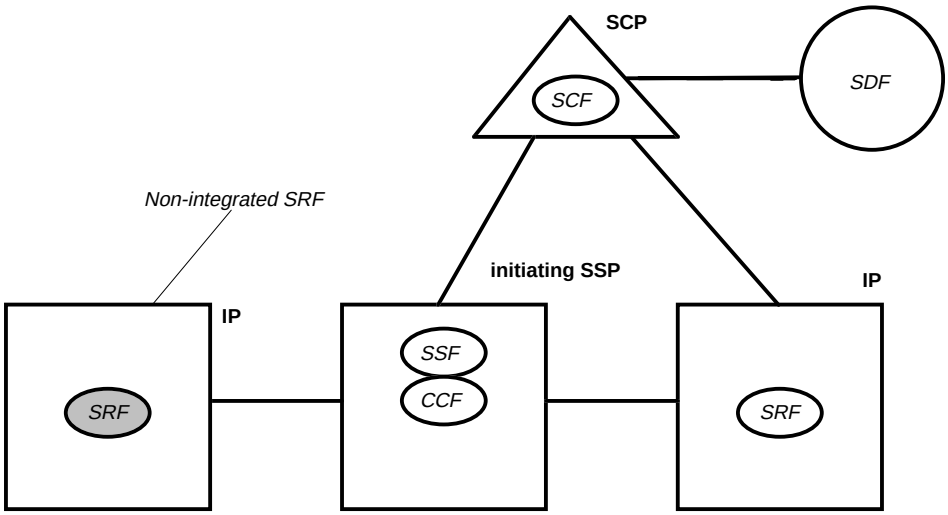


Figure A.22: Configuration 4_5: IUT= SRF of initiating SSP (non integrated SRF)

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

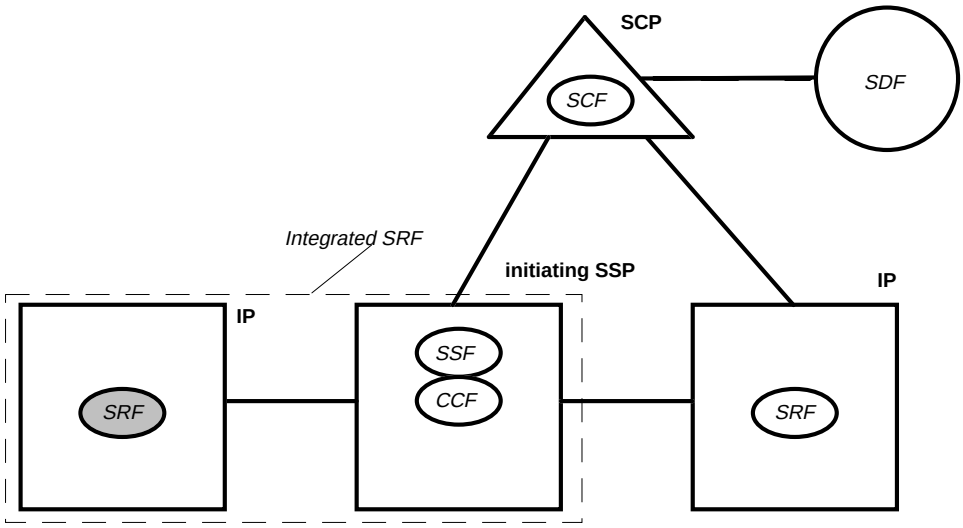


Figure A.23: Configuration 4_6: IUT= SRF of initiating SSP (integrated SRF)

