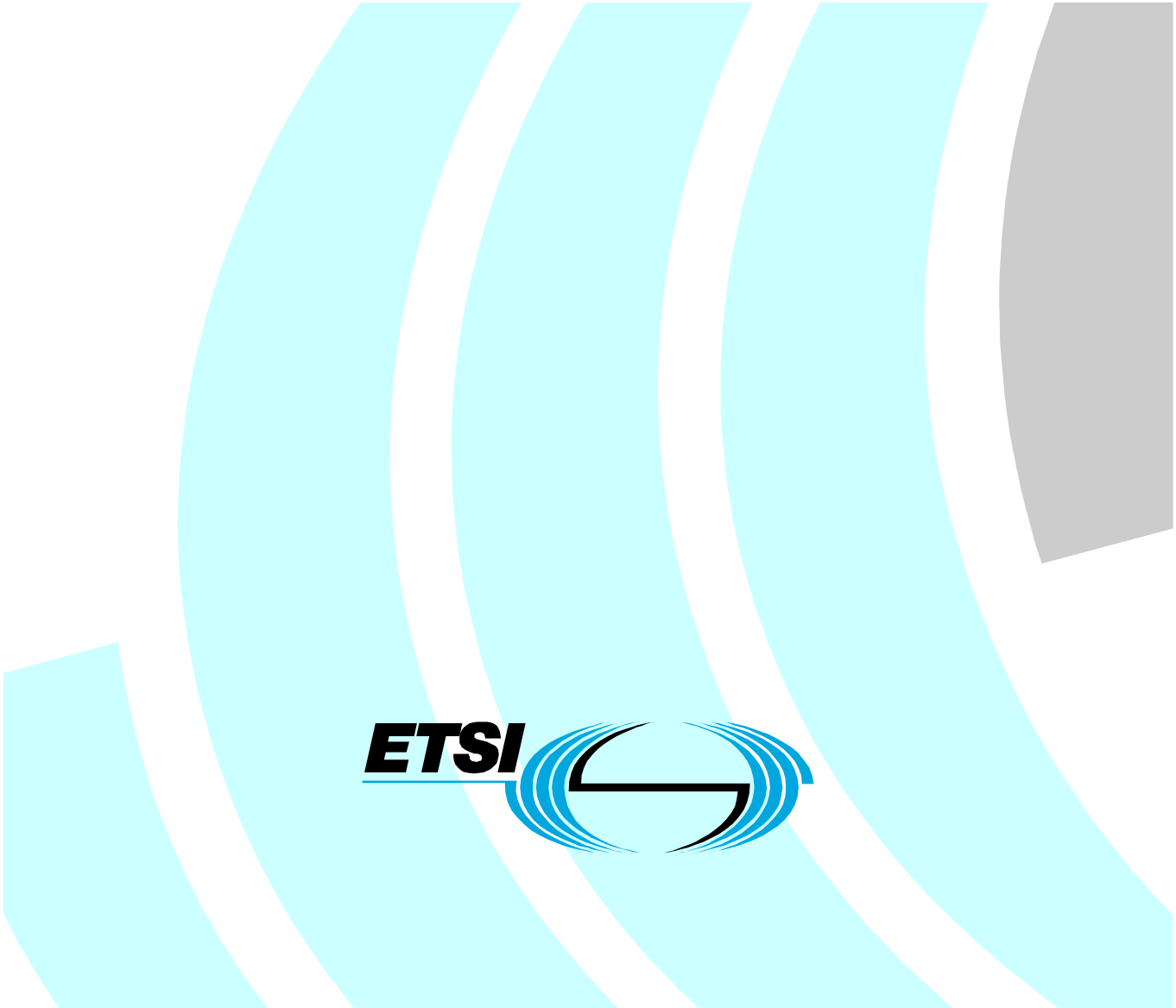
ETSI EN 301 933-1 V1.1.1 (2003-01)

European Standard (Telecommunications series)

Intelligent Network (IN);
Intelligent Network Capability Set 3 (CS3);
Intelligent Network Application Protocol (INAP);
Test Suite Structure and Test Purposes (TSS&TP)
specification for Service Switching Function (SSF);
Part 1: Basic capability set of CS3



Reference
DEN/SPAN-120063-3-1

Keywords
CS3, CTM, IN, INAP, SSF, TSS&TP, UPT

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - 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

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 ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

All published ETSI deliverables shall include information which directs the reader to the above source of information.

Foreword

This European Standard (Telecommunications series) has been produced by ETSI Technical Committee Services and Protocols for Advanced Networks (SPAN).

The present document is part 1 of a multi-part deliverable covering the Intelligent Network Capability Set 3 (CS3); Intelligent Network Application Protocol (INAP); Test Suite Structure and Test Purposes (TSS&TP) specification for Service Switching Function (SSF), as identified below:

Part 1: "Basic capability set of CS3";

Part 2: "Call Party Handling (CPH)";

Part 3: "Specialized Resource Function (SRF)".

National transposition dates	
Date of adoption of this EN:	10 January 2003
Date of latest announcement of this EN (doa):	30 April 2003
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	31 October 2003
Date of withdrawal of any conflicting National Standard (dow):	31 October 2003

1 Scope

The present document provides the Test Suite Structure and Test Purposes (TSS&TP) for the testing of the "Basic" operations of the Service Switching Function (SSF), defined for the Intelligent Network Application Protocol (INAP) of Intelligent Network (IN) Capability Set 3 (CS3) according to EN 301 931-1 [1] and EN 301 931-2 [2].

The present document is completed by other parts constituting the testing of the CS3 Core INAP specifications: EN 301 933-2 [5] (Call party handling functions) and EN 301 933-3 [6] (Specialized Resource Function).

ISO/IEC 9646-1 [8] and ISO/IEC 9646-2 [9] are used as the basis for the testing 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 and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] ETSI EN 301 931-1: "Intelligent Network (IN); Intelligent Network Capability Set 3 (CS3); Intelligent Network Application Protocol (INAP); Protocol specification; Part 1: Common aspects".

[2] ETSI EN 301 931-2: "Intelligent Network (IN); Intelligent Network Capability Set 3 (CS3); Intelligent Network Application Protocol (INAP); Protocol specification; Part 2: SCF-SSF interface".

[3] ETSI EN 301 931-3: "Intelligent Network (IN); Intelligent Network Capability Set 3 (CS3); Intelligent Network Application Protocol (INAP); Protocol specification; Part 3: SCF-SRF interface".

[4] ETSI EN 301 931-4: "Intelligent Network (IN); Intelligent Network Capability Set 3 (CS3); Intelligent Network Application Protocol (INAP); Protocol specification; Part 4: SDLs for SCF-SSF interface".

[5] ETSI EN 301 933-2: "Intelligent Network (IN); Intelligent Network capability Set 3 (CS3); Intelligent Network Application protocol (INAP); Test Suite Structure and Test Purposes (TSS&TP) specification for Service Switching Function (SSF); Part 2: Call Party Handling (CPH)".

[6] ETSI EN 301 933-3: "Intelligent Network (IN); Intelligent Network Capability Set 3 (CS3); Intelligent Network Application Protocol (INAP); Test Suite Structure and Test Purposes (TSS&TP) specification for Service Switching Function (SSF); Part 3: Specialized Resource Function (SRF)".

[7] ETSI ES 201 296 (V1.2.2): "Integrated Services Digital Network (ISDN); Signalling System No.7; ISDN User Part (ISUP); Signalling aspects of charging".

[8] ISO/IEC 9646-1: "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General concepts".

[9] 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 301 931-1 [1];
- terms defined in ISO/IEC 9646-1 [8] and in ISO/IEC 9646-2 [9].

In particular, the following terms defined in ISO/IEC 9646-1 [8] 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
BV	Valid Behaviour tests
CA	Capability tests
CPH	Call Party Handling
CS	Call Segment
CS	Capability Set
EDP-R	Event Detection Point - Request
FSM	Finite State Machine
IN	Intelligent Network
INAP	Intelligent Network Application Protocol
IUT	Implementation Under Test
MSC	Message Sequence Chart
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
PIXIT	Protocol Implementation eXtra Information for Testing
SCF	Service Control Function
SCP	Service Control Point
SDL	Specification and Description Language
SRF	Specialized Resource Function
SSF	Service Switching Function
SSP	Service Switching Point
SUT	System Under Test
TCAP	Transaction Capabilities Application Part
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. It is expected, that each TP will result in a test case keeping the same name, specified in the ATS.

4.2 Grouping of Test Purposes per elementary procedures

The Test Purposes are grouped by elementary procedures. A procedure groups elementary INAP operations which it is possible to test together. For each elementary procedure, are defined: how to invoke it; and what are the possible return results and return error(s) at the INAP interface.

NOTE: Some have no results at all at this INAP interface. In these cases, and to have a "visible" result, the PCO will be at the signalling control interface.

4.3 Source of test purpose definitions

The Test Purposes are based on the requirement documented in EN 301 931-1 [1] and EN 301 931-2 [2].

4.4 Method used for developing TPs

4.4.1 Use of MSCs generated by the SDL model of Core INAP CS3

The SDL model of INAP CS3 is specified with object oriented SDL (SDL'92) and specifies the behaviour of the SSF. The SDL specification is the normative specification of the INAP behaviour and is contained in EN 301 931-4 [4].

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.

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 shown in table 1.

Table 1: Test purpose description sample

	TP name, e.g. IN3_A_BASIC_FC_BV_01
Work item no.:	Temporary work item number; to be deleted when the TPs are stable
IN2 Ref(tmp)	Reference to INAP CS2 TP (optional)
Purpose:	Textual phrasing of the TP to be achieved.
Requirements refs	Reference to clause(s) of EN 301 931-2 [2]. For TPs related to the SRF function: also reference to clause(s) of EN 301 931-3 [3]. In the latter case the part numbers are explicitly indicated (part 2 and/or part 3).
Selection Cond.	Reference to a formal selection expression, if the TP is related to an optional INAP feature. If the field is empty, the TP is unconditional (mandatory requirement(s)).
Preamble:	Reference to a preamble or "None".
Test description	Sequence of transmitted and received events and timeouts (see clause "TTCN-like notation"). Textual description is also used, as appropriate.
Pass criteria	Indication of reception (or assured non-reception) of decisive message(s) related to the TP.
Postamble:	Reference to a postamble or "None".

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

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 2.

Table 2: TP identifier naming convention scheme

Identifier: **IN3_<i>_<sss>_<pp>_<cc>_<nn>**

IN3 indicates IN Capability Set 3

<i>= interface: A SSF-SCF interface
B SSF-SRF interface
C SCF-SCF interface

<sss> = common set BASIC Basic set for CS3
CPH Call Party Handling from Capability Set 3
SRF SRF-related functions from Capability Set 3

<pp> = procedure name like
SF ServiceFiltering

<cc> = test category:
BVValid Behaviour tests
BI Invalid Behaviour tests
BOInopportune Behaviour tests

<nn> = sequential number: (01-99)

Example of test purpose and test case name: **IN3_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 CS3 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_Set-up OS

Terminating_Set-up TS

Originating_ 1_Party_Setup O1PS

Stable_1_Party S1P

Stable_2_Party S2P

Forward FW

Stable_Multi_Passive_Party (no. of passive legs n) SnPP

Stable_Multi_Party (no. of passive legs n) SnP

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_S1P_S1P.

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

TCAP primitives from SCF to SSF:

TC_InvokeReq (InvokeID, Class, DialogueID, OperationCode, OperationArg, Timeout);

TC_BeginReq (DialogueID, OriginatingAddress);

TC_ContinueReq (DialogueID, OriginatingAddress);

TC_EndReq (DialogueID, Termination);

TC_AbortReq (DialogueID).

TCAP primitives from SSF to SCF:

TC_InvokeInd (InvokeID, DialogueID, OperationCode, OperationArg, 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.

Some preambles contain references to an ASP Mgt_SetTriggerTable. This does not exist in the protocol, but in the SDL model it allows which Trigger Detection points need to be set before commencing the test case.

4.6 Test purpose parametrization and selection

In order to define an appropriate set of TPs for all functions and operations, but to enable deselection of TPs not applicable to particular IUTs, the following Test Parameters are defined in table 3.

NOTE: It is assumed, that these Test Parameters are mapped to corresponding PIXIT/Test Suite Parameters.

Table 3: Test Parameters applicable to TP selection

Test Parameter name	Type	Explanation
RCSR_INTERRUPTABLE	BOOLEAN	TRUE if the RequestCurrentStatusReport operation is interruptable, i.e. when after receiving the operation the SSF remains for some time in a state, where other operations can be received before the returnResult component is sent. Otherwise FALSE.
SINGLE_POINT_OF_CONTROL	BOOLEAN	TRUE if **only** Single Point of Control is implemented in the SSF.
BASIC_TARIFF_METHOD	IA5String	The possible values are "Currency" and "Pulse", depending on which method is implemented in the SSF for charging supervision.
TSPX_SCI_DESTB_ROUTE_CH	BOOLEAN	True if there is a destination routing address (given by TSPX_SCI_DESTB1_ADDR_CH) which forces the SSF to determine a charging tariff (act as CDP).
TSPX_SCI_DESTC_ROUTE_CH	BOOLEAN	True if there is a destination routing address (given by TSPX_SCI_DESTC1_ADDR_CH) which forces the SSF to determine a charging tariff (act as CDP).
TSPX_SCI_CLD_IN_CH	BOOLEAN	True if there is a called IN address (given by TSPX_SCI_IN_ADDR_CH) which forces the SSF to determine a charging tariff (act as CDP).

The following Test Parameters used to parameterize the TP descriptions, when necessary, are defined in table 4.

NOTE: It is assumed, that these Test Parameters are mapped to corresponding PIXIT/Test Suite Parameters.

Table 4: Test Parameters applicable to TP parametrization

Test Parameter name	Type	Explanation
STAT_RESOURCE_1	ResourceID	When describing TPs for StatusReport operations (RequestCurrentStatusReport, RequestEveryStatusChangeReport and RequestFirstStatusMatchReport), calls with 2 legs are made. STAT_RESOURCE_1 identifies the resources associated with leg 1 (controlling; user A). The association is independent of whether the StatusReport operations are performed inside this call context or not. In each case call-related operations are performed to bring the associated resources in a specified status ("busy" or "idle").
STAT_RESOURCE_2	ResourceID	Like STAT_RESOURCE_1, but for leg 2 (passive; user B).
STAT_RESOURCE_U	ResourceID	Syntactically valid Resource ID not identifying any resource known to the SSF.
STAT_MON_DUR_ANY	Duration	Duration value to be sent in the RequestEveryStatusChangeReport or RequestFirstStatusMatchReport invoke component. Any positive duration in the range specified by Duration or OMIT is allowed.

Test Parameter name	Type	Explanation
STAT_MON_DUR	Duration	Positive duration value to be sent in the RequestEveryStatusChangeReport or RequestFirstStatusMatchReport invoke component. OMIT is not allowed.
PROFILE_ID_1	ProfileIdentifier	Profile Identifier value compatible with the call-related data sent by the SCF/tester in a SetupInd message where CALL-DATA-1 is indicated as parameter for the SetupInd message. The parameter is used in TPs for Trigger Management operations. The parameter must be seen in connection with parameters TRIGGER_DATA_1 and CALL-DATA-1. It is assumed that a call with CALL-DATA-1 can be made to go through all DPs except **originationAttemptDenied**, **routeSelectFailure**, **authorizeRouteFailure** and **terminationAttemptDenied** by appropriate call operations like release or suspend etc. It is also assumed that a trigger can be set on all DPs except the mentioned ones, if the trigger uses PROFILE_ID_1 and TRIGGER_DATA_1 and that the trigger criteria will be fulfilled for a call with call-related data CALL-DATA-1.
TRIGGER_DATA_1	TriggerData	Trigger Data value (trigger criteria etc.) compatible with the call-related data sent by the SCF/tester in a "standard" SetupInd message, or where CALL-DATA-1 is indicated as parameter for the SetupInd message. See also PROFILE_ID_1.
CALL-DATA-1	Informal	This parameter denotes call-related data sent in a SetupInd message/ASP in an informal way. These data cover the called address, and depending on the kind of call and the signalling protocol, data like calling address, calling party's category, facilities etc. The important point is, that a call with these call-related data can be made to match trigger conditions and trigger criteria defined by parameters PROFILE_ID_1 and TRIGGER_DATA_1 at all DPs mentioned under PROFILE_ID_1.
SERVICE_KEY1	ServiceKey	Service key set for triggers defined by PROFILE_ID_1 and TRIGGER_DATA_1.
PROFILE_ID_2	ProfileIdentifier	Similar as PROFILE_ID_1, but a matching trigger condition can be set at DP **originationAttemptDenied**, if also TRIGGER_DATA_2 and CALL-DATA-2 are set appropriately.
TRIGGER_DATA_2	TriggerData	Similar as TRIGGER_DATA_1, but a matching trigger condition can be set at DP **originationAttemptDenied**, if also PROFILE_ID_2 and CALL-DATA-2 are set appropriately.
CALL-DATA-2	Informal	Similar as CALL-DATA-1, but a matching trigger condition can be set at DP **originationAttemptDenied**, if also TRIGGER_DATA_2 and PROFILE_ID_2 are set appropriately.
SERVICE_KEY2	ServiceKey	Service key set for triggers defined by PROFILE_ID_2 and TRIGGER_DATA_2.

Test Parameter name	Type	Explanation
PROFILE_ID_3	ProfileIdentifier	Similar as PROFILE_ID_1, but a matching trigger condition can be set at DP **routeSelectFailure**, if also TRIGGER_DATA_3 and CALL-DATA-3 are set appropriately.
TRIGGER_DATA_3	TriggerData	Similar as TRIGGER_DATA_1, but a matching trigger condition can be set at DP **routeSelectFailure**, if also PROFILE_ID_3 and CALL-DATA-3 are set appropriately.
CALL-DATA-3	Informal	Similar as CALL-DATA-1, but a matching trigger condition can be set at DP **routeSelectFailure**, if also TRIGGER_DATA_3 and PROFILE_ID_3 are set appropriately.
SERVICE_KEY3	ServiceKey	Service key set for triggers defined by PROFILE_ID_3 and TRIGGER_DATA_3.
PROFILE_ID_4	ProfileIdentifier	Similar as PROFILE_ID_1, but a matching trigger condition can be set at DP **authorizeRouteFailure**, if also TRIGGER_DATA_4 and CALL-DATA-4 are set appropriately.
TRIGGER_DATA_4	TriggerData	Similar as TRIGGER_DATA_1, but a matching trigger condition can be set at DP **authorizeRouteFailure**, if also PROFILE_ID_4 and CALL-DATA-4 are set appropriately.
CALL-DATA-4	Informal	Similar as CALL-DATA-1, but a matching trigger condition can be set at DP **authorizeRouteFailure**, if also TRIGGER_DATA_4 and PROFILE_ID_4 are set appropriately.
SERVICE_KEY4	ServiceKey	Service key set for triggers defined by PROFILE_ID_4 and TRIGGER_DATA_4.
PROFILE_ID_5	ProfileIdentifier	Similar as PROFILE_ID_1, but a matching trigger condition can be set at DP **terminationAttemptDenied**, if also TRIGGER_DATA_5 and CALL-DATA-5 are set appropriately.
TRIGGER_DATA_5	TriggerData	Similar as TRIGGER_DATA_1, but a matching trigger condition can be set at DP **terminationAttemptDenied**, if also PROFILE_ID_5 and CALL-DATA-5 are set appropriately.
CALL-DATA-5	Informal	Similar as CALL-DATA-1, but a matching trigger condition can be set at DP **terminationAttemptDenied**, if also TRIGGER_DATA_5 and PROFILE_ID_5 are set appropriately.
SERVICE_KEY5	ServiceKey	Service key set for triggers defined by PROFILE_ID_5 and TRIGGER_DATA_5.
PROFILE_ID_1A	ProfileIdentifier	Profile different from PROFILE_ID_1, matching other calls.
TRIGGER_DATA_1A	TriggerData	Similar as TRIGGER_DATA_1, but used to trigger calls of profile PROFILE_ID_1A.
CALL-DATA-1A	Informal	Call related data used to be able to trigger a call, when triggering conditions according to PROFILE_ID_1A and TRIGGER_DATA_1A have been set.
RFSM_MAND	BOOLEAN	TRUE if the resourceID and resourceStatus parameters are implemented as mandatory parameters in the **RequestFirstStatusMatchReport Argument**. Tmp.NOTE: resourceID and resourceStatus parameters are currently defined to be OPTIONAL in the **RequestFirstStatusMatchReportArg** but this is considered to be an error, because a status match cannot be decided when these parameters are missing.

Test Parameter name	Type	Explanation
TSSF_TIMEOUT_IDP_DEFAULT	Integer	Default timeout value (seconds) for the IDP operation, i.e. time after which the SSF aborts the TCAP dialog, when no operation is received from the SCF after having sent an InitialDP invoke component and being in the "Waiting for instructions" state.
TSSF_TIMEOUT_DEFAULT	Integer	Default timeout value (seconds) after which the SSF aborts the TCAP dialog or sends an EntityReleased invoke component, when no operation is received from the SCF while being in the "Waiting for instructions" state (not after having sent the InitialDP operation).
TSSF_TIMEOUT_RESET	Integer	Timeout value (seconds) the tester uses in the ResetTimer invoke component. Is required to be greater than 10 % of TSSF_TIMEOUT_DEFAULT and greater than 10 % of TSSF_TIMEOUT_IDP_DEFAULT.
DEFAULT_OMIT_GAP_TREATMENT	IA5String	The default gap treatment to be performed when the gapTreatment parameter is missing in the CallGap invoke component can be: "Release call", "Continue call processing (non-IN)" or "Other". In case "Continue call processing (non-IN)" is selected and the gap criteria match for an incoming call, it is assumed, that the call is being processed further without SCF interaction, i.e. no InitialDP operation is invoked by the SSF. The call is successfully transferred to the called user, if the non-IN facilities of the SSF and the call-related data present in the SetupInd signal allow this.
DEFAULT_ITS_GAP_TREATMENT	IA5String	The default gap treatment to be performed when the gapTreatment parameter contained in the CallGap invoke component indicates "informationToSend" (not "both/informationToSend") can be: "Release call", "Continue call processing (non-IN)" or "Other". In case "Continue call processing (non-IN)" is selected and the gap criteria match for an incoming call, it is assumed, that the call is being processed further without SCF interaction, i.e. no InitialDP operation is invoked by the SSF. The indicated "informationToSend" is sent to the calling user and the call is successfully transferred to the called user, if the non-IN facilities of the SSF and the call-related data present in the SetupInd signal allow this.
SCF_ID	OCTET STRING	SCF ID assigned to the tester and known by the SSF.
FCI_IN_CRI1	INRecordIndicators	Value of INRecordIndicators used in the first FurnishChargingInformation operation (in the FurnishChargingInformation TPs)
FCI_IN_CRI2	INRecordIndicators	Value of INRecordIndicators used in the second FurnishChargingInformation operation (in the FurnishChargingInformation TPs)
FCI_PARTY_TO_CHARGE	PartyToCharge	Value of PartyToCharge used in the FurnishChargingInformation TPs
CHARGING_TARIFF	chargingTariff (TariffCurrency | TariffPulse) (see ASN.1 type ChargingTariffInformation)	chargingTariff to be used in APM(ChargingTariffInformation) messages, e.g. in RNC TPs.
ADD_ON_CHARGE	addOnCharge (addOnChargeCurrency | addOnChargePulse) (see ASN.1 type AddOnCharginInformation)	addOnCharge to be used in APM(AddOnChargingInformation) messages, e.g. in RNC TPs.

Test Parameter name	Type	Explanation
TSPX_EXCEPT_CRGT_RELEASE	BOOLEAN	TRUE if the SSF releases the connection, when performing exception handling on a received APM(crgt, "subscriber charge") message. FALSE otherwise (the connection is continued).
TSPX_EXCEPT_AOCRG_RELEASE	BOOLEAN	TRUE if the SSF releases the connection, when performing exception handling on a received APM(aocrg, "subscriber charge") message. FALSE otherwise (the connection is continued).
TSPX_EXCEPT_CRGA_RELEASE	BOOLEAN	TRUE if the SSF releases the connection, when performing exception handling on a received APM(crga, "not accepted") message. FALSE otherwise (the connection is continued).
TSPX_EXCEPT_CRGT_TOUT_RELEASE	BOOLEAN	TRUE if the SSF releases the connection, when performing exception handling after timeout (no APM(crga) message received after sending APM(crgt, "subscriber charge")). FALSE otherwise (the connection is continued).
TSPX_EXCEPT_AOCRG_TOUT_RELEASE	BOOLEAN	TRUE if the SSF releases the connection, when performing exception handling after timeout (no APM(crga) message received after sending APM(aocrg, "subscriber charge")). FALSE otherwise (the connection is continued).
TSPX_SCI_DESTB1_ADDR	Address	Address of called user 1 (B-party) in a network different from the network of the SSF under test. The address does not enable the SSF to determine a charging tariff on its own.
TSPX_SCI_DESTC1_ADDR	Address	Address of called user 2 (C-party) in a network different from the network of the SSF under test. The address does not enable the SSF to determine a charging tariff on its own.
TSPX_SCI_DESTB1_ADDR_CH	Address	Address of called user 1 (B-party) in a network different from the network of the SSF under test. The address enables the SSF to determine a charging tariff on its own.
TSPX_SCI_DESTC1_ADDR_CH	Address	Address of called user 2 (C-party) in a network different from the network of the SSF under test. The address enables the SSF to determine a charging tariff on its own.
TSPX_SCI_IN_ADDR_CH	Address	IN Address enabling the SSF to determine a charging tariff on its own.
TSPX_SCI_DESTC_ADDR	Address	Address of called user (C-party) in a network different from the network of the SSF under test.
TSPX_SCI_DESTB_IN_NUMBER	Address	Called IN number (B-party) which requires special charging.

The Selection Expressions listed in table 5 are used to select/deselect TPs in a formal way, when necessary.

Table 5: Selection Expressions

Selection Expression name	Expression	Description
RCSR_Interruptable	RCSR_INTERRUPTABLE	
SinglePointOfControl	SINGLE_POINT_OF_CONTROL	
RFSMMandatory	RFSM_MAND	
DefaultOmitGapRelease	DEFAULT_OMIT_GAP_TREATMENT = "Release call"	
DefaultOmitGapContinue	DEFAULT_OMIT_GAP_TREATMENT = "Continue call processing (non-IN)"	
DefaultItsGapRelease	DEFAULT_ITS_GAP_TREATMENT = "Release call"	
DefaultItsGapContinue	DEFAULT_ITS_GAP_TREATMENT = "Continue call processing (non-IN)"	
CurrencyTariff	BASIC_TARIFF_METHOD = "Currency"	
PulseTariff	BASIC_TARIFF_METHOD = "Pulse"	
SelCDPRoutingB	TSPX_SCI_DESTB_ROUTE_CH	True if there is a destination routing address to the first called user (given by TSPX_SCI_DESTB1_ADDR_ CH) which forces the SSF to determine a charging tariff (act as CDP).
SelCDPRoutingC	TSPX_SCI_DESTC_ROUTE_CH	True if there is a destination routing address to the second called user (given by TSPX_SCI_DESTC1_ADDR_ CH) which forces the SSF to determine a charging tariff (act as CDP).
SelCDPINAddress	TSPX_SCI_CLD_IN_CH	True if there is a called IN address (given by TSPX_SCI_IN_ADDR_CH) which forces the SSF to determine a charging tariff (act as CDP).

5 Test configuration

Figure 1 shows the test configuration assumed for the basic Test Purposes.

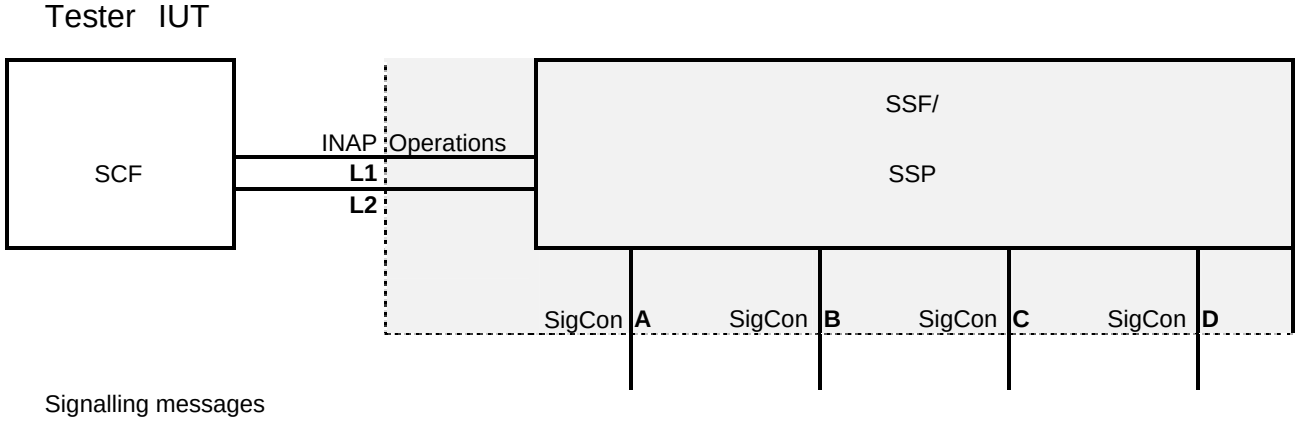


Figure 1: Test configuration for the basic Test Purposes

This test configuration covers a single SCP and a single SSP, where the SCP is simulated by the tester and the SSF is the implementation under test (IUT).

INAP PDUs (operations) are exchanged between the tester and the IUT across the interface named **L1** (or L2 etc.), which corresponds to a PCO in the TTCN-like notation used for the description of the test behaviour (see clause 6.4.2). INAP PDUs are embedded in TCAP messages as described in clause 10 of EN 301 931-1 [1] and clause 15 of EN 301 931-2 [2].

When call-related operations are tested, signalling messages are exchanged between the tester and the IUT, where the signalling terminations in the IUT are named SigCon A to SigCon D. Depending on the implementation, the signalling messages can be messages of the DSS1 protocol, ISUP protocol or another protocol (see e.g. clause 6.2.2.1 of EN 301 931-2 [2]). To be independent of any particular signalling protocol, **Abstract Signalling Primitives** are used in the test descriptions instead of signalling messages. For the definition of the Abstract Signalling Primitives see clause 6.2.2.2 of EN 301 931-2 [2].

NOTE: Test configurations including SRF entities are defined in EN 301 933-3 [6].

6 TSS and TPs for CS3 basic capabilities

6.1 Introduction

The Test Purposes related to the basic capabilities of the SSF are grouped according to the INAP operations or procedures.

6.2 Basic procedures

6.2.1 List of procedures

The Test Purposes for Basic CS3 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.

AT	ActivityTest
AC	ApplyCharging
CA	Cancel
CF	CallInformation
CG	CallGap
CI	CollectInformation
CO	Connect
CT	CreateOrRemoveTriggerData
CU	Continue
CW	ContinueWithArgument
DP	InitialDP
ER	EntityReleased
FC	FurnishChargingInformation
IC	InitiateCallAttempt
MT	ManageTriggerData
RC	ReleaseCall
RE	RequestEveryStatusChangeReport
RF	RequestFirstStatusMatchReport
RN	RequestNotificationChargingEvent
RP	ReportUTSI
RR	RequestReportBCSMEvent
RS	ResetTimer
RT	RequestCurrentStatusReport

SY	SelectFacility
SF	ServiceFiltering
SCI	SendChargingInformation
SP	SetServiceProfile

6.2.2 Definitions of the procedures

ActivityTest procedure (AT)

```
Invoke: ActivityTest
Return Result:  ActivityTest
Return Error:   None
```

ApplyCharging procedure (AC)

```
Invoke: ApplyCharging
Return Result:  ApplyChargingReport
Return Error:   ApplyCharging
    ApplyChargingReport
```

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:  None
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
```

Continue procedure (CU)

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

ContinueWithArgument procedure (CW)

```
Invoke: ContinueWithArgument
Return Result:  None
Return Error:   ContinueWithArgument
```

CreateOrRemoveTriggerData procedure (CT)

```
Invoke: CreateOrRemoveTriggerData
Return Result:  CreateOrRemoveTriggerData
Return Error:   CreateOrRemoveTriggerData
```

EntityReleased (ER)

```
Invoke: EntityReleased
Return Result:  None
Return Error:   None
```

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
```

ManageTriggerData procedure (MT)

```
Invoke: ManageTriggerData
Return Result:  ManageTriggerData
Return Error:   ManageTriggerData
```

ReleaseCall procedure (RC)

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

ReportUTSI procedure (RP)

```
Invoke: RequestReportUTSI
    ReportUTSI
Return Result:  None
Return Error:   RequestReportUTSI
```

RequestCurrentStatusReport procedure (RT)

```
Invoke: RequestCurrentStatusReport
Return Result:  RequestCurrentStatusReport
Return Error:   RequestCurrentStatusReport
```

RequestEveryStatusChangeReport procedure (RE)

```
Invoke: RequestEveryStatusChangeReport
    StatusReport
    CancelStatusReportRequest
Return Result:  RequestEveryStatusChangeReport
Return Error:   RequestEveryStatusChangeReport
```

RequestFirstStatusMatchReport procedure (RF)

```
Invoke: RequestFirstStatusMatchReport
    StatusReport
    CancelStatusReportRequest
Return Result:  RequestFirstStatusMatchReport
Return Error:   RequestFirstStatusMatchReport
```

RequestNotificationChargingEvent procedure (RN)

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

RequestReportBCSMEvent procedure (RR)

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

ResetTimer procedure (RS)

```
Invoke: ResetTimer
Return Result:  None
Return Error:   ResetTimer
```

SelectFacility procedure (SY)

```
Invoke: SelectFacility
Return Result:  None
Return Error:   SelectFacility
```

SendChargingInformation procedure (SN)

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

ServiceFiltering procedure (SF)

```
Invoke: ActivateServiceFiltering
Return Result:  ServiceFilteringResponse
    ActivateServiceFiltering
Return Error:   ActivateServiceFiltering
```

SetServiceProfile procedure (SP)

```
Invoke: SetServiceProfile
Return Result:  None
Return Error:   SetServiceProfile
```

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

Table 6 shows the structure of the test suites for SSF functions and the number of Test Purposes produced.

Table 6: Test suite structure of the SSF test

Test purposes for the relay method			
Procedure/Group	**Group identifier**	**Category**	**Number**
ActivityTest	IN3_A_BASIC_AT	BV	3
		BI	1
		BO	0
ApplyCharging	IN3_A_BASIC_AC	BV	22
		BI	1
		BO	1
Cancel	IN3_A_BASIC_CA	BV	4
		BI	1
		BO	1
CallInformation	IN3_A_BASIC_CF	BV	6
		BI	1
		BO	2
CallGap	IN3_A_BASIC_CG	BV	33
		BI	2
		BO	0
CollectInformation	IN3_A_BASIC_CI	BV	1
		BI	1
		BO	3
Connect	IN3_A_BASIC_CO	BV	10
		BI	1
		BO	1
Continue	IN3_A_BASIC_CU	BV	2
		BI	0
		BO	0
ContinueWithArgument	IN3_A_BASIC_CW	BV	0
		BI	0
		BO	0
CreateOrRemoveTriggerData	IN3_A_BASIC_CT	BV	40
		BI	4
		BO	1
EntityReleased	IN3_A_BASIC_ER	BV	2
		BI	0
		BO	0
InitialDP	IN3_A_BASIC_DP	BV	7
		BI	2
		BO	0
EventNotificationCharging	IN3_A_BASIC_EN	BV	0
		BI	0
		BO	0
FurnishChargingInformation	IN3_A_BASIC_FC	BV	22
		BI	1
		BO	1
InitiateCallAttempt	IN3_A_BASIC_IC	BV	5
		BI	0
		BO	2
ManageTriggerData	IN3_A_BASIC_MT	BV	21
		BI	3
		BO	1
ReleaseCall	IN3_A_BASIC_RC	BV	3
		BI	0
		BO	1
RequestEveryStatusChangeReport	IN3_A_BASIC_RE	BV	11
		BI	9
		BO	3

Test purposes for the relay method			
Procedure/Group	**Group identifier**	**Category**	**Number**
RequestFirstStatusMatchReport	IN3_A_BASIC_RF	BV	11
		BI	10
		BO	2
RequestNotificationChargingEvent	IN3_A_BASIC_RN	BV	10
		BI	2
		BO	1
ReportUTSI	IN3_A_BASIC_RP	BV	0
		BI	0
		BO	0
RequestReportBCSMEvent	IN3_A_BASIC_RR	BV	34
		BI	2
		BO	1
ResetTimer	IN3_A_BASIC_RS	BV	6
		BI	2
		BO	2
RequestCurrentStatusReport	IN3_A_BASIC_RT	BV	5
		BI	2
		BO	1
SelectFacility	IN3_A_BASIC_SY	BV	2
		BI	2
		BO	1
ServiceFiltering	IN3_A_BASIC_SF	BV	3
		BI	2
		BO	2
SendChargingInformation	IN3_A_BASIC_SN	BV	0
		BI	0
		BO	0
SetServiceProfile	IN3_A_BASIC_SP	BV	2
		BI	3
		BO	1
Total:			345

6.4 Notations

6.4.1 Names of preambles and postambles

INAP requires a large set of preambles. Due to the complexity of their description, the Connection View (CV) model is used for an understanding of the configuration, referring to the following CV objects:

- CallSegmentAssociation (always initial);
- CallSegment;
- Connection point;
- Legs.

NOTE 1: The controlling leg can be either joined or surrogate. The controlling leg identifies the physical access to the end user.

NOTE 2: The legs are named by the legID, and there is a unique correspondence between a legID and a BCSM.

Restrictions: The test configuration is limited to four passive legs within a call segment, and three call segments within a call segment association and two call segment associations.

Comment on T preambles: The preamble T_TS (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.

Based on these considerations, the naming convention indicated in clause 4.5 is used.

6.4.2 TTCN-like notation for preamble, test case and postamble description

The notation used to describe the trees containing the required INAP operations and signalling control primitives, is a TTCN-like notation, showing what is sent (character !) and received (character ?) by the main tester (playing the role of the SCF).

The IUT is an SSF while the main lower tester represents an SCF.

The PCO-like notation **L1** (or in addition **L2**, etc.) are used to send and receive INAP operations. L1 is used with the first dialog opened in a test case (including the preambles). L2 is used for the second dialog etc. (even when a previous dialog has closed in between).

The co-ordination points (CPs) address the Signalling Control entities (SigCon*X*, X=A, B, C etc.) of the legs in the related Call Segment (only one CSA is used).

CP1_1 addresses SigConA (in CSA1), CP1_4 addresses SigConD in CSA1 etc.

Each preamble shows the state from where it starts (idle or a different state reached by the execution of another preamble), then it shows the operations executed in this preamble and finally the state or configuration reached, using the notation described above.

Each test purpose description shows the state from where it starts by identifying a preamble.

6.4.3 Representation of preambles, postambles and Test Purposes using MSCs

In addition to the TTCN-like notation, an MSC is drawn from the SDL simulator to represent each preamble or postamble. For each test purpose, an MSC is also given, in addition to the tabular description of each TP.

Each MSC shows the interface between SCF and SSF using TCAP primitives, and the signalling control points. As there can be any number of signalling control points (from 1 up to 8), the MSC shows SigCon A in one column, while all the other SigCon are merged in a second column. The parameter CallRef number makes it possible to identify the SigCon concerned, SigCon B being number 2, SigCon C being 3, etc.

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

6.4.4.1 General

The MSCs show the exchanges of PDUs of the TCAP protocol, as well as the Core INAP protocol and the signalling connection service primitives. PDUs of both protocols and the service primitives generally make use of parameters.

The complete list of the parameters for the Core INAP protocol is given in clause 12 of EN 301 931-2 [2]. Only a limited list of parameters is used in the MSCs, as described in clause 6.4.4.2.

6.4.4.2 INAP operation parameters and their values

Each preamble, test purpose description and postamble uses a limited set of operations (listed below) with the indication of the main parameters used:

- IDP (Initial Detection Point): trigger;
- CWA (ContinueWithArgument): legID or csID;
- CON (Connect): legToBeCreated (default = 2), csID;
- DL (DisconnectLeg): legToBeReleased;
- ERB (EventReportBCSM): legID, eventTypeBCSM;

NOTE: The presence of the legID is required for some eventTypes (e.g.oDisconnect), and is optional for other eventTypes. When the legID is not shown in connection with an ERB operation, the presence is considered to be optional.

- ICA (InitiateCallAttempt): legToBeCreated (default = 1), newCallSegment (default = 1);
- RRB (RequestReportBCSMEvent): legID, monitorMode, eventTypeBCSM;
- SL (SplitLeg): legToBeSplit, newCallSegmentID;
- CIRQ (CallInformationRequest): legID;
- CIRP (CallInformationReport): legID;
- ACRQ (ApplyChargingRequest): legID;

NOTE: The releaseWhenLimitReached parameter should not be specified.

- ACRP (ApplyChargingReport): legID;

Each operation shows the value of the required parameters, knowing that the leg numbers successively take the values 1, 2, 3, etc. (in case of an ICA performed in a CSA without CSs, leg numbering normally starts with 2).

6.4.4.3 Cause values related to signalling connection release messages

The following cause value indications appear in the TPs:

- Normal clearing
  Cause recognized with a clearing event detected at the O_Disconnect or T_Disconnect DP;
- Busy cause;
- RouteSelectFailure cause.

6.4.4.4 TCAP operation parameters and their values

The list of parameters for the TCAP protocol is repeated here for each TCAP primitive. 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.

6.5 Preambles and postambles used

6.5.1 Preamble descriptions

6.5.1.1 O_OS preamble

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

O_null

CP1_1!SetupInd

L1?InitialDP

O_OS

6.5.1.2 O_OS_ColIn preamble

This preamble is used to bring the IUT from the idle or Null state to the 1 party state, where it is assumed that TDP CollectedInfo is set.

O_null

CP1_1!SetupInd

L1?InitialDP(CollectedInfo)

O_OS

6.5.1.3 O_S2P preamble

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

O_OS

L1!RequestReportBCSMEvent(2,interrupted,oDisconnect)

L1!Connect(2,1)

CP1_2?SetUpReq

CP1_2!SetUpConf

CP1_1!SetUpResp

O_S2P

6.5.1.4 I_S1P preamble

null

L1!InitiateCallAttempt(1,1)

L1!RequestReportBCSMEvent(1,interrupted,oDisconnect)

L1!ContinueWithArgument(legID=1)

CP1_1?SetUpReq

CP1_1!SetUpConf

I_S1P

6.5.1.5 I_S1P_S1P_S1P preamble

I_S1P

L1!InitiateCallAttempt(2,2)

L1!RequestReportBCSMEvent(2,notifyAndContinue,oDisconnect)

L1!ContinueWithArgument(legID=2)

CP1_2?SetUpReq

CP1_2!SetUpConf

L1!InitiateCallAttempt(3,3)

L1!RequestReportBCSMEvent(3,notifyAndContinue,oDisconnect)

L1!ContinueWithArgument(legID=3)

CP1_3?SetUpReq

CP1_3!SetUpConf

I_S1P_S1P_S1P

6.5.1.6 T_TS preamble

T_null

CP1_1!SetupInd

L1?IDP(termAttemptAuthorized)

T_TS

6.5.1.7 T_S2P preamble

T_TS

L1!RequestReportBCSMEvent(1,interrupted,tDisconnect)

L1!ContinueWithArgument(legID=1)

CP1_2?SetUpReq

CP1_2!SetUpConf

CP1_1?SetUpResp

T_S2P

6.5.1.8 STAT_S2P

This preamble sets up a 2-party call, where it is assumed, that the resources associated with leg 1 and leg 2 can be identified by the SSF with the resourceIDs contained in Test Parameters STAT_RESOURCE_1 and STAT_RESOURCE_1 respectively (see table 4 on page 13). This is informally indicated by passing the keywords resource1 and resource2 to SetupInd and Connect respectively.