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

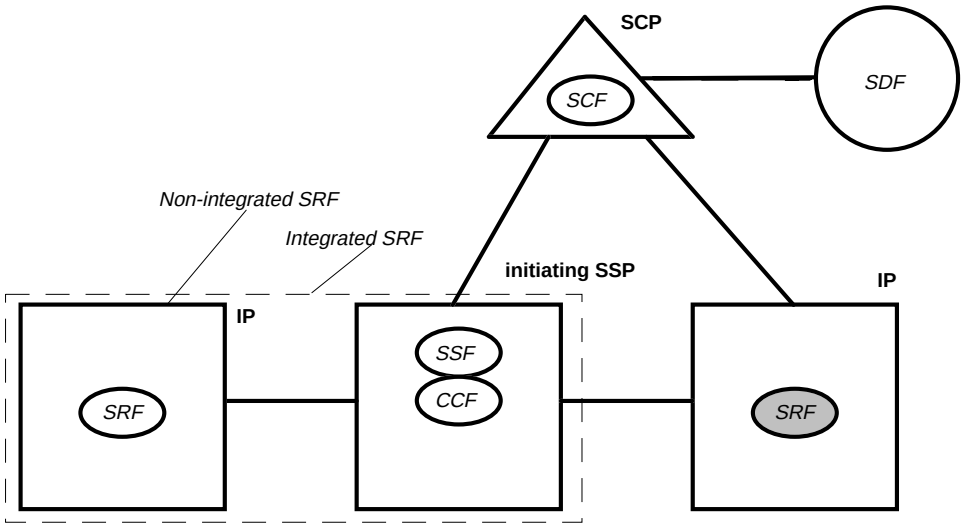


Figure A.24: Configuration 4_7: IUT= SRF as IP

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

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

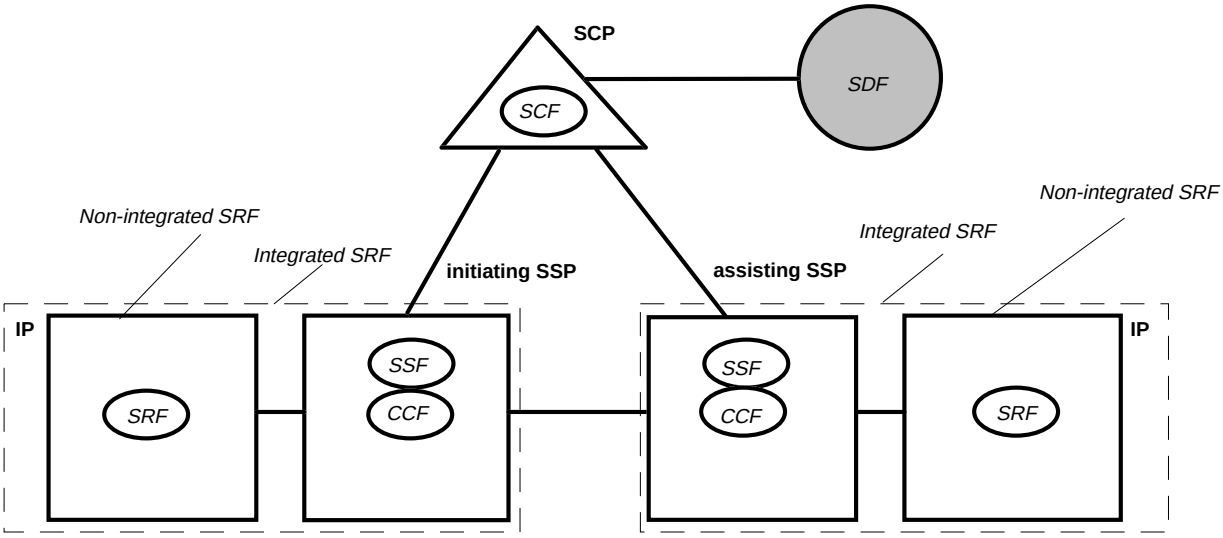


Figure A.25: Configuration 5_1: IUT= SDF