STAT_S2P

CP1_1!SetupInd(resource1)

L1?InitialDP

L1!RequestReportBCSMEvent(1,notifyAndContinue,oDisconnect)

L1!RequestReportBCSMEvent(2,interrupted,oDisconnect)

L1!Connect(2,1,resource2)

CP1_2?SetUpReq

CP1_2!SetUpConf

CP1_1?SetUpResp

STAT_S2P

6.5.1.9 STAT_1P

STAT_S2P

L1!DisconnectLeg(2)

L1?DisconnectLeg ReturnResult

CP1_2?ReleaseReq

L1?EventReportBCSM(2,oDisconnect)

STAT_1P

6.5.1.10 STAT_S1P_1P

STAT_S2P

L1!SplitLeg(1,2)

L1?SplitLegReturnResult

L1!ContinueWithArgument (csID = 1)

STAT_S1P_1P

6.5.1.11 PRE_TRIG

This preamble is used in TPs for the CreateAndRemoveTriggerData, ManageTriggerData and SetServiceProfile procedures. An "initialized" SSF trigger table is required before any of these procedures is performed. The best initialization would be the removal of all Triggers from the SSF. If this is not possible: the minimum requirement is, that the table is initialized in a way, that only such trigger criteria survive, that do not apply to the call-related data that are provided with the SetupInd messages/ASPs used in these TPs. See table 4 on page 13 (Test Parameters) for more information on this subject.

NOTE: Manual operation could be necessary for formal ASP "InitializeTriggerTable". Therefore it could be necessary to install an additional PCO in the ATS realizing these TPs. It could also be necessary to wait for the result of the execution of this ASP, which could also require manual intervention. For the purposes of the present document, no care is taken about the realization of the ASP.

InitializeTriggerTable

PRE_TRIG

6.5.1.12 PRE_ACTIVATE_TRIG1

This preamble is used in TPs for the CreateAndRemoveTriggerData, ManageTriggerData and SetServiceProfile procedures. After trigger table initialization a single trigger is created and activated. The resulting triggerIdentifier is tDPIdentifier1, which is used in TP descriptions applying this preamble.

PRE_TRIG

L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L2!TC-BEGIN

L2?TC-END

L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1)

L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1)

L3!TC-BEGIN

L3?TC-END

L3?ManageTriggerData returnResult(activated, tDPId1))

PRE_ACTIVATE_TRIG1

6.5.1.13 PRE_DEACTIVATE_TRIG1

This preamble is used in TPs for the CreateAndRemoveTriggerData, ManageTriggerData and SetServiceProfile procedures. After trigger table initialization a single trigger is created and deactivated. The resulting triggerIdentifier is triggerIdentifier1, which is used in TP descriptions applying this preamble.

PRE_TRIG

L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L2!TC-BEGIN

L2?TC-END

L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1)

NOTE: The trigger is deactivated

PRE_DEACTIVATE_TRIG1

6.5.1.14 PRE_ACTIVATE_TRIG_PROF_2

This preamble is used in TPs for the CreateAndRemoveTriggerData, ManageTriggerData and SetServiceProfile procedures. After trigger table initialization two triggers are created and activated. The triggers are different in profile and trigger data, but apply to the same DP. The resulting triggerIdentifiers (Integers) are tDPId1 and tDPId2, which are used in TP descriptions applying this preamble.

PRE_TRIG

L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L2!TC-BEGIN

L2?TC-END

L2?CreateOrRemoveTriggerData returnResult(created, tDPId1)

L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1A, PROFILE_ID_1A, TRIGGER_DATA_1A, defaultFaultHandling omitted)

L3!TC-BEGIN

L3?TC-END

L3?CreateOrRemoveTriggerData returnResult(created, tDPId2)

L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), triggers ((tDPId1),(tDPId2))

L4!TC-BEGIN

L4?TC-END

L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPId1), (activated, tDPId2)))

PRE_ACTIVATE_TRIG_PROF_2

6.5.1.15 PRE_DEACTIVATE_TRIG_PROF_2

This preamble is used in TPs for the CreateAndRemoveTriggerData, ManageTriggerData and SetServiceProfile procedures. After trigger table initialization two triggers are created and deactivated. The triggers are different in profile and trigger data, but apply to the same DP. The resulting triggerIdentifiers (Integers) are tDPId1 and tDPId2, which are used in TP descriptions applying this preamble.

PRE_TRIG

L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L2!TC-BEGIN

L2?TC-END

L2?CreateOrRemoveTriggerData returnResult(created, tDPId1)

L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1A, PROFILE_ID_1A, TRIGGER_DATA_1A, defaultFaultHandling omitted)

L3!TC-BEGIN

L3?TC-END

L3?CreateOrRemoveTriggerData returnResult(created, tDPId2)

NOTE: The triggers are deactivated.

PRE_DEACTIVATE_TRIG_PROF_2

6.5.1.16 PRE_ACTIVATE_TRIG_DP_2

This preamble is used in TPs for the CreateAndRemoveTriggerData, ManageTriggerData and SetServiceProfile procedures. After trigger table initialization two triggers are created and activated. The triggers have common profile and trigger data, but apply to different DPs: **oOrigAttemptAuthorized** and **oAnswer** respectively. The resulting triggerIdentifiers (Integers) are tDPId1 and tDPId2, which are used in TP descriptions applying this preamble.

PRE_TRIG

L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L2!TC-BEGIN

L2?TC-END

L2?CreateOrRemoveTriggerData returnResult(created, tDPId1)

L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAnswer, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L3!TC-BEGIN

L3?TC-END

L3?CreateOrRemoveTriggerData returnResult(created, tDPId2)

L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), triggers ((tDPId1),(tDPId2))

L4!TC-BEGIN

L4?TC-END

L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPId1), (activated, tDPId2)))

PRE_ACTIVATE_TRIG_DP_2

6.5.1.17 PRE_DEACTIVATE_TRIG_DP_2

This preamble is used in TPs for the CreateAndRemoveTriggerData, ManageTriggerData and SetServiceProfile procedures. After trigger table initialization two triggers are created and deactivated. The triggers have common profile and trigger data, but apply to different DPs: **oOrigAttemptAuthorized** and **oAnswer** respectively. The resulting triggerIdentifiers (Integers) are tDPId1 and tDPId2, which are used in TP descriptions applying this preamble.

PRE_TRIG

L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L2!TC-BEGIN

L2?TC-END

L2?CreateOrRemoveTriggerData returnResult(created, tDPId1)

L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAnswer, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L3!TC-BEGIN

L3?TC-END

L3?CreateOrRemoveTriggerData returnResult(created, tDPId2)

NOTE: The triggers are deactivated.

PRE_DEACTIVATE_TRIG_DP_2

6.5.1.18 PRE_ACTIVATE_TRIG_DP_3

This preamble is used in TPs for the CreateAndRemoveTriggerData, ManageTriggerData and SetServiceProfile procedures. After trigger table initialization three triggers are created. The triggers have common profile and trigger data, but apply to different DPs: **oOrigAttemptAuthorize**, **doAnswer** and **tDisconnect** respectively. The resulting triggerIdentifiers (Integers) are tDPId1, tDPId2 and tDPId3, which are used in TP descriptions applying this preamble. The triggers with identifiers tDPId1 and tDPId2 are activated while the trigger with identifier tDPId3 is deactivated.

PRE_TRIG

L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L2!TC-BEGIN

L2?TC-END

L2?CreateOrRemoveTriggerData returnResult(created, tDPId1)

L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAnswer, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L3!TC-BEGIN

L3?TC-END

L3?CreateOrRemoveTriggerData returnResult(created, tDPId2)

L4!CreateOrRemoveTriggerData invoke(createOrRemove omitted, tDisconnect, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted)

L4!TC-BEGIN

L4?TC-END

L4?CreateOrRemoveTriggerData returnResult(created, tDPId3)

L5!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), triggers ((tDPId1),(tDPId2))

L5!TC-BEGIN

L5?TC-END

L5?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPId1), (activated, tDPId2)))

NOTE: The trigger identified by tDPId3 is deactivated by default.

PRE_ACTIVATE_TRIG_DP_3

6.5.1.19 O_OSA_AC_CURR

This preamble is used to test the ApplyCharging procedure. An incoming call is initiated and the FurnishChargingInformation operation is issued when the IDP message is received from the SSF, before call setup continues. The sCFchargingTariff "TariffCurrency" is transferred to the SSF, to charge leg 2.

The objective is to "register" the following values:

a) currentTariffCurrency,

b) nextTariffCurrency,

c) tariffSwitchOverTime.

NOTE 1: The term "register" in this context does not necessarily mean that the SSF performs the Charging Registration function.

NOTE 2: The tariffSwitchoverTime is an absolute time. In the realization of this preamble in an ATS, the tariffDuration of the currentTariffCurrency should be set, to ensure tariff switch-over within a suitable time for the test case.

O_null

CP1_1!SetupInd

L1?InitialDP

L1!RequestReportBCSMEvent(1,interrupted,oDisconnect)

L1!FurnishChargingInformation invoke(sCFChargingTariff=TariffCurrency, chargingAddress = leg 2, callAttemtChargeCurrency omitted, callSetupChargeCurrency omitted,)

L1!Connect(2,1)

CP1_2?SetUpReq

O_OSA_AC_CURR

6.5.1.20 O_OSA_AC_PULSE

This preamble is used to test the ApplyCharging procedure. An incoming call is initiated and the FurnishChargingInformation operation is issued when the IDP message is received from the SSF, before call setup continues. The sCFchargingTariff "TariffPulse" is transferred to the SSF, to charge leg 2.

The objective is to "register" the following values:

a) currentTariffPulse,

b) nextTariffPulse,

c) tariffSwitchOverTime.

NOTE 1: The term "register" in this context does not necessarily mean that the SSF performs the Charging Registration function.

NOTE 2: The tariffSwitchoverTime is an absolute time. In the realization of this preamble in an ATS, an appropriate TS Operation should be used to adjust the TariffSwitchoverTime to a value being near to 15 minutes after the "current time" (absolute time when the preamble is executed).

O_null

CP1_1!SetupInd

L1?InitialDP

L1!RequestReportBCSMEvent(1,interrupted,oDisconnect)

L1!FurnishChargingInformation invoke(sCFChargingTariff=TariffPulse, chargingAddress = leg 2, callAttemtChargePulse omitted, callSetupChargePulse omitted)

L1!Connect(2,1)

CP1_2?SetUpReq

O_OSA_AC_PULSE

6.5.1.21 O_OSA_AC

This preamble is used to test the ApplyCharging procedure, when the TP is independent of whether the "TariffCurrency" or the "TariffPulse" method (or both) is implemented in the IUT. An incoming call is initiated and the FurnishChargingInformation operation is issued when the IDP message is received from the SSF, before call setup continues.

Depending on Test Parameter BASIC_TARIFF_METHOD, sCFchargingTariff "TariffCurrency" or "TariffPulse" is transferred to the SSF, to charge leg 2.

The objective is to "register" the following values (if the "TariffCurrency" method is implemented):

a) currentTariffCurrency,

b) nextTariffCurrency,

c) tariffSwitchOverTime,

or (if only the "TariffPulse" method is implemented):

a') currentTariffPulse,

b') nextTariffPulse,

c') tariffSwitchOverTime.

NOTE 1: The term "register" in this context does not necessarily mean that the SSF performs the Charging Registration function.

NOTE 2: The tariffSwitchoverTime is an absolute time. In the realization of this preamble in an ATS, an appropriate TS Operation should be used to adjust the TariffSwitchoverTime to a value being near to 15 minutes after the "current time" (absolute time when the preamble is executed).

[BASIC_TARIFF_METHOD = "Currency"]

O_OSA_AC_CURR

[BASIC_TARIFF_METHOD = "Pulse"]

O_OSA_AC_PULSE

6.5.1.22 O_OSA_FCI

This preamble is used to test the FurnishChargingInformation procedure. An incoming call is initiated which is completed in the individual TPs.

O_null

CP1_1!SetupInd

L1?InitialDP

L1!RequestReportBCSMEvent(1,interrupted,oDisconnect)

O_OSA_FCI

6.5.1.23 O_OSA_RNC

This preamble is used to test the RequestNotificationChargingEvent (RN) procedure. An incoming call is initiated, leaving the SSF in the "Waiting for instructions" state. The call establishment is continued in the individual TPs.

O_null

CP1_1!SetupInd

L1?InitialDP

L1!RequestReportBCSMEvent(1,interrupted,oDisconnect)

O_OSA_RNC

6.5.1.24 Preamble PRE_SCI_1

Objective: Initiate a call from the A-party, on which the SCI operation is going to be applied. The preamble is parameterized with parameter DestinationAddress, which identifies either a called user in a network different from the network of the SSF, or an IN number.

PRE_SCI_1(DestinationAddress:Address)

O_null

CP1_1!SetUpInd(DestinationAddress)

L1?InitialDP(DestinationAddress)

L1!RequestReportBCSMEvent(1,interrupted,oDisconnect)

L1!RequestReportBCSMEvent(2,notifyAndContinue,oDisconnect)

PRE_SCI_1

6.5.1.25 Preamble PRE_SCI_2

Objective: Initiate a call from the A-party to the B-party without applying the SCI operation. Then add a third party (C-party) where the SCI operation is going to be used. The preamble is parameterized with parameter DestinationAddressB, which identify the called B-party (each either a called user in a network different from the network of the SSF, or an IN number). The connection to the C-party is performed within the Test Purposes.

PRE_SCI_2(DestinationAddressB:Address)

O_null

CP1_1!SetUpInd(DestinationAddressB)

L1?InitialDP(DestinationAddressB)

L1!RequestReportBCSMEvent(1,interrupted,oDisconnect)

L1!RequestReportBCSMEvent(2,notifyAndContinue,oDisconnect)

L1!Continue

CP1_2?SetUpReq

CP1_2!SetUpConf

CP1_1?SetUpResp

CP1_1!(FeatureInd)

L1!SplitLeg(1,2)

L1?SplitLegReturnResult

L1!RequestReportBCSMEvent(3,notifyAndContinue,oDisconnect)

PRE_SCI_1

6.5.2 Postamble descriptions

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

6.5.2.1 SigConA_Release postamble

CP1_1!ReleaseInd(Normal clearing)

SigConA_Release

6.5.2.2 SigConA_Release_thenB postamble

CP1_1!ReleaseInd(Normal clearing)

CP1_2?ReleaseReq

SigConA_Release_thenB

6.5.2.3 ReleaseCallA postamble

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

ReleaseCallA

6.5.2.4 ReleaseICA postamble

CP1_2!ReleaseInd(Normal clearing)

ReleaseICA

6.5.2.5 ReleaseCallAB_cause_00 postamble

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

ReleaseCallAB_cause_00

6.5.2.6 ReleaseCallAB_cause_0F postamble

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

ReleaseCallAB_cause_0F

6.5.2.7 SigConB_Release postamble

CP1_2!ReleaseInd

SigConB_Release

6.5.2.8 SigConB_Release_cause_0D postamble

CP1_2!ReleaseInd

SigConB_Release_cause_0D

6.5.2.9 SigConA_Release_thenB_cause10 postamble

CP1_1!ReleaseInd

CP1_2?ReleaseReq

SigConA_Release_thenB_cause10

6.5.2.10 ReleaseCallB postamble

L1!ReleaseCall(allCallSegments)

CP1_2?ReleaseReq

ReleaseCallB

6.5.2.11 ReleaseCallA2 postamble

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

ReleaseCallA2

6.5.2.12 ReleaseCallA3 postamble

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

CP1_3?ReleaseReq

ReleaseCallA3

6.5.2.13 ReleaseToAB postamble

This postamble is used when the SSF initiates release to users A and B. Note that the two ReleaseReq primitives may be received in any order.

CP1_1?ReleaseReq

CP1_2?ReleaseReq

ReleaseToAB

6.5.2.14 ReleaseAandIgnoreStat

This postamble is used in TPs testing StatusReport operations, i.e. StatusReport operations may be pending, when the call is released (independently of whether the StatusReport operations are performed inside or outside the call context). Only one leg (leg 1) exists and is released. Possible received StatusReport components are ignored.

L1!ReleaseCall(allCallSegments)

L1?StatusReport: ignored

L2?StatusReport: ignored

CP1_1?ReleaseReq

L1?StatusReport: ignored

L2?StatusReport: ignored

Idle

6.5.2.15 ReleaseABandIgnoreStat

This postamble is used in TPs testing StatusReport operations, i.e. StatusReport operations may be pending, when the call is released (independently of whether the StatusReport operations were performed inside or outside the call context). Two legs (leg 1 and leg 2) exist and are released. Possible received StatusReport components are ignored. Note that the release signals of the 2 legs may appear in any order.

L1!ReleaseCall(allCallSegments)

L1?StatusReport: ignored

L2?StatusReport: ignored

CP1_1?ReleaseReq

L1?StatusReport: ignored

L2?StatusReport: ignored

CP1_2?ReleaseReq

L1?StatusReport: ignored

L2?StatusReport: ignored

Idle

6.5.2.16 TrigReleaseA postamble

This postamble is used in TPs testing Trigger Management operations, and where a call on only one leg is initiated. No release cause is considered. Possible EventReportBCSM components that could be issued from the SSF (before the ReleaseCall has been received by the SSF) are ignored.

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

Idle

6.5.2.17 TrigReleaseB postamble

This postamble is used in TPs testing Trigger Management operations, and where a call on only one leg is initiated. No release cause is considered. Possible EventReportBCSM components that could be issued from the SSF (before the ReleaseCall has been received by the SSF) are ignored.

L1!ReleaseCall(allCallSegments)

CP1_2?ReleaseReq

Idle

6.5.2.18 TrigReleaseAB postamble

This postamble is used in TPs testing Trigger Management operations, and where a 2-party call is initiated on leg 1 and is answered by the user on leg 2. No release cause is considered. Possible EventReportBCSM components that could be issued from the SSF (before the ReleaseCall has been received by the SSF) are ignored.

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

Idle

6.5.2.19 TrigReleaseA2 postamble

This postamble is used in TPs testing Trigger Management operations, and where two calls are initiated at CP1_1, with two different call references. No release cause is considered. Possible EventReportBCSM components that could be issued from the SSF (before the ReleaseCall has been received by the SSF) are ignored.

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq(call reference 1)

CP1_1?ReleaseReq(call reference 2)

Idle

6.5.2.20 StopGapCld(cldDigits,scf,ctrl,clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "calledAddressValue" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating called address digits, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **one** CallGap invoke component is issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (cldDigits,scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L3!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

[clear=4]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

CP1_3!ReleaseInd

CP1_4?ReleaseReq

StopGapCld

6.5.2.21 StopGapCld2(cldDigits,scf,ctrl,clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "calledAddressValue" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating called address digits, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **two** CallGap invoke components are issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (cldDigits,scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L4!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapCld2

6.5.2.22 StopGapCld3(cldDigits,scf,ctrl,clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "calledAddressValue" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating called address digits, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **three** CallGap invoke components are issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (cldDigits,scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L4!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L4!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L5!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapCld3

6.5.2.23 StopGapCldService(skey,cldDigits,scf,ctrl,clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "calledAddressAndService" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating service key, called address digits, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **one** CallGap invoke component is issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (skey, cldDigits, scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L3!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapCldService

6.5.2.24 StopGapCldService2(skey, cldDigits, scf, ctrl, clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "calledAddressAndService" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating service key, called address digits, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **two** CallGap invoke component are issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (skey, cldDigits, scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L4!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapCldService2

6.5.2.25 StopGapClgService(skey, clgDigits, locNum, scf, ctrl, clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "callingAddressAndService" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating service key, calling address digits, location number, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **one** CallGap invoke component is issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (skey, clgDigits, locNum, scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L3!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapClgService

6.5.2.26 StopGapClgService2(skey, clgDigits, locNum, scf, ctrl, clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "callingAddressAndService" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating service key, calling address digits, location number, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **two** CallGap invoke components are issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (skey, clgDigits, locNum, scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L4!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapClgService2

6.5.2.27 StopGapGos(sKey, scf, ctrl, clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "gapOnService" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating serviceKey, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **one** CallGap invoke component is issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (sKey,scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L3!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

[clear=4]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

CP1_3!ReleaseInd

CP1_4?ReleaseReq

StopGapGos

6.5.2.28 StopGapGos2(sKey, scf, ctrl, clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "gapOnService" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating serviceKey, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **two** CallGap invoke components are issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (sKey,scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L4!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapGos2

6.5.2.29 StopGapGos3(sKey,scf,ctrl,clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "gapOnService" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating serviceKey, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **three** CallGap invoke components are issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (sKey, scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L4!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L4!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L5!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapGos3

6.5.2.30 StopGapGosOS(sKey,scf,ctrl,clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "gapOnService" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating serviceKey, scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when the CallGap invoke component is issued within the context of a dialog where an InitialDP is invoked.

Ln!CallGap invoke(basic or compound gap criteria (sKey,scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L2!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

CP1_4?ReleaseReq

StopGapGosOS

6.5.2.31 StopGapAllIn(scf, ctrl, clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "gapAllInTraffic" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **one** CallGap invoke component is issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L2!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L3!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapAllIn

6.5.2.32 StopGapAllIn2(scf, ctrl, clear) postamble

This postamble is used to stop call gapping in TPs testing call gapping, where the "gapAllInTraffic" alternative is used as "gap criteria". The postamble is parameterized with formal parameters indicating scfID and control type to be used. If parameter scf is OMIT, then the basic gap criteria are used, otherwise the compound gap criteria are used. If parameter ctrl is OMIT, then the controlType is omitted, otherwise the indicated controlType is used. The signalling connections not yet being cleared are released according to parameter "clear".

The CallGap invoke component is sent with the next available DialogId; since the number of dialogs being started in the TP is variable, the PCO is indicated by "Ln".

This preamble is used when **two** CallGap invoke component are issued before the first InitialDP with matching gap criteria is invoked.

Ln!CallGap invoke(basic or compound gap criteria (scf), gapIndicators(duration null), controlType(ctrl), gapTreatment omitted)

Ln!TCAP-BEGIN

[clear=2]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[clear=3]

L3!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

L4!ReleaseCall(allCallSegments)

CP1_3?ReleaseReq

StopGapAllIn2

6.5.2.33 Postamble POST_SCI_RELEASE_B

Objective: The postamble is applicable to configurations where an A-party and a B-party are involved. All connections are released.

POST_SCI_RELEASE_B

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

Idle

6.5.2.34 Postamble POST_SCI_RELEASE_BC

Objective: The postamble is applicable to configurations where an A-party, B-party and C-party are involved. All connections are released.

POST_SCI_RELEASE_BC

L1!ReleaseCall(allCallSegments)

CP1_1?ReleaseReq

CP1_2?ReleaseReq

CP1_3?ReleaseReq

Idle

6.5.2.35 Postamble POST_SCI_EXCEPT_RELEASE_B

Objective: The postamble is applicable to configurations where an A-party and a B-party are involved. It is parameterized with boolean parameter **BReleased**, which defines whether the release of the connection to the B-party has already been performed (TRUE) or not (FALSE). The remaining connections are released.

POST_SCI_EXCEPT_RELEASE_B(BReleased:BOOLEAN)

L1!ReleaseCall(allCallSegments)

[BReleased = TRUE]

CP1_1?ReleaseReq

[BReleased = FALSE]

CP1_1?ReleaseReq

CP1_2?ReleaseReq

Idle

6.5.2.36 Postamble POST_SCI_EXCEPT_RELEASE_BC

Objective: The postamble is applicable to configurations where an A-party, B-party and C-party are involved. It is parameterized with boolean parameter CReleased, which defines whether the release of the connection to the C-party has already been performed (TRUE) or not (FALSE). The remaining connections are released.

POST_SCI_EXCEPT_RELEASE_BC(CReleased:BOOLEAN)

L1!ReleaseCall(allCallSegments)

[CReleased = TRUE]

CP1_1?ReleaseReq

CP1_2?ReleaseReq

[CReleased = FALSE]

CP1_1?ReleaseReq

CP1_2?ReleaseReq

CP1_3?ReleaseReq

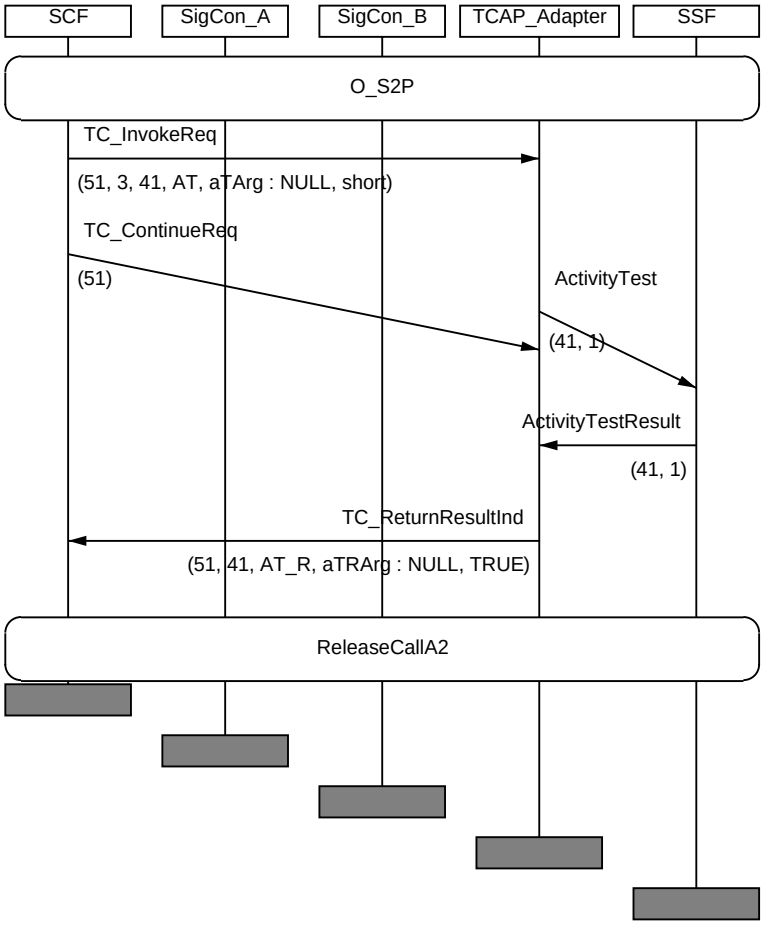
Idle

6.6 Test Purposes (TP) description

6.6.1 ActivityTest (AT) procedure

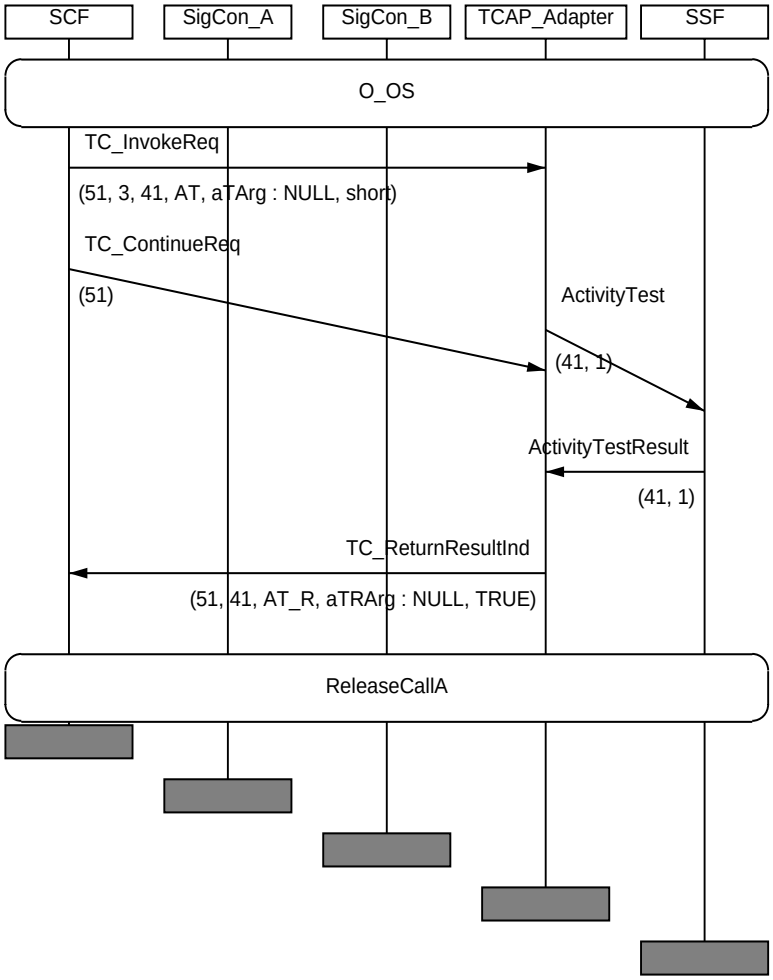
	IN3_A_BASIC_AT_BV_01
Work item no.:	ITEM_BASIC_1
IN2 Ref(tmp)	IN2_A_BASIC_AT_CA_01
Purpose:	Verify that the SSF, being in the "Stable_2_Party" CSCV state and in the "Monitoring" FSM for CS state, sends an **ActivityTest** returnResult component to the SCF, when it receives an **ActivityTest** invoke component from SCF (within the same dialog).
Requirements refs	8.2.1.2, 9.2, 9.3.3, 11.2.1, 11.2.3.1
Selection Cond.	
Preamble:	O_S2P
Test description	L1!ActivityTest invoke L1?ActivityTest returnResult
Pass criteria	L1?ActivityTest returnResult
Postamble:	ReleaseCallA2

MSC IN3_A_BASIC_AT_BV_01



	IN3_A_BASIC_AT_BV_02
Work item no.:	ITEM_BASIC_2
IN2 Ref(tmp)	IN2_A_BASIC_AT_BV_01
Purpose:	Verify that the SSF, being in the "Originating_Setup" CSCV state and in the "Waiting for Instructions" FSM for CS state, sends an **ActivityTest** returnResult component to the SCF, when it receives an **ActivityTest** invoke component from SCF (within the same dialog).
Requirements refs	8.2.1.2, 9.2, 9.3.3, 11.2.1, 11.2.3.1
Preamble:	O_OS
Selection Cond.	
Test description	L1!ActivityTest invoke L1?ActivityTest returnResult
Pass criteria	L1?ActivityTest returnResult
Postamble:	ReleaseCallA

MSC IN3_A_BASIC_AT_BV_02



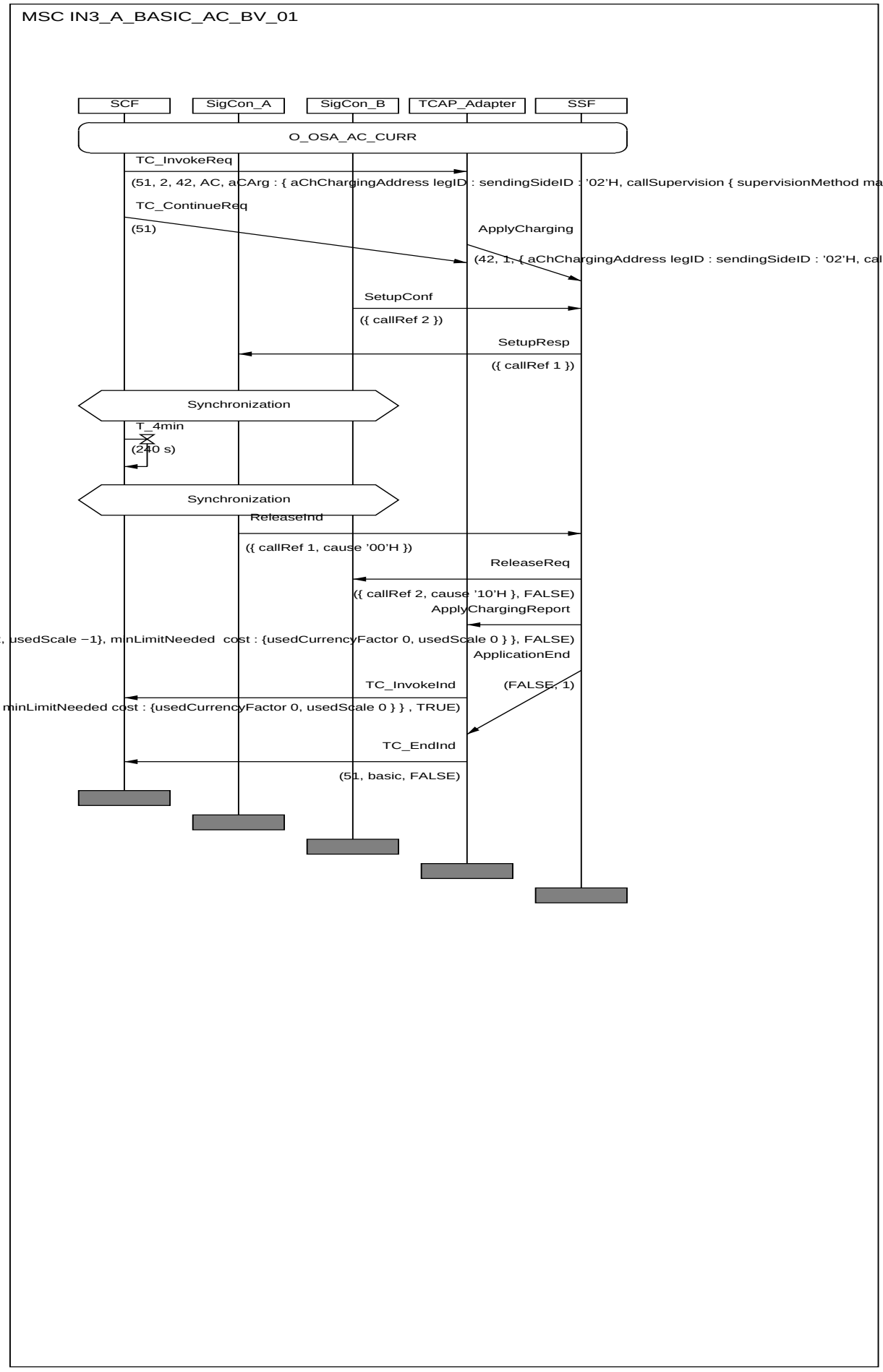
6.6.2 ApplyCharging (AC) procedure

NOTE 1: Only Test Purposes using call supervision are specified.

NOTE 2: Only Test Purposes related to the ApplyCharging procedure can be found in the present document under the FurnishChargingInformation and SendChargingInformation procedures.

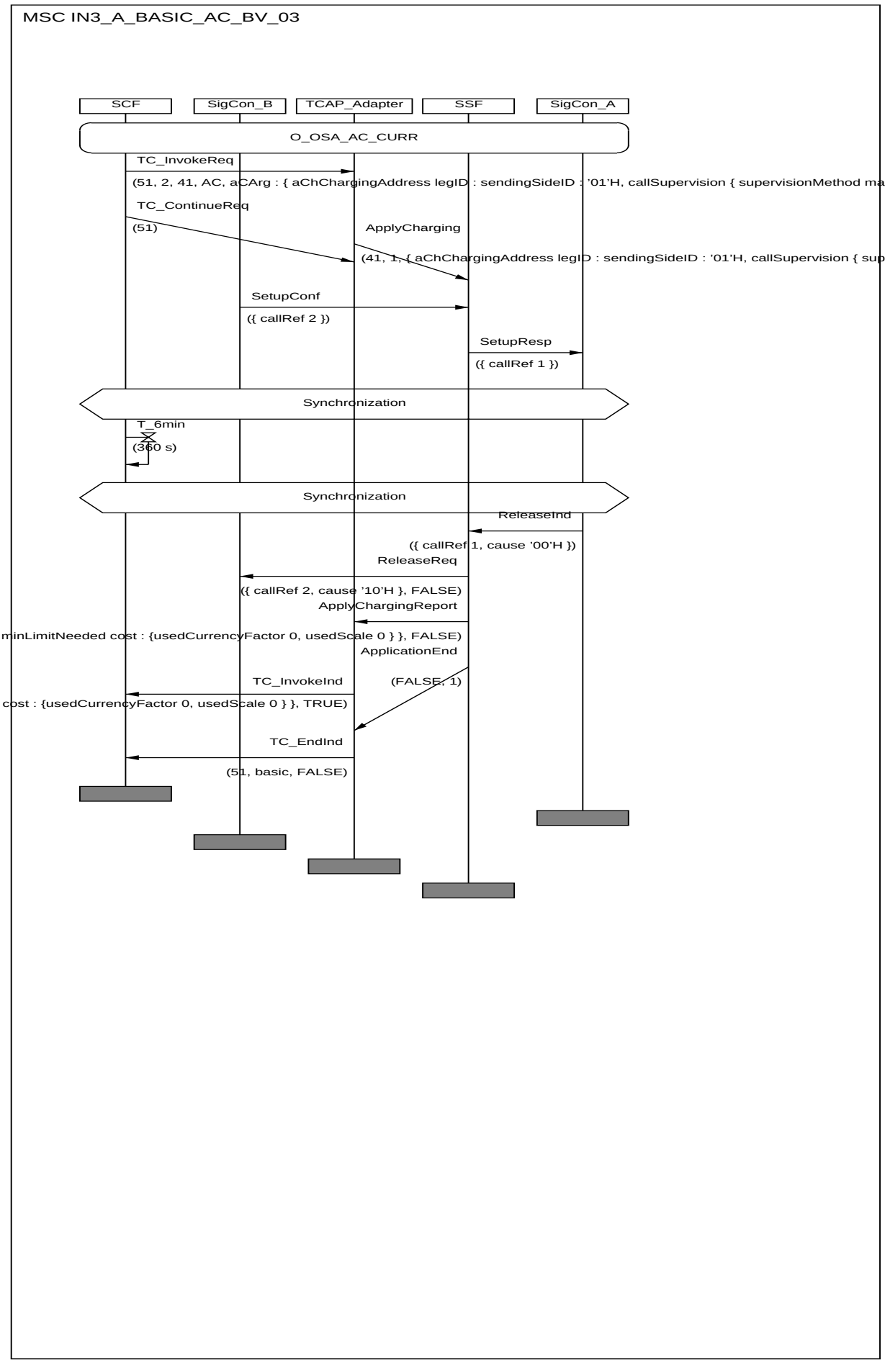
NOTE 3: Test purposes making use of the ApplyCharging procedure in connection with CPH can be found in EN 301 933-2 [5].

	IN3_A_BASIC_AC_BV_01
Work item no.:	ITEM_BASIC_310
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTariffCurrency", and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedCost", where the evaluation of the sub-parameters of "usedCost" results in a value less than the value resulting from AC parameter "maximumTariffCurrency", when the call is released (by a user) earlier than the point in time calculated from "start of charging" and "maximumTariffCurrency", and earlier than the registered tariffSwitchOverTime.
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_CURR
Selection Cond.	CurrencyTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffCurrency (preamble) and sub-parameters of maximumTariffCurrency TR := Time from now to user-initiated release Timing conditions: TR<TM < TS L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1?ReleaseReq(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value UsedCost(TR, currentTariffCurrency)
Postamble:	None

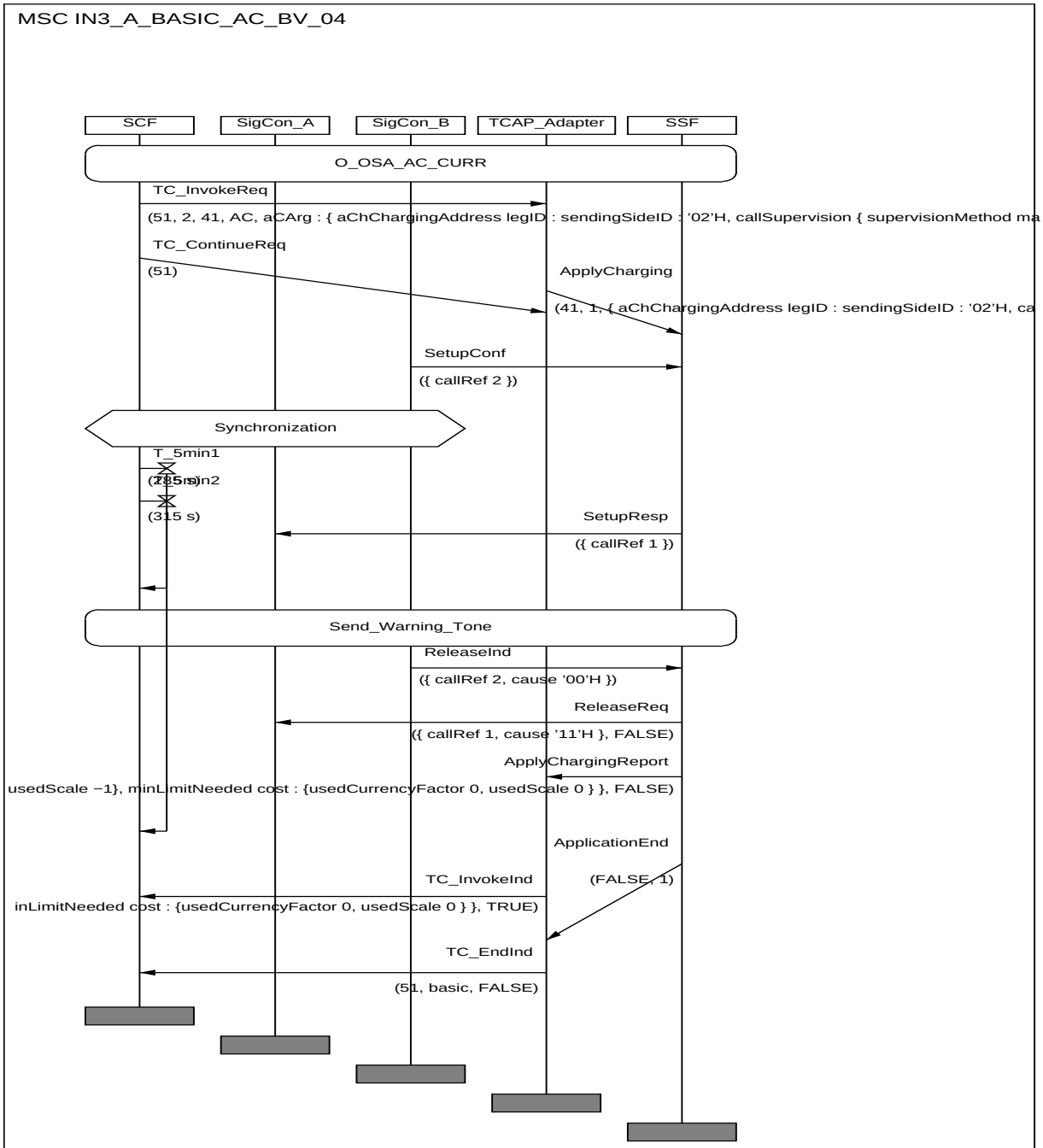

MSC IN3_A_BASIC_AC_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OSA_AC_CURR
TC_InvokeReq
(51, 2, 42, AC, aCArg : { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { supervisionMethod maximumTariffCurrency : { cu
TC_ContinueReq
(51)
ApplyCharging
(42, 1, { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { supervision
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
Synchronization
T_4min
(240 s)
Synchronization
ReleaseInd
({ callRef 1, cause '00'H })
ReleaseReq
({ callRef 2, cause '10'H }, FALSE)
ApplyChargingReport
rencyFactor 72, usedScale −1}, minLimitNeeded cost : {usedCurrencyFactor 0, usedScale 0 } }, FALSE)
ApplicationEnd
TC_InvokeInd
(FALSE, 1)
usedScale −1}, minLimitNeeded cost : {usedCurrencyFactor 0, usedScale 0 } } , TRUE)
TC_EndInd
(51, basic, FALSE)

	IN3_A_BASIC_AC_BV_02
Work item no.:	ITEM_BASIC_311
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTariffCurrency", and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallActive" and supervisionResult "usedCost", where the evaluation of the sub-parameters of "usedCost" results in a value reaching the value resulting from AC parameter "maximumTariffCurrency", when the point in time calculated from "start of charging" and "maximumTariffCurrency" is reached (being earlier than the registered tariffSwitchOverTime).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_CURR
Selection Cond.	CurrencyTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffCurrency (preamble) and sub-parameters of maximumTariffCurrency Timing conditions: TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TM-5 % CP1_1?SetupResp ?Timeout TM L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallActive, supervisionResult = usedCost(TM, currentTariffCurrency))
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallActive, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value UsedCost(TM, currentTariffCurrency)
Postamble:	ReleaseCallA2

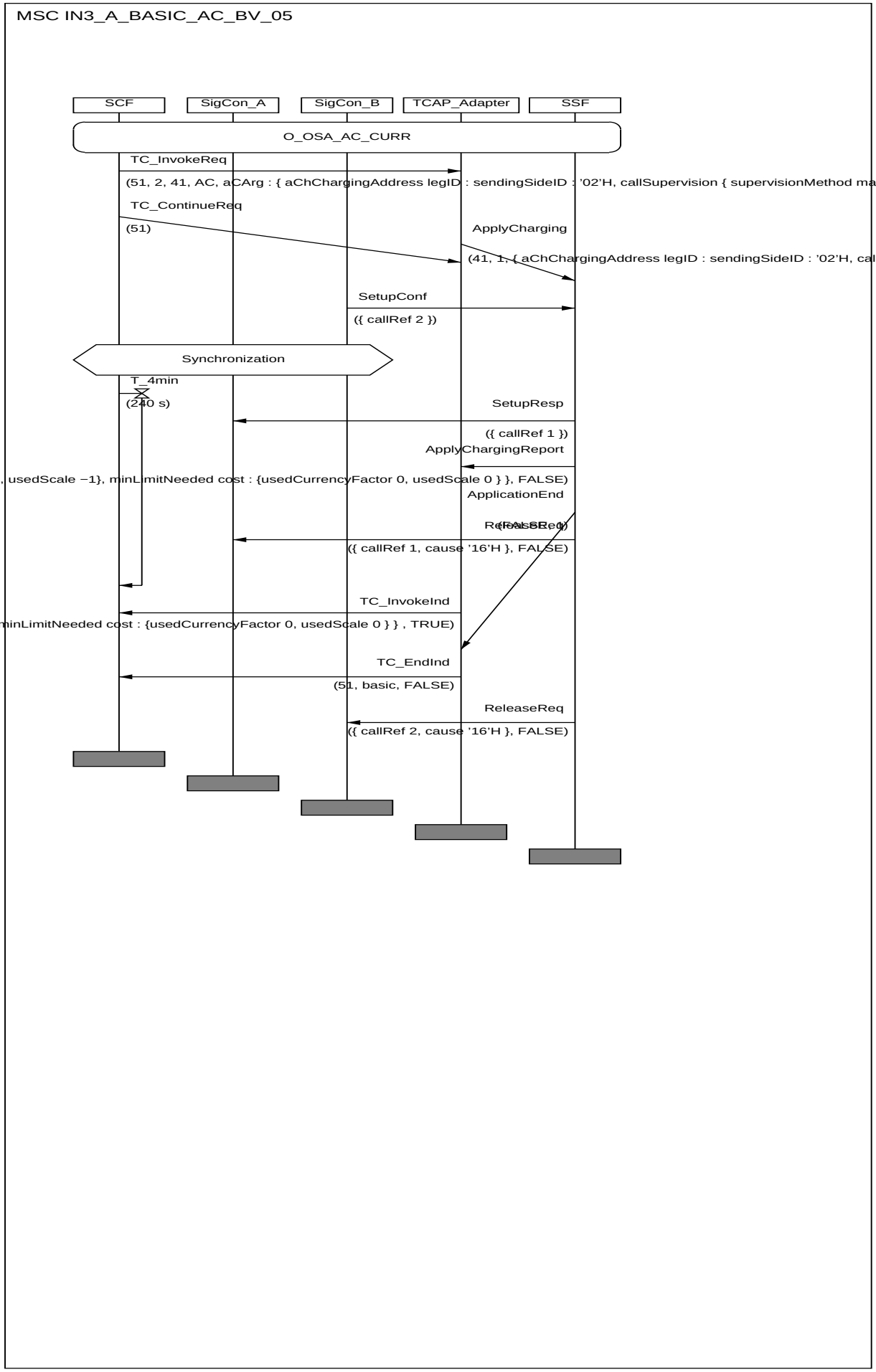
	IN3_A_BASIC_AC_BV_03
Work item no.:	ITEM_BASIC_312
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTariffCurrency", and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedCost", where the evaluation of the sub-parameters of "usedCost" results in a value less than the value resulting from AC parameter "maximumTariffCurrency", when the call is released (by a user) later than the registered "tariffSwitchOverTime", but earlier than the point in time calculated from "start of charging" and "maximumTariffCurrency", taking into account the registered "currentTariffCurrency" (=initial Tariff Currency), "nextTariffCurrency" and "tariffSwitchOverTime".
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_CURR
Selection Cond.	CurrencyTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime (preamble) and sub-parameters of maximumTariffCurrency TR := Time from now to user-initiated release Timing conditions: TS<TR < TM L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1?ReleaseReq(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost within ±5 % of the calculatory value UsedCost(TR, TS, currentTariffCurrency, nextTariffCurrency)) = TS*currentTariffCurrency + (TR-TS)*nextTariffCurrency
Postamble:	None


MSC IN3_A_BASIC_AC_BV_03
SCF
SigCon_B
TCAP_Adapter
SSF
SigCon_A
O_OSA_AC_CURR
TC_InvokeReq
(51, 2, 41, AC, aCArg : { aChChargingAddress legID : sendingSideID : '01'H, callSupervision { supervisionMethod maximumTariffCurrency
TC_ContinueReq
(51)
ApplyCharging
(41, 1, { aChChargingAddress legID : sendingSideID : '01'H, callSupervision { supervisionMethod maxi
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
Synchronization
T_6min
(360 s)
Synchronization
ReleaseInd
({ callRef 1, cause '00'H })
ReleaseReq
({ callRef 2, cause '10'H }, FALSE)
ApplyChargingReport
2, usedScale −1 }, minLimitNeeded cost : {usedCurrencyFactor 0, usedScale 0 } }, FALSE)
ApplicationEnd
TC_InvokeInd
(FALSE, 1)
}, minLimitNeeded cost : {usedCurrencyFactor 0, usedScale 0 } }, TRUE)
TC_EndInd
(51, basic, FALSE)

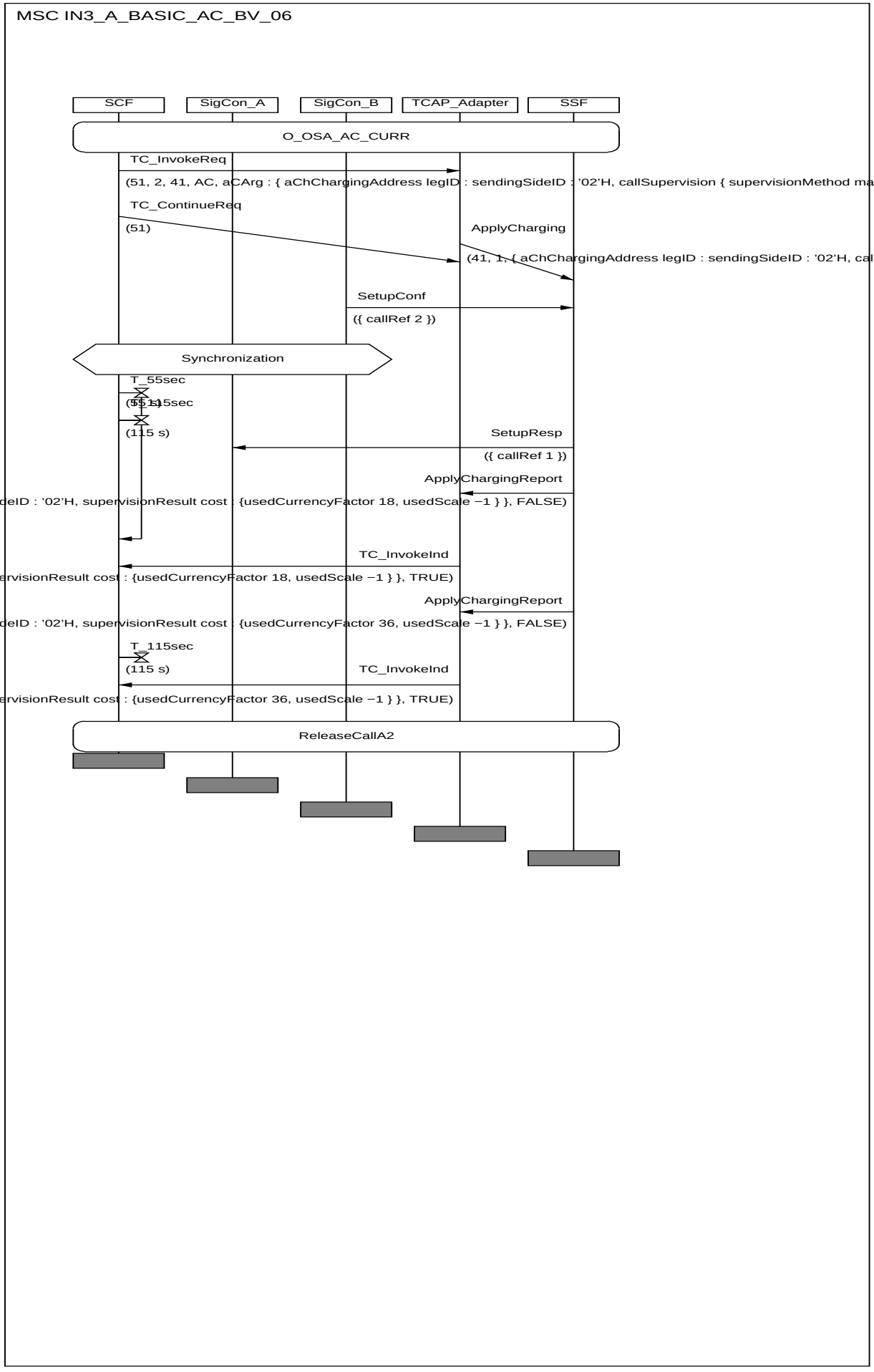
	IN3_A_BASIC_AC_BV_04
Work item no.:	ITEM_BASIC_313
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTariffCurrency", indicating parameter warningBeforeLimitReached with a valid "durationBeforeLimitReached", and having omitted parameters "releaseWhenLimitReached" and "reportConditions", sends a warning tone to the specified warningDirection, when the "durationBeforeLimitReached" expires (before the call is released). Verify also that the SSF sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedCost", where the evaluation of the sub-parameters of "usedCost" results in a value less than the value resulting from AC parameter "maximumTariffCurrency", when the call is released (by a user) earlier than the point in time calculated from "start of charging" and "maximumTariffCurrency", and earlier than the registered tariffSwitchOverTime.
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_CURR
Selection Cond.	CurrencyTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime (preamble) and sub-parameters of maximumTariffCurrency TR := Time from now to user-initiated release TL := durationBeforeLimitReached Timing conditions: TL<TR<TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = (durationBeforeLimitReached,warningDirection), releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR Start TL-5 % CP1_1?SetupResp ?Timeout TL-5 % <Warning tone received in specified direction> ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1?ReleaseReq(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost within ±5 % of the calculatory value UsedCost(TR,currentTariffCurrency))
Postamble:	None


MSC IN3_A_BASIC_AC_BV_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OSA_AC_CURR
TC_InvokeReq
(51, 2, 41, AC, aCArg : { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { supervisionMethod ma
TC_ContinueReq
(51)
ApplyCharging
(41, 1, { aChChargingAddress legID : sendingSideID : '02'H, ca
SetupConf
({ callRef 2 })
Synchronization
T_5min1
(285 s)
T_5min2
(315 s)
SetupResp
({ callRef 1 })
Send_Warning_Tone
ReleaseInd
({ callRef 2, cause '00'H })
ReleaseReq
({ callRef 1, cause '11'H }, FALSE)
ApplyChargingReport
usedScale −1}, minLimitNeeded cost : {usedCurrencyFactor 0, usedScale 0 } }, FALSE)
ApplicationEnd
(FALSE, 1)
TC_InvokeInd
inLimitNeeded cost : {usedCurrencyFactor 0, usedScale 0 } }, TRUE)
TC_EndInd
(51, basic, FALSE)

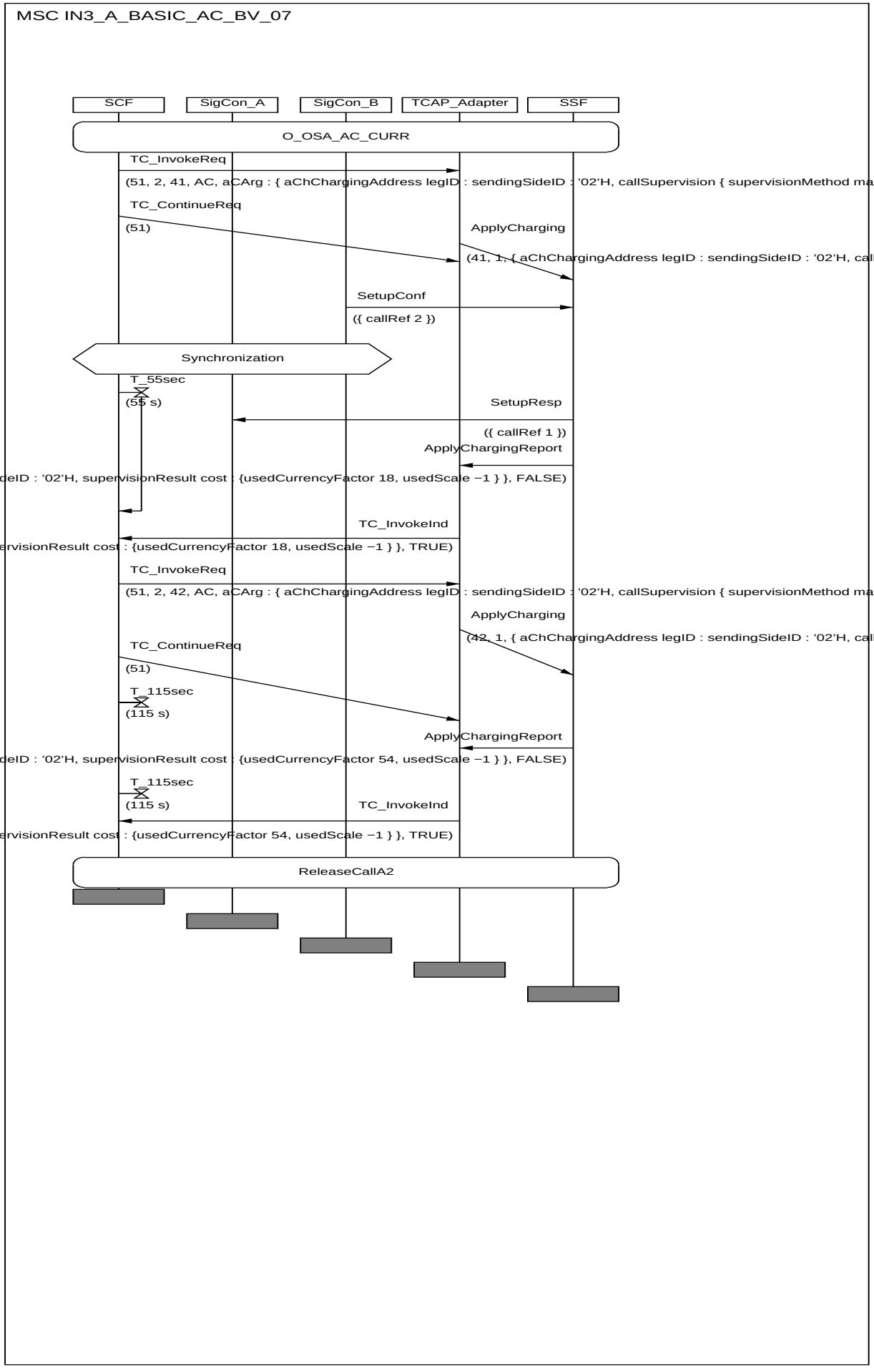
	IN3_A_BASIC_AC_BV_05
Work item no.:	ITEM_BASIC_314
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTariffCurrency", "releaseWhenLimitReached" with a cause different from "normal release", and having omitted parameters "warningBeforeLimitReached", and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedCost", where the evaluation of the sub-parameters of "usedCost" results in a value reaching the value resulting from AC parameter "maximumTariffCurrency", when the call is not released (by a user) before the point in time calculated from "start of charging" and "maximumTariffCurrency" is reached (earlier than the registered tariffSwitchOverTime). Verify also that the call is released by the SSF with the indicated cause (to users A and B).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_CURR
Selection Cond.	CurrencyTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime (preamble) and sub-parameters of maximumTariffCurrency Timing conditions: TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = releaseCause, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TM-5 % CP1_1?SetupResp ?Timeout TM-5 % CP1_1?ReleaseReq(releaseCause) CP1_2?ReleaseReq(releaseCause) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost within ±5 % of the calculatory value UsedCost(TM,currentTariffCurrency)) CP1_1?ReleaseReq(releaseCause) CP1_2?ReleaseReq(releaseCause) NOTE: The 3 events may occur in any order.
Postamble:	None


MSC IN3_A_BASIC_AC_BV_05
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OSA_AC_CURR
TC_InvokeReq
(51, 2, 41, AC, aCArg : { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { supervisionMethod maximumTariffCurrency
TC_ContinueReq
(51)
ApplyCharging
(41, 1, { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { superv
SetupConf
({ callRef 2 })
Synchronization
T_4min
(240 s)
SetupResp
({ callRef 1 })
ApplyChargingReport
dCurrencyFactor 72, usedScale −1}, minLimitNeeded cost : {usedCurrencyFactor 0, usedScale 0 } }, FALSE)
ApplicationEnd
ReleaseReq
({ callRef 1, cause '16'H }, FALSE)
TC_InvokeInd
72, usedScale −1}, minLimitNeeded cost : {usedCurrencyFactor 0, usedScale 0 } } , TRUE)
TC_EndInd
(51, basic, FALSE)
ReleaseReq
({ callRef 2, cause '16'H }, FALSE)

	IN3_A_BASIC_AC_BV_06
Work item no.:	ITEM_BASIC_315
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTariffCurrency" and parameter "reportConditions" with a valid "periodicReportInterval" value, and having omitted parameters "warningBeforeLimitReached" and "releaseWhenLimitReached", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedCost", where the evaluation of the sub-parameters of "usedCost" results in the value calculated from "currentTariffCurrency" and "periodicReportInterval", when the call is not released (by a user) until the point in time defined by the first "periodicReportInterval" value is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumTariffCurrency", and earlier than the registered tariffSwitchOverTime). Verify also that the SSF sends to the SCF a second ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedCost", where the evaluation of the sub-parameters of "usedCost" results in the value calculated from "currentTariffCurrency" and "periodicReportInterval" times two, when the call is not released (by a user) until the point in time defined by the second "periodicReportInterval" value is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumTariffCurrency", and earlier than the registered tariffSwitchOverTime).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_CURR
Selection Cond.	CurrencyTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime (preamble) and sub-parameters of maximumTariffCurrency TI := reportInterval Timing conditions: 2*TI<TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = periodicReportInterval(reportInterval)) CP1_2!SetupConf (This indicates start of charging) Start TI-5 % Start 2*TI-5 % CP1_1?SetupResp ?Timeout TI-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedCost1) ?Timeout 2*TI-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedCost2)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedCost1) L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedCost2) usedCost1 within ±5 % of the calculatory value TI*currentTariffCurrency usedCost2 within ±5 % of the calculatory value 2*TI*currentTariffCurrency
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_AC_BV_06
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OSA_AC_CURR
TC_InvokeReq
(51, 2, 41, AC, aCArg : { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { supervisionMethod maximumTariffCurrency : { c
TC_ContinueReq
(51)
ApplyCharging
(41, 1, { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { supervision
SetupConf
({ callRef 2 })
Synchronization
T_55sec
(55 s)
T_115sec
(115 s)
SetupResp
({ callRef 1 })
ApplyChargingReport
ID : receivingSideID : '02'H, supervisionResult cost : {usedCurrencyFactor 18, usedScale −1 } }, FALSE)
TC_InvokeInd
eID : '02'H, supervisionResult cost : {usedCurrencyFactor 18, usedScale −1 } }, TRUE)
ApplyChargingReport
ID : receivingSideID : '02'H, supervisionResult cost : {usedCurrencyFactor 36, usedScale −1 } }, FALSE)
T_115sec
(115 s)
TC_InvokeInd
eID : '02'H, supervisionResult cost : {usedCurrencyFactor 36, usedScale −1 } }, TRUE)
ReleaseCallA2

	IN3_A_BASIC_AC_BV_07
Work item no.:	ITEM_BASIC_316
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffCurrency, nextTariffCurrency and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component (no. 1) indicating parameter supervisionMethod = "maximumTariffCurrency" and parameter "reportConditions" with a valid "periodicReportInterval" value (no 1), and having omitted parameters "warningBeforeLimitReached" and "releaseWhenLimitReached", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedCost", where the evaluation of the sub-parameters of "usedCost" results in the value calculated from "currentTariffCurrency" and "periodicReportInterval" no. 1, when the call is not released (by a user) until the point in time defined by the first "periodicReportInterval" value no. 1 is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumTariffCurrency", and earlier than the registered tariffSwitchOverTime). Verify also that the SSF, having received from the SCF an ApplyCharging invoke component (no. 2) indicating parameter supervisionMethod = "maximumTariffCurrency" and parameter "reportConditions" with a valid "periodicReportInterval" value (no. 2), "maximumTariffCurrency" and "periodicReportInterval" values being doubled with respect to no. 1, does not send to the SCF an ApplyChargingReport invoke component, when the point in time defined by "currentTariffCurrency" and (("periodicReportInterval" no. 1) times 2) is reached. Verify also that the SSF sends to the SCF a second ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedCost", where the evaluation of the sub-parameters of "usedCost" results in the value calculated from "currentTariffCurrency" and ("periodicReportInterval" no. 1 plus "periodicReportInterval" no. 2), when the call is not released (by a user) until the point in time defined by "periodicReportInterval" no. 1 plus "periodicReportInterval" no. 2 is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumTariffCurrency", and earlier than the registered tariffSwitchOverTime).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_CURR
Selection Cond.	CurrencyTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM1 := maximum Time calculated from currentTariffCurrency and sub-parameters of maximumTariffCurrency1 TM2 := maximum Time calculated from currentTariffCurrency and sub-parameters of maximumTariffCurrency2 TI1 := reportInterval1, TI2 := reportInterval2 Timing conditions: TI1<TM1, TI2 = 2*TI1, TM2 = 2*TM2, TI2<TM2, TI1+TI2<TS L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency(maximumTariffCurrency1), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = periodicReportInterval(reportInterval1)) CP1_2!SetupConf (This indicates start of charging) Start TI1-5 % CP1_1?SetupResp ?Timeout TI1-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedCost1) L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency(maximumTariffCurrency2), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = periodicReportInterval(reportInterval2)) Start TI2-5 % ?Timeout TI2-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedCost2)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedCost1) L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedCost2) usedCost1 within ±5 % of the calculatory value TI1*currentTariffCurrency usedCost2 within ±5 % of the calculatory value (TI1 +TI2)*currentTariffCurrency
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_AC_BV_07
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OSA_AC_CURR
TC_InvokeReq
(51, 2, 41, AC, aCArg : { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { supervisionMethod maximumTariffCurrency
TC_ContinueReq
(51)
ApplyCharging
(41, 1, { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { superv
SetupConf
({ callRef 2 })
Synchronization
T_55sec
(55 s)
SetupResp
({ callRef 1 })
ApplyChargingReport
s legID : receivingSideID : '02'H, supervisionResult cost : {usedCurrencyFactor 18, usedScale −1 } }, FALSE)
TC_InvokeInd
gSideID : '02'H, supervisionResult cost : {usedCurrencyFactor 18, usedScale −1 } }, TRUE)
TC_InvokeReq
(51, 2, 42, AC, aCArg : { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { supervisionMethod maximumTariffCurrency
ApplyCharging
(42, 1, { aChChargingAddress legID : sendingSideID : '02'H, callSupervision { superv
TC_ContinueReq
(51)
T_115sec
(115 s)
ApplyChargingReport
s legID : receivingSideID : '02'H, supervisionResult cost : {usedCurrencyFactor 54, usedScale −1 } }, FALSE)
T_115sec
(115 s)
TC_InvokeInd
gSideID : '02'H, supervisionResult cost : {usedCurrencyFactor 54, usedScale −1 } }, TRUE)
ReleaseCallA2

	IN3_A_BASIC_AC_BV_08
Work item no.:	ITEM_BASIC_317
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffPulse, nextTariffPulse and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumPulseUnits", and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedPulses", where "usedPulses" has a value less than the value of AC parameter "maximumPulseUnits", when the call is released (by a user) earlier than the point in time calculated from "start of charging" and "maximumPulseUnits", and earlier than the registered tariffSwitchOverTime.
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_PULSE
Selection Cond.	PulseTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffPulse (preamble) and sub-parameters of maximumTariffPulse TR := Time from now to user-initiated release Timing conditions: TR<TM < TS L1!ApplyCharging invoke(supervisionMethod = maximumPulseUnits, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1?ReleaseReq(Normal clearing L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value UsedPulses (TR,currentTariffPulse)
Postamble:	None

	IN3_A_BASIC_AC_BV_09
Work item no.:	ITEM_BASIC_318
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffPulse, nextTariffPulse and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumPulseUnits", and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallActive" and supervisionResult "usedPulses", where "usedPulses" has a value reaching the value of AC parameter "maximumPulseUnits", when the point in time calculated from "start of charging" and "maximumPulseUnits" is reached (being earlier than the registered tariffSwitchOverTime).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_PULSE
Selection Cond.	PulseTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffPulse (preamble) and sub-parameters of maximumPulseUnits Timing conditions: TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumPulseUnits, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TM-5 % CP1_1?SetupResp ?Timeout TM L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallActive, supervisionResult = usedPulses (TM, currentTariffPulse))
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallActive, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value UsedPulses (TM, currentTariffPulse)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_AC_BV_10
Work item no.:	ITEM_BASIC_319
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffPulse, nextTariffPulse and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumPulseUnits", and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedPulses", where "usedPulses" has a value less than the value of AC parameter "maximumPulseUnits", when the call is released (by a user) later than the registered "tariffSwitchOverTime", but earlier than the point in time calculated from "start of charging" and "maximumPulseUnits", taking into account the registered "currentTariffPulse" (=initial Tariff Pulse), "nextTariffPulse" and "tariffSwitchOverTime".
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_PULSE
Selection Cond.	PulseTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffPulse, nextTariffPulse and tariffSwitchOverTime (preamble) and sub-parameters of maximumPulseUnits TR := Time from now to user-initiated release Timing conditions: TS<TR < TM L1!ApplyCharging invoke(supervisionMethod = maximumPulseUnits, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1?ReleaseReq(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulsest within ±5 % of the calculatory value UsedPulsest(TR, TS, currentTariffPulse, nextTariffPulse)) = TS*currentTariffPulse + (TR-TS)*nextTariffPulse
Postamble:	None

	IN3_A_BASIC_AC_BV_11
Work item no.:	ITEM_BASIC_320
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffPulse, nextTariffPulse and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumPulseUnits", indicating parameter warningBeforeLimitReached with a valid "durationBeforeLimitReached", and having omitted parameters "releaseWhenLimitReached" and "reportConditions", sends a warning tone to the specified warningDirection, when the "durationBeforeLimitReached" expires (before the call is released by a user). Verify also that the SSF sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedPulses", where "usedPulses" has a value less than the value of parameter "maximumPulseUnits", when the call is released (by a user) earlier than the point in time calculated from "start of charging" and "maximumPulseUnits", and earlier than the registered tariffSwitchOverTime.
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_PULSE
Selection Cond.	PulseTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffPulse, nextTariffPulse and tariffSwitchOverTime (preamble) and sub-parameters of maximumPulseUnits TR := Time from now to user-initiated release TL := durationBeforeLimitReached Timing conditions: TL<TR<TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumPulseUnits, warningBeforeLimitReached = (durationBeforeLimitReached,warningDirection), releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR Start TL-5 % CP1_1?SetupResp ?Timeout TL-5 % <Warning tone received in specified direction> ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses within ±5 % of the calculatory value UsedPulses (TR,currentTariffPulse))
Postamble:	None

	IN3_A_BASIC_AC_BV_12
Work item no.:	ITEM_BASIC_321
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffPulse, nextTariffPulse and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumPulseUnits", "releaseWhenLimitReached" with a cause different from "normal release", and having omitted parameters "warningBeforeLimitReached", and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedPulses", where the evaluation of "usedPulses" has a value reaching the value of AC parameter "maximumPulseUnits", when the call is not released (by a user) until the point in time calculated from "start of charging" and "maximumPulseUnits" is reached (earlier than the registered tariffSwitchOverTime). Verify also that the call is released by the SSF with the indicated cause (to users A and B).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_PULSE
Selection Cond.	PulseTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffPulse, nextTariffPulse and tariffSwitchOverTime (preamble) and sub-parameters of maximumPulseUnits Timing conditions: TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumPulseUnits, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = releaseCause, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TM-5 % CP1_1?SetupResp ?Timeout TM-5 % CP1_1?ReleaseReq(releaseCause) CP1_2?ReleaseReq(releaseCause) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses within ±5 % of the calculatory value UsedPulses (TM, currentTariffPulse)) CP1_1?ReleaseReq(releaseCause) CP1_2?ReleaseReq(releaseCause) NOTE: The 3 events may occur in any order.
Postamble:	None

	IN3_A_BASIC_AC_BV_13
Work item no.:	ITEM_BASIC_322
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffPulse, nextTariffPulse and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumPulseUnits" and parameter "reportConditions" with a valid "periodicReportInterval" value, and having omitted parameters "warningBeforeLimitReached" and "releaseWhenLimitReached", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedPulses", where the evaluation of "usedPulses" results in the value calculated from "currentTariffPulse" and "periodicReportInterval", when the call is not released (by a user) until the point in time defined by the first "periodicReportInterval" value is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumPulseUnits", and earlier than the registered tariffSwitchOverTime). Verify also that the SSF sends to the SCF a second ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedPulses", where the evaluation of "usedPulses" results in the value calculated from "currentTariffPulse" and "periodicReportInterval" times two, when the call is not released (by a user) until the point in time defined by the second "periodicReportInterval" value is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumPulseUnits", and earlier than the registered tariffSwitchOverTime).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_PULSE
Selection Cond.	PulseTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximum Time calculated from currentTariffPulse, nextTariffPulse and tariffSwitchOverTime (preamble) and sub-parameters of maximumPulseUnits TI := reportInterval Timing conditions: 2*TI<TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumPulseUnits, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = periodicReportInterval(reportInterval)) CP1_2!SetupConf (This indicates start of charging) Start TI-5 % Start 2*TI-5 % CP1_1?SetupResp ?Timeout TI-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedPulses1) ?Timeout 2*TI-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedPulses2)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedPulses1) L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedPulses2) usedPulses1 within ±5 % of the calculatory value TI*currentTariffPulse usedPulses2 within ±5 % of the calculatory value 2*TI*currentTariffPulse
Postamble:	ReleaseCallA2

	IN3_A_BASIC_AC_BV_14
Work item no.:	ITEM_BASIC_323
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered currentTariffPulse, nextTariffPulse and tariffSwitchOverTime, and having received from the SCF an ApplyCharging invoke component (no. 1) indicating parameter supervisionMethod = "maximumPulseUnits" and parameter "reportConditions" with a valid "periodicReportInterval" value (no. 1), and having omitted parameters "warningBeforeLimitReached" and "releaseWhenLimitReached", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedPulses", where "usedPulses" has the value calculated from "currentTariffPulse" and "periodicReportInterval" no. 1, when the call is not released (by a user) until the point in time defined by the first "periodicReportInterval" value no. 1 is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumPulseUnits", and earlier than the registered tariffSwitchOverTime). Verify also that the SSF, having received from the SCF an ApplyCharging invoke component (no. 2) indicating parameter supervisionMethod = "maximumPulseUnits" and parameter "reportConditions" with a valid "periodicReportInterval" value (no. 2), "maximumPulseUnits" and "periodicReportInterval" values being doubled with respect to no. 1, does not send to the SCF an ApplyChargingReport invoke component, when the point in time defined by "currentTariffPulse" and (("periodicReportInterval" no. 1) times 2) is reached. Verify also that the SSF sends to the SCF a second ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedPulses", where "usedPulses" has the value calculated from "currentTariffPulse" and ("periodicReportInterval" no. 1 plus "periodicReportInterval" no. 2), when the call is not released (by a user) until the point in time defined by "periodicReportInterval" no. 1 plus "periodicReportInterval" no. 2 is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumPulseUnits", and earlier than the registered tariffSwitchOverTime).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC_PULSE
Selection Cond.	PulseTariff
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM1 := maximum Time calculated from currentTariffPulse and sub-parameters of maximumPulseUnits1 TM2 := maximum Time calculated from currentTariffPulse and sub-parameters of maximumPulseUnits2 TI1 := reportInterval1 TI2 := reportInterval2 Timing conditions: TI1<TM1, TI2 = 2*TI1, TM2 = 2*TM2, TI2<TM2, TI1+TI2<TS L1!ApplyCharging invoke(supervisionMethod = maximumPulseUnits(maximumPulseUnits1), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = periodicReportInterval(reportInterval1)) CP1_2!SetupConf (This indicates start of charging) Start TI1-5 % CP1_1?SetupResp ?Timeout TI1-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedPulses1) L1!ApplyCharging invoke(supervisionMethod = maximumPulseUnits(maximumPulseUnits2), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = periodicReportInterval(reportInterval2)) Start TI2-5 % ?Timeout TI2-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedPulses2)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedPulses1) L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedPulses2) usedPulses1 within ±5 % of the calculatory value TI1*currentTariffPulse usedPulses2 within ±5 % of the calculatory value (TI1 +TI2)*currentTariffPulse
Postamble:	ReleaseCallA2

	IN3_A_BASIC_AC_BV_15
Work item no.:	ITEM_BASIC_324
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTimeDuration" (switchOverTime not specified), and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedTime", where "usedDuration" has a value less than the value of AC parameter "maximumTimeDuration", when the call is released (by a user) earlier than the point in time calculated from "start of charging" and "maximumTimeDuration".
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC
Selection Cond.	
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximumDuration TR := Time from now to user-initiated release Timing conditions: TR<TM < TS L1!ApplyCharging invoke(supervisionMethod = maximumTimeDuration(maximumDuration, switchOverTime=OMIT), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime) usedTime is within ±5 % of calculatory value TR
Postamble:	None

	IN3_A_BASIC_AC_BV_16
Work item no.:	ITEM_BASIC_325
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTimeDuration" (switchOverTime not specified), and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallActive" and supervisionResult "usedTime", where "usedDuration" has a value reaching the value of AC parameter "maximumTimeDuration", when the point in time calculated from "start of charging" and "maximumTimeDuration" is reached.
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC
Selection Cond.	
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximumDuration Timing conditions: TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumTimeDuration(maximumDuration, switchOverTime=OMIT), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TM-5 % CP1_1?SetupResp ?Timeout TM L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallActive, supervisionResult = usedTime)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallActive, supervisionResult = usedTime) usedTime is within ±5 % of calculatory value TM
Postamble:	ReleaseCallA2

	IN3_A_BASIC_AC_BV_17
Work item no.:	ITEM_BASIC_326
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered "tariffSwitchOverTime", and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTimeDuration" (switchOverTime not specified), and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedTime", where "usedDuration" has a value less than the value of AC parameter "maximumTimeDuration" and the value of "usedDurationAfterSwitchOverTime" is omitted, when the call is released (by a user) later than the registered "tariffSwitchOverTime", but earlier than the point in time calculated from "start of charging" and "maximumTimeDuration".
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC
Selection Cond.	
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximumDuration TR := Time from now to user-initiated release Timing conditions: TS<TR < TM L1!ApplyCharging invoke(supervisionMethod = maximumTimeDuration(maximumDuration, switchOverTime=OMIT), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime) usedTime within ±5 % of the calculatory value TR, usedDurationAfterSwitchOverTime is omitted in ACR
Postamble:	None

	IN3_A_BASIC_AC_BV_18
Work item no.:	ITEM_BASIC_327
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having registered "tariffSwitchOverTime", and having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTimeDuration" (switchOverTime being specified and having the value corresponding to "tariffSwitchOverTime"), and having omitted parameters "warningBeforeLimitReached", "releaseWhenLimitReached" and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedTime", where "usedDuration" has a value less than the value of AC parameter "maximumTimeDuration" and the value of "usedDurationAfterSwitchOverTime" is according to the difference of release time and "tariffSwitchOverTime", when the call is released (by a user) later than the registered "tariffSwitchOverTime", but earlier than the point in time calculated from "start of charging" and "maximumTimeDuration".
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC
Selection Cond.	
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximumDuration TR := Time from now to user-initiated release Timing conditions: TS<TR < TM L1!ApplyCharging invoke(supervisionMethod = maximumTimeDuration(maximumDuration, switchOverTime=TS), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime) usedTime within ±5 % of the calculatory value TR, usedDurationAfterSwitchOverTime within ±5 % of (TR-TS)
Postamble:	None

	IN3_A_BASIC_AC_BV_19
Work item no.:	ITEM_BASIC_328
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTimeDuration" (switchOverTime not specified), indicating parameter warningBeforeLimitReached with a valid "durationBeforeLimitReached", and having omitted parameters "releaseWhenLimitReached" and "reportConditions", sends a warning tone to the specified warningDirection, when the "durationBeforeLimitReached" expires (before the call is released by a user). Verify also that the SSF sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedTime", where "usedDuration" has a value less than the value of AC parameter "maximumTimeDuration", when the call is released (by a user) earlier than the point in time calculated from "start of charging" and "maximumTimeDuration".
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC
Selection Cond.	
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximumDuration TR := Time from now to user-initiated release TL := durationBeforeLimitReached Timing conditions: TL<TR<TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumTimeDuration(maximumDuration, switchOverTime=OMIT), warningBeforeLimitReached = (durationBeforeLimitReached,warningDirection), releaseWhenLimitReached = OMIT, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TR Start TL-5 % CP1_1?SetupResp ?Timeout TL-5 % <Warning tone received in specified direction> ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime) usedTime within ±5 % of the calculatory value TR
Postamble:	None

	IN3_A_BASIC_AC_BV_20
Work item no.:	ITEM_BASIC_329
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTimeDuration" (switchOverTime not specified), "releaseWhenLimitReached" with a cause different from "normal release", and having omitted parameters "warningBeforeLimitReached", and "reportConditions", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "finalReportCallReleased" and supervisionResult "usedTime", where "usedDuration" has a value reaching the value of AC parameter "maximumTimeDuration", when the call is not released (by a user) until the point in time calculated from "start of charging" and "maximumTimeDuration" is reached. Verify also that the call is released by the SSF with the indicated cause (to users A and B).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC
Selection Cond.	
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximumDuration Timing conditions: TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumTimeDuration(maximumDuration, switchOverTime=OMIT), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = releaseCause, reportCondition = OMIT) CP1_2!SetupConf (This indicates start of charging) Start TM-5 % CP1_1?SetupResp ?Timeout TM-5 % CP1_1?ReleaseReq(releaseCause) CP1_2?ReleaseReq(releaseCause) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedTime) usedTime within ±5 % of the calculatory value TM CP1_1?ReleaseReq(releaseCause) CP1_2?ReleaseReq(releaseCause) NOTE: The 3 events may occur in any order.
Postamble:	None

	IN3_A_BASIC_AC_BV_21
Work item no.:	ITEM_BASIC_330
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF an ApplyCharging invoke component indicating parameter supervisionMethod = "maximumTimeDuration" (switchOverTime not specified) and parameter "reportConditions" with a valid "periodicReportInterval" value, and having omitted parameters "warningBeforeLimitReached" and "releaseWhenLimitReached", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedTime", where "usedDuration" has the value of "periodicReportInterval", when the call is not released (by a user) until the point in time defined by the first "periodicReportInterval" value is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumTimeDuration"). Verify also that the SSF sends to the SCF a second ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedTime", where "usedDuration" has the value of "periodicReportInterval" times two, when the call is not released (by a user) until the point in time defined by the second "periodicReportInterval" value is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumTimeDuration").
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC
Selection Cond.	
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM := maximumDuration TI := reportInterval Timing conditions: 2*TI<TM<TS L1!ApplyCharging invoke(supervisionMethod = maximumTimeDuration(maximumDuration, switchOverTime=OMIT), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = periodicReportInterval(reportInterval)) CP1_2!SetupConf (This indicates start of charging) Start TI-5 % Start 2*TI-5 % CP1_1?SetupResp ?Timeout TI-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedTime1) ?Timeout 2*TI-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedTime2)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedTime1) L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedTime2) usedTime1 within ±5 % of the calculatory value TI usedTime2 within ±5 % of the calculatory value 2*TI
Postamble:	ReleaseCallA2