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

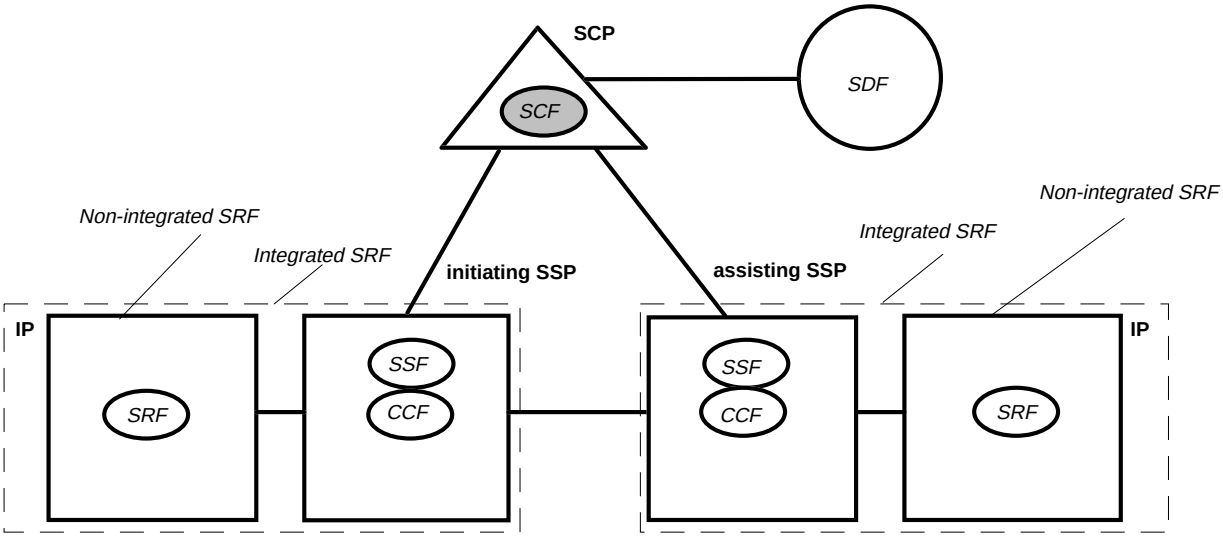


Figure A.26: Configuration 5_2: IUT= SCF

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

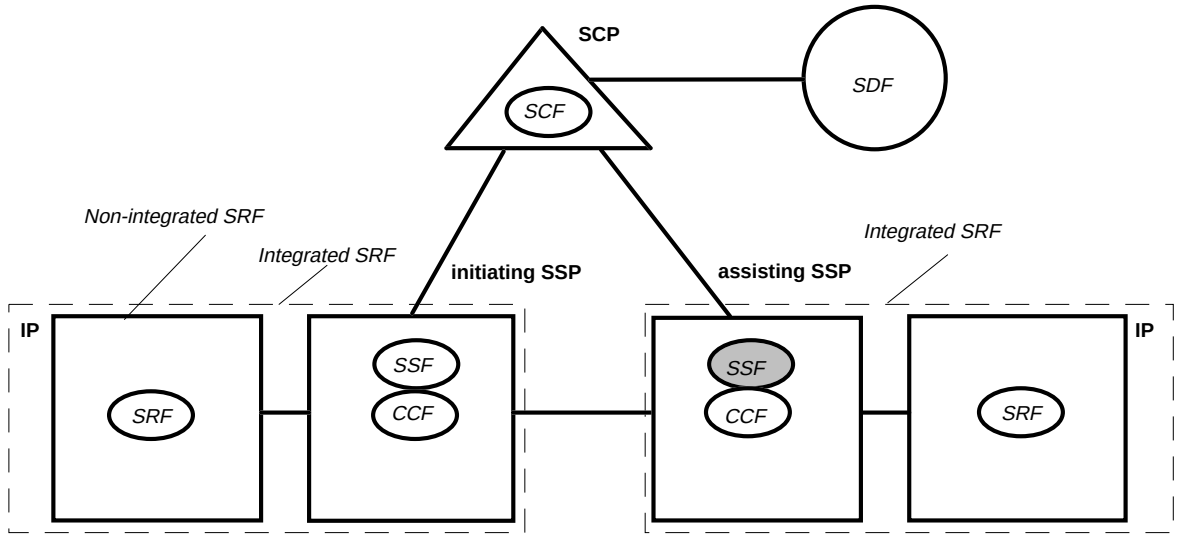


Figure A.27: Configuration 5_3: IUT= SSF of assisting SSP(with integrated SRF)