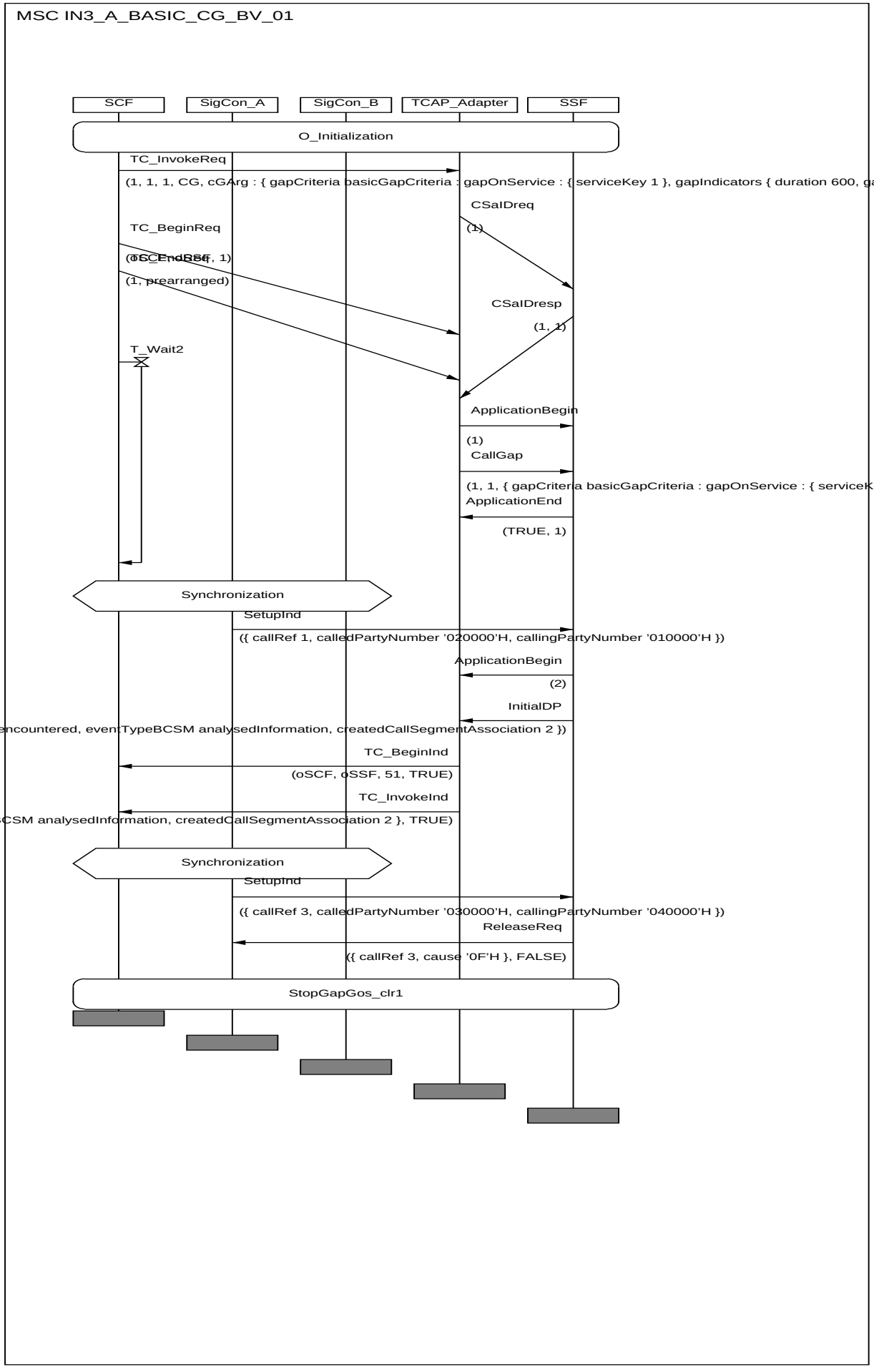
	IN3_A_BASIC_AC_BV_22
Work item no.:	ITEM_BASIC_331
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF an ApplyCharging invoke component (no. 1) indicating parameter supervisionMethod = "maximumTimeDuration" (switchOverTime not specified) and parameter "reportConditions" with a valid "periodicReportInterval" value (no. 1), and having omitted parameters "warningBeforeLimitReached" and "releaseWhenLimitReached", sends to the SCF an ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedTime", where "usedDuration" has the value of "periodicReportInterval" no. 1, when the call is not released (by a user) until the point in time defined by the first "periodicReportInterval" value no. 1 is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumTimeDuration"). Verify also that the SSF, having received from the SCF an ApplyCharging invoke component (no. 2) indicating parameter supervisionMethod = "maximumTimeDuration" and parameter "reportConditions" with a valid "periodicReportInterval" value (no. 2), "maximumTimeDuration" and "periodicReportInterval" values being doubled with respect to no. 1, does **not** send to the SCF an ApplyChargingReport invoke component, when the point in time defined by (("periodicReportInterval" no. 1) times 2) is reached. Verify also that the SSF sends to the SCF a second ApplyChargingReport invoke component indicating reportConditionInformation "intermediateReport" and supervisionResult "usedTime", where "usedDuration" has the value of ("periodicReportInterval" no. 1 plus "periodicReportInterval" no. 2), when the call is not released (by a user) until the point in time defined by "periodicReportInterval" no. 1 plus "periodicReportInterval" no. 2 is reached (and this is earlier than the point in time calculated from "start of charging" and "maximumTimeDuration").
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Preamble:	O_OSA_AC
Selection Cond.	
Test description	TS := Time from now to tariffSwitchOverTime (preamble) TM1 := maximumDuration1 TM1 := maximumDuration2 TI1 := reportInterval1 TI2 := reportInterval2 Timing conditions: TI1<TM1, TI2 = 2*TI1, TM2 = 2*TM2, TI2<TM2, TI1+TI2<TS L1!ApplyCharging invoke(supervisionMethod = maximumTimeDuration(maximumDuration1,switchOverTime=OMIT), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = periodicReportInterval(reportInterval1)) CP1_2!SetupConf (This indicates start of charging) Start TI1-5 % CP1_1?SetupResp ?Timeout TI1-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedTime1) L1!ApplyCharging invoke(supervisionMethod = maximumTimeDuration(maximumDuration2,switchOverTime=OMIT), warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = periodicReportInterval(reportInterval2)) Start TI2-5 % ?Timeout TI2-5 % L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedTime2)
Pass criteria	L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedTime1) L1?ApplyChargingReport invoke(reportConditionInformation = intermediateReport, supervisionResult = usedTime2) usedTime1 within ±5 % of the calculatory value TI1 usedTime2 within ±5 % of the calculatory value TI1 +TI2
Postamble:	ReleaseCallA2

	IN3_A_BASIC_AC_BI_01
Work item no.:	ITEM_BASIC_11
IN2 Ref(tmp)	IN2_A_BASIC_AC_BI_02
Purpose:	Verify that the SSF sends an **ApplyCharging** returnError component with errorCode "unknownLegID", after having received an **ApplyCharging** invoke component with aChChargingAddress = leg 3 (not existing leg).
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Selection Cond.	
Preamble:	O_OSA_AC
Test description	L1!ApplyCharging invoke(aChChargingAddress = leg 3, supervisionMethod = maximumTimeDuration, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT)
Pass criteria	L1?ApplyCharging returnError(unknownLegID)
Postamble:	ReleaseCallA

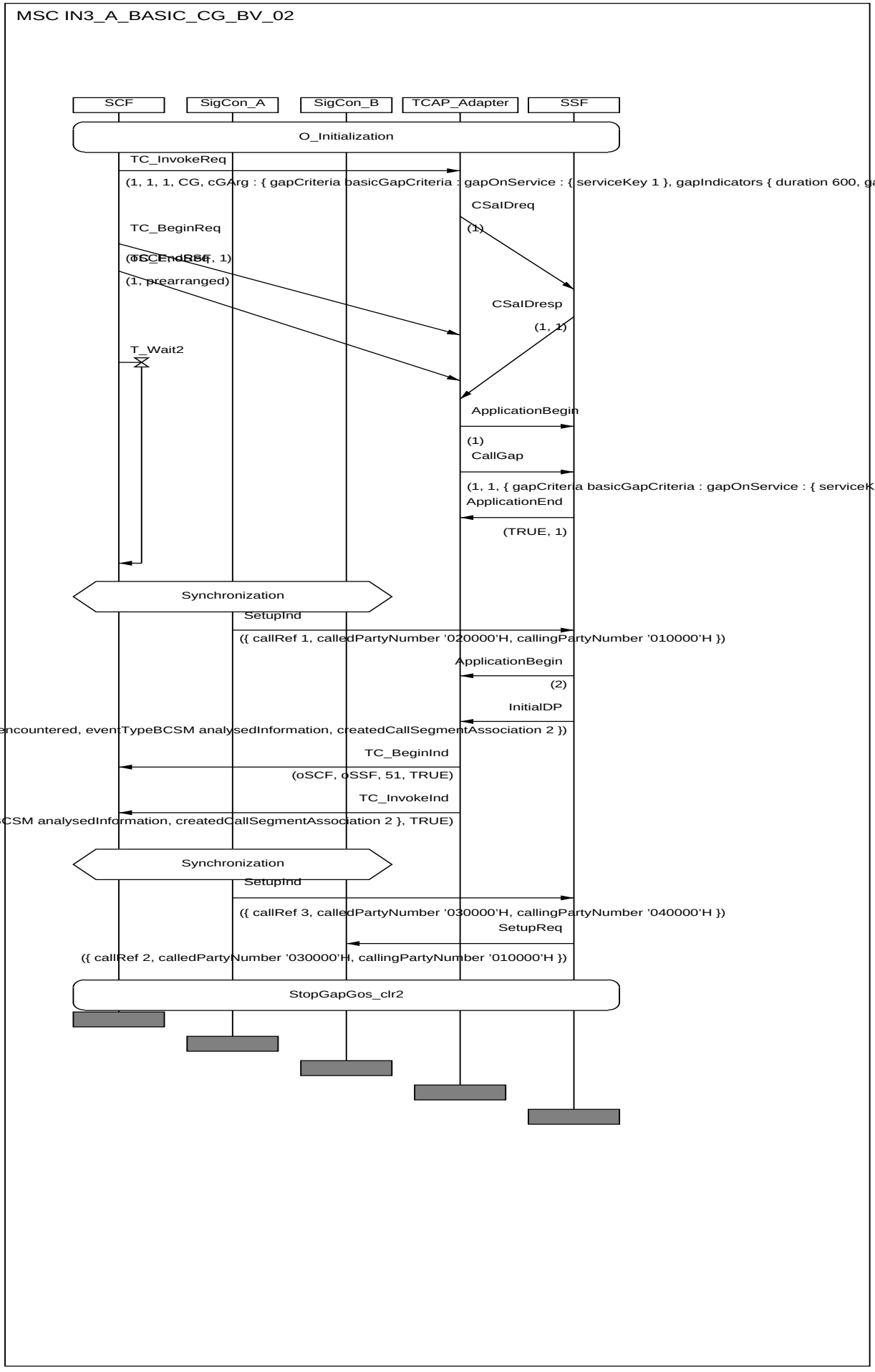
	IN3_A_BASIC_AC_BO_01
Work item no.:	ITEM_BASIC_12
IN2 Ref(tmp)	IN2_A_BASIC_AC_BO_01
Purpose:	Verify that the SSF sends an **ApplyCharging** returnError component with errorCode "taskRefused" or "unexpectedComponentSequence", after having received an **ApplyCharging** invoke component in state "Idle".
Requirements refs	11.3, 11.4, 11.12.1, 12.24, 12.30
Selection Cond.	
Preamble:	None
Test description	L1!ApplyCharging invoke(aChChargingAddress = leg 2, supervisionMethod = maximumTimeDuration, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!TCAP-BEGIN
Pass criteria	L1?ApplyCharging returnError("taskRefused" or "unexpectedComponentSequence")
Postamble:	None

6.6.3 CallGap (CG) procedure

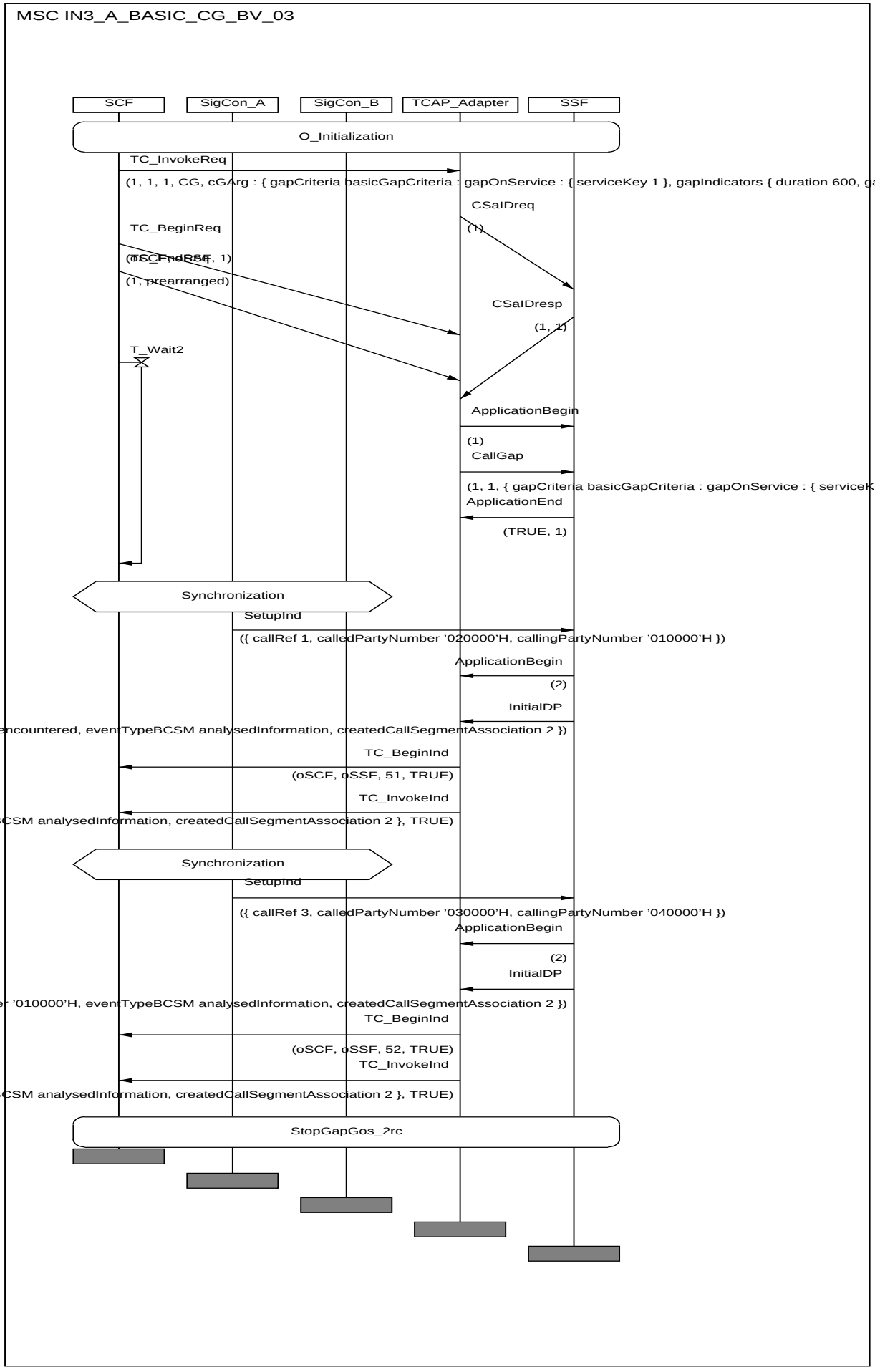
	IN3_A_BASIC_CG_BV_01
Work item no.:	ITEM_BASIC_306
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains only mandatory parameters a) **gapCriteria**, and b) **gapIndicators**, with values: a) **gapOnService** with any valid value for serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds); c) gapTreatment omitted. Check that an incoming call (during an active gap interval) is released (default gap treatment) and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	DefaultOmitGapRelease
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: gapOnService with any valid value for serviceKey, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey, cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_3?ReleaseReq(any cause) No further InitialDP received
Pass criteria	CP1_3?ReleaseReq(any cause) No further InitialDP received
Postamble:	StopGapGos(serviceKey, scf=OMIT, ctrl=OMIT, clear=2)


MSC IN3_A_BASIC_CG_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
TC_InvokeReq
(1, 1, 1, CG, cGArg : { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators { duration 600, gapInterval 30000 }, c
CSalDreq
(1)
TC_BeginReq
(oSCF, 1)
(1, prearranged)
CSalDresp
(1, 1)
T_Wait2
ApplicationBegin
(1)
CallGap
(1, 1, { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators
ApplicationEnd
(TRUE, 1)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
untered manualCGencountered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ntered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
Synchronization
SetupInd
({ callRef 3, calledPartyNumber '030000'H, callingPartyNumber '040000'H })
ReleaseReq
({ callRef 3, cause '0F'H }, FALSE)
StopGapGos_clr1

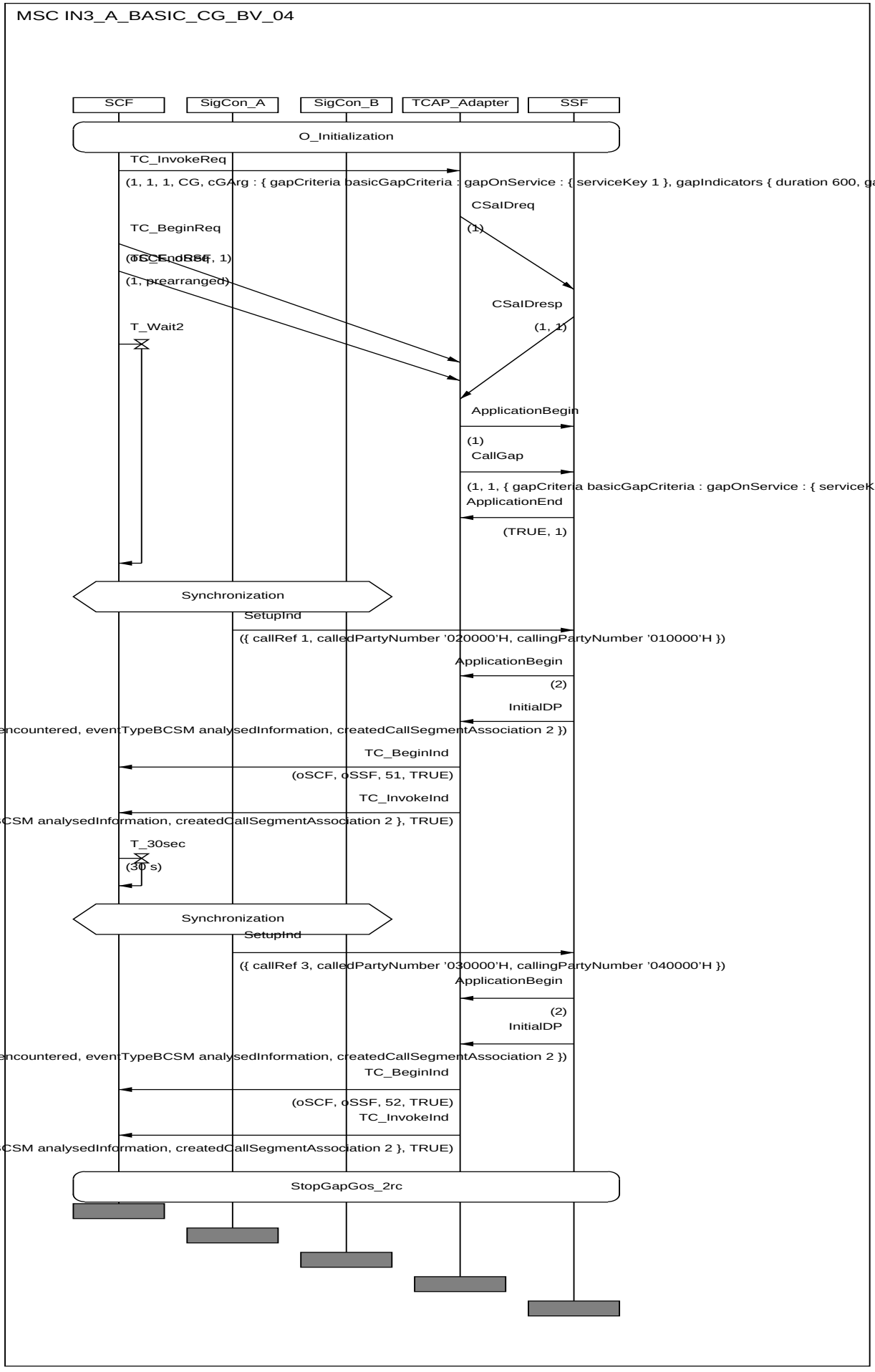
	IN3_A_BASIC_CG_BV_02
Work item no.:	ITEM_BASIC_275
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains only mandatory parameters a) **gapCriteria**, and b) **gapIndicators**, with values: a) **gapOnService** with any valid value for serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds); c) gapTreatment omitted. Check that an incoming call (during an active gap interval) is transferred to SigConB (default gap treatment) and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	DefaultOmitGapContinue
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: gapOnService with any valid value for serviceKey, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_4?SetupReq No further InitialDP received
Pass criteria	CP1_4?SetupReq No further InitialDP received
Postamble:	StopGapGos(serviceKey, scf=OMIT, ctrl=OMIT, clear=4)


MSC IN3_A_BASIC_CG_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
TC_InvokeReq
(1, 1, 1, CG, cGArg : { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators { duration 600, gapInterval 30000 }, c
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
(1, prearranged)
CSalDresp
(1, 1)
T_Wait2
ApplicationBegin
(1)
CallGap
(1, 1, { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators
ApplicationEnd
(TRUE, 1)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
untered manualCGencountered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ntered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
Synchronization
SetupInd
({ callRef 3, calledPartyNumber '030000'H, callingPartyNumber '040000'H })
SetupReq
({ callRef 2, calledPartyNumber '030000'H, callingPartyNumber '010000'H })
StopGapGos_clr2

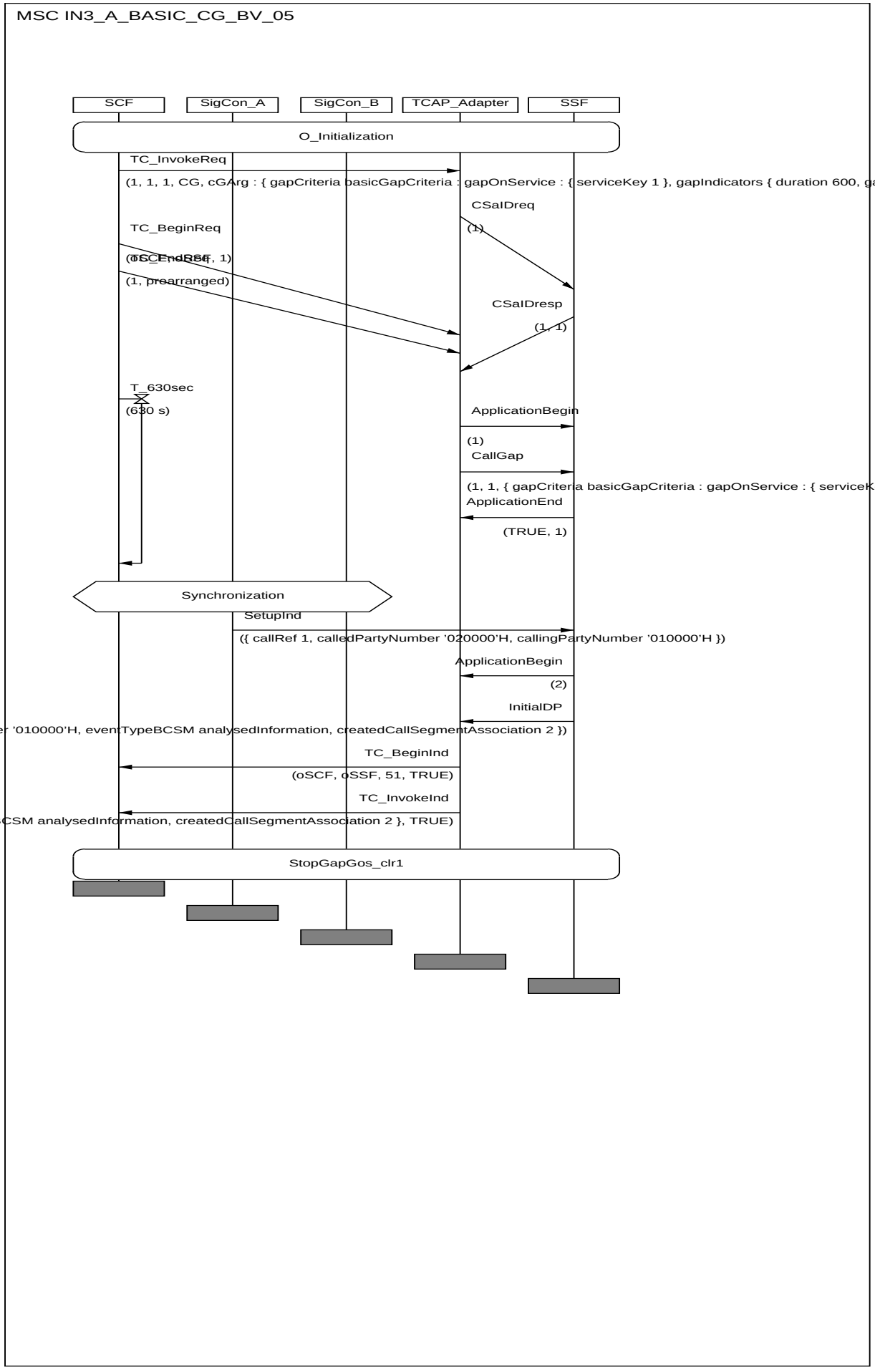
	IN3_A_BASIC_CG_BV_03
Work item no.:	ITEM_BASIC_276
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains only mandatory parameters a) **gapCriteria**, and b) **gapIndicators**, with values: a) **gapOnService** with any valid value for serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds); c) gapTreatment omitted. Check that an InitialDP invoke component is sent by the SSF, when an incoming call is received (during an active gap interval) and the call-related data received in the SetupInd signal are **not** compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: gapOnService with any valid value for serviceKey, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(**in**compatible call-related data) L3?TCAP-BEGIN L3?InitialDP invoke(serviceKey, no "cGEncountered" indicated)
Pass criteria	L3?InitialDP invoke(serviceKey, no "cGEncountered" indicated)
Postamble:	StopGapGos(serviceKey, scf=OMIT, ctrl=OMIT, clear=3)


MSC IN3_A_BASIC_CG_BV_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
TC_InvokeReq
(1, 1, 1, CG, cGArg : { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators { duration 600, gapInterval 30000 }, c
CSalDreq
(1)
TC_BeginReq
(oSCFEndRef, 1)
(1, prearranged)
CSalDresp
(1, 1)
T_Wait2
ApplicationBegin
(1)
CallGap
(1, 1, { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators
ApplicationEnd
(TRUE, 1)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
untered manualCGencountered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ntered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
Synchronization
SetupInd
({ callRef 3, calledPartyNumber '030000'H, callingPartyNumber '040000'H })
ApplicationBegin
(2)
InitialDP
, callingPartyNumber '010000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 52, TRUE)
TC_InvokeInd
0000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
StopGapGos_2rc

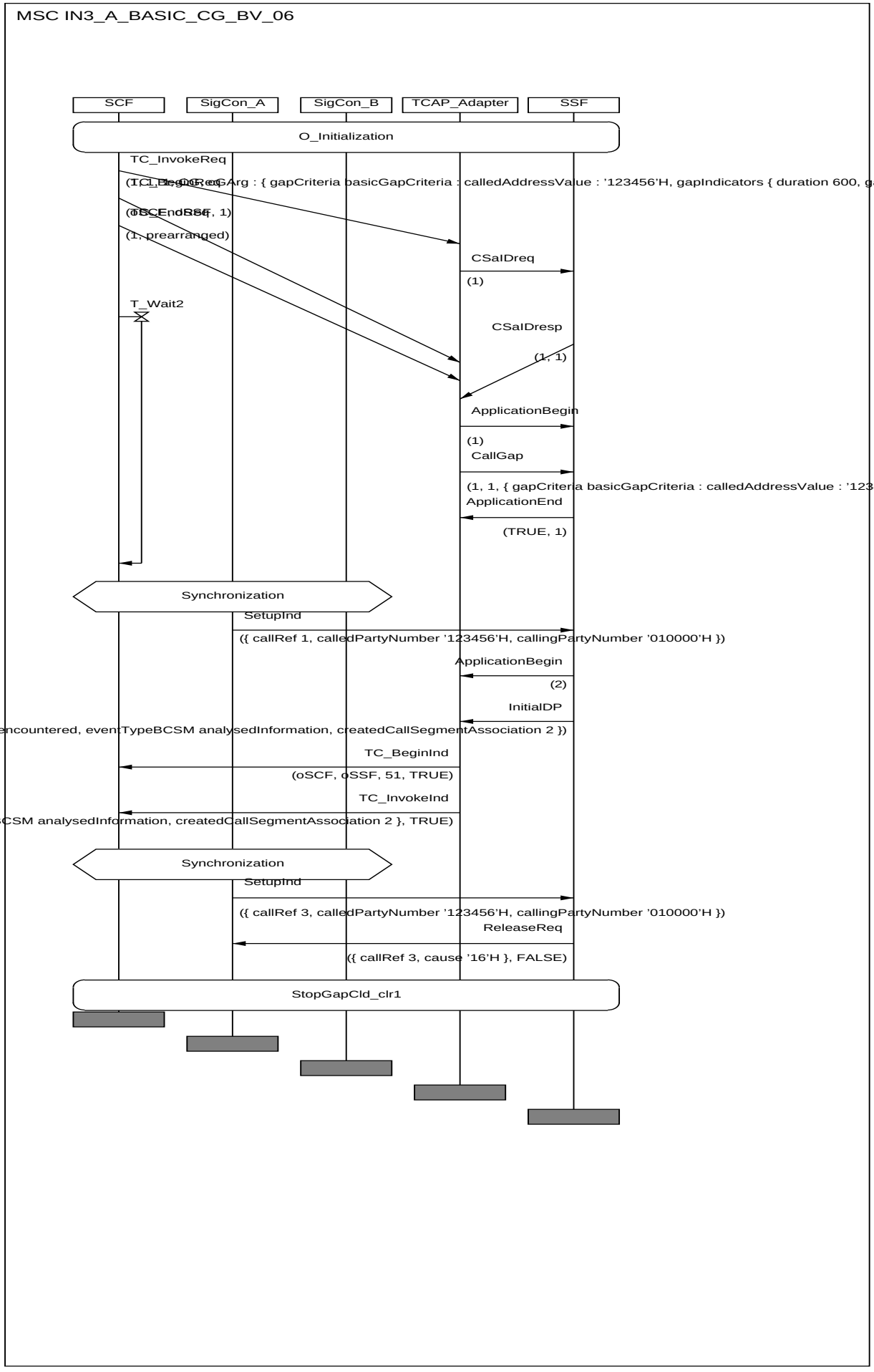
	IN3_A_BASIC_CG_BV_04
Work item no.:	ITEM_BASIC_277
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains only mandatory parameters a) **gapCriteria**, and b) **gapIndicators**, with values: a) **gapOnService** with any valid value for serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds); c) gapTreatment omitted. Check that an InitialDP invoke component is sent by the SSF containing all mandatory parameters and indicating call gapping encountered, with at least the parameters: serviceKey and cGEncountered, when an incoming call is received after gap **interval timer** expiration, and the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: gapOnService with any valid value for serviceKey, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp Wait some time to let the **interval timer** expire CP1_3!SetupInd(compatible call-related data) L3?TCAP-BEGIN L3?InitialDP invoke (**serviceKey,cGEncountered**)
Pass criteria	L3?InitialDP invoke (serviceKey,cGEncountered)
Postamble:	StopGapGos(serviceKey, scf=OMIT, ctrl=OMIT, clear=3)


MSC IN3_A_BASIC_CG_BV_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
TC_InvokeReq
(1, 1, 1, CG, cGArg : { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators { duration 600, gapInterval 30000 }, c
CSalDreq
TC_BeginReq
(1)
(oSCF, oSSF, 1)
(1, prearranged)
CSalDresp
(1, 1)
T_Wait2
ApplicationBegin
(1)
CallGap
(1, 1, { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators
ApplicationEnd
(TRUE, 1)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
untered manualCGencountered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ntered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
T_30sec
(30 s)
Synchronization
SetupInd
({ callRef 3, calledPartyNumber '030000'H, callingPartyNumber '040000'H })
ApplicationBegin
(2)
InitialDP
untered manualCGencountered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 52, TRUE)
TC_InvokeInd
ntered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
StopGapGos_2rc

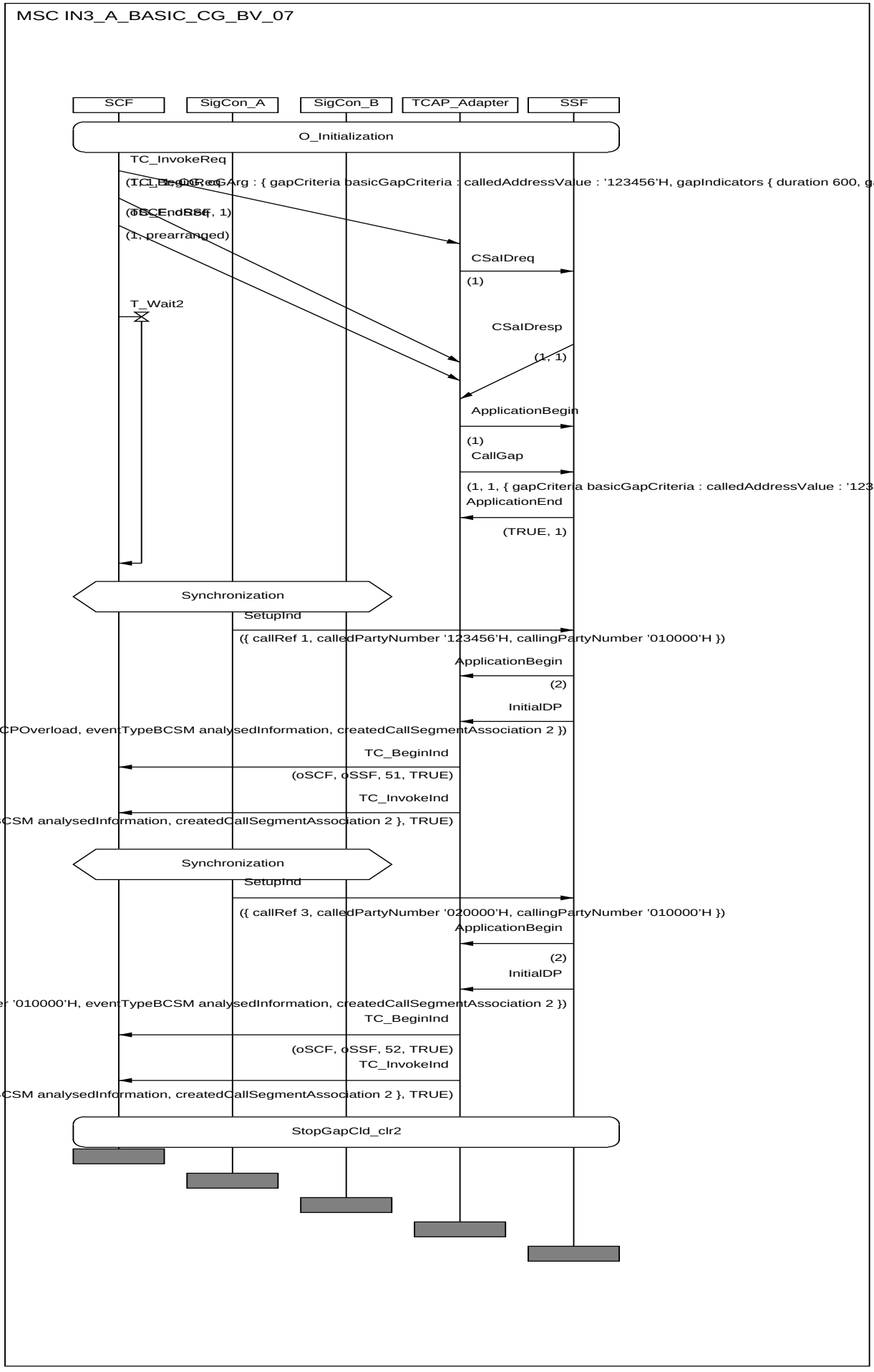
	IN3_A_BASIC_CG_BV_05
Work item no.:	ITEM_BASIC_278
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains only mandatory parameters a) **gapCriteria**, and b) **gapIndicators**, with values: a) **gapOnService** with any valid value for serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds); c) gapTreatment omitted. Check that an InitialDP invoke component is sent by the SSF, containing all mandatory parameters but not indicating call gapping encountered, when an incoming call is received after gap **duration timer** expiration, and the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: gapOnService with any valid value for serviceKey, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp Wait some time to let the **duration timer** expire CP1_3!SetupInd(compatible call-related data) L3?TCAP-BEGIN L3?InitialDP invoke(serviceKey, cGEncountered=OMIT)
Pass criteria	L3?InitialDP invoke(serviceKey, cGEncountered=OMIT)
Postamble:	StopGapGos(serviceKey, scf=OMIT, ctrl=OMIT, clear=3) NOTE: This postamble is used although CG duration has expired.