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

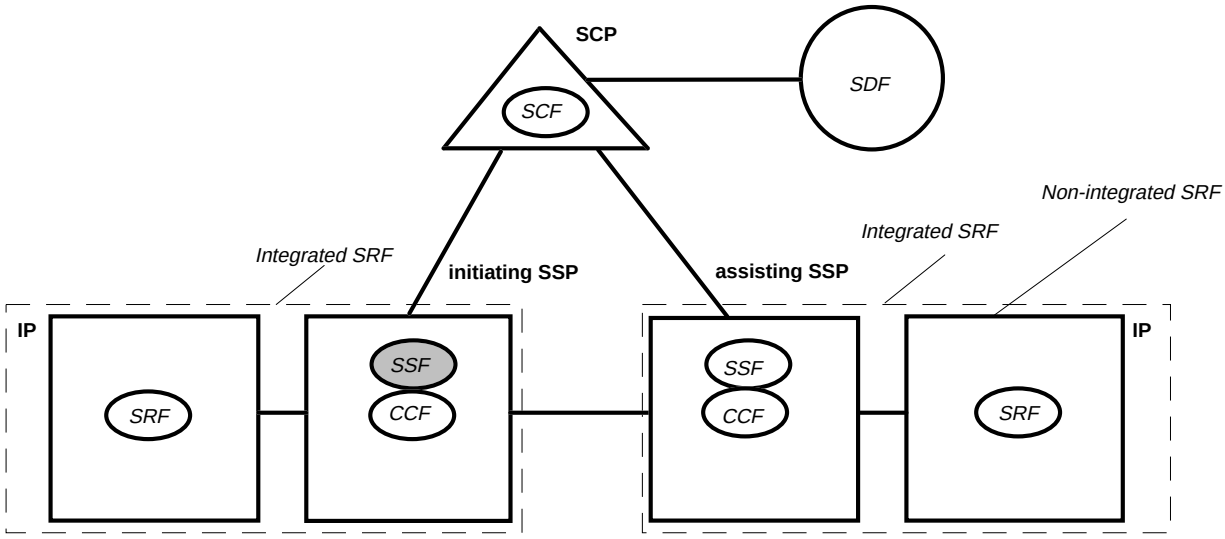


Figure A.28: Configuration 5_4: IUT= SSF of initiating SSP(with integrated SRF)

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

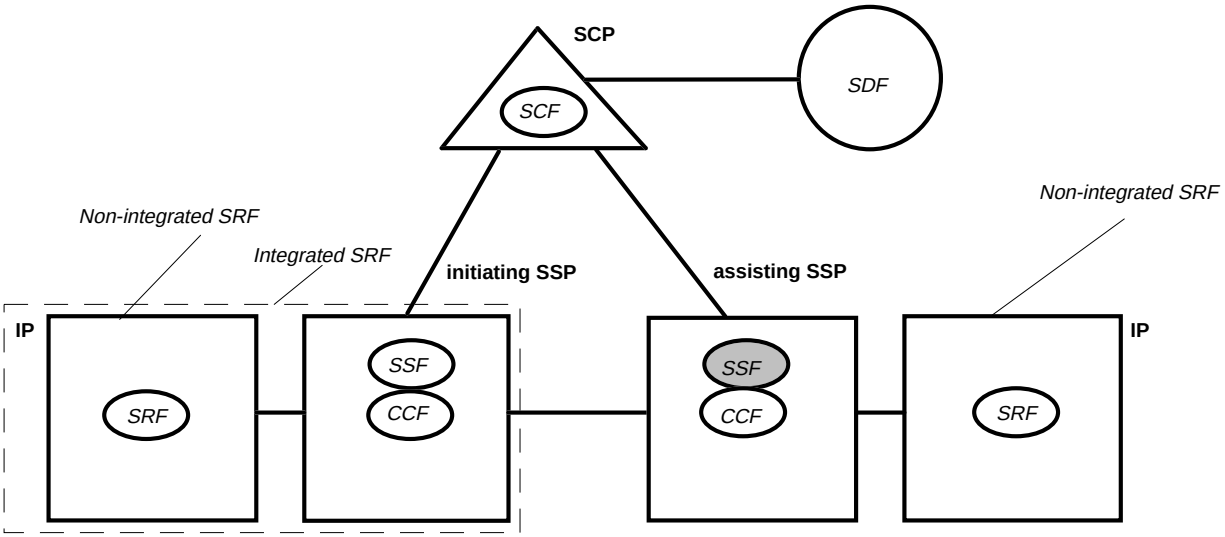


Figure A.29: Configuration 5_5: IUT= SSF of assisting SSP(with non integrated SRF)