MSC IN3_A_BASIC_CG_BV_05
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
TC_InvokeReq
(1, 1, 1, CG, cGArg : { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators { duration 600, gapInterval 30000 }, c
CSalDreq
(1)
TC_BeginReq
(oSCF, 1)
(1, prearranged)
CSalDresp
(1, 1)
T_630sec
(630 s)
ApplicationBegin
(1)
CallGap
(1, 1, { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators
ApplicationEnd
(TRUE, 1)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
, callingPartyNumber '010000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
0000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
StopGapGos_clr1

	IN3_A_BASIC_CG_BV_06
Work item no.:	ITEM_BASIC_279
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) gapTreatment, with values: a) **calledAddressValue** with any valid value, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). Check that an incoming call (during an active gap interval) is released (cause = "gapCause") and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: calledAddressValue with any valid value, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_3?ReleaseReq("gapCause") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("gapCause") No further InitialDP received
Postamble:	StopGapCld(calledAddressValue,scf=OMIT, ctrl=OMIT, clear=2)


MSC IN3_A_BASIC_CG_BV_06
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
TC_InvokeReq
(1, prearranged)
CSalDreq
(1)
T_Wait2
CSalDresp
(1, 1)
ApplicationBegin
(1)
CallGap
ApplicationEnd
(TRUE, 1)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '123456'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
Synchronization
SetupInd
({ callRef 3, calledPartyNumber '123456'H, callingPartyNumber '010000'H })
ReleaseReq
({ callRef 3, cause '16'H }, FALSE)
StopGapCld_clr1

	IN3_A_BASIC_CG_BV_07
Work item no.:	ITEM_BASIC_280
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) gapTreatment, d) controlType, with values: a) **calledAddressValue** with any valid value, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType **sCPOverloaded**. Check that an InitialDP invoke component is sent by the SSF, when an incoming call is received (during an active gap interval) and the call-related data received in the SetupInd signal are **not** compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: calledAddressValue with any valid value, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") - controlType sCPOverloaded L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(**in**compatible call-related data) cannot be sent L3?TCAP-BEGIN L3?InitialDP invoke(serviceKey, cGEncountered=OMIT)
Pass criteria	L3?InitialDP invoke(serviceKey, cGEncountered=OMIT)
Postamble:	StopGapCld(calledAddressValue,scf=OMIT, ctrl=sCPOverloaded, clear=3)


MSC IN3_A_BASIC_CG_BV_07
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
TC_InvokeReq
(1, prearranged)
CSalDreq
(1)
T_Wait2
CSalDresp
(1, 1)
ApplicationBegin
(1)
CallGap
ApplicationEnd
(TRUE, 1)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '123456'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
Synchronization
SetupInd
({ callRef 3, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
TC_BeginInd
(oSCF, oSSF, 52, TRUE)
TC_InvokeInd
StopGapCld_clr2

	IN3_A_BASIC_CG_BV_08
Work item no.:	ITEM_BASIC_281
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** compound procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria** (compound), b) **gapIndicators**, c) gapTreatment and d) controlType, with values: a) **calledAddressValue** with any valid value, scfID = SCF_ID, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) **controlType destinationOverload**. Check that an InitialDP invoke component is sent by the SSF containing all mandatory parameters and indicating call gapping encountered, when an incoming call is received after gap **interval** timer expiration, and the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria (compound): calledAddressValue with any valid value, scfID = SCF_ID, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") - controlType **destinationOverload** L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp Wait some time to let the **interval** timer expire CP1_3!SetupInd(compatible call-related data) L3?TCAP-BEGIN L3?InitialDP invoke (**serviceKey,cGEncountered**)
Pass criteria	L3?InitialDP invoke (serviceKey,cGEncountered)
Postamble:	StopGapCld(calledAddressValue,scf=SCF_ID, ctrl=destinationOverload, clear=3)

	IN3_A_BASIC_CG_BV_09
Work item no.:	ITEM_BASIC_282
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** compound procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria** (compound), b) **gapIndicators**, c) gapTreatment and d) controlType, with values: a) **calledAddressValue** with any valid value, scfID = SCF_ID, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) **controlType** manuallyInitiated. Check that an InitialDP invoke component is sent by the SSF containing all mandatory parameters and indicating call gapping encountered, when an incoming call is received after gap interval timer expiration, and the call-related data received in the SetupInd signal are compatible with the gap criteria. Check also that the interval timer is started again: verify that the next incoming call (during the active gap interval) is released and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria (compound): calledAddressValue with any valid value, scfID = SCF_ID, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") - controlType **manuallyInitiated** L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp Wait some time to let the interval timer expire CP1_3!SetupInd(compatible call-related data) L3?TCAP-BEGIN L3?InitialDP invoke (serviceKey,cGEncountered) CP1_4!SetupInd(compatible call-related data) CP1_4?ReleaseReq("gapCause") No further InitialDP received
Pass criteria	L3?InitialDP invoke (serviceKey,cGEncountered) CP1_3?ReleaseReq("gapCause") No further InitialDP received
Postamble:	StopGapCld(calledAddressValue,scf=SCF_ID, ctrl=manuallyInitiated, clear=3)

	IN3_A_BASIC_CG_BV_10
Work item no.:	ITEM_BASIC_283
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **calledAddressValue** with any valid value, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = **informationToSend** (not "both/informationToSend"). d) controlType omitted. Check that an incoming call (during an active gap interval) is released and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria. Check also that the indicated "informationToSend" is provided to the calling user.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	DefaultItsGapRelease
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: calledAddressValue with any valid value, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment **informationToSend**(any valid value) L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_3?CallProgressReq(informationToSend) CP1_3?ReleaseReq(any cause) No further InitialDP received
Pass criteria	CP1_3?ReleaseReq(any cause) No further InitialDP received informationToSend provided to the calling user
Postamble:	StopGapCld(calledAddressValue,scf=OMIT, ctrl=OMIT, clear=2)

	IN3_A_BASIC_CG_BV_11
Work item no.:	ITEM_BASIC_284
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **calledAddressValue** with any valid value, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = informationToSend (not "both/informationToSend"). d) controlType omitted. Check that an incoming call (during an active gap interval) is processed further as a non-IN call and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria. Check also that the indicated "informationToSend" is provided to the calling user.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	DefaultItsGapContinue
Preamble:	None
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: calledAddressValue with any valid value, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment informationToSend(any valid value) L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_3?CallProgressReq(informationToSend) CP1_4?SetupReq No further InitialDP received
Pass criteria	CP1_4?SetupReq No further InitialDP received informationToSend provided to the calling user
Postamble:	StopGapCld(calledAddressValue,scf=OMIT, ctrl=OMIT, clear=4)

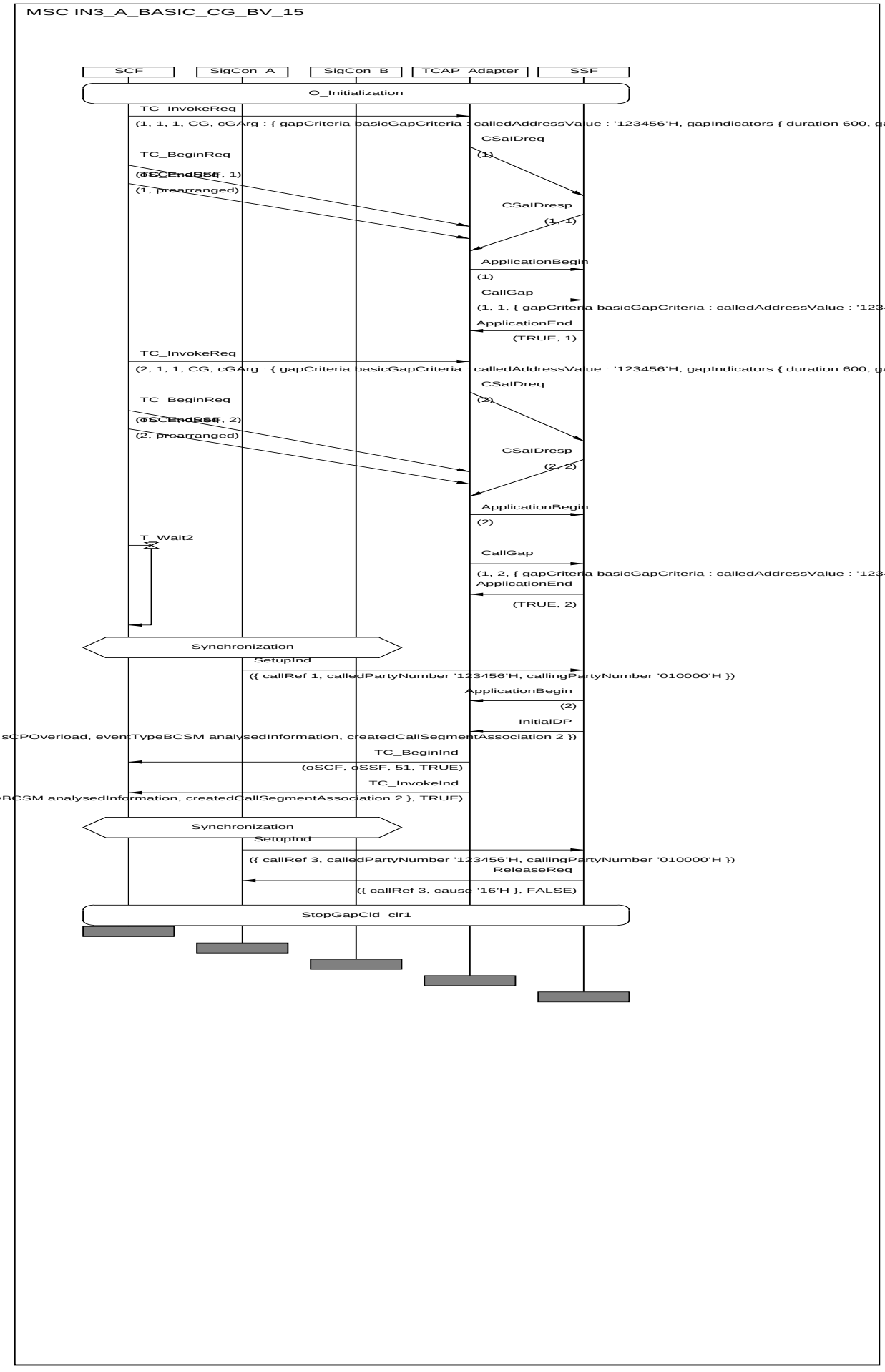
	IN3_A_BASIC_CG_BV_12
Work item no.:	ITEM_BASIC_285
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **calledAddressValue** with any valid value, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = **both**(valid informationToSend, "gapCause"). d) controlType omitted. Check that an incoming call (during an active gap interval) is released (cause = "gapCause") and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria. Check also that the indicated "informationToSend" is provided to the calling user.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF a **CallGap** invoke component with parameters/values: - gapCriteria: calledAddressValue with any valid value, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment **both**(valid informationToSend, "gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_3?CallProgressReq(informationToSend) CP1_3?ReleaseReq("gapCause") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("gapCause") No further InitialDP received informationToSend provided to the calling user
Postamble:	StopGapCld(calledAddressValue,scf=OMIT, ctrl=OMIT, clear=2)

	IN3_A_BASIC_CG_BV_13
Work item no.:	ITEM_BASIC_286
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the **CallGap** procedure twice for the same gapCriteria, but for different controlTypes **sCPOverloaded** and **manuallyInitiated**, performs the **manuallyInitiated** control on an incoming call matching the gapCriteria: Check that an incoming call (while both gap intervals are not exceeded) is released (cause = "gapCauseMan" = gapCause of Treatment specified for manuallyInitiated control) and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria. Check also that the "informationToSend" specified for manuallyInitiated control is provided to the calling user.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to the SSF a **CallGap** invoke component with parameters/values: - gapCriteria (basic): calledAddressValue with any valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment **releaseCause**(any cause value referred to as "sCPOverloadedCause") - controlType **sCPOverloaded** L1!TCAP-BEGIN SCF sends to the SSF a **CallGap** invoke component with parameters/values: - gapCriteria (basic): calledAddressValue with valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment **both** (releaseCause(any cause value different from "sCPOverloadedCause", referred to as "gapCauseMan", valid informationToSend) - controlType **manuallyInitiated** L2!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L3?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L3!RequestReportBCSMEvent(2,interrupted,oDisconnect) L3!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_3?CallProgressReq(informationToSend) CP1_3?ReleaseReq("gapCauseMan") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("gapCauseMan") No further InitialDP received informationToSend provided to the calling user
Postamble:	StopGapCld2(calledAddressValue,scf=OMIT, ctrl=sCPOverloaded, clear=0) StopGapCld2(calledAddressValue,scf=OMIT, ctrl=manuallyInitiated, clear=2)

	IN3_A_BASIC_CG_BV_14
Work item no.:	ITEM_BASIC_287
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the **CallGap** procedure twice for the same gapCriteria, but for different controlTypes **sCPOverloaded** and **manuallyInitiated**, and having stopped the **manuallyInitiated** callGap procedure, performs the **sCPOverloaded** control on an incoming call matching the gapCriteria (the manuallyInitiated control was appended and did not overwrite the sCPOverloaded control): Check that an incoming call (while the gap interval is active) is released (cause = "sCPOverloadedCause" = gapCause of Treatment specified for sCPOverloaded control) and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria. Check also that the "informationToSend" specified for manuallyInitiated control is not provided to the calling user.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to the SSF a **CallGap** invoke component with parameters/values: - gapCriteria (basic): calledAddressValue with any valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment releaseCause(any cause value referred to as "sCPOverloadedCause") - controlType sCPOverloaded L1!TCAP-BEGIN SCF sends to the SSF a **CallGap** invoke component with parameters/values: - gapCriteria (basic): calledAddressValue with valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment both (releaseCause(any cause value different from "sCPOverloadedCause", referred to as "gapCauseMan", valid informationToSend) - controlType manuallyInitiated L2!TCAP-BEGIN SCF sends to the SSF a **CallGap** invoke component with parameters/values (actually stopping the second callGap procedure): - gapCriteria (basic): calledAddressValue = calledAddress1, scfID = OMIT, - gapIndicators duration = 0 gapInterval = **interval1** (milliseconds) - gapTreatment omitted - controlType manuallyInitiated L3!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L4?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L4!RequestReportBCSMEvent(2,interrupted,oDisconnect) L4!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_3?ReleaseReq("sCPOverloadedCause") No further InitialDP received no informationToSend provided to the calling user

Pass criteria	CP1_3?ReleaseReq("sCPOverloadedCause") No further InitialDP received no informationToSend provided to the calling user
Postamble:	StopGapCld3(calledAddressValue,scf=OMIT, ctrl=sCPOverloaded, clear=2)

	IN3_A_BASIC_CG_BV_15
Work item no.:	ITEM_BASIC_288
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the **CallGap** procedure twice: for the same gapCriteria and for the same controlType (sCPOverloaded), but for different gapTreatment, performs the treatment specified for the second CallGap procedure on an incoming call matching the gapCriteria (the first gapTreatment is overwritten): Check that an incoming call (while both gap intervals are not exceeded) is released (cause = "gapCause2" = gapCause of Treatment specified for the second CallGap procedure) and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	None
Test description	SCF sends to the SSF a **CallGap** invoke component with parameters/values: - gapCriteria (basic): calledAddressValue with any valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment both (releaseCause(any cause value different from "sCPOverloadedCause", referred to as "sCPOverloadedCause", valid informationToSend) - controlType sCPOverloaded L1!TCAP-BEGIN SCF sends to the SSF a **CallGap** invoke component with parameters/values: - gapCriteria (basic): calledAddressValue with valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment releaseCause(any cause value referred to as "gapCause2") - controlType **sCPOverloaded** L2!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L3?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L3!RequestReportBCSMEvent(2,interrupted,oDisconnect) L3!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_3?CallProgressReq(informationToSend) CP1_3?ReleaseReq("gapCause2") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("gapCause2") No further InitialDP received informationToSend provided to the calling user
Postamble:	StopGapCld2(calledAddressValue,scf=OMIT, ctrl=sCPOverloaded, clear=2)


MSC IN3_A_BASIC_CG_BV_15
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
TC_InvokeReq
(1, 1, 1, CG, cGArg : { gapCriteria basicGapCriteria : calledAddressValue : '123456'H, gapIndicators { duration 600, gapInterval 30000 },
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
(1, prearranged)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CallGap
(1, 1, { gapCriteria basicGapCriteria : calledAddressValue : '123456'H, gapIndicators
ApplicationEnd
(TRUE, 1)
TC_InvokeReq
(2, 1, 1, CG, cGArg : { gapCriteria basicGapCriteria : calledAddressValue : '123456'H, gapIndicators { duration 600, gapInterval 30000 },
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
(2, prearranged)
CSalDresp
(2, 2)
ApplicationBegin
(2)
T_Wait2
CallGap
(1, 2, { gapCriteria basicGapCriteria : calledAddressValue : '123456'H, gapIndicators
ApplicationEnd
(TRUE, 2)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '123456'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
, cGEncountered sCPOverload, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
erload, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
Synchronization
SetupInd
({ callRef 3, calledPartyNumber '123456'H, callingPartyNumber '010000'H })
ReleaseReq
({ callRef 3, cause '16'H }, FALSE)
StopGapCld_clr1

	IN3_A_BASIC_CG_BV_16
Work item no.:	ITEM_BASIC_289
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the **CallGap** procedure twice: for the same gapCriteria and for the same controlType (sCPOverloaded), but for different gapIndicators and gapTreatment, overwrites the gapIndicators and gapTreatment values specified for the first CallGap procedure with the gapIndicators and gapTreatment values specified for the second CallGap procedure: Stop the second callGap procedure and check that an InitialDP invoke component not containing "cGEncountered" is sent by the SSF, when receiving an incoming call (during the gap interval specified for the first CallGap procedure).
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to the SSF a **CallGap** invoke component with parameters/values: - gapCriteria (basic): calledAddressValue with any valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment releaseCause(any cause value referred to as "sCPOverloadedCause") - controlType sCPOverloaded L1!TCAP-BEGIN SCF sends to the SSF **CallGap** invoke component with parameters/values: - gapCriteria (basic): calledAddressValue with valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = **interval2** (= half of interval1) - gapTreatment both (releaseCause(any cause value different from "sCPOverloadedCause", referred to as "gapCause2", valid informationToSend) - controlType sCPOverloaded L2!TCAP-BEGIN SCF sends to the SSF **CallGap** invoke component with parameters/values (stopping the second callGap procedure): - gapCriteria (basic): calledAddressValue with valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = 0 gapInterval = **interval2** (= half of interval1) - gapTreatment **both** (releaseCause(any cause value different from "sCPOverloadedCause", referred to as "gapCause2", valid informationToSend) - controlType sCPOverloaded L3!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L4?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L4!RequestReportBCSMEvent(2,interrupted,oDisconnect) L4!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) L5?TCAP-BEGIN L5?InitialDP(serviceKey,"cGEncountered" not contained)
Pass criteria	L5?TCAP-BEGIN L5?InitialDP(serviceKey,"cGEncountered" not contained)
Postamble:	StopGapCld3(calledAddressValue,scf=OMIT, ctrl=sCPOverloaded, clear=3)

	IN3_A_BASIC_CG_BV_17
Work item no.:	ITEM_BASIC_290
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **calledAddressAndService** with valid values for called address and serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an incoming call (during an active gap interval) is released (cause = "gapCause") and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria (called address and serviceKey).
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: **calledAddressAndService**(valid values for called address and serviceKey), - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd((call-related data compatible with called address and serviceKey) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible with called address and serviceKey) CP1_3?ReleaseReq("gapCause") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("gapCause") No further InitialDP received
Postamble:	StopGapCldService(serviceKey, calledAddressValue, scf=OMIT, ctrl=OMIT, clear=2)

	IN3_A_BASIC_CG_BV_18
Work item no.:	ITEM_BASIC_291
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **calledAddressAndService** with valid values for called address and serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an InitialDP invoke component (not containing parameter "cGEncountered") is sent by the SSF, when an incoming call is received (during an active gap interval) and the call-related data received in the SetupInd signal are compatible with the gap criteria for the called address, but not compatible with the serviceKey).
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: calledAddressAndService(valid values for called address and serviceKey), - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible with called address but **not** compatible with the serviceKey specified for callGap) L3?TCAP-BEGIN L3?InitialDP(serviceKey,"cGEncountered" not contained)
Pass criteria	L3?InitialDP(serviceKey,"cGEncountered" not contained)
Postamble:	StopGapCldService(serviceKey, calledAddressValue, scf=OMIT, ctrl=OMIT, clear=3)

	IN3_A_BASIC_CG_BV_19
Work item no.:	ITEM_BASIC_292
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **calledAddressAndService** with valid values for called address and serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an InitialDP invoke component (not containing parameter "cGEncountered") is sent by the SSF, when an incoming call is received (during an active gap interval) and the call-related data received in the SetupInd signal are compatible with the gap criteria for the serviceKey, but not compatible with the called address).
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: calledAddressAndService(valid values for called address and serviceKey), - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible with serviceKey but **not** compatible with the called address specified for callGap) L3?TCAP-BEGIN L3?InitialDP(serviceKey,"cGEncountered" not contained)
Pass criteria	L3?InitialDP(serviceKey,"cGEncountered" not contained)
Postamble:	StopGapCldService(serviceKey, calledAddressValue, scf=OMIT, ctrl=OMIT, clear=3)

	IN3_A_BASIC_CG_BV_20
Work item no.:	ITEM_BASIC_293
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **calledAddressAndService** with valid values for called address and serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an InitialDP invoke component is sent by the SSF containing all mandatory parameters and indicating call gapping encountered, when an incoming call is received after gap interval timer expiration, and the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: calledAddressAndService(valid values for called address and serviceKey), - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp Wait some time to let the interval **timer** expire CP1_3!SetupInd(call-related data compatible with serviceKey and called address specified for callGap) L3?TCAP-BEGIN L3?InitialDP(serviceKey,"cGEncountered")
Pass criteria	L3?InitialDP(serviceKey,"cGEncountered")
Postamble:	StopGapCldService(serviceKey, calledAddressValue, scf=OMIT, ctrl=OMIT, clear=3)

	IN3_A_BASIC_CG_BV_21
Work item no.:	ITEM_BASIC_294
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **callingAddressAndService** with valid values for calling address and serviceKey, locationNumber being omitted, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an incoming call (during an active gap interval) is released (cause = "gapCause") and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria (calling address and serviceKey).
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: call**ing**AddressAndService(valid values for calling address and serviceKey, locationNumber omitted), - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible with call**ing** address and serviceKey) CP1_3?ReleaseReq("gapCause") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("gapCause") No further InitialDP received
Postamble:	StopGap**Clg**Service(serviceKey, callingAddressValue, locationNumber=OMIT, scf=OMIT, ctrl=OMIT, clear=2)

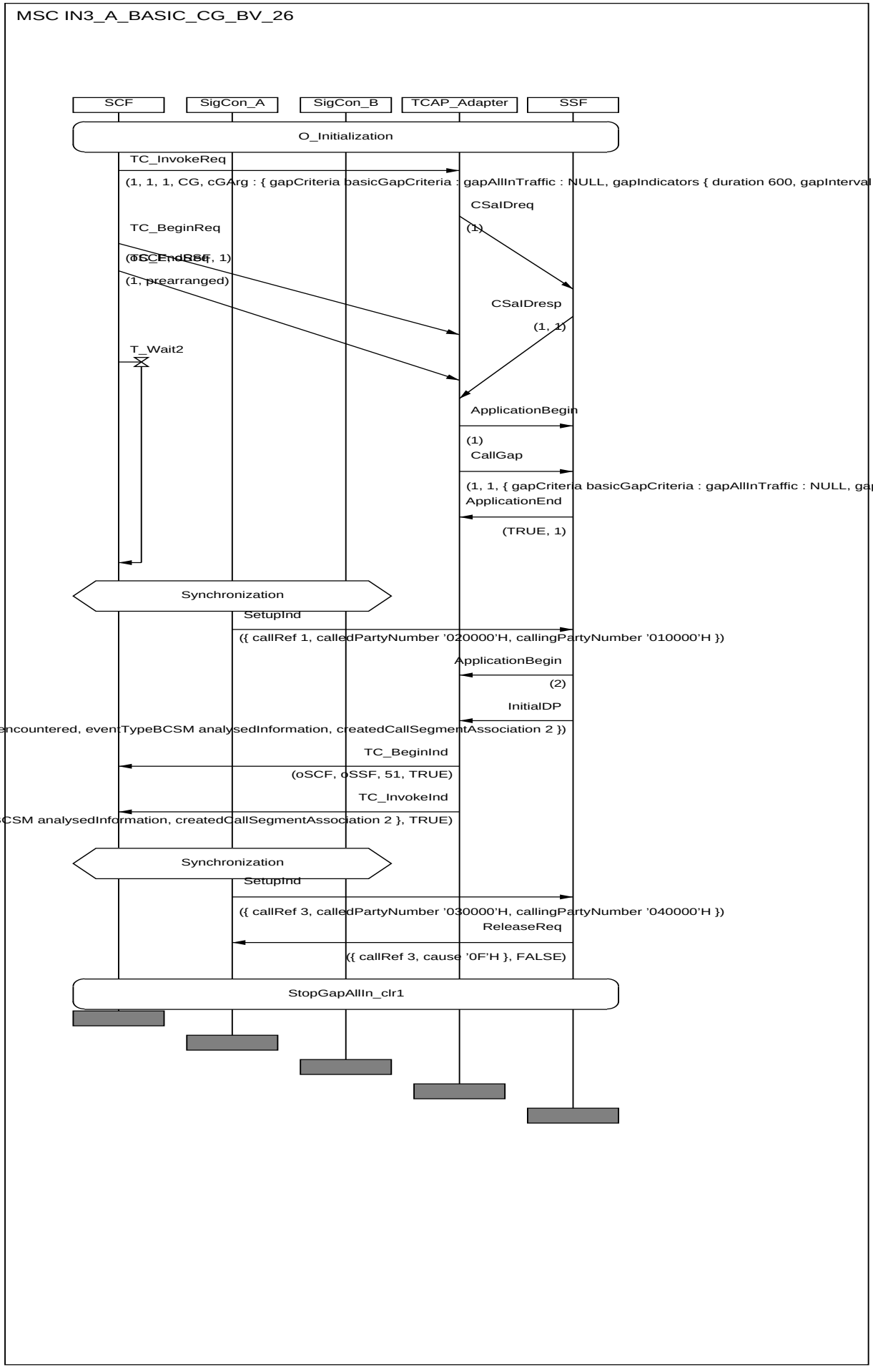
	IN3_A_BASIC_CG_BV_22
Work item no.:	ITEM_BASIC_295
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **callingAddressAndService** with valid values for serviceKey, calling address and locationNumber, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an incoming call (during an active gap interval) is released (cause = "gapCause") and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria serviceKey and locationNumber, but not compatible with the calling address.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: callingAddressAndService(valid values for calling address, serviceKey and **locationNumber**), - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible with serviceKey and locationNumber, but **not** compatible with the calling address) CP1_3?ReleaseReq("gapCause") No further InitialDP received
Pass criteria	CP1_2?ReleaseReq("gapCause") No further InitialDP received
Postamble:	StopGapClgService(serviceKey, callingAddressValue**, locationNumber**, scf=OMIT, ctrl=OMIT, clear=2)

	IN3_A_BASIC_CG_BV_23
Work item no.:	ITEM_BASIC_296
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **callingAddressAndService** with valid values for serviceKey, calling address and locationNumber, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an InitialDP invoke component (not containing parameter "cGEncountered") is sent by the SSF, when an incoming call is received (during an active gap interval) and the call-related data received in the SetupInd signal are compatible with the gap criteria for calling address and locationNumber, but not compatible with the serviceKey).
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: callingAddressAndService(valid values for calling address and serviceKey, locationNumber omitted), - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible with calling address and locationNumber, but **not** compatible with the serviceKey) L3?TCAP-BEGIN L3?InitialDP(serviceKey,"cGEncountered" not contained)
Pass criteria	L3?InitialDP(serviceKey,"cGEncountered" not contained)
Postamble:	StopGapClgService(serviceKey, callingAddressValue, locationNumber omitted, scf=OMIT, ctrl=OMIT, clear=3)

	IN3_A_BASIC_CG_BV_24
Work item no.:	ITEM_BASIC_297
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **callingAddressAndService** with valid values for serviceKey, calling address and locationNumber, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an InitialDP invoke component (not containing parameter "cGEncountered") is sent by the SSF, when an incoming call is received (during an active gap interval) and the call-related data received in the SetupInd signal are compatible with the gap criteria for calling address and serviceKey, but not compatible with the locationNumber).
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	None
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: callingAddressAndService(valid values for calling address, serviceKey and locationNumber), - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible with calling address and serviceKey, but **not** compatible with the locationNumber) L3?TCAP-BEGIN L3?InitialDP(serviceKey,"cGEncountered" not contained)
Pass criteria	L3?InitialDP(serviceKey,"cGEncountered" not contained)
Postamble:	StopGapClgService(serviceKey, callingAddressValue, locationNumber, scf=OMIT, ctrl=OMIT, clear=3)

	IN3_A_BASIC_CG_BV_25
Work item no.:	ITEM_BASIC_298
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **callingAddressAndService** with valid values for serviceKey, calling address and locationNumber, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an InitialDP invoke component is sent by the SSF containing all mandatory parameters and indicating call gapping encountered, when an incoming call is received after gap interval timer expiration, and the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	None
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: callingAddressAndService(valid values for calling address and serviceKey, locationNumber), - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp Wait some time to let the **interval** timer expire CP1_3!SetupInd(call-related data compatible with calling address, serviceKey and locationNumber) L3?TCAP-BEGIN L3?InitialDP(serviceKey,**"cGEncountered"**)
Pass criteria	L3?InitialDP(serviceKey,"cGEncountered")
Postamble:	StopGapClgService(serviceKey, callingAddressValue, locationNumber, scf=OMIT, ctrl=OMIT, clear=3)

	IN3_A_BASIC_CG_BV_26
Work item no.:	ITEM_BASIC_299
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **gapAllInTraffic**, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an incoming call (during an active gap interval) is released (cause = "gapCause") and no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria (any IN service).
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - **gapAllInTraffic**, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) CP1_3?ReleaseReq("gapCause") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("gapCause") No further InitialDP received
Postamble:	StopGapAllIn(scf=OMIT, ctrl=OMIT, clear=2)


MSC IN3_A_BASIC_CG_BV_26
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
TC_InvokeReq
(1, 1, 1, CG, cGArg : { gapCriteria basicGapCriteria : gapAllInTraffic : NULL, gapIndicators { duration 600, gapInterval 30000 }, controlType
CSalDreq
(1)
TC_BeginReq
(oSCF, 1)
(1, prearranged)
CSalDresp
(1, 1)
T_Wait2
ApplicationBegin
(1)
CallGap
(1, 1, { gapCriteria basicGapCriteria : gapAllInTraffic : NULL, gapIndicators { duration
ApplicationEnd
(TRUE, 1)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
untered manualCGencountered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ntered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
Synchronization
SetupInd
({ callRef 3, calledPartyNumber '030000'H, callingPartyNumber '040000'H })
ReleaseReq
({ callRef 3, cause '0F'H }, FALSE)
StopGapAllIn_clr1

	IN3_A_BASIC_CG_BV_27
Work item no.:	ITEM_BASIC_300
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **Idle** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **gapAllInTraffic**, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds). c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an InitialDP invoke component is sent by the SSF containing all mandatory parameters and indicating call gapping encountered, when an incoming call is received after gap interval timer expiration, and the call-related data received in the SetupInd signal are compatible with the gap criteria (any IN service).
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapAllInTraffic, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-BEGIN CP1_1!SetupInd(compatible call-related data) L2?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp Wait some time to let the **interval** timer expire CP1_3!SetupInd(compatible call-related data) L3?TCAP-BEGIN L3?InitialDP(serviceKey,"cGEncountered")
Pass criteria	L3?InitialDP(serviceKey,"cGEncountered")
Postamble:	StopGapAllIn(scf=OMIT, ctrl=OMIT, clear=3)

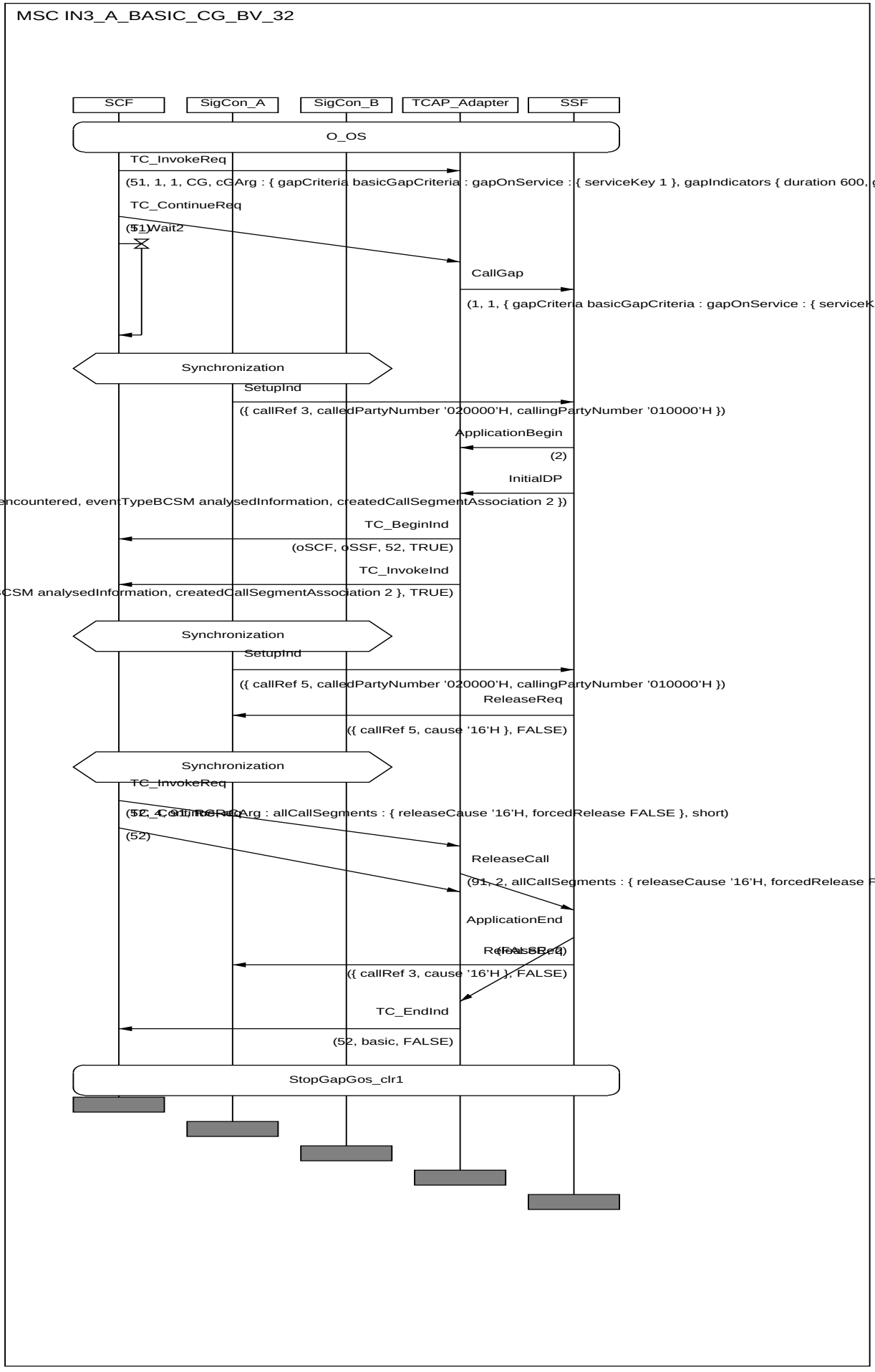
	IN3_A_BASIC_CG_BV_28
Work item no.:	ITEM_BASIC_301
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the **CallGap** procedure twice: once for gapCriteria "calledAddressValue", and once for gapCriteria "callingAddressAndService", performs the gapTreatment of the "calledAddressValue" on an incoming call matching both gapCriteria: Check that an incoming call (while both gap intervals are active) is released (cause = "calledAddressValueCause" = gapCause of Treatment specified for gapCriteria "calledAddressValue"), and that no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with both gap criteria. Check also that the "informationToSend" specified for gapCriteria "calledAddressValue" is provided to the calling user.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to the SSF **CallGap** invoke component with parameters/values: - gapCriteria (basic) calledAddressValue: calledAddressValue with any valid value referred to as calledAddress1, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment both (releaseCause(any valid cause value referred to as "calledAddressValueCause", valid informationToSend) - controlType OMIT L1!TCAP-BEGIN SCF sends to the SSF **CallGap** invoke component with parameters/values: - gapCriteria (basic): call**ing**AddressAndService with valid values for callingAddress and serviceKey, locationNumber=OMIT, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment releaseCause(any cause value different from "calledAddressValueCause", referred to as "callingAddressAndServiceCause") - controlType OMIT L2!TCAP-BEGIN CP1_1!SetupInd(call-related data compatible to all gap criteria) L3?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the intervals, starts the intervals and is not gapped) L3!RequestReportBCSMEvent(2,interrupted,oDisconnect) L3!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible to all gap criteria) CP1_3?CallProgressReq(informationToSend) CP1_3?ReleaseReq("calledAddressValueCause") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("calledAddressValueCause") No further InitialDP received informationToSend provided to the calling user
Postamble:	StopGapCld2(calledAddressValue,scf=OMIT, ctrl=OMIT, clear=0) StopGapClgService2(serviceKey, callingAddressDigits, locationNumber=OMIT, scf=OMIT, ctrl=OMIT, clear=2)

	IN3_A_BASIC_CG_BV_29
Work item no.:	ITEM_BASIC_302
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the **CallGap** procedure twice: once for gapCriteria "calledAddressAndService", and once for gapCriteria "callingAddressAndService", performs the gapTreatment of the "calledAddressAndService" on an incoming call matching both gapCriteria: Check that an incoming call (while both gap intervals are active) is released (cause = "calledAddressAndServiceCause" = gapCause of Treatment specified for gapCriteria "calledAddressAndService"), and that no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with both gap criteria. Check also that the "informationToSend" specified for gapCriteria "calledAddressAndService" is provided to the calling user.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to the SSF a **CallGap** invoke component with parameters/values: - gapCriteria (basic) calledAddressAndService: valid calledAddress and serviceKey values, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment both (releaseCause(any valid cause value referred to as "calledAddressAndServiceCause", valid informationToSend) - controlType OMIT L1!TCAP-BEGIN SCF sends to the SSF **CallGap** invoke component with parameters/values: - gapCriteria (basic): callingAddressAndService with valid values for callingAddress and serviceKey, locationNumber=OMIT, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment releaseCause(any cause value different from "calledAddressAndServiceCause", referred to as "callingAddressAndServiceCause") - controlType OMIT L2!TCAP-BEGIN CP1_1!SetupInd(call-related data compatible to all gap criteria) L3?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the intervals, starts the intervals and is not gapped) L3!RequestReportBCSMEvent(2,interrupted,oDisconnect) L3!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible to all gap criteria) CP1_3?CallProgressReq(informationToSend) CP1_3?ReleaseReq("calledAddressAndServiceCause") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("calledAddressAndServiceCause") No further InitialDP received informationToSend provided to the calling user
Postamble:	StopGapCldService2(serviceKey, calledAddressValue, scf=OMIT, ctrl=OMIT, clear=0) StopGapClgService2(serviceKey, callingAddressDigits, locationNumber=OMIT, scf=OMIT, ctrl=OMIT, clear=2)

	IN3_A_BASIC_CG_BV_30
Work item no.:	ITEM_BASIC_303
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the **CallGap** procedure twice: once for gapCriteria "callingAddressAndService", and once for gapCriteria "gapOnService", performs the gapTreatment of the "callingAddressAndService" on an incoming call matching both gapCriteria: Check that an incoming call (while both gap intervals are active) is released (cause = "callingAddressAndServiceCause" = gapCause of Treatment specified for gapCriteria "callingAddressAndService"), and that no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with both gap criteria. Check also that the "informationToSend" specified for gapCriteria "callingAddressAndService" is provided to the calling user.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to the SSF **CallGap** invoke component with parameters/values: - gapCriteria (basic) callingAddressAndService: with valid values for callingAddress and serviceKey, locationNumber=OMIT, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment both (releaseCause(any valid cause value referred to as "callingAddressAndServiceCause", referred to as "gapOnServiceCause"), valid informationToSend) - controlType OMIT L1!TCAP-BEGIN SCF sends to the SSF **CallGap** invoke component with parameters/values: - gapCriteria (basic) gapOnService: valid serviceKey value, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment releaseCause(any cause value different from "callingAddressAndServiceCause") - controlType OMIT L2!TCAP-BEGIN CP1_1!SetupInd(call-related data compatible to all gap criteria) L3?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the intervals, starts the intervals and is not gapped) L3!RequestReportBCSMEvent(2,interrupted,oDisconnect) L3!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible to all gap criteria) CP1_3?CallProgressReq(informationToSend) CP1_3?ReleaseReq("callingAddressAndServiceCause") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("callingAddressAndServiceCause") No further InitialDP received informationToSend provided to the calling user
Postamble:	StopGapClgService2(serviceKey, callingAddressDigits, locationNumber=OMIT, scf=OMIT, ctrl=OMIT, clear=0) StopGapGos2(serviceKeyscf=OMIT, ctrl=OMIT, clear=2)

	IN3_A_BASIC_CG_BV_31
Work item no.:	ITEM_BASIC_304
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the **CallGap** procedure twice: once for gapCriteria "gapOnService", and once for gapCriteria "gapAllInService", performs the gapTreatment of the "gapOnService" on an incoming call matching both gapCriteria: Check that an incoming call (while both gap intervals are active) is released (cause = "gapOnServiceCause" = gapCause of Treatment specified for gapCriteria "gapOnService"), and that no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with both gap criteria. Check also that the "informationToSend" specified for gapCriteria "gapOnService" is provided to the calling user.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	none
Test description	SCF sends to the SSF **CallGap** invoke component with parameters/values: - gapCriteria (basic) gapOnService: valid serviceKey value, scfID = OMIT, - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment both (releaseCause(any cause value different from "gapAllInServiceCause", referred to as "gapOnServiceCause", valid informationToSend) - controlType OMIT L1!TCAP-BEGIN SCF sends to the SSF **CallGap** invoke component with parameters/values: - gapCriteria (basic) gapAllInService - gapIndicators duration = duration1 (seconds) gapInterval = interval1 (milliseconds) - gapTreatment releaseCause(any valid cause value referred to as "gapAllInServiceCause")) - controlType OMIT L2!TCAP-BEGIN CP1_1!SetupInd(call-related data compatible to all gap criteria) L3?InitialDP invoke (serviceKey,cGEncountered) (the first call falls outside the intervals, starts the intervals and is not gapped) L3!RequestReportBCSMEvent(2,interrupted,oDisconnect) L3!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(call-related data compatible to all gap criteria) CP1_3?CallProgressReq(informationToSend) CP1_3?ReleaseReq("gapOnServiceCause") No further InitialDP received
Pass criteria	CP1_3?ReleaseReq("gapOnServiceCause") No further InitialDP received informationToSend provided to the calling user
Postamble:	StopGapAllIn2(scf=OMIT, ctrl=OMIT, clear=0) StopGapGos2(serviceKeyscf=OMIT, ctrl=OMIT, clear=2)

	IN3_A_BASIC_CG_BV_32
Work item no.:	ITEM_BASIC_305
IN2 Ref(tmp)	None
Purpose:	Verify the **CallGap** base procedure performed by the SSF in the **WaitingForInstructions** state when the **CallGap** invoke component sent by the SCF contains parameters a) **gapCriteria**, b) **gapIndicators** and c) **gapTreatment**, with values: a) **gapOnService** with any valid value for serviceKey, and b) **duration** (being a valid positive duration value in seconds) and **gapInterval** (being a valid positive interval value in milliseconds); c) **gapTreatment** = releaseCause("gapCause"). d) controlType omitted. Check that an incoming call (during an active gap interval) is released and that no InitialDP invoke component is sent by the SSF, when the call-related data received in the SetupInd signal are compatible with the gap criteria.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: gapOnService with any valid value for serviceKey, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L1!TCAP-CONTINUE (the CallGap invoke component is sent as first response to the IDP received in the preamble) L1!RequestReportBCSMEvent(2,interrupted,oDisconnect) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_3!SetupInd(compatible call-related data) L2?InitialDP invoke(serviceKey,cGEncountered) (the first call matching call gap criteria falls outside the interval, starts the interval and is not gapped) L2!RequestReportBCSMEvent(2,interrupted,oDisconnect) L2!Connect(2,1) CP1_4?SetUpReq CP1_4!SetUpConf CP1_3?SetUpResp CP1_5!SetupInd(compatible call-related data) CP1_5?ReleaseReq("gapCause") No further InitialDP received
Pass criteria	CP1_5?ReleaseReq("gapCause") No further InitialDP received
Postamble:	StopGapGosOS(serviceKey, scf=OMIT, ctrl=OMIT, clear=2)


MSC IN3_A_BASIC_CG_BV_32
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 1, 1, CG, cGArg : { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators { duration 600, gapInterval 30000 },
TC_ContinueReq
(51)
Wait2
CallGap
(1, 1, { gapCriteria basicGapCriteria : gapOnService : { serviceKey 1 }, gapIndicators
Synchronization
SetupInd
({ callRef 3, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
InitialDP
untered manualCGencountered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 })
TC_BeginInd
(oSCF, oSSF, 52, TRUE)
TC_InvokeInd
ntered, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 2 }, TRUE)
Synchronization
SetupInd
({ callRef 5, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseReq
({ callRef 5, cause '16'H }, FALSE)
Synchronization
TC_InvokeReq
(52, 4, cGArg : allCallSegments : { releaseCause '16'H, forcedRelease FALSE }, short)
(52)
ReleaseCall
(91, 2, allCallSegments : { releaseCause '16'H, forcedRelease FALSE }
ApplicationEnd
(FALSE)
({ callRef 3, cause '16'H }, FALSE)
TC_EndInd
(52, basic, FALSE)
StopGapGos_clr1

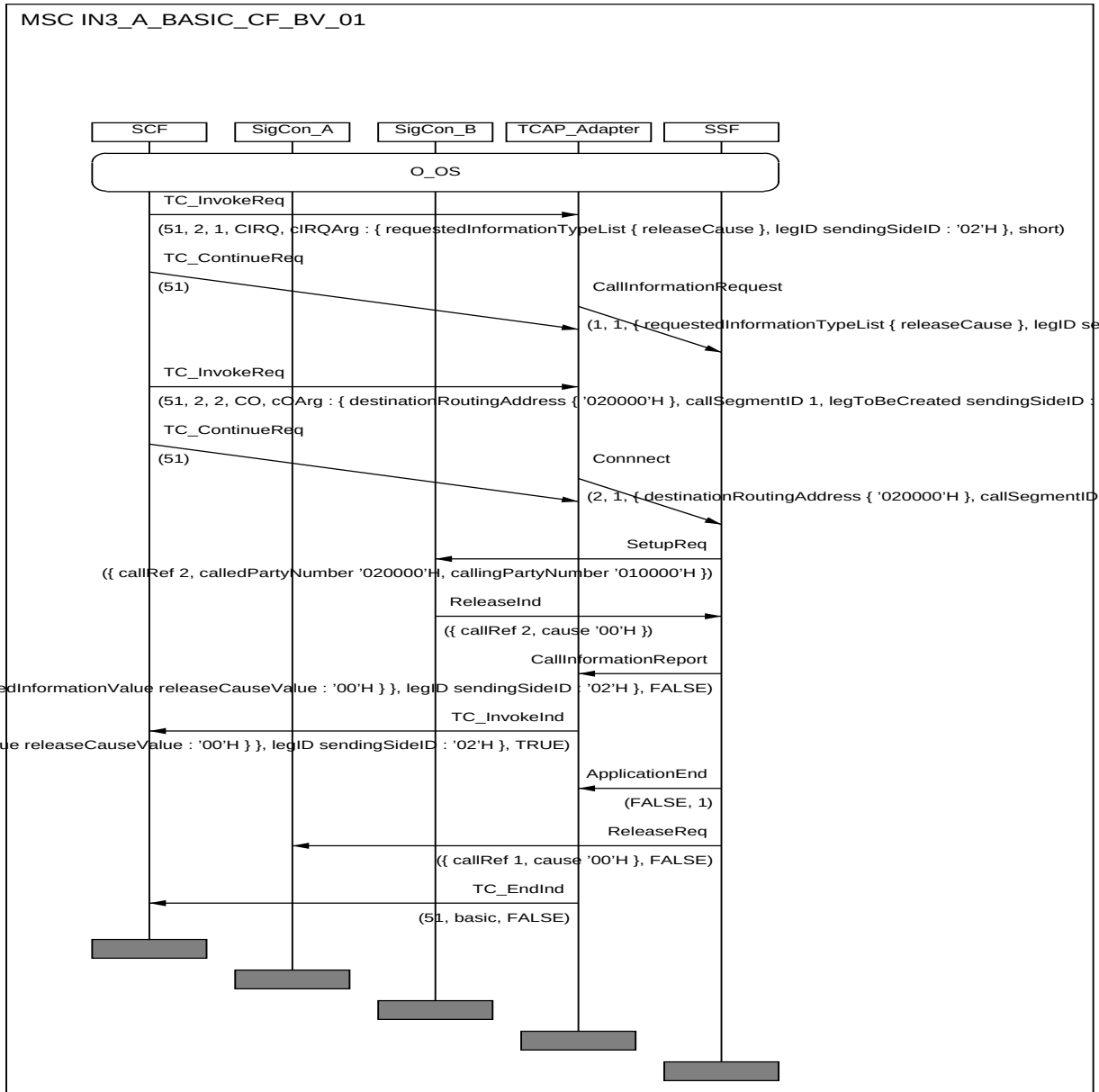
	IN3_A_BASIC_CG_BI_01
Work item no.:	ITEM_BASIC_308
IN2 Ref(tmp)	None
Purpose:	Verify that the **CallGap** procedure is not started, when the **CallGap** invoke component sent by the SCF does not contain mandatory parameter **gapCriteria**. Check that an InitialDP invoke component is sent by the SSF containing all mandatory parameters, but not indicating call gapping encountered, when an incoming call is received.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	None
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: omitted, - gapIndicators duration (positive duration value (seconds)) gapInterval (valid positive interval value in milliseconds) - gapTreatment releaseCause("gapCause") L2!TCAP-BEGIN CP1_1!SetupInd L1?InitialDP(serviceKey,"cGEncountered" not contained)
Pass criteria	L1?InitialDP(serviceKey,"cGEncountered" not contained)
Postamble:	ReleaseCallA