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

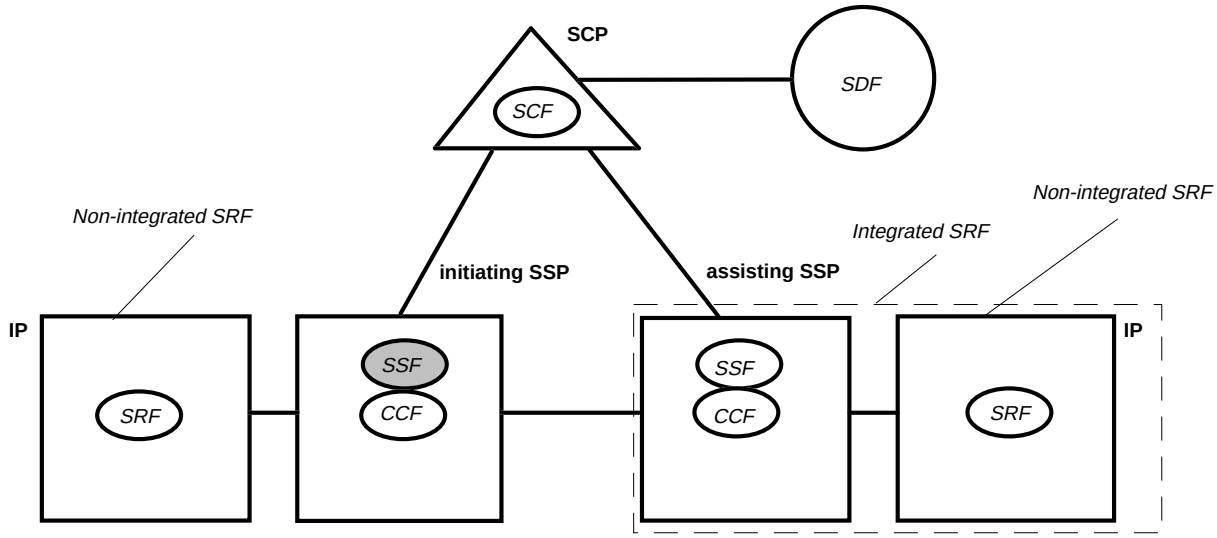


Figure A.30: Configuration 5_6: IUT= SSF of initiating SSP(with non integrated SRF)

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

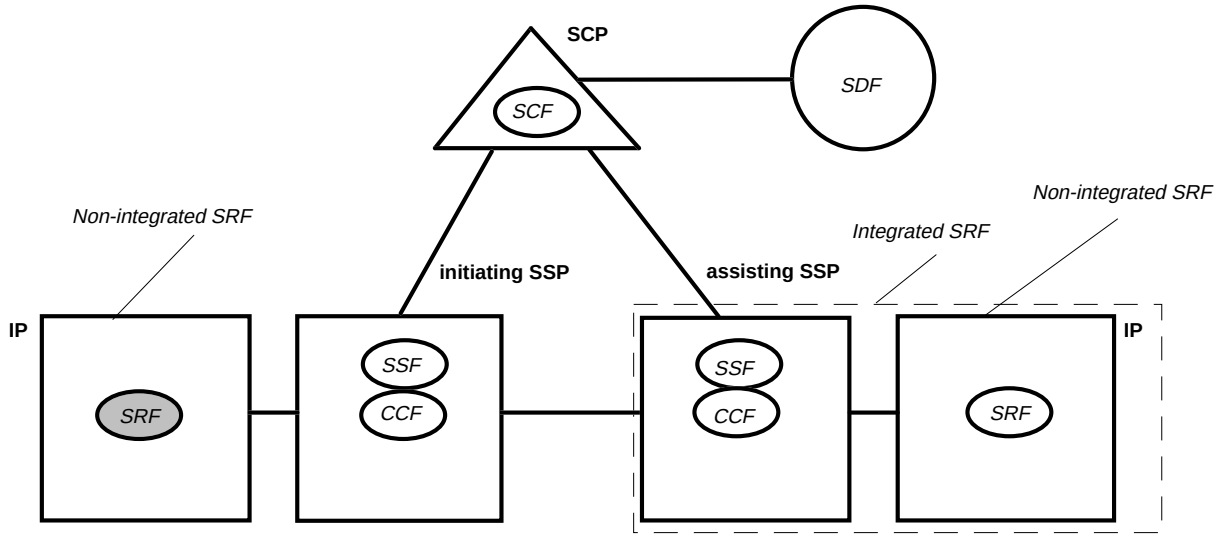


Figure A.31: Configuration 5_7: IUT= SRF of initiating SSP(non integrated SRF)