	IN3_A_BASIC_CG_BI_02
Work item no.:	ITEM_BASIC_309
IN2 Ref(tmp)	None
Purpose:	Verify that the **CallGap** procedure is not started, when the **CallGap** invoke component sent by the SCF does not contain mandatory parameter **gapIndicators**. Check that an InitialDP invoke component is sent by the SSF containing all mandatory parameters, but not indicating call gapping encountered, when an incoming call is received.
Requirements refs	6.5.1.2.1, 8.2.2, 8.2.2.1, 11.6, 11.26
Selection Cond.	
Preamble:	None
Test description	SCF sends to SSF **CallGap** invoke component with parameters/values: - gapCriteria: gapOnService with any valid value for serviceKey, - gapIndicators omitted - gapTreatment releaseCause("gapCause") L2!TCAP-BEGIN CP1_1!SetupInd L1?InitialDP(serviceKey,"cGEncountered" not contained)
Pass criteria	L1?InitialDP(serviceKey,"cGEncountered" not contained)
Postamble:	ReleaseCallA

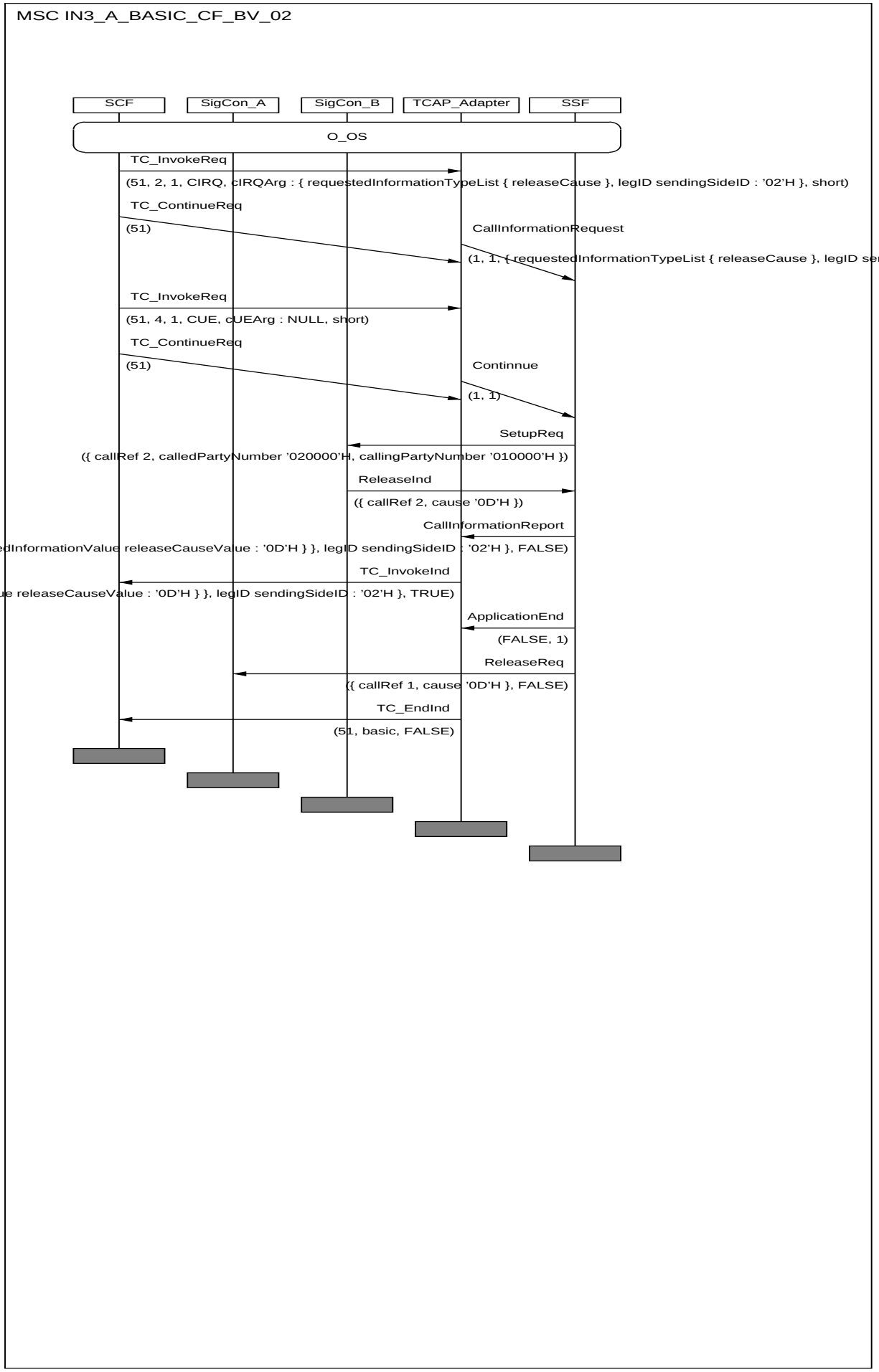
6.6.4 CallInformation (CF) procedure

NOTE: Other Test Purposes related to the CallInformation procedure are contained in EN 301 933-2 [5].

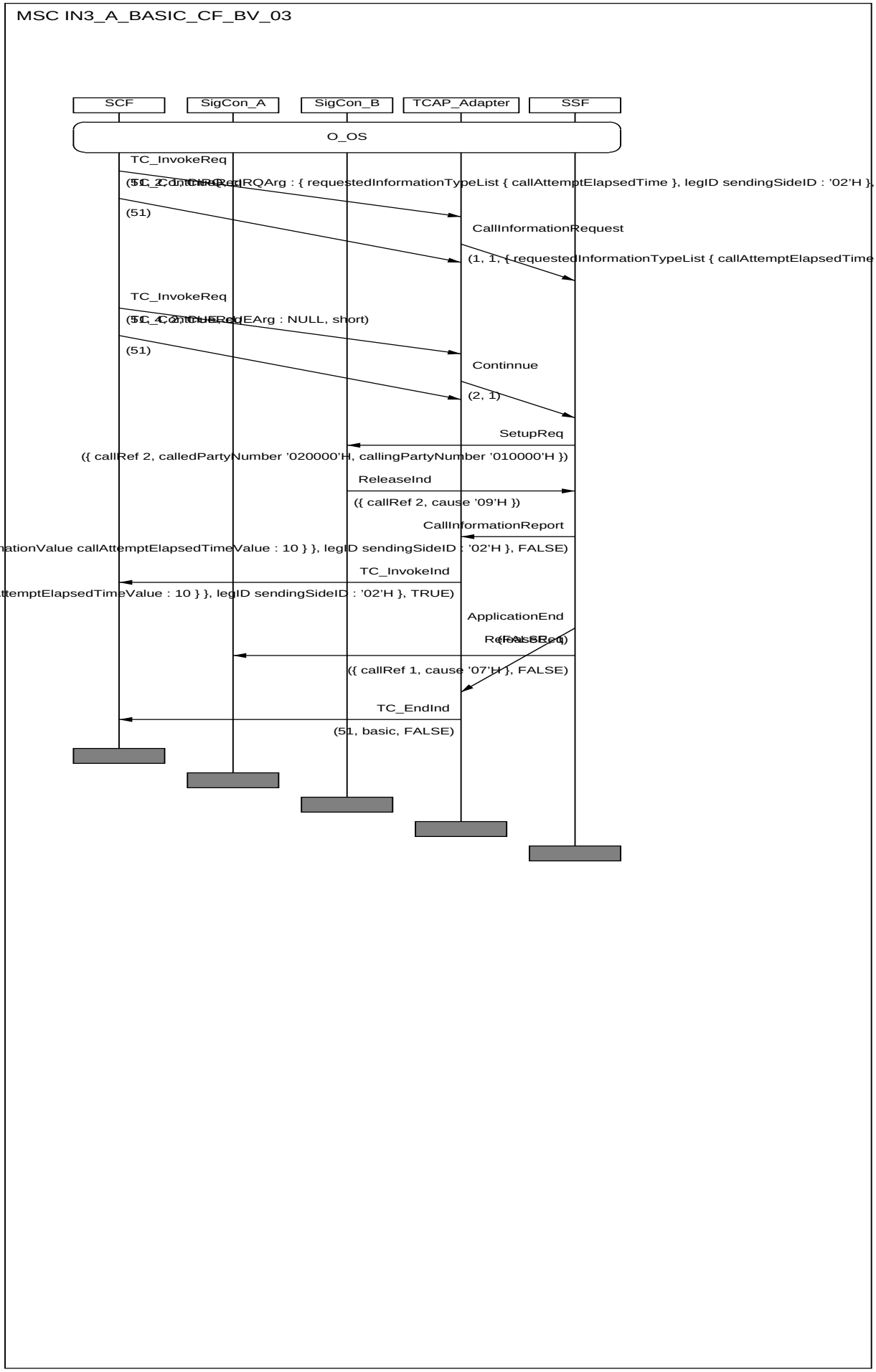
	IN3_A_BASIC_CF_BV_01
Work item no.:	ITEM_BASIC_25
IN2 Ref(tmp)	IN2_A_BASIC_CF_CA_01
Purpose:	Verify that upon detection of call release, the SSF sends a **CallInformationReport** invoke component with the **requestedInformationList** including the information requested by the SCF in the previously issued **CallInformationRequest** invoke component. The **requestedInformationTypeList** includes **requestedInformationType** = releaseCause.
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.7.1, 11.7.1.1.1, 11.7.2.1, 11.8.1, 11.8.1.1.1, 11.8.3.1, 11.12
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 IN3_A_BASIC_CF_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, CIRQ, cIRQArg : { requestedInformationTypeList { releaseCause }, legID sendingSideID : '02'H }, short)
TC_ContinueReq
(51)
CallInformationRequest
(1, 1, { requestedInformationTypeList { releaseCause }, legID sendingSideID : '02'H }
TC_InvokeReq
(51, 2, 2, CO, cOArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
TC_ContinueReq
(51)
Connnect
(2, 1, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '00'H })
CallInformationReport
aseCause, requestedInformationValue releaseCauseValue : '00'H } }, legID sendingSideID : '02'H }, FALSE)
TC_InvokeInd
estedInformationValue releaseCauseValue : '00'H } }, legID sendingSideID : '02'H }, TRUE)
ApplicationEnd
(FALSE, 1)
ReleaseReq
({ callRef 1, cause '00'H }, FALSE)
TC_EndInd
(51, basic, FALSE)

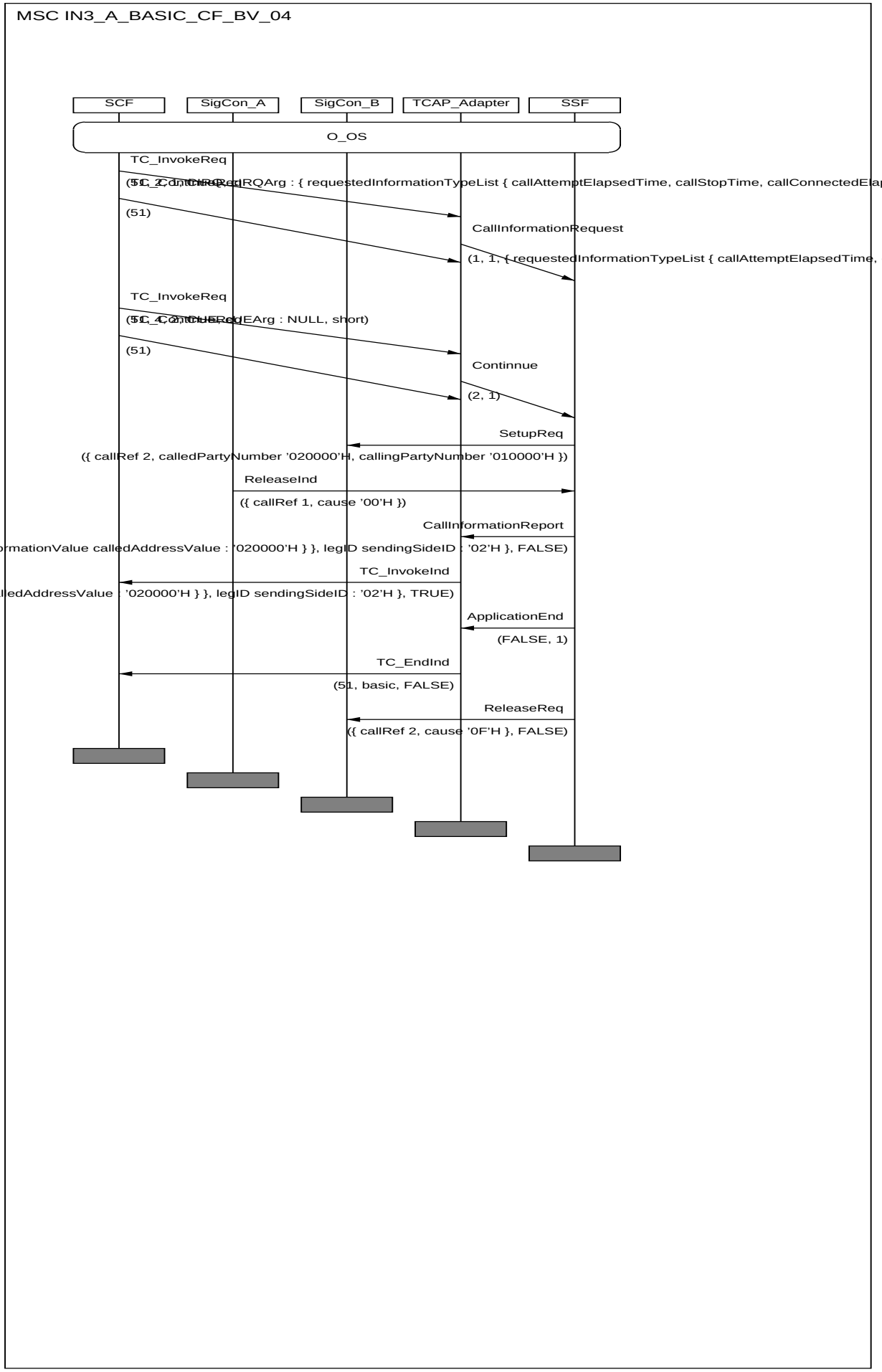
	IN3_A_BASIC_CF_BV_02
Work item no.:	ITEM_BASIC_26
IN2 Ref(tmp)	IN2_A_BASIC_CF_BV_01
Purpose:	Verify that upon detection of SigConB busy, the SSF sends a **CallInformationReport** invoke component with the **requestedInformationList** including the information requested by the SCF in the previously issued **CallInformationRequest** invoke component. The **requestedInformationTypeList** includes **requestedInformationType** = releaseCause.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.7.1, 11.7.1.1.1, 11.7.2.1, 11.8.1, 11.8.1.1.1, 11.8.3.1, 11.14
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 **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 IN3_A_BASIC_CF_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, CIRQ, cIRQArg : { requestedInformationTypeList { releaseCause }, legID sendingSideID : '02'H }, short)
TC_ContinueReq
(51)
CallInformationRequest
(1, 1, { requestedInformationTypeList { releaseCause }, legID sendingSideID : '02'H }
TC_InvokeReq
(51, 4, 1, CUE, cUEArg : NULL, short)
TC_ContinueReq
(51)
Continnue
(1, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '0D'H })
CallInformationReport
aseCause, requestedInformationValue releaseCauseValue : '0D'H } }, legID sendingSideID : '02'H }, FALSE)
TC_InvokeInd
stedInformationValue releaseCauseValue : '0D'H } }, legID sendingSideID : '02'H }, TRUE)
ApplicationEnd
(FALSE, 1)
ReleaseReq
({ callRef 1, cause '0D'H }, FALSE)
TC_EndInd
(51, basic, FALSE)

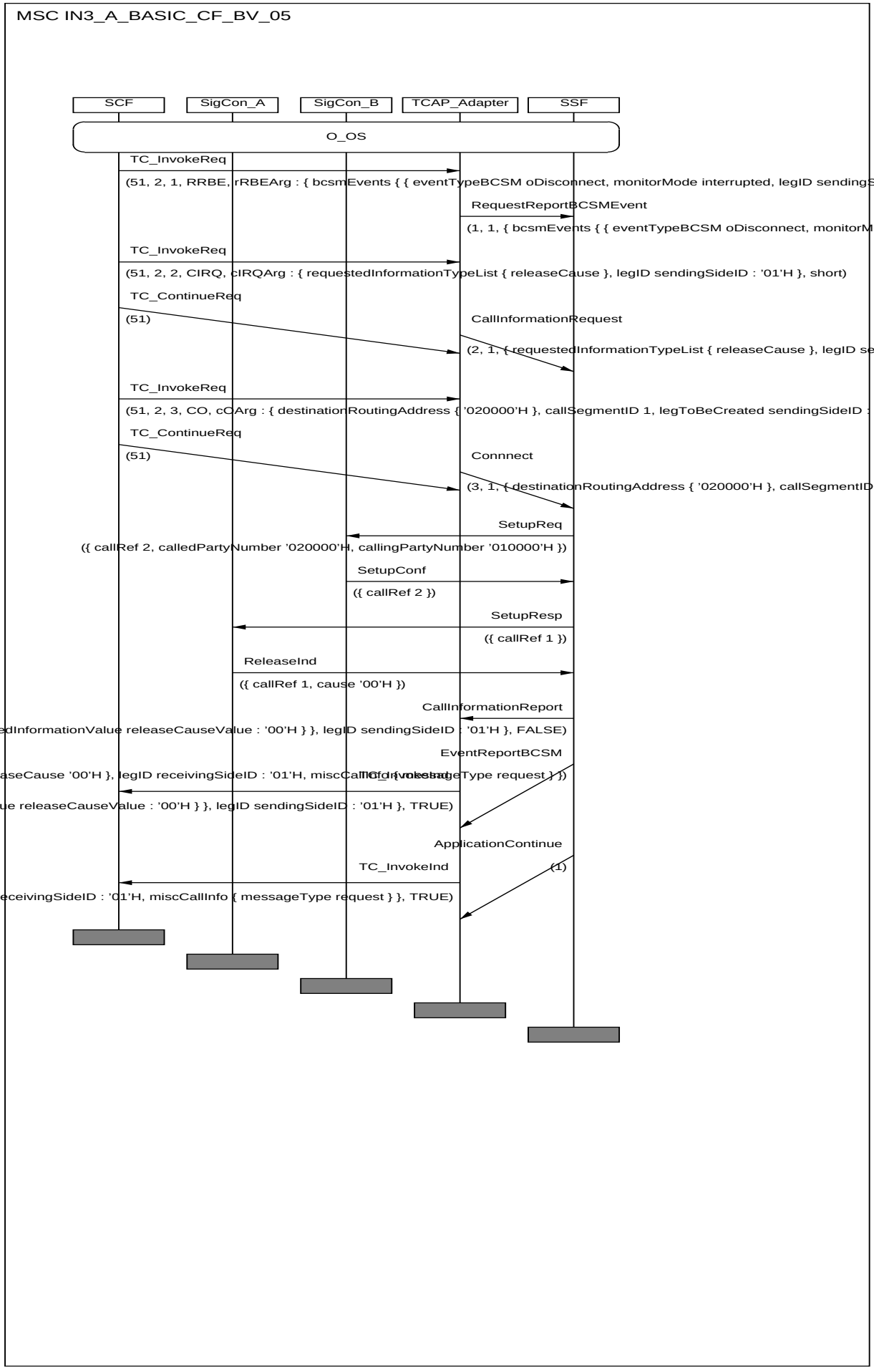
	IN3_A_BASIC_CF_BV_03
Work item no.:	ITEM_BASIC_27
IN2 Ref(tmp)	IN2_A_BASIC_CF_BV_02
Purpose:	Verify that upon detection of **timeout** (no answer from SigConB**)**, the SSF sends a **CallInformationReport** invoke component with the **requestedInformationList** including the information requested by the SCF in the previously issued **CallInformationRequest** invoke component. The **requestedInformationTypeList** includes **requestedInformationType** = callAttemptElapsedTime.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.7.1, 11.7.1.1.1, 11.7.2.1, 11.8.1, 11.8.1.1.1, 11.8.3.1, 11.14
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 (callAttemptElapsedTime), followed by a **Continu**e 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 IN3_A_BASIC_CF_BV_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, CallInformationRequest, ARG : { requestedInformationTypeList { callAttemptElapsedTime }, legID sendingSideID : '02'H }, short
(51)
CallInformationRequest
(1, 1, { requestedInformationTypeList { callAttemptElapsedTime }, legID sendingSide
TC_InvokeReq
(51, 4, Continue, ARG : NULL, short)
(51)
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '09'H })
CallInformationReport
me, requestedInformationValue callAttemptElapsedTimeValue : 10 } }, legID sendingSideID : '02'H }, FALSE)
TC_InvokeInd
ormationValue callAttemptElapsedTimeValue : 10 } }, legID sendingSideID : '02'H }, TRUE)
ApplicationEnd
ReleaseReq
({ callRef 1, cause '07'H }, FALSE)
TC_EndInd
(51, basic, FALSE)

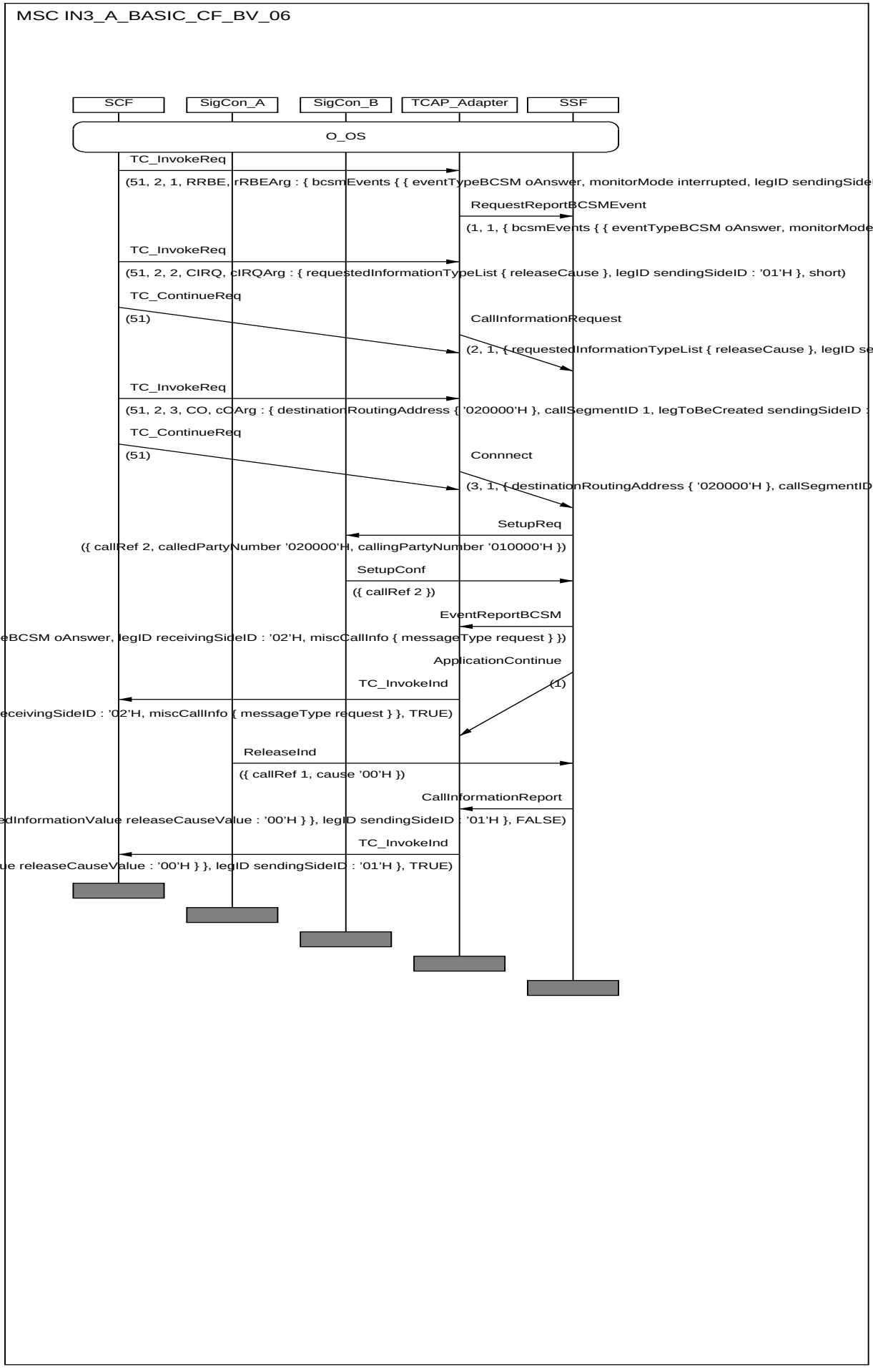
	IN3_A_BASIC_CF_BV_04
Work item no.:	ITEM_BASIC_28
IN2 Ref(tmp)	IN2_A_BASIC_CF_BV_03
Purpose:	Verify that upon detection of **release from SigConA**, the SSF sends a **CallInformationReport** invoke component with the **requestedInformationList** including the information requested by the SCF in the previously issued **CallInformationRequest** invoke component. The multiple **requestedInformationTypeList** includes **requestedInformationType** = callAttemptElapsedTime, callStopTime, callConnectedElapsedTime and calledAddress.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.7.1, 11.7.1.1.1, 11.7.2.1, 11.8.1, 11.8.1.1.1, 11.8.3.1, 11.14
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 callConnectedElapsedTime Value, and including: - requestedInformationType (calledAddress), - requestedInformationValue being calledAddressValue
Postamble:	none


MSC IN3_A_BASIC_CF_BV_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, CallInformationRQArg : { requestedInformationTypeList { callAttemptElapsedTime, callStopTime, callConnectedElapsedTime, calledAd
(51)
CallInformationRequest
(1, 1, { requestedInformationTypeList { callAttemptElapsedTime, callStopTime, callC
TC_InvokeReq
(51, 4, ContinueEArg : NULL, short)
(51)
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 1, cause '00'H })
CallInformationReport
dress, requestedInformationValue calledAddressValue : '020000'H } }, legID sendingSideID : '02'H }, FALSE)
TC_InvokeInd
InformationValue calledAddressValue : '020000'H } }, legID sendingSideID : '02'H }, TRUE)
ApplicationEnd
(FALSE, 1)
TC_EndInd
(51, basic, FALSE)
ReleaseReq
({ callRef 2, cause '0F'H }, FALSE)

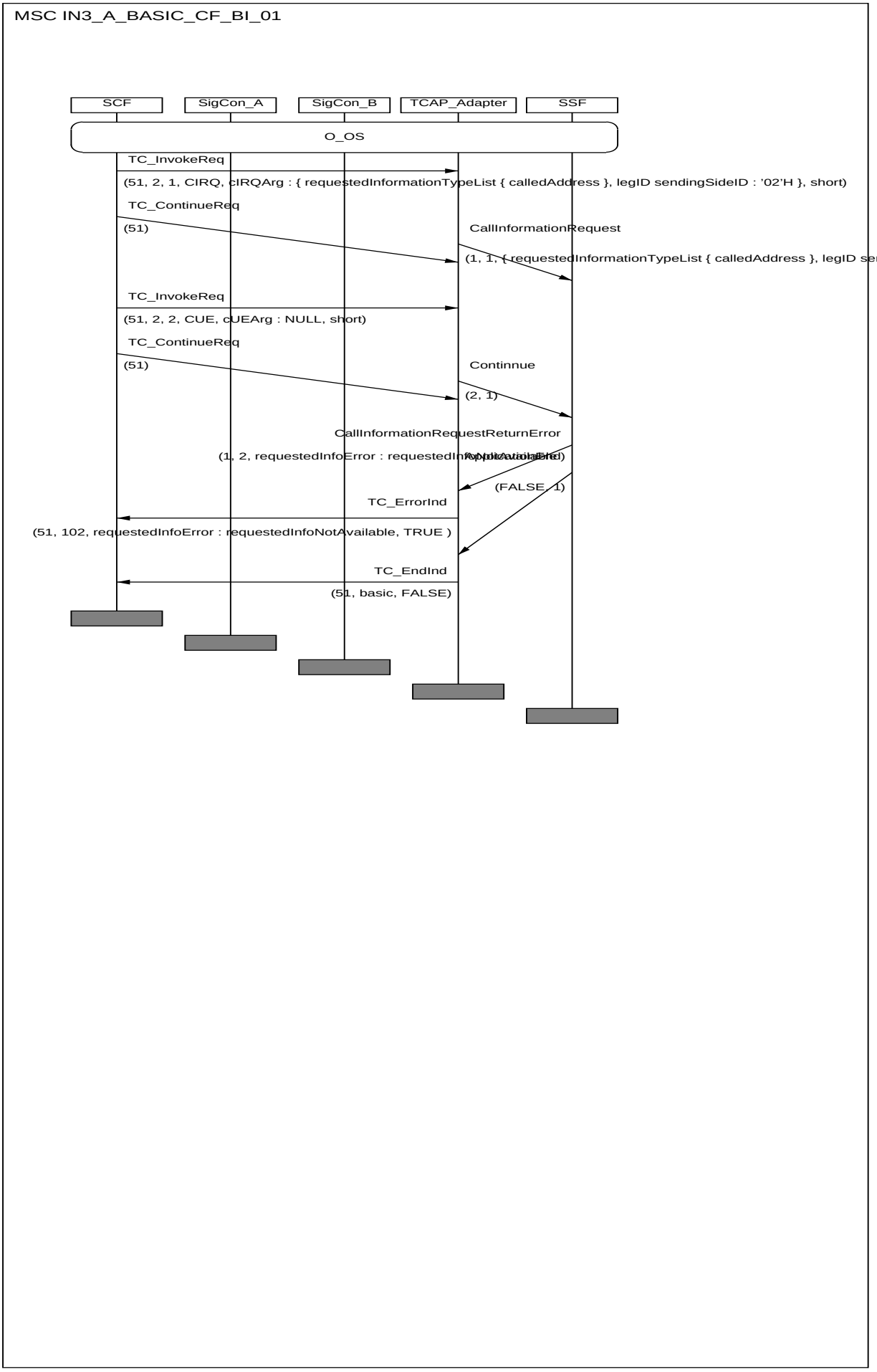
	IN3_A_BASIC_CF_BV_05
Work item no.:	ITEM_BASIC_29
IN2 Ref(tmp)	IN2_A_BASIC_CF_BV_04
Purpose:	Verify that upon detection of **release from SigConA**, the SSF sends a **CallInformationReport** invoke component with the **requestedInformationList** including the information requested by the SCF in the previously issued **CallInformationRequest** invoke component, followed by an **EventReportBCSM** with the indication of **eventTypeBCSM=** oDisconnect, if the O_Disconnect DP has been armed by the SCF. The **requestedInformationTypeList** includes **requestedInformationType** = releaseCause.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.7.1, 11.7.1.1.1, 11.7.2.1, 11.8.1, 11.8.1.1.1, 11.8.3.1, 11.12, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg1) - eventTypeBCSM= oDisconnect - monitorMode=interrupted then SCF sends a **CallInformationRequest** invoke, containing mandatory parameters only, with at least the parameters: (leg2) - 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 IN3_A_BASIC_CF_BV_05
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID
TC_InvokeReq
(51, 2, 2, CIRQ, cIRQArg : { requestedInformationTypeList { releaseCause }, legID sendingSideID : '01'H }, short)
TC_ContinueReq
(51)
CallInformationRequest
(2, 1, { requestedInformationTypeList { releaseCause }, legID sendingSideID : '01'H }
TC_InvokeReq
(51, 2, 3, CO, cOArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
TC_ContinueReq
(51)
Connnect
(3, 1, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ReleaseInd
({ callRef 1, cause '00'H })
CallInformationReport
aseCause, requestedInformationValue releaseCauseValue : '00'H } }, legID sendingSideID : '01'H }, FALSE)
EventReportBCSM
tSpecificInfo : { releaseCause '00'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
estedInformationValue releaseCauseValue : '00'H } }, legID sendingSideID : '01'H }, TRUE)
ApplicationContinue
(1)
TC_InvokeInd
ause '00'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)

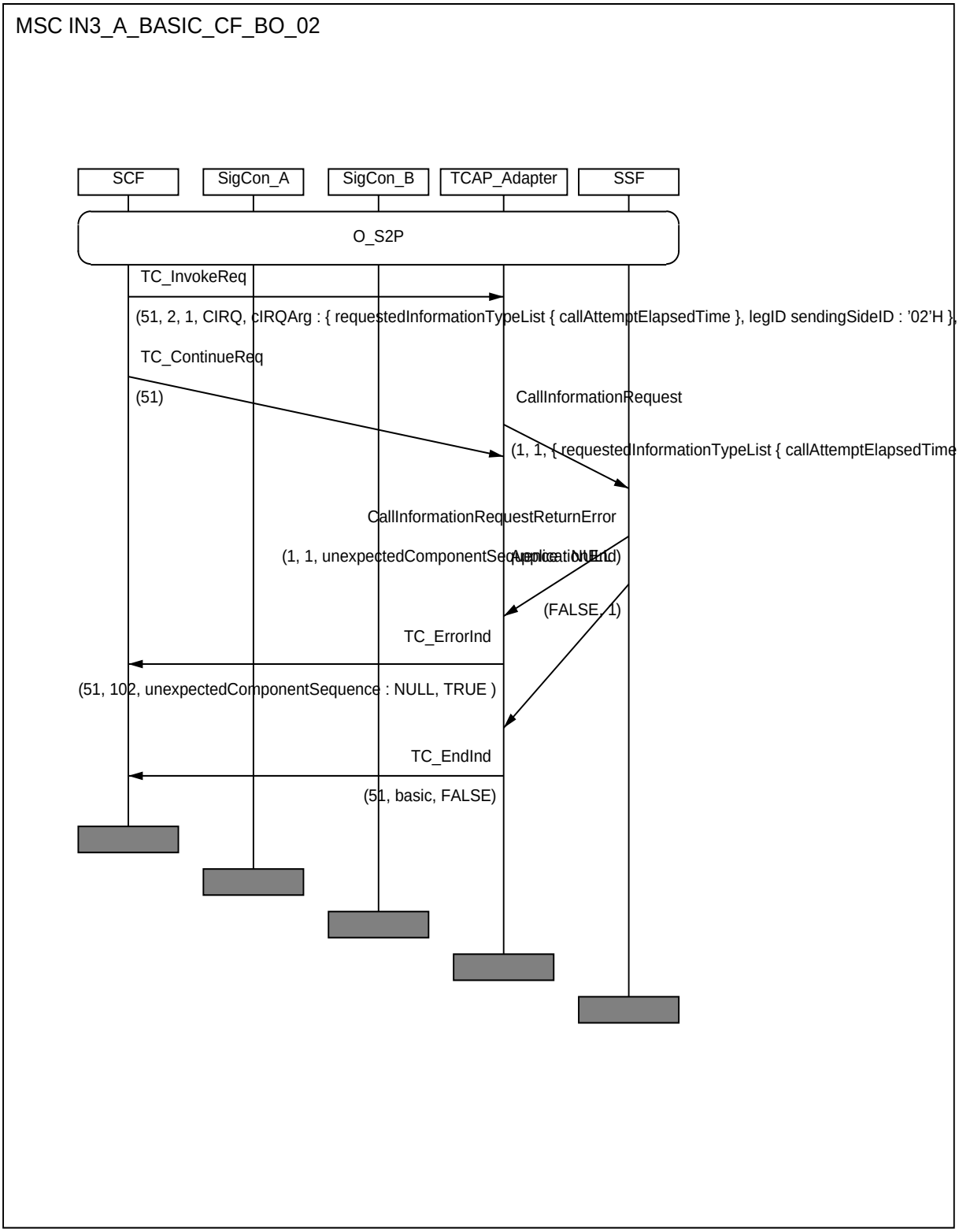
	IN3_A_BASIC_CF_BV_06
Work item no.:	ITEM_BASIC_30
IN2 Ref(tmp)	IN2_A_BASIC_CF_BV_05
Purpose:	Verify that upon connection of a call the SSF sends an **EventReportBCSM** with the indication of **eventTypeBCSM** = oAnswer, and that upon detection of **release from SigConA**, the SSF sends a **CallInformationReport** invoke component with the **requestedInformationList** including the information requested by the SCF in the previously issued **CallInformationRequest** invoke component, if the O_Answer DP has been armed by the SCF. The **requestedInformationTypeList** includes **requestedInformationType** = releaseCause.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.7.1, 11.7.1.1.1, 11.7.2.1, 11.8.1, 11.8.1.1.1, 11.8.3.1, 11.12, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg2) - eventTypeBCSM= **oAnswer** - monitorMode=interrupted then SCF sends a **CallInformationRequest** invoke, containing mandatory parameters only, with at least the parameters: (leg1) - 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 IN3_A_BASIC_CF_BV_06
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, short
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID se
TC_InvokeReq
(51, 2, 2, CIRQ, cIRQArg : { requestedInformationTypeList { releaseCause }, legID sendingSideID : '01'H }, short)
TC_ContinueReq
(51)
CallInformationRequest
(2, 1, { requestedInformationTypeList { releaseCause }, legID sendingSideID : '01'H }
TC_InvokeReq
(51, 2, 3, CO, cOArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
TC_ContinueReq
(51)
Connnect
(3, 1, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
1, { eventTypeBCSM oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
(1)
TC_InvokeInd
M oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
ReleaseInd
({ callRef 1, cause '00'H })
CallInformationReport
aseCause, requestedInformationValue releaseCauseValue : '00'H } }, legID sendingSideID : '01'H }, FALSE)
TC_InvokeInd
estedInformationValue releaseCauseValue : '00'H } }, legID sendingSideID : '01'H }, TRUE)

	IN3_A_BASIC_CF_BI_01
Work item no.:	ITEM_BASIC_31
IN2 Ref(tmp)	IN2_A_BASIC_CF_BI_01
Purpose:	Verify that the SSF sends a **CallInformationRequest** returnError component indicating: **requestedInfoError** to the SCF, when the **CallInformationRequest** invoke component sent by the SCF contains an invalid parameter.
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 11.8.1, 11.8.1.1.1, 11.8.3.2, 11.14, 13.1.6.1, 15.1.1.2.2
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 and "requestedInfoNotAvailable"
Postamble:	SigConA_Release


MSC IN3_A_BASIC_CF_BI_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, CIRQ, cIRQArg : { requestedInformationTypeList { calledAddress }, legID sendingSideID : '02'H }, short)
TC_ContinueReq
(51)
CallInformationRequest
(1, 1, { requestedInformationTypeList { calledAddress }, legID sendingSideID : '02'H }
TC_InvokeReq
(51, 2, 2, CUE, cUEArg : NULL, short)
TC_ContinueReq
(51)
Continnue
(2, 1)
CallInformationRequestReturnError
(1, 2, requestedInfoError : requestedInfoNotAvailable)
(FALSE, 1)
TC_ErrorInd
(51, 102, requestedInfoError : requestedInfoNotAvailable, TRUE)
TC_EndInd
(51, basic, FALSE)

	IN3_A_BASIC_CF_BO_02
Work item no.:	ITEM_BASIC_33
IN2 Ref(tmp)	IN2_A_BASIC_CF_BO_02
Purpose:	Verify that the SSF sends **CallInformationRequest** returnError component with an indication of "UnexpectedComponentSequence" to the SCF, when the **CallInformationRequest** invoke component has been received from the SCF in the **Monitoring** state.
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 11.8.1, 11.8.1.1.1, 11.8.3.2, 13.1.9.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	O_S2P
Test description	SCF sends **CallInformationRequest** invoke to SSF
Pass criteria	Check that SSF sends to SCF a **CallInformationRequest** returnError component with an indication of "UnexpectedComponentSequence".
Postamble:	SigConA_Release_thenB


MSC IN3_A_BASIC_CF_BO_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_S2P
TC_InvokeReq
(51, 2, 1, CIRQ, cIRQArg : { requestedInformationTypeList { callAttemptElapsedTime }, legID sendingSideID : '02'H }, short
TC_ContinueReq
(51)
CallInformationRequest
(1, 1, { requestedInformationTypeList { callAttemptElapsedTime }, legID sendingSide
CallInformationRequestReturnError
(1, 1, unexpectedComponentSequence, NULL)
(FALSE, 1)
TC_ErrorInd
(51, 102, unexpectedComponentSequence : NULL, TRUE)
TC_EndInd
(51, basic, FALSE)

6.6.5 Cancel (CA) procedure

NOTE: Other Test Purposes related to the Cancel operation can be found in EN 301 933-3 [6].

	IN3_A_BASIC_CA_BV_01
Work item no.:	ITEM_BASIC_34
IN2 Ref(tmp)	IN2_A_BASIC_CA_CA_01
Purpose:	Verify that the SSF returns to idle state when it receives from the SCF a **Cancel** invoke component indicating **allRequests**.
Requirements refs	8.2.1.2, 8.2.2 (table 21), 8.3.3, 8.4.3, 11.2.1, 11.2.3.1, 11.9.1, 11.9.1.1.1, 11.9.3.1
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 sends a TC-P-Abort as the dialogue is not used any more
Postamble:	SigConA_Release_thenB

	IN3_A_BASIC_CA_BV_02
Work item no.:	ITEM_BASIC_35
IN2 Ref(tmp)	IN2_A_BASIC_CA_BV_02
Purpose:	Verify that the SSF cancels an **RequestNotificationChargingEvent** invoked by the SCF, when it receives from the SCF a **Cancel** invoke component indicating **allRequests.**
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2 (table 21), 8.3.3, 8.4.3, 11.9.1, 11.9.1.1.1, 11.9.3.1, 11.37.1, 11.37.1.1.1, 11.37.3.1
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

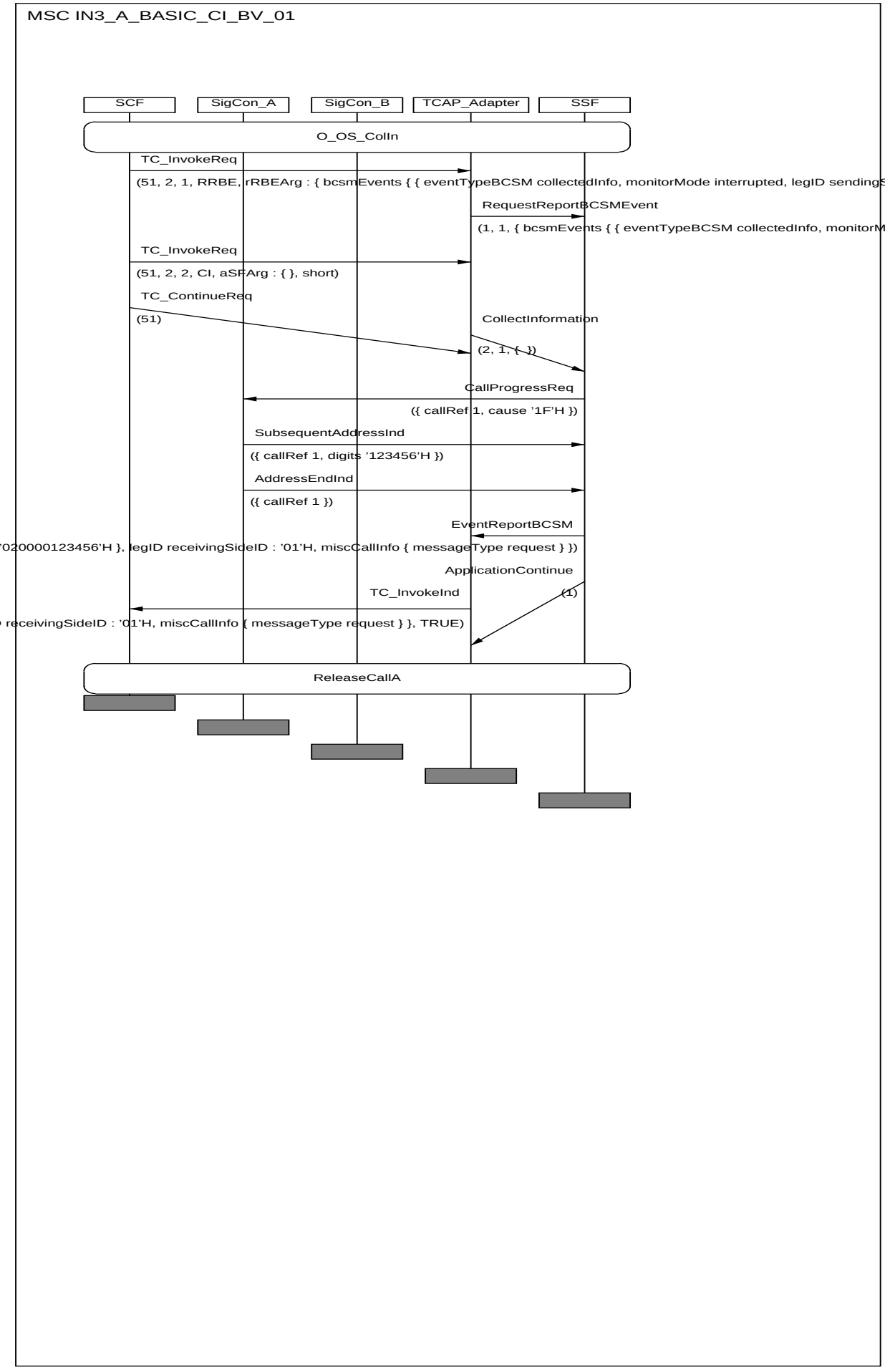
	IN3_A_BASIC_CA_BV_03
Work item no.:	ITEM_BASIC_36
IN2 Ref(tmp)	IN2_A_BASIC_CA_BV_03
Purpose:	Verify that the SSF cancels an **ApplyCharging** invoked by the SCF, when it receives from the SCF a **Cancel** invoke component indicating **allRequests.**
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2 (table 21), 8.3.2, 8.3.3, 8.4.2, 8.4.3, 11.3.1, 11.3.1.1.1, 11.3.3.1, 11.9.1, 11.9.1.1.1, 11.9.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **ApplyCharging** invoke containing mandatory parameters only 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

	IN3_A_BASIC_CA_BV_04
Work item no.:	ITEM_BASIC_37
IN2 Ref(tmp)	IN2_A_BASIC_CA_BV_04
Purpose:	Verify that the SSF cancels a **CallInformationRequest** invoked by the SCF, when it receives from the SCF a **Cancel** invoke component indicating **allRequests.**
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2 (table 21), 8.3.3, 8.4.3, 11.8.1, 11.8.1.1.1, 11.8.3.1, 11.9.1, 11.9.1.1.1, 11.9.3.1
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

	IN3_A_BASIC_CA_BI_01
Work item no.:	ITEM_BASIC_38
IN2 Ref(tmp)	IN2_A_BASIC_CA_BI_01
Purpose:	Verify that the SSF sends to the SCF a **Cancel** returnError component indicating **cancelFailed**, when it receives from the SCF a **Cancel** invoke component containing an invokeID not being the InvokeID of the existing operation.
Requirements refs	8.2.1.2, 8.2.2 (table 21), 8.3.3, 8.4.3, 11.9.1, 11.9.1.1.1, 11.9.3.2, 13.1.1.1.2, 15.1.1.2.2
Selection Cond.	
Preamble:	O_OS
Test description	**Cancel** invoke is sent by SCF to SSF, containing invokeID being not existing operation InvokeID
Pass criteria	- Check that SSF sends to SCF **Cancel** with error cancelFailed
Postamble:	SigConA_Release

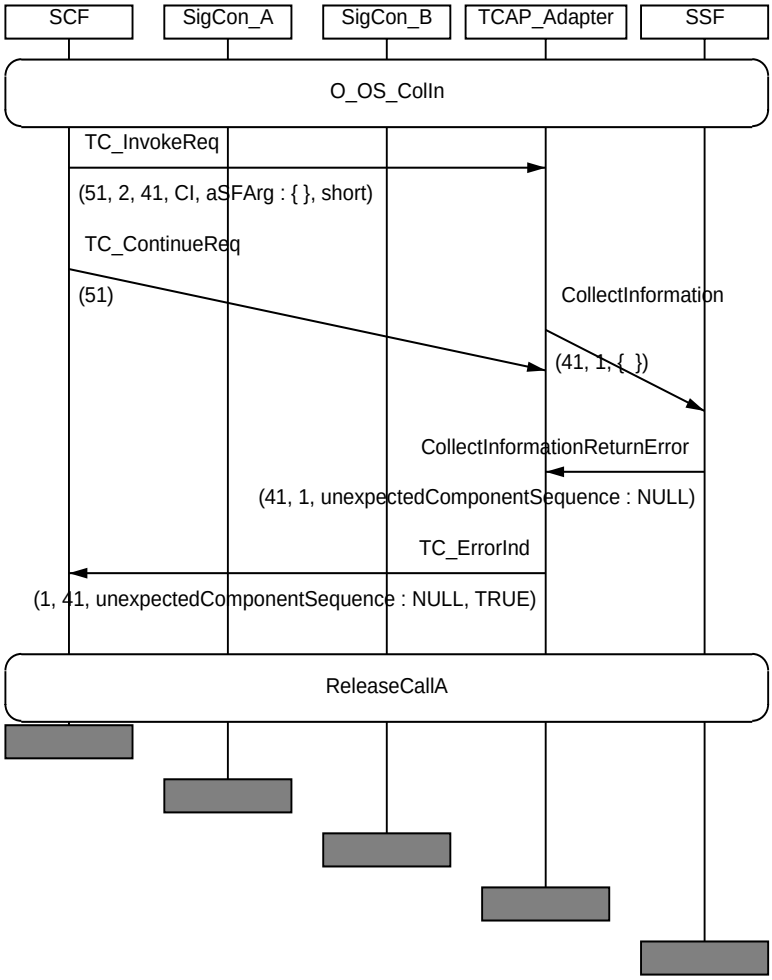
6.6.6 CollectInformation (CI) procedure

	IN3_A_BASIC_CI_BV_01
Work item no.:	ITEM_BASIC_40
IN2 Ref(tmp)	IN2_A_BASIC_CI_CA_01
Purpose:	Verify that the SSF, while progressing a call and after having received a **RequestReportBCSMEvent** invoke component containing parameters eventTypeBCSM=**collectedInfo** and monitorMode=**interrupted**, followed by a **CollectInformation** invoke component, sends to the SCF an **EventReportBCSM** invoke component with the indication of eventTypeBCSM=**collectedInfo**, together with the requested information, when this is available from the calling party.
Requirements refs	6.6.3.4.2.2, 6.6.3.5.2, 8.2, 8.2.1.2, 8.2.2, 11.11.1, 11.11.3.1, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS_ColIn Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=collectedInfo; leg1 - monitorMode=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 IN3_A_BASIC_CI_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS_ColIn
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM collectedInfo, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM collectedInfo, monitorMode interrupted, legI
TC_InvokeReq
(51, 2, 2, CI, aSFArg : { }, short)
TC_ContinueReq
(51)
CollectInformation
(2, 1, { })
CallProgressReq
({ callRef 1, cause '1F'H })
SubsequentAddressInd
({ callRef 1, digits '123456'H })
AddressEndInd
({ callRef 1 })
EventReportBCSM
alledPartynumber '020000123456'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
(1)
TC_InvokeInd
0123456'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)
ReleaseCallA

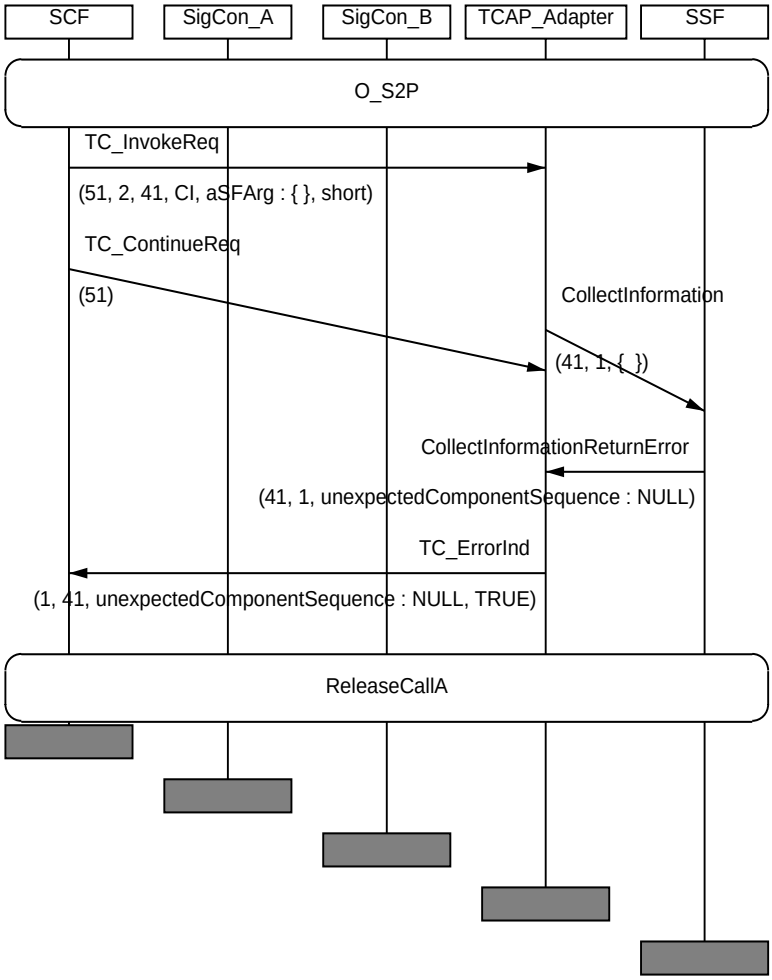
	IN3_A_BASIC_CI_BO_01
Work item no.:	ITEM_BASIC_42
IN2 Ref(tmp)	IN2_A_BASIC_CI_BO_01
Purpose:	Verify that the SSF sends to the SCF a **CollectInformation** returnError component indicating **unexpectedComponentSequence**, when it receives from the SCF a **CollectInformation** invoke component without the DP **Collected_Information** being armed.
Requirements refs	6.6.3.5.2, 8.2, 8.2.1.2, 8.2.2, 11.11.1, 11.11.3.2, 13.1.9.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	O_OS_ColIn Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF sends **CollectInformation** invoke to SSF without sending before any **RequestReportBCSMEvent** invoke
Pass criteria	Check that SSF sends to SCF a **CollectInformation** error with an indication of UnexpectedComponentSequence
Postamble:	ReleaseCallA

MSC IN3_A_BASIC_CI_BO_01



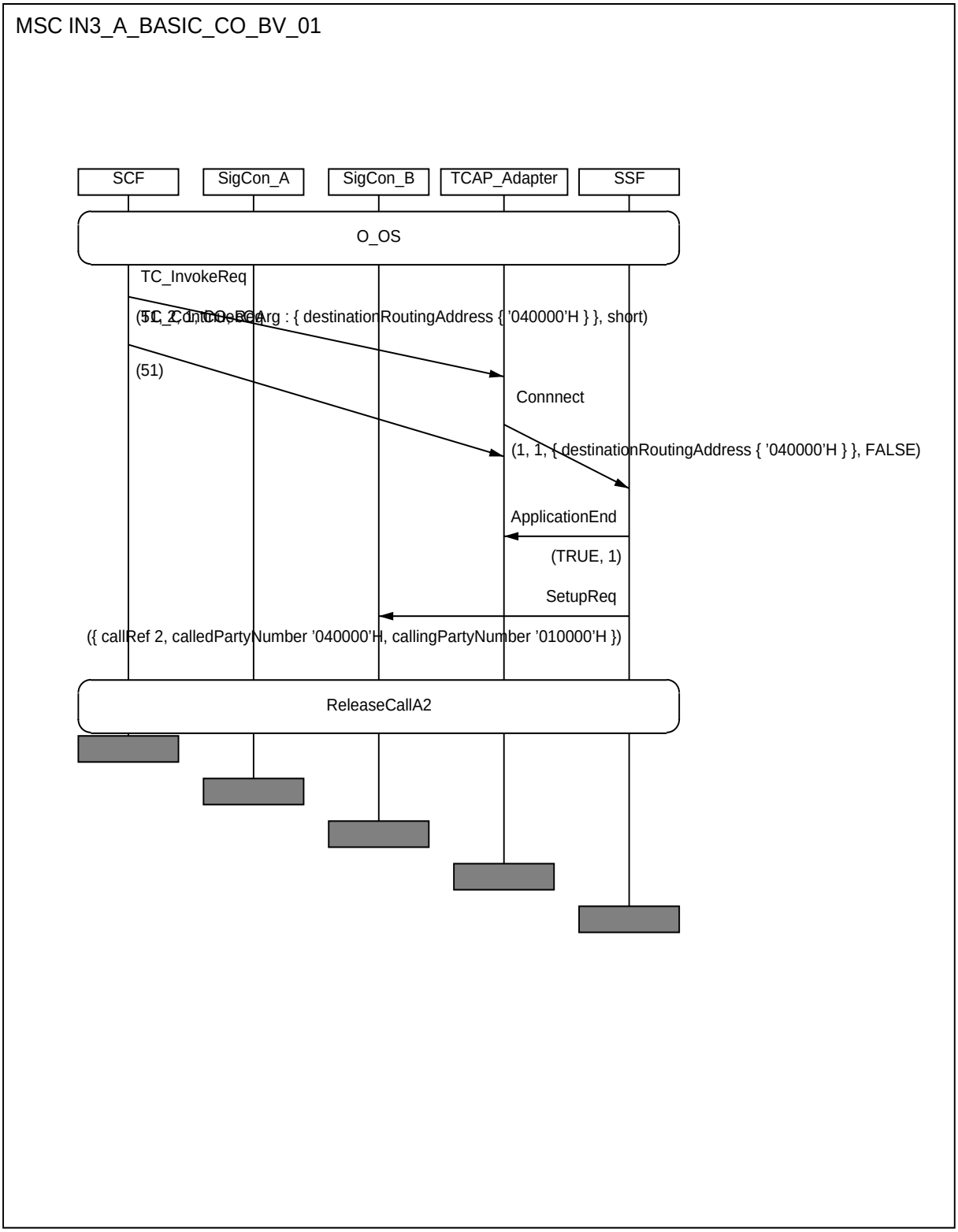
	IN3_A_BASIC_CI_BO_03
Work item no.:	ITEM_BASIC_44
IN2 Ref(tmp)	IN2_A_BASIC_CI_BO_03
Purpose:	Verify that the SSF sends to the SCF a **CollectInformation returnE**rror, when it receives from the SCF a **CollectInformation** invoke component in the **Monitoring** state.
Requirements refs	6.6.3.5.2, 8.2, 8.2.1.2, 8.2.2, 11.11.1, 11.11.3.2, 13.1.9.2.2, 15.1.1.2.2
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

MSC IN3_A_BASIC_CI_BO_03

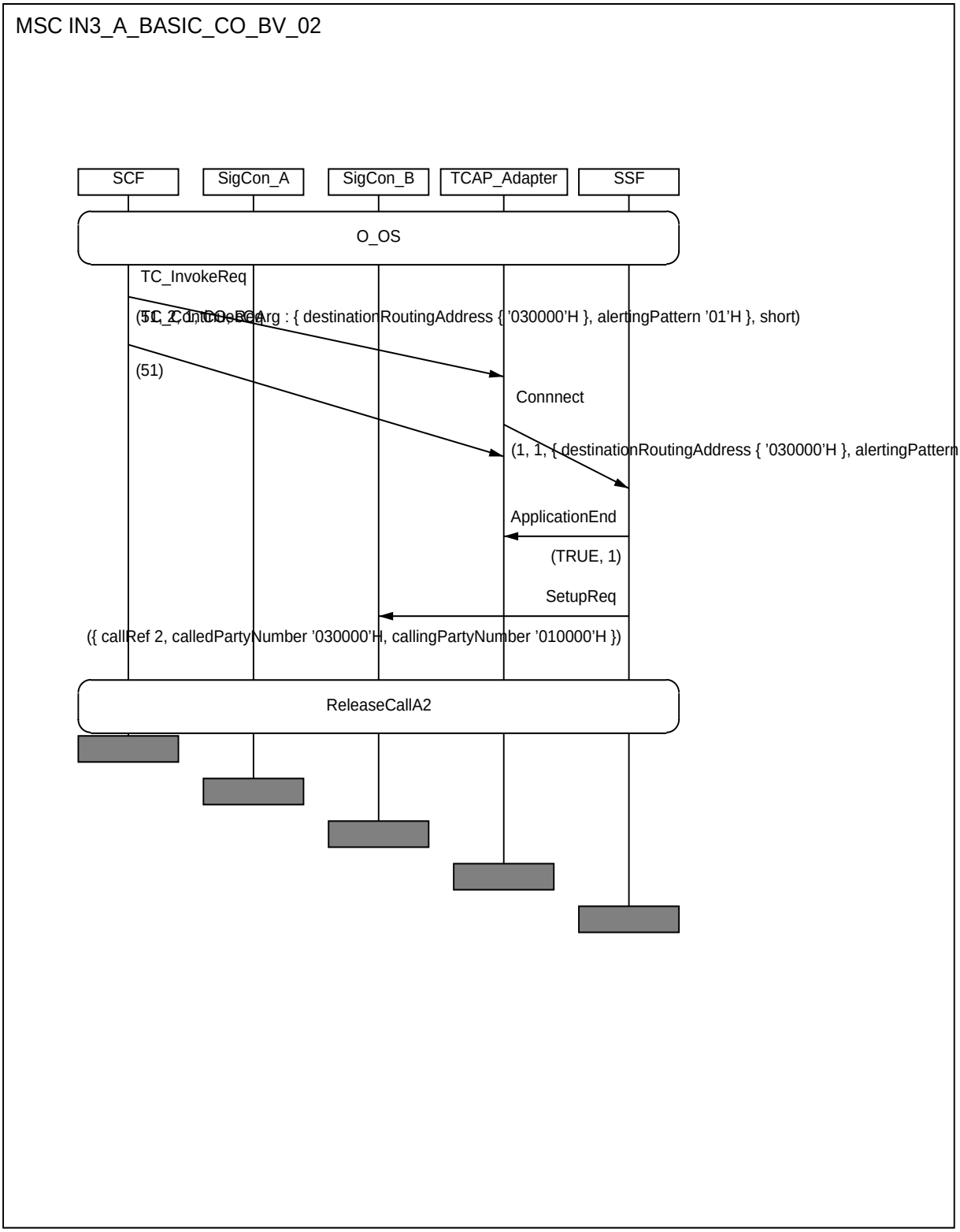


6.6.7 Connect (CO) procedure

	IN3_A_BASIC_CO_BV_01
Work item no.:	ITEM_BASIC_45
IN2 Ref(tmp)	IN2_A_BASIC_CO_CA_01
Purpose:	Verify that the SSF sends a SetupRequest to the B side and maps the destinationRoutingAddress into this **Setup** request, when it receives from the SCF a **Connect** invoke component containing only the mandatory parameter **destinationRoutingAddress**, having any valid value.
Requirements refs	8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends a **Connect** invoke to SSF with mandatory parameters only, with destinationRoutingAddress contained by default
Pass criteria	Check that the relevant parameters are mapped from Connect into the **Setup** request
Postamble:	ReleaseCallA2

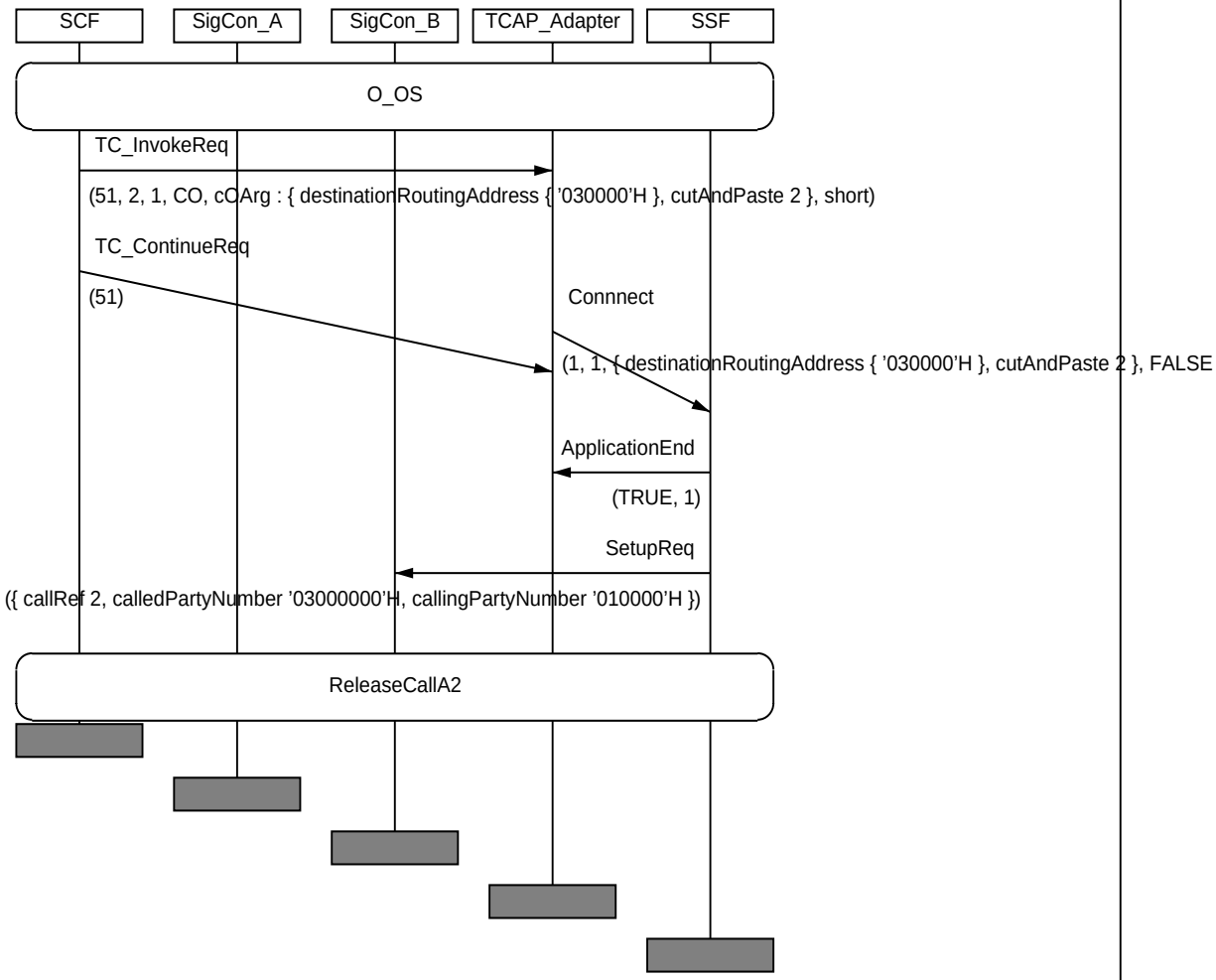

MSC IN3_A_BASIC_CO_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, InitCCReq, { destinationRoutingAddress { '040000'H } }, short)
(51)
Connnect
(1, 1, { destinationRoutingAddress { '040000'H } }, FALSE)
ApplicationEnd
(TRUE, 1)
SetupReq
({ callRef 2, calledPartyNumber '040000'H, callingPartyNumber '010000'H })
ReleaseCallA2

	IN3_A_BASIC_CO_BV_02
Work item no.:	ITEM_BASIC_46
IN2 Ref(tmp)	IN2_A_BASIC_CO_BV_01
Purpose:	Verify that the SSF does not reject a **Connect** invoke component received from the SCF, containing only mandatory parameter **destinationRoutingAddress** and optional parameter **alertingPattern**, and sends a SetupRequest to the B side.
Requirements refs	8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Connect** invoke with mandatory and optional parameters, with destinationRoutingAddress contained by default alertingPattern SSF sends a SetupRequest to B side
Pass criteria	Check that the connect operation is not rejected and that the relevant parameters are mapped from Connect into the **Setup** request : destinationRoutingAddress----------> calledPartyNumber and check the special tone indicated in **alertingPattern**
Postamble:	ReleaseCallA2

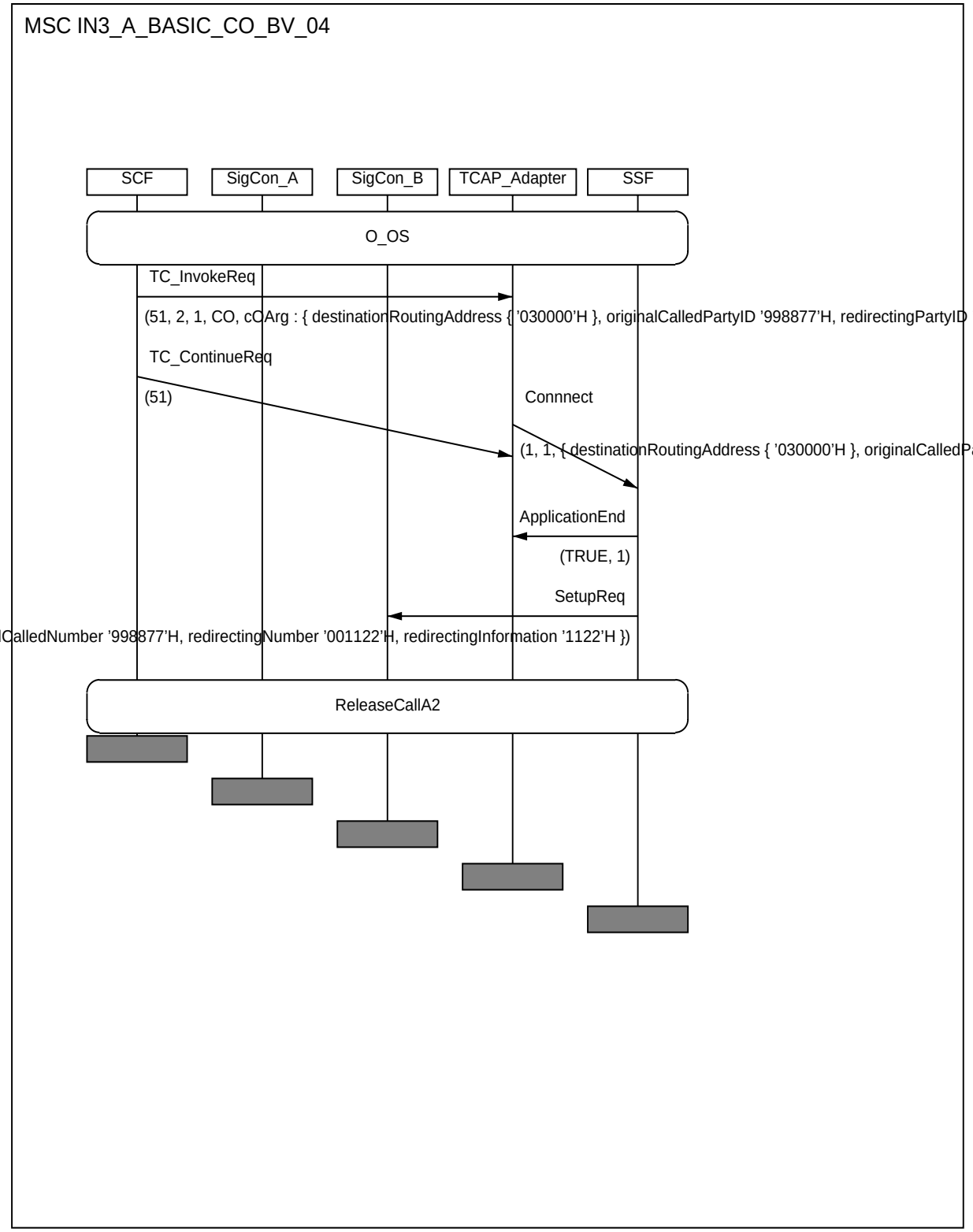

MSC IN3_A_BASIC_CO_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, InitCOeReq
TC_InvokeReq
Arg : { destinationRoutingAddress { '030000'H }, alertingPattern '01'H }, short)
(51)
Connnect
(1, 1, { destinationRoutingAddress { '030000'H }, alertingPattern '01'H }, FALSE
ApplicationEnd
(TRUE, 1)
SetupReq
({ callRef 2, calledPartyNumber '030000'H, callingPartyNumber '010000'H })
ReleaseCallA2

	IN3_A_BASIC_CO_BV_03
Work item no.:	ITEM_BASIC_47
IN2 Ref(tmp)	IN2_A_BASIC_CO_BV_02
Purpose:	Verify that the SSF does not reject a **Connect** invoke component received from the SCF, containing only mandatory parameter **destinationRoutingAddress** and optional parameter **cutAndPaste**, and sends a SetupRequest to the B side.
Requirements refs	8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.1
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 and that the relevant parameters are mapped from Connect into the **Setup** request : destinationRoutingAddress+remainig digits of the former cldPartyNumber --→ cldPartyNumber
Postamble:	ReleaseCallA2

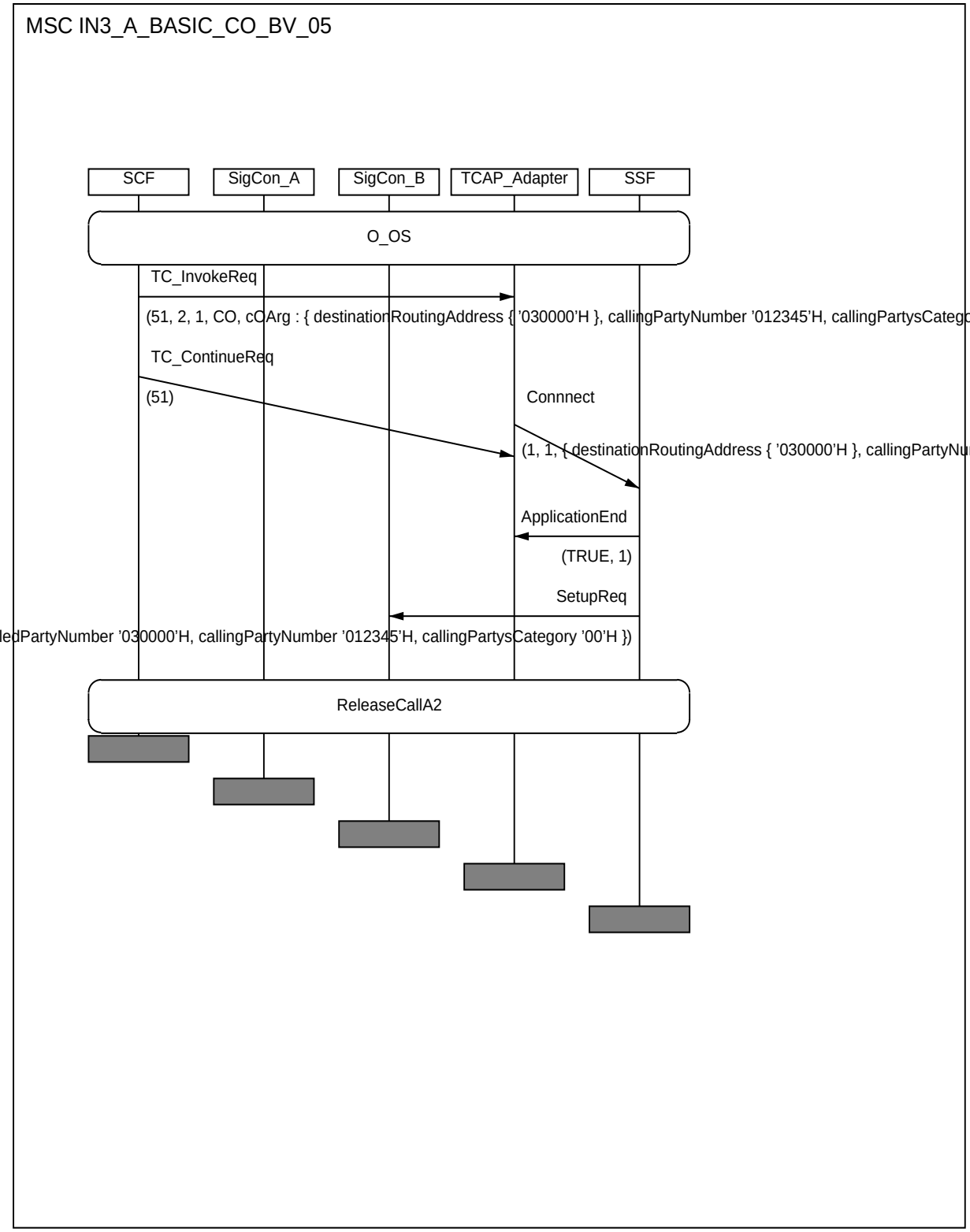
MSC IN3_A_BASIC_CO_BV_03



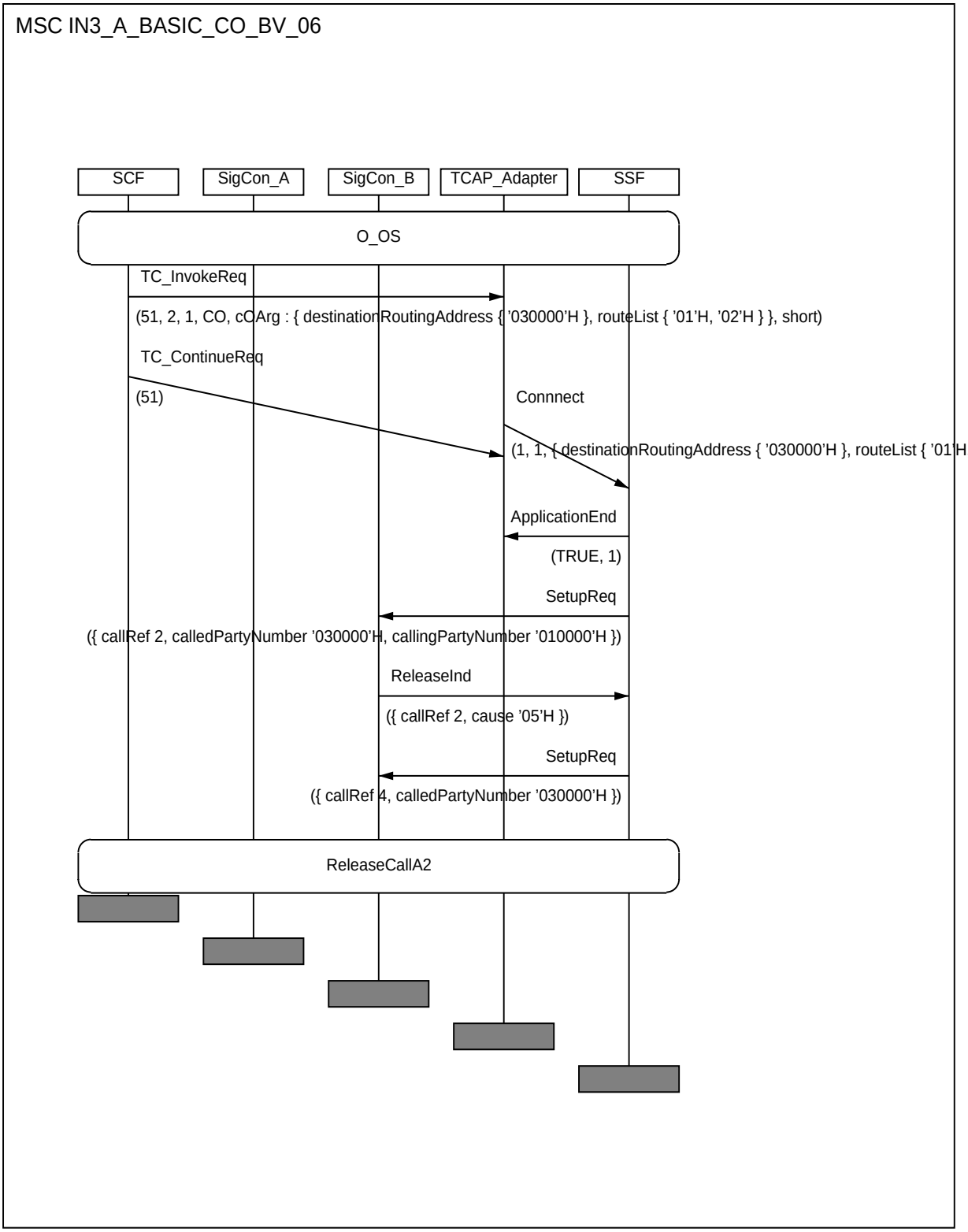
	IN3_A_BASIC_CO_BV_04
Work item no.:	ITEM_BASIC_48
IN2 Ref(tmp)	IN2_A_BASIC_CO_BV_03
Purpose:	Verify that the SSF accepts a **Connect** invoke component received from the SCF, containing mandatory parameter **destinationRoutingAddress** and optional parameters **originalCalledPartyID**, **redirectingPartyID** and **redirectionInformation**, and sends a SetupRequest to the B side. Check also that these parameters are mapped to **Setup request** parameters **calledPartyNumber**, **originalCalledNumber**, **redirectingNumber** and **redirectionInformation** respectively.
Requirements refs	8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Connect** invoke containing parameters related to supplementary services, with: - destinationRoutingAddress, - originalCalledPartyID, - redirectingPartyID, - redirectionInformation SSF sends a SetupRequest to B side
Pass criteria	Check that the above parameters are mapped from Connect into the Set-up request: - destinationRoutingAddress ----------> calledPartyNumber - originalCalledPartyID----------> originalCalledNumber - redirectingPartyID----------> redirectingNumber - redirectionInformation----------> redirectionInformation
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CO_BV_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, CO, cOArg : { destinationRoutingAddress { '030000'H }, originalCalledPartyID '998877'H, redirectingPartyID '001122'H, redirectio
TC_ContinueReq
(51)
Connnect
(1, 1, { destinationRoutingAddress { '030000'H }, originalCalledPartyID '998877'H, re
ApplicationEnd
(TRUE, 1)
SetupReq
'010000'H, originalCalledNumber '998877'H, redirectingNumber '001122'H, redirectingInformation '1122'H })
ReleaseCallA2