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

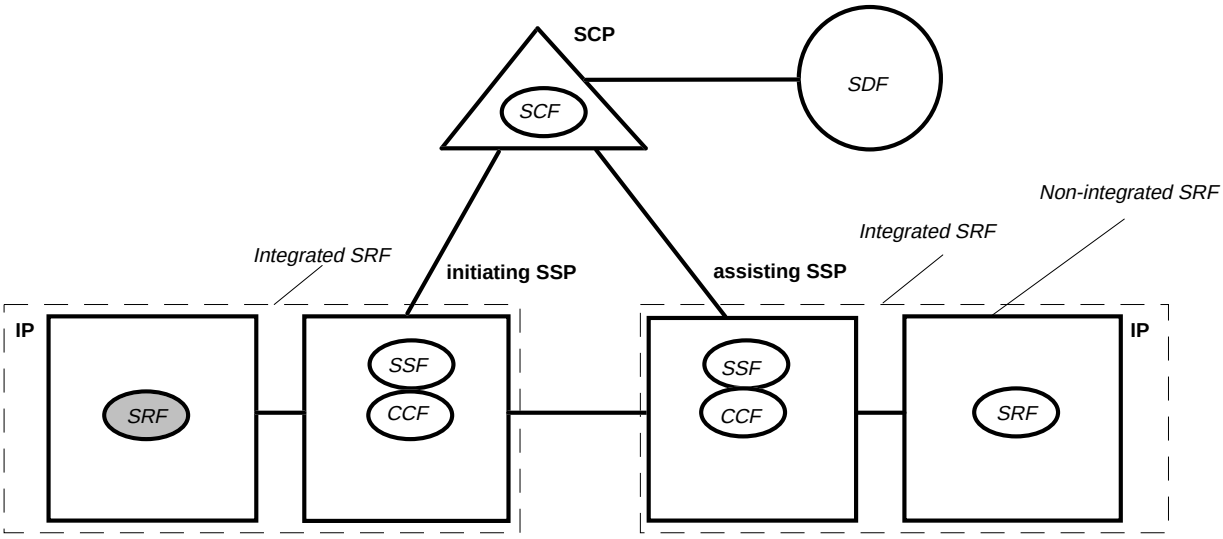


Figure A.32: Configuration 5_8: IUT= SRF of initiating SSP(integrated SRF)

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

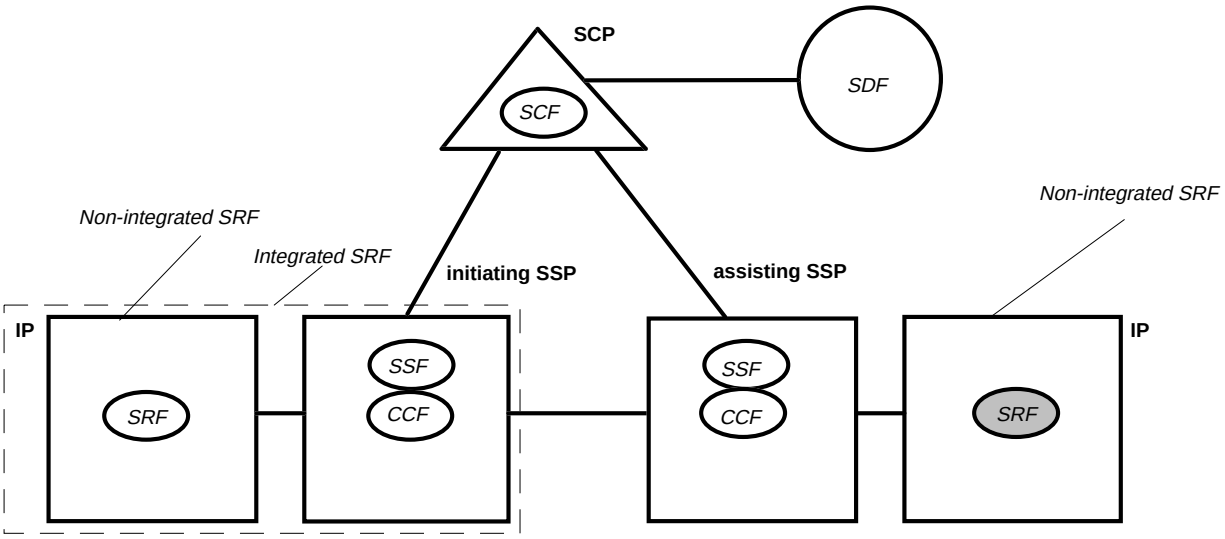


Figure A.33: Configuration 5_9: IUT= SRF of assisting SSP(non integrated SRF)

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

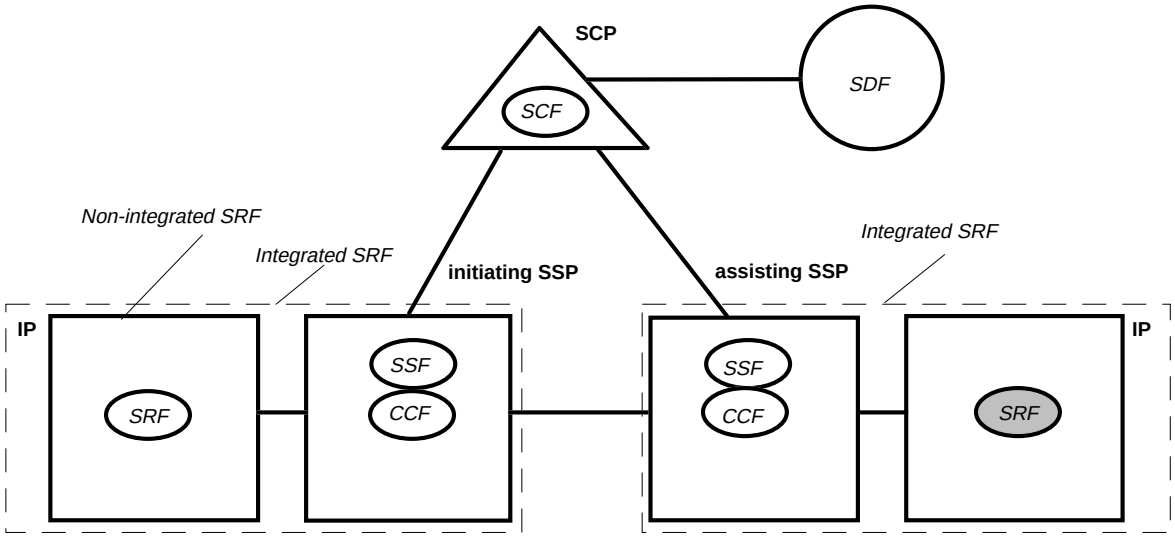


Figure A.34: Configuration 5_10: IUT= SRF of assisting SSP(integrated SRF)

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

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

Item	Parameter	Parameter type	Explanation/Format	Value
	PIX_AChBillingChargingCharacteristics	AChBillingChargingCharacteristics	"xx"H	44
	PIX_AlertingPattern	**AlertingPattern**	"xxx"H	123
	PIX_AlertingPattern_ICA	**AlertingPattern**	"xxx"H	124
	PIX_APtyAbandonCause	**Cause**	"xx"H	0F
	PIX_APtyDiscCause	**Cause**	"xx"H	10
	PIX_AssistingSSPIRoutingAddress	**AssistingSSPIRoutingAddress**	"xxxx"H	7755
	PIX_BPtyBusy_UDUBCause	Cause	"xx"H	0D
	PIX_BPtyNoAnswerCause	Cause	"xx"H	09
	PIX_CalledPartyNumber1_CON	**CalledPartyNumber**	LegId 2 "xxxx"H	2001
	PIX_CalledPartyNumber2_CON	**CalledPartyNumber**	LegId 3 "xxxx"H	2003
	PIX_CalledPartyNumber3_CON	**CalledPartyNumber**	LegId 4 "xxxx"H	2005
	PIX_CalledPartyNumber4_CON	**CalledPartyNumber**	LegId 5 "xxxx"H	2007
	PIX_CalledPartyNumber5_CON	**CalledPartyNumber**	LegId 6 "xxxx"H	2009
	PIX_CalledPartyNumber6_CON	**CalledPartyNumber**	LegId 7 "xxxx"H	2011
	PIX_CalledPartyNumber7_CON	**CalledPartyNumber**	LegId 8 "xxxx"H	2013
	PIX_CalledPartyNumber8_CON	**CalledPartyNumber**	LegId 9 "xxxx"H	2015
	PIX_CalledPartyNumberInvalid_CON	**CalledPartyNumber**	"xxxx"H	AA20
	PIX_CalledPartyNumber1_ICA	**CalledPartyNumber**	LegId 2 "xxxx"H	2100
	PIX_CalledPartyNumber2_ICA	**CalledPartyNumber**	LegId 3 "xxxx"H	2101
	PIX_CalledPartyNumber3_ICA	**CalledPartyNumber**	LegId 4 "xxxx"H	2102
	PIX_CalledPartyNumber4_ICA	**CalledPartyNumber**	LegId 5 "xxxx"H	2103
	PIX_CalledPartyNumber5_ICA	**CalledPartyNumber**	LegId 6 "xxxx"H	2104
	PIX_CalledPartyNumber6_ICA	**CalledPartyNumber**	LegId 7 "xxxx"H	2105
	PIX_CalledPartyNumber7_ICA	**CalledPartyNumber**	LegId 8 "xxxx"H	2106
	PIX_CalledPartyNumber1_SetupInd	**CalledPartyNumber**	"xxxx"H	2000
	PIX_CalledPartyNumber2_SetupInd	**CalledPartyNumber**	"xxxx"H	2002
	PIX_CallingPartyNumber1	**CallingPartyNumber**	"xxxx"H	1000
	PIX_CallingPartyNumber2	**CallingPartyNumber**	"xxxx"H	1002
	PIX_CallingPartysCategory_CON	**CallingPartysCategory**	"xx"H	BB
	PIX_CallingPartysCategory_SetupInd	**CallingPartysCategory**	"xx"H	CC
	PIX_DateAndTime	DateAndTime	YYMMDDHHMMSS	980115123030
	PIX_Duration	Duration	Seconds	66
	PIX_EventTypeCharging1	EventTypeCharging		"AAAA"
	PIX_EventTypeCharging2	EventTypeCharging		"CCCC"
	PIX_FCIBillingChargingCharacteristics	FCIBillingChargingCharacteristics		55
	PIX_InbandInfo_message	InbandInfo	InformationToSend	"AABB"
	PIX_Interval	Integer	Seconds	33
	PIX_IPRoutingAddress	IPRoutingAddress	"xxx"H	400
	PIX_LocationNumber	LocationNumber	"xxxx"H	9001
	PIX_MaximumNumberOfCounters	MaximumNumberOfCounters	"xx"H	14
	PIX_NumberOfCalls	Integer	xx	13
	PIX_OriginalCalledPartyNumber	**CalledPartyNumber**	"xxxx"H	2211
	PIX_RedirectingPartyNumber	**CalledPartyNumber**	"xxxx"H	3000
	PIX_RedirectionInformation	RedirectionInformation	"xx"H	AA
	PIX_ReleaseCause	Cause	"xx"H	00
	PIX_RouteSelectFailure1Cause	Cause	"xx"H	0B
	PIX_RouteSelectFailure2Cause	Cause	"xx"H	0C
	PIX_ScfID	ScfID	"xxxx"H	8881
	PIX_ServiceInteractionIndicators	ServiceInteractionIndicators	"xx"H	22
	PIX_ServiceKey1	ServiceKey	"xx"H	27
	PIX_ServiceKey2	ServiceKey	"xx"H	28
	PIX_SFBillingChargingCharacteristics	SFBillingChargingCharacteristics	"xxxx"H	BBBB
	PIX_StartTime	DateAndTime	YYMMDDHHMMSS	971128113015
	PIX_StopTime	DateAndTime	YYMMDDHHMMSS	971212113015

Item	Parameter	Parameter type	Explanation/Format	Value
	PIX_ElementaryMessageID	integer	xxx	191
	PIX_CorrelationId	correlationID	"xxx"H	AAA
	PIX_UiScriptID1	integer	xxx	202
	PIX_UiScriptID2	integer	xxx	203
	PIX_UiScriptIDInvalid	integer	xxx	210
	PIX_UiScriptResult	UiScriptResult	"xxxx"H	5110
	PIX_UiScriptSpecificInfo	UiScriptSpecificInfo	"xxxx"H	5220

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

Item	Parameter	Parameter type	Explanation	Value
	PIX_InvokeId	InvokeIDType	Direction SCF ->SSF Direction SSF->SCF	1-100 101-200
	PIX_DialogueId	DialogueIDType	Direction SCF ->SSF Direction SSF->SCF	1-50 51-100

History

Document history			
V1.1.2	June 1999	Public Enquiry	PE 9947: 1999-06-23 to 1999-11-19