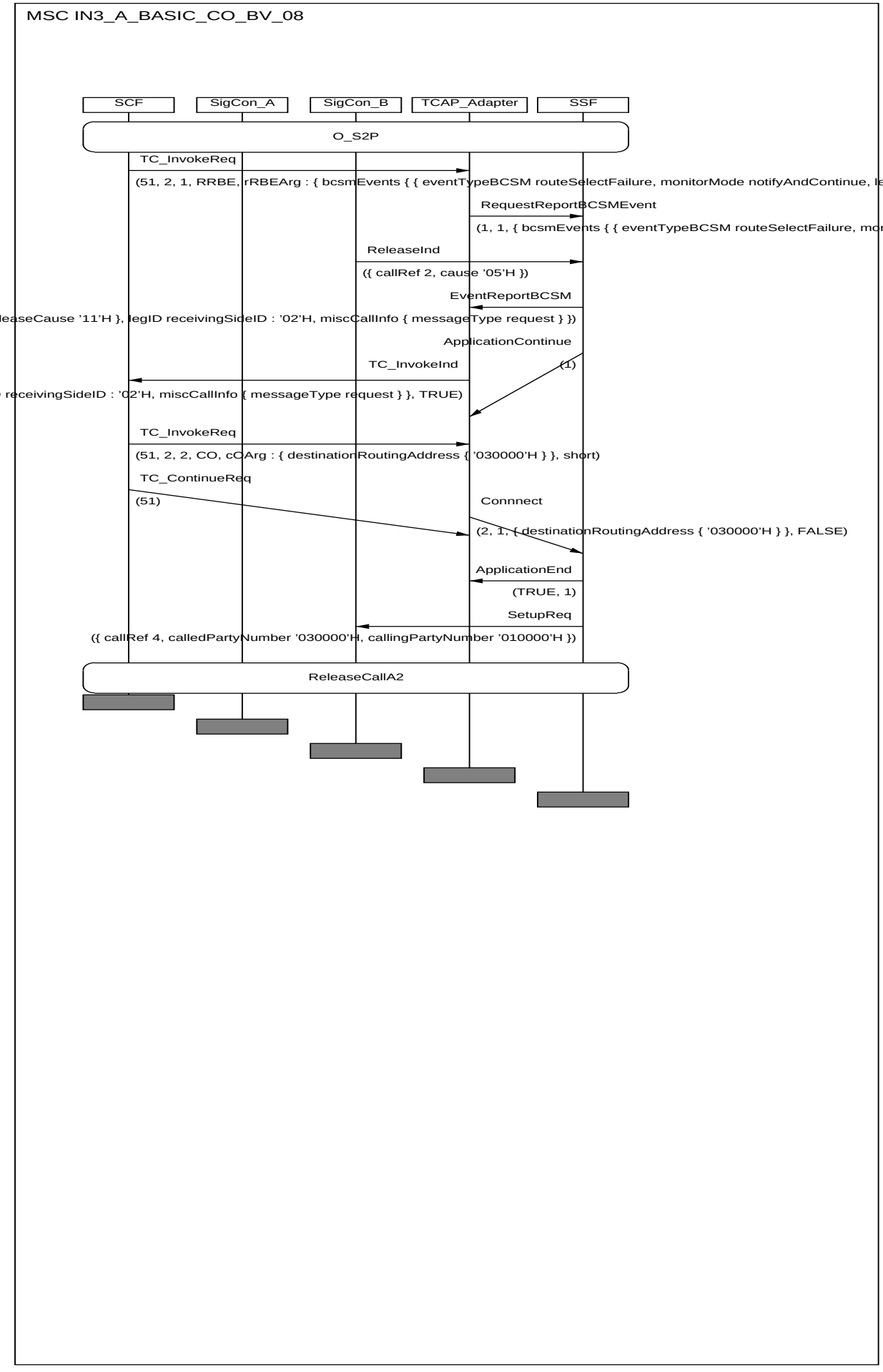
	IN3_A_BASIC_CO_BV_05
Work item no.:	ITEM_BASIC_49
IN2 Ref(tmp)	IN2_A_BASIC_CO_BV_04
Purpose:	Verify that the SSF accepts a **Connect** invoke component received from the SCF, containing mandatory parameter **destinationRoutingAddress** and optional parameters **callingPartyNumber** and **callingPartysCategory**, and sends a SetupRequest to the B side. Check also that these parameters are mapped to **Setup request** parameters **calledPartyNumber**, **callingPartyNumber** and **callingPartysCategory** respectively.
Requirements refs	8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Connect** invoke containing mandatory and optional parameters - destinationRoutingAddress, - callingPartyNumber, - callingPartysCategory SSF sends a SetupRequest to B side
Pass criteria	Check that the above parameters are mapped from Connect into the Set-up request: - destinationRoutingAddress----------> calledPartyNumber - callingPartyNumber----------> callingPartyNumber - callingPartysCategory----------> callingPartysCategory
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CO_BV_05
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, CO, cOArg : { destinationRoutingAddress { '030000'H }, callingPartyNumber '012345'H, callingPartysCategory '00'H }, short
TC_ContinueReq
(51)
Connnect
(1, 1, { destinationRoutingAddress { '030000'H }, callingPartyNumber '012345'H, calli
ApplicationEnd
(TRUE, 1)
SetupReq
{ callRef 2, calledPartyNumber '030000'H, callingPartyNumber '012345'H, callingPartysCategory '00'H })
ReleaseCallA2

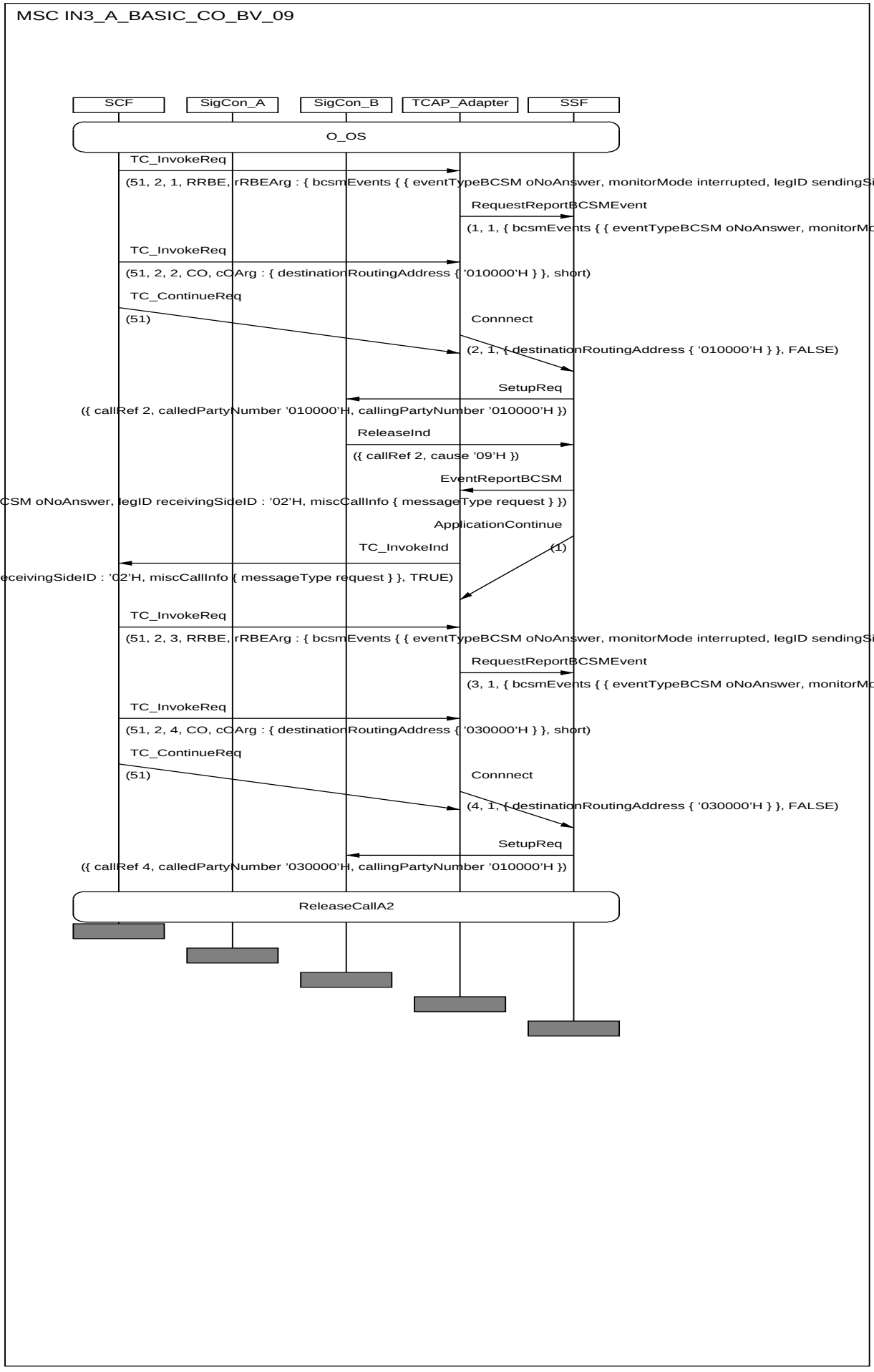
	IN3_A_BASIC_CO_BV_06
Work item no.:	ITEM_BASIC_50
IN2 Ref(tmp)	IN2_A_BASIC_CO_BV_05
Purpose:	Verify that the SSF accepts a **Connect** invoke component received from the SCF, containing mandatory parameter **destinationRoutingAddress** and optional parameter **routeList** (with two different routes), and sends a SetupRequest for the first route to the B side. Check also, when the first route fails, that the IUT sends a second SetUpReq to B side using the second route.
Requirements refs	8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Connect** invoke containing mandatory and optional parameters - destinationRoutingAddress, - routeList (with two different routes) SSF sends a SetupRequest to B side B Side sends a RelInd with release cause being routeSelectFailure
Pass criteria	Check that if the first route fails the IUT sends a second SetUpReq to B side using the second route.
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CO_BV_06
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, CO, cCArg : { destinationRoutingAddress { '030000'H }, routeList { '01'H, '02'H } }, short)
TC_ContinueReq
(51)
Connnect
(1, 1, { destinationRoutingAddress { '030000'H }, routeList { '01'H, '02'H } }, FALSE
ApplicationEnd
(TRUE, 1)
SetupReq
({ callRef 2, calledPartyNumber '030000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '05'H })
SetupReq
({ callRef 4, calledPartyNumber '030000'H })
ReleaseCallA2

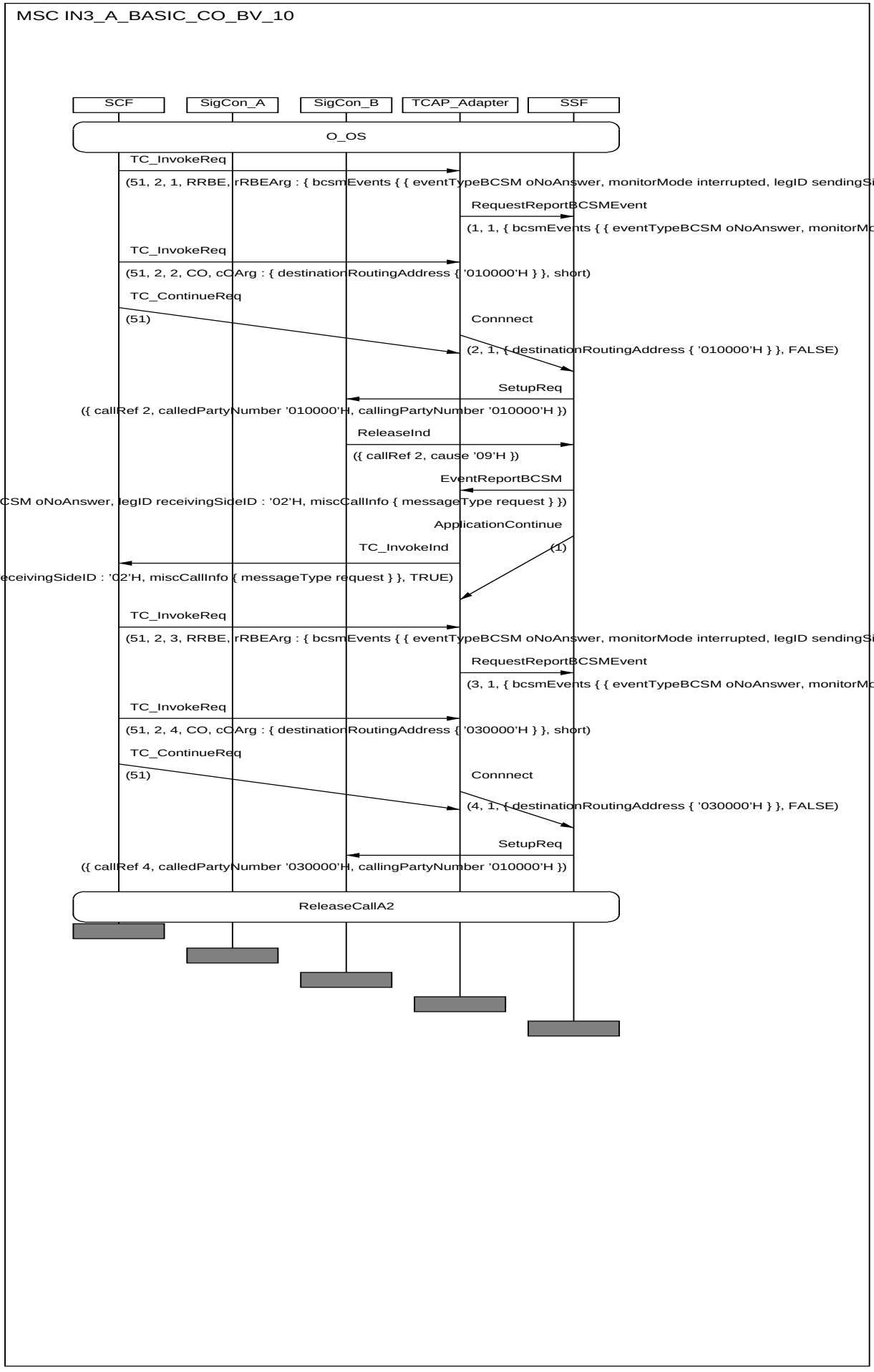
	IN3_A_BASIC_CO_BV_08
Work item no.:	ITEM_BASIC_52
IN2 Ref(tmp)	IN2_A_BASIC_CO_BV_07
Purpose:	Verify that the SSF, having armed the **Route_Select_Failure** DP and receiving ReleaseInd with release cause being **routeSelectFailure**, accepts a **Connect** invoke component from the SCF, containing mandatory parameter **destinationRoutingAddress**, and sends a SetupRequest to the B side containing the new destinationRoutingAddress as calledPartyNumber.
Requirements refs	6.4.2.1.7, 6.4.2.2.4, 6.6.3.3.2.1, 8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.1, 11.24.1, 11.24.1.1.1, 11.24.2.1
Selection Cond.	
Preamble:	O_S2P
Test description	The SCF sends a **RequestReportBCSM** operation to arm routeSelectFailure event SigConB sends a ReleaseInd with release cause being routeSelectFailure ("05"H) => ERB(oDisc) SCF sends to SSF **Connect** invoke containing mandatory parameter - destinationRoutingAddress
Pass criteria	Check that SSF sends a SetupReq to B side and that the relevant parameters are mapped from Connect into the **Setup** request : destinationRoutingAddress----------> calledPartyNumber
Postamble:	ReleaseCallAB_cause_0F


MSC IN3_A_BASIC_CO_BV_08
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_S2P
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode notifyAndContinue, legID sendingSideID :
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode notifyAndC
ReleaseInd
({ callRef 2, cause '05'H })
EventReportBCSM
tSpecificInfo : { releaseCause '11'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
(1)
TC_InvokeInd
ause '11'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
TC_InvokeReq
(51, 2, 2, CO, cCArg : { destinationRoutingAddress { '030000'H } }, short)
TC_ContinueReq
(51)
Connnect
(2, 1, { destinationRoutingAddress { '030000'H } }, FALSE)
ApplicationEnd
(TRUE, 1)
SetupReq
({ callRef 4, calledPartyNumber '030000'H, callingPartyNumber '010000'H })
ReleaseCallA2

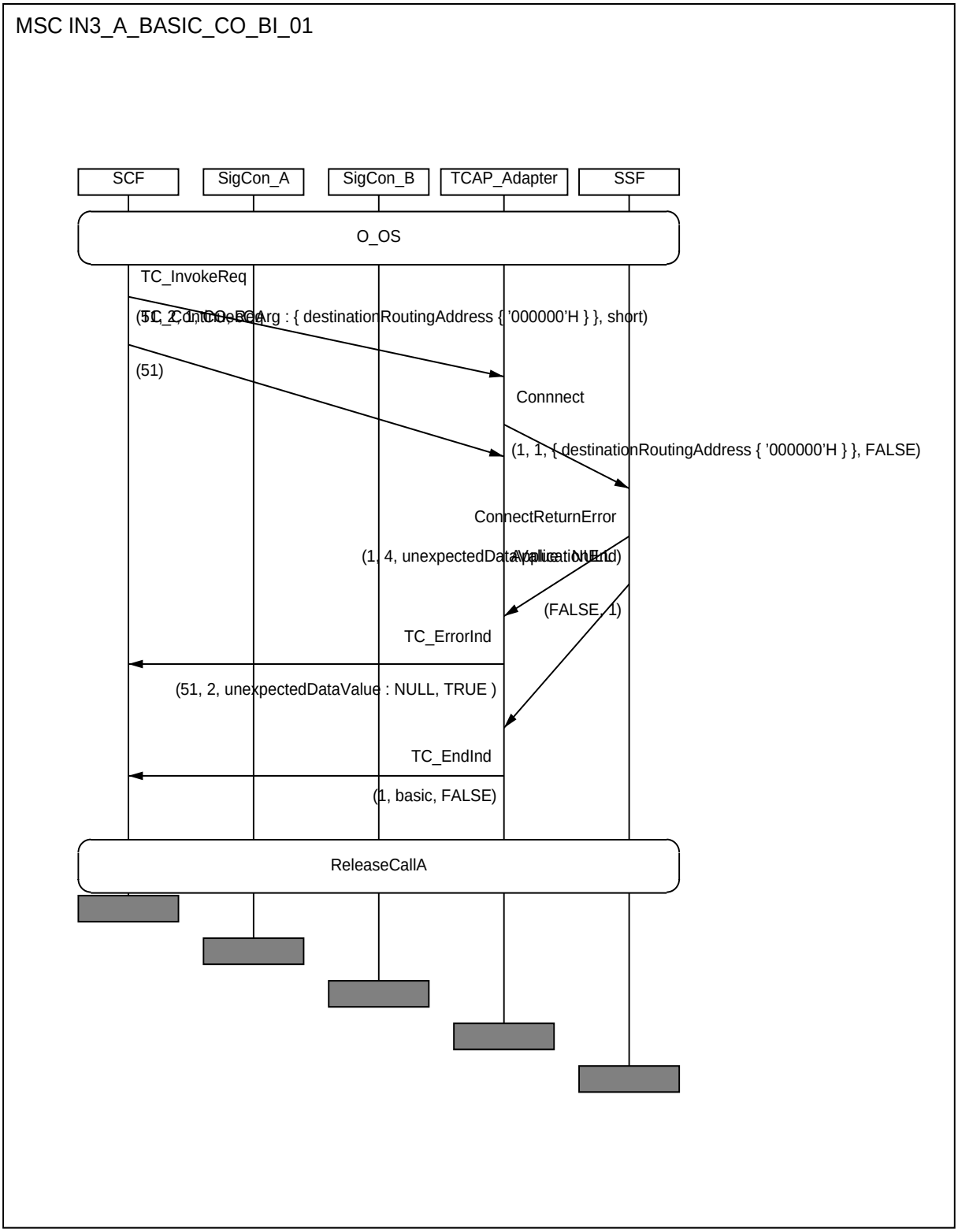
	IN3_A_BASIC_CO_BV_09
Work item no.:	ITEM_BASIC_53
IN2 Ref(tmp)	IN2_A_BASIC_CO_BV_08
Purpose:	Verify that the SSF, having armed the **O_No_Answer** DP and receiving ReleaseInd with release cause being **bPtyNoAnswer**, accepts a **Connect** invoke component from the SCF, containing mandatory parameter **destinationRoutingAddress**, and sends a SetupRequest to the B side containing the new destinationRoutingAddress as calledPartyNumber.
Requirements refs	6.4.2.1.7, 6.4.2.2.4, 6.6.3.3.2.1, 8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.1, 11.24.1, 11.24.1.1.1, 11.24.2.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends a **RequestReportBCSMEvent** operation to arm oNoAnswer event SCF sends a Connect operation with mandatory parameters => ?SetupReq SigConB sends a ReleaseInd with release cause being bPtyNoAnswer "09"H SSF sends an EventReportBCSM operation with at least parameter :eventTypeBCSM = oNoAnswerSCF sends a **RequestReportBCSMEvent** operation to arm oNoAnswer event Scs SCF sends to SSF **Connect** invoke containing mandatory parameter - destinationRoutingAddress,
Pass criteria	Check that SSF sends a SetupReq to B side and that the relevant parameters are mapped from Connect into the **Setup** request : destinationRoutingAddress----------> calledPartyNumber
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CO_BV_09
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, sho
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID
TC_InvokeReq
(51, 2, 2, CO, cOArg : { destinationRoutingAddress { '010000'H } }, short)
TC_ContinueReq
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H } }, FALSE)
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '09'H })
EventReportBCSM
1, { eventTypeBCSM oNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
(1)
TC_InvokeInd
oNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
TC_InvokeReq
(51, 2, 3, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, sho
RequestReportBCSMEvent
(3, 1, { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID
TC_InvokeReq
(51, 2, 4, CO, cOArg : { destinationRoutingAddress { '030000'H } }, short)
TC_ContinueReq
(51)
Connnect
(4, 1, { destinationRoutingAddress { '030000'H } }, FALSE)
SetupReq
({ callRef 4, calledPartyNumber '030000'H, callingPartyNumber '010000'H })
ReleaseCallA2

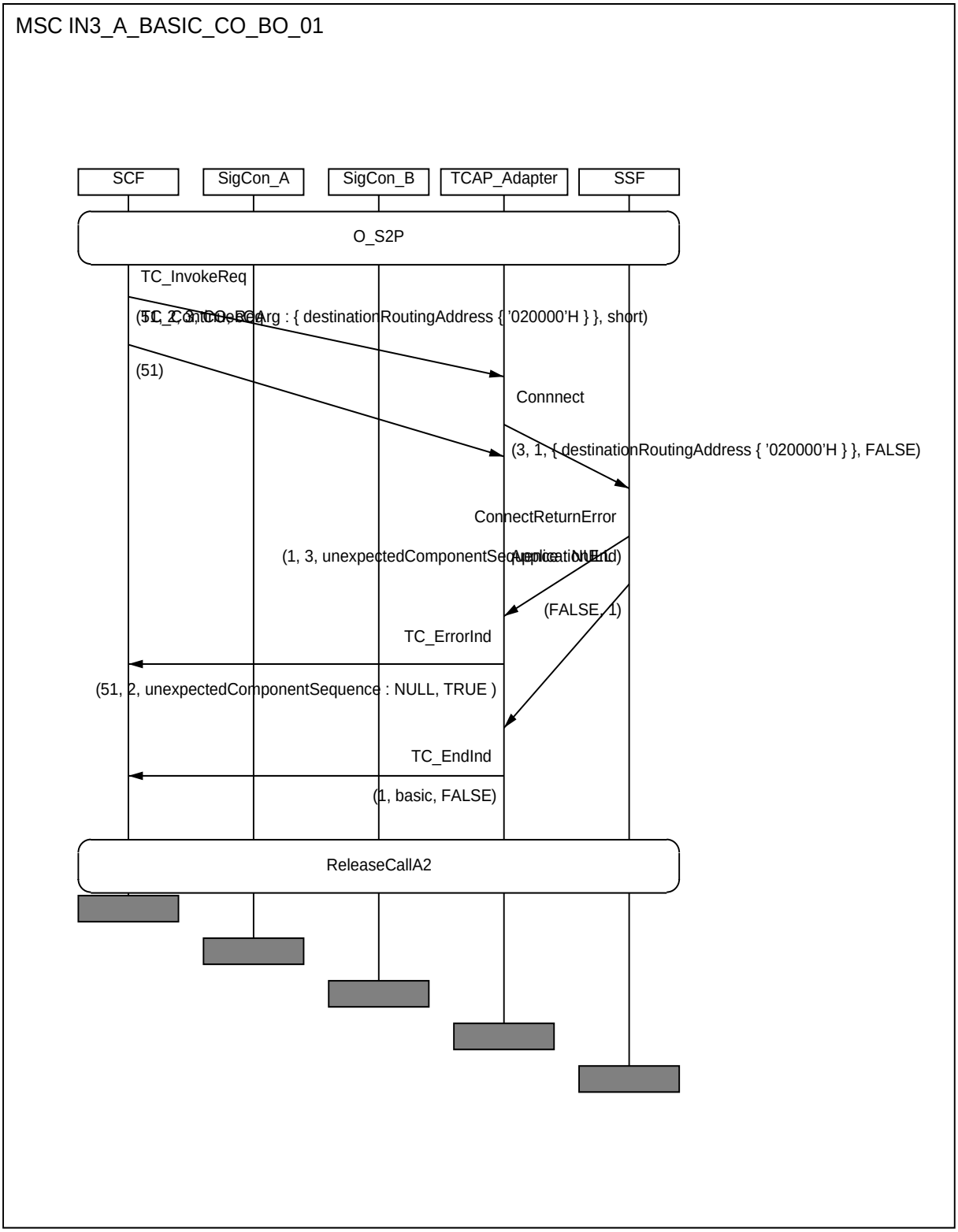
	IN3_A_BASIC_CO_BV_10
Work item no.:	ITEM_BASIC_54
IN2 Ref(tmp)	IN2_A_BASIC_CO_BV_09
Purpose:	Verify that the SSF, having armed the **O_Called_Party_Busy** DP and receiving ReleaseInd with release cause being **bPtyBusy_UDUB**, accepts a **Connect** invoke component from the SCF, containing mandatory parameter **destinationRoutingAddress**, and sends a SetupRequest to the B side containing the new destinationRoutingAddress as calledPartyNumber.
Requirements refs	6.4.2.1.7, 6.4.2.2.4, 6.6.3.3.2.1, 8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.1, 11.24.1, 11.24.1.1.1, 11.24.2.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends a **RequestReportBCSM** operation to arm oCalledPartyBusy event SC SCF sends a **Connect** operation with mandatory parameters => ?SetupReq SigConB sends a ReleaseInd with release cause being bPtyBusy_UDUB SSF sends an EventReportBCSM operation with at least parameter :eventTypeBCSM = oCalledPartyBusy SCF sends a **RequestReportBCSMEvent** operation to re-arm oCalledPartyBusy event SCF sends to SSF **Connect** invoke containing mandatory parameter - destinationRoutingAddress
Pass criteria	Check that SSF sends a SetupReq to B side and that the relevant parameters are mapped from Connect into the **Setup** request : destinationRoutingAddress-----> calledPartyNumber
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CO_BV_10
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, sho
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID
TC_InvokeReq
(51, 2, 2, CO, cOArg : { destinationRoutingAddress { '010000'H } }, short)
TC_ContinueReq
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H } }, FALSE)
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '09'H })
EventReportBCSM
1, { eventTypeBCSM oNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
(1)
TC_InvokeInd
oNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
TC_InvokeReq
(51, 2, 3, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, sho
RequestReportBCSMEvent
(3, 1, { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID
TC_InvokeReq
(51, 2, 4, CO, cOArg : { destinationRoutingAddress { '030000'H } }, short)
TC_ContinueReq
(51)
Connnect
(4, 1, { destinationRoutingAddress { '030000'H } }, FALSE)
SetupReq
({ callRef 4, calledPartyNumber '030000'H, callingPartyNumber '010000'H })
ReleaseCallA2

	IN3_A_BASIC_CO_BI_01
Work item no.:	ITEM_BASIC_55
IN2 Ref(tmp)	IN2_A_BASIC_CO_BI_01
Purpose:	Verify that the SSF sends back a **Connect** returnError component indicating **UnexpectedDataValue**, when the **Connect** invoke component received from the SCF has an invalid value for mandatory parameter **destinationRoutingAddress**.
Requirements refs	8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.2, 13.1.10.1, 15.1.1.2.2
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **Connect** invoke containing mandatory parameters with an invalid value in - destinationRoutingAddress
Pass criteria	Check that SSF sends back **Connect** error, with error parameter UnexpectedDataValue
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CO_BI_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, InitDPArg : { destinationRoutingAddress { '000000'H } }, short)
(51)
Connnect
(1, 1, { destinationRoutingAddress { '000000'H } }, FALSE)
ConnectReturnError
(1, 4, unexpectedDataValue, NULL)
(FALSE, 1)
TC_ErrorInd
(51, 2, unexpectedDataValue : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)
ReleaseCallA

	IN3_A_BASIC_CO_BO_01
Work item no.:	ITEM_BASIC_56
IN2 Ref(tmp)	IN2_A_BASIC_CO_BO_01
Purpose:	Verify that the SSF sends back a **Connect** returnError component indicating **UnexpectedComponentSequence**, when a **Connect** invoke component is received from the SCF in the **Monitoring** state.
Requirements refs	8.2.1.2, 8.2.2, 11.12.1, 11.12.1.1.1, 11.12.3.2, 13.1.9.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	O_S2P
Test description	SCF sends to SSF **Connect** invoke containing mandatory parameters, while in monitoring state
Pass criteria	Check that SSF sends back **Connect** error, with error parameter UnexpectedComponentSequence
Postamble:	ReleaseCallAB_cause_00


MSC IN3_A_BASIC_CO_BO_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_S2P
TC_InvokeReq
(51, 2, connectArg : { destinationRoutingAddress { '020000'H } }, short)
(51)
Connnect
(3, 1, { destinationRoutingAddress { '020000'H } }, FALSE)
ConnectReturnError
(1, 3, unexpectedComponentSequence : NULL)
(FALSE, 1)
TC_ErrorInd
(51, 2, unexpectedComponentSequence : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)
ReleaseCallA2

6.6.8 EntityReleased (ER) procedure

	IN3_A_BASIC_ER_BV_01
Work item no.:	ITEM_BASIC_273
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed a MergeCallSegments operation (destinationCSId = 2) and being in the "Waiting for instructions" state, sends an EntityReleased invoke component indicating cSFailure with csID=2, when the default Tssf timeout value expires. Verify also that the SSF sends a ReleaseRequest signal to the affected SigCons.
Requirements refs	8.2.2, 11.21
Selection Cond.	
Preamble:	I_S1P_S1P_S1P
Test description	L1!MergeCallSegments(3,2) L1?MergeCallSegments ReturnResult Start timer(TSSF_TIMEOUT_DEFAULT + 5 %) Wait L1?EntityReleased invoke(cSFailure, csID=2) (before timeout TSSF_TIMEOUT_DEFAULT + 5 %) CP1_2?ReleaseReq CP1_3?ReleaseReq
Pass criteria	L1?EntityReleased invoke(cSFailure, csID=2) (before timeout TSSF_TIMEOUT_DEFAULT + 5 %) CP1_3?ReleaseReq CP1_4?ReleaseReq NOTE: The 3 events may appear in any order.
Postamble:	ReleaseICA

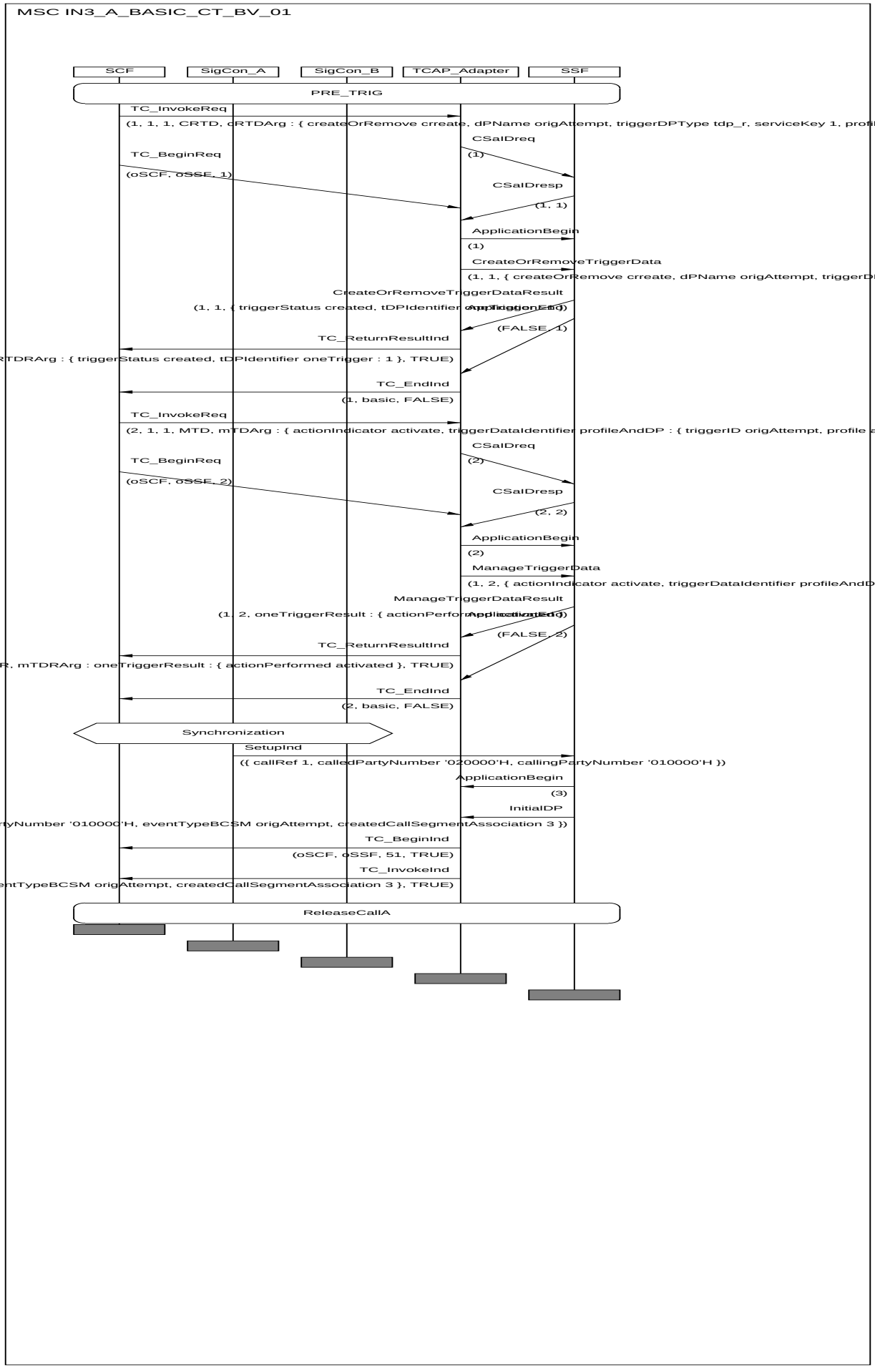
	IN3_A_BASIC_ER_BV_02
Work item no.:	ITEM_BASIC_274
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having suspended call processing at DP O_Answer when connecting a new leg (legID=3) to the controlling leg in the second CS, sends an EntityReleased invoke component after receiving a SelectFacility invoke component indicating legID=3. Verify also that the SSF sends a ReleaseRequest signal to the affected SigCon.
Requirements refs	8.2.2, 11.21
Selection Cond.	
Preamble:	O_S2P
Test description	L1!SplitLeg(1,2) L1?SplitLegReturnResult L1!RequestReportBCSMEvent(3,notifyAndContinue,oAnswer) L1!RequestReportBCSMEvent(3,notifyAndContinue,oDisconnect) L1!ContinueWithArgument (csID = 1) L1!Connect(3,2) CP1_3?SetUpReq CP1_3!SetUpConf L1?EventReportBCSM(3,oAnswer) L1!SelectFacility invoke(legID = 3) L1?EntityReleased invoke(bCSMFailure, legID=3) CP1_3?ReleaseReq
Pass criteria	L1?EntityReleased invoke(bCSMFailure, legID=3) CP1_3?ReleaseReq NOTE: These events may appear in any order.
Postamble:	ReleaseCallA2

6.6.9 CreateOrRemoveTriggerData (CT) procedure

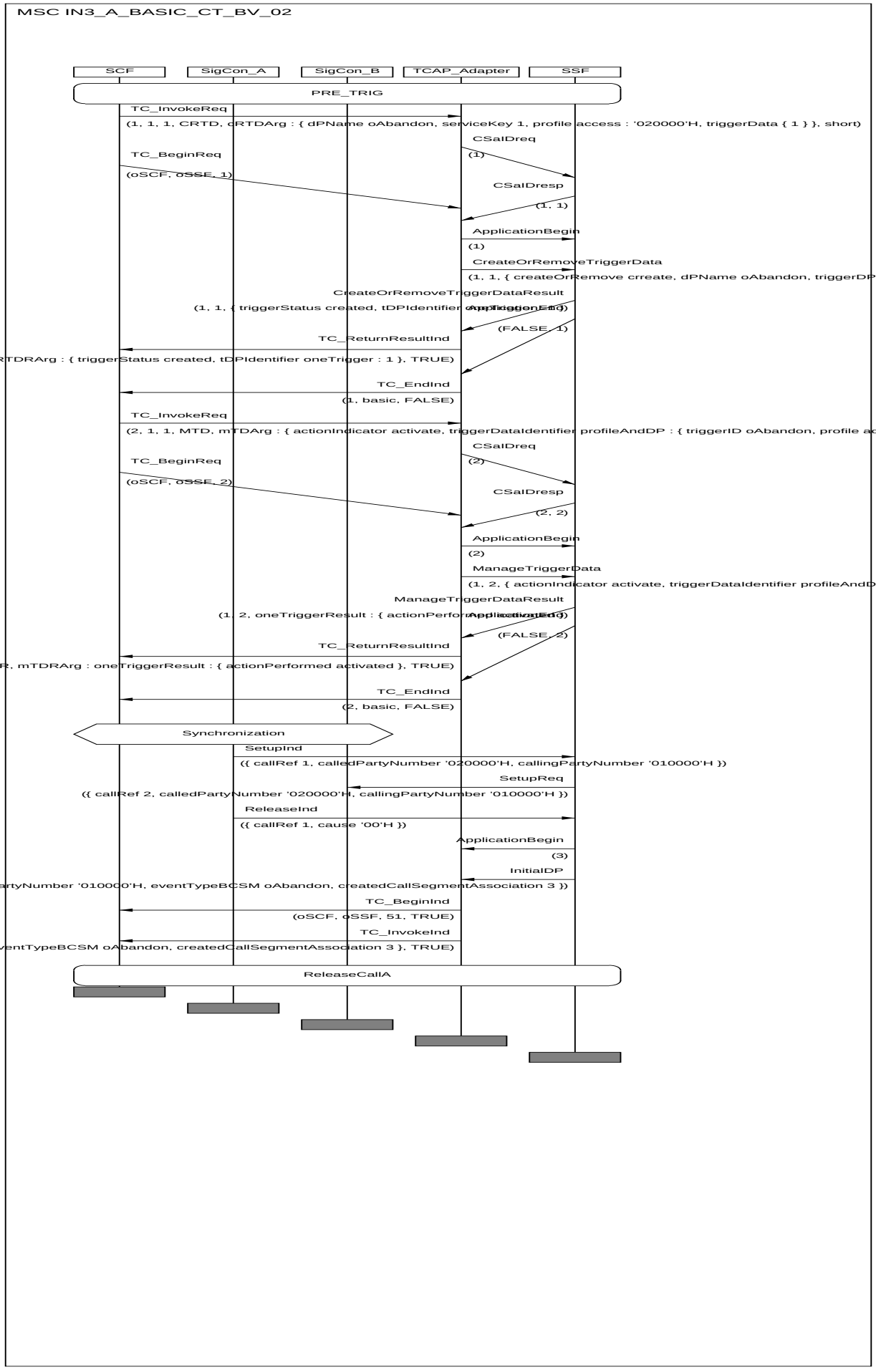
In the TPs of this group, "PCO" L1 is normally used for the first operation used inside a call context, while L2, L3 etc. are normally (except when 2 or more InitialDP operations occur) used for the procedures used outside a call context

(CreateOrRemoveTriggerData and ManageTriggerData).

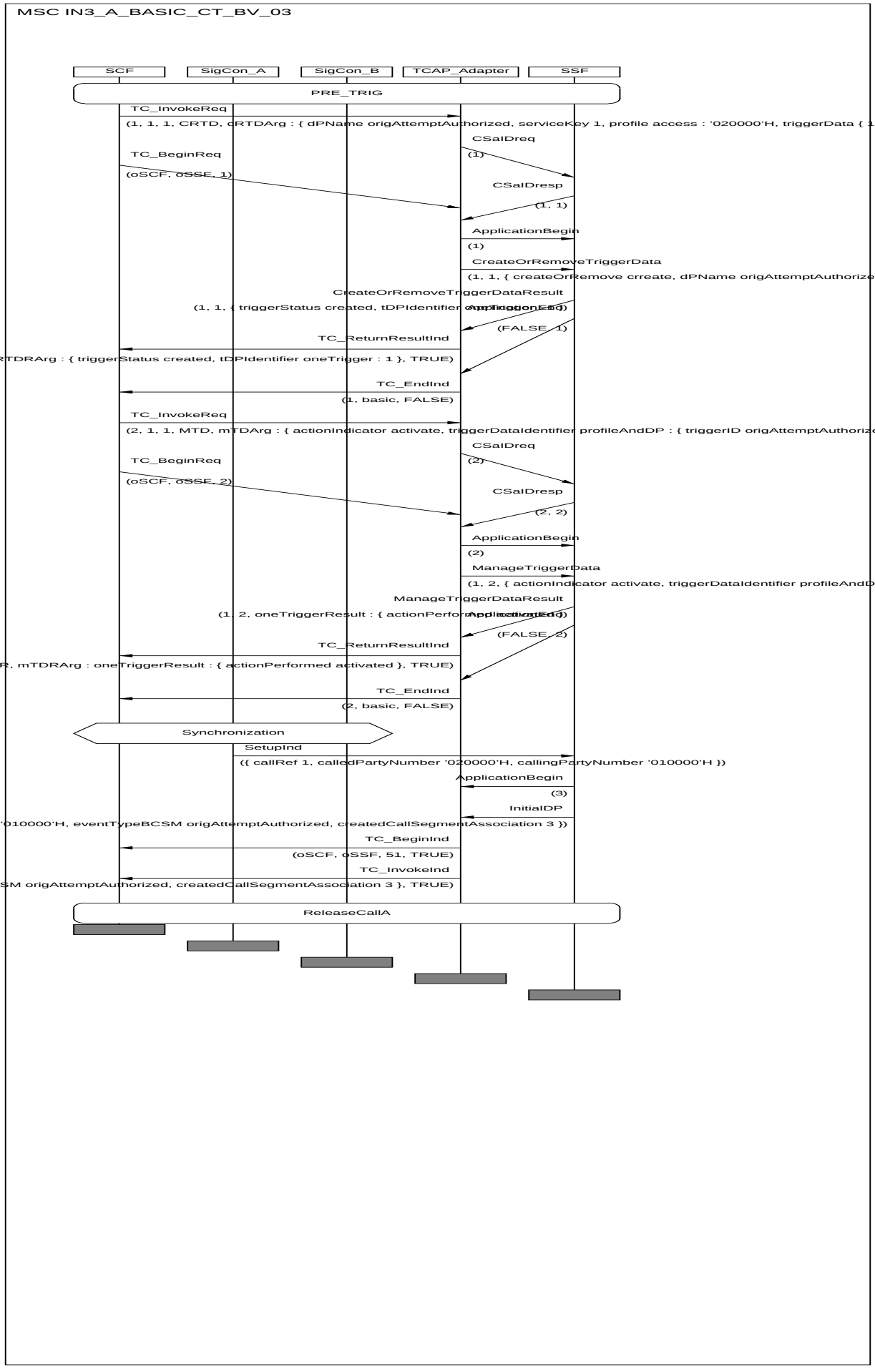
	IN3_A_BASIC_CT_BV_01
Work item no.:	ITEM_BASIC_182
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters createOrRemove = "create", dPName = "**origAttempt**", triggerDPType = "tdp-r", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1, triggerData = TRIGGER_DATA_1 and defaultFaultHandling = "resumeCallProcessing", sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**origAttempt**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(create, origAttempt, tdp-r, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, resumeCallProcessing) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated))CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttempt)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttempt)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { createOrRemove crreate, dPName origAttempt, triggerDPType tdp_r, serviceKey 1, profile access : '020000'H
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttempt, triggerDPType tdp_r, service
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttempt, profile access : '020000'H },
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(3)
InitialDP
20000'H, callingPartyNumber '010000'H, eventTypeBCSM origAttempt, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
mber '010000'H, eventTypeBCSM origAttempt, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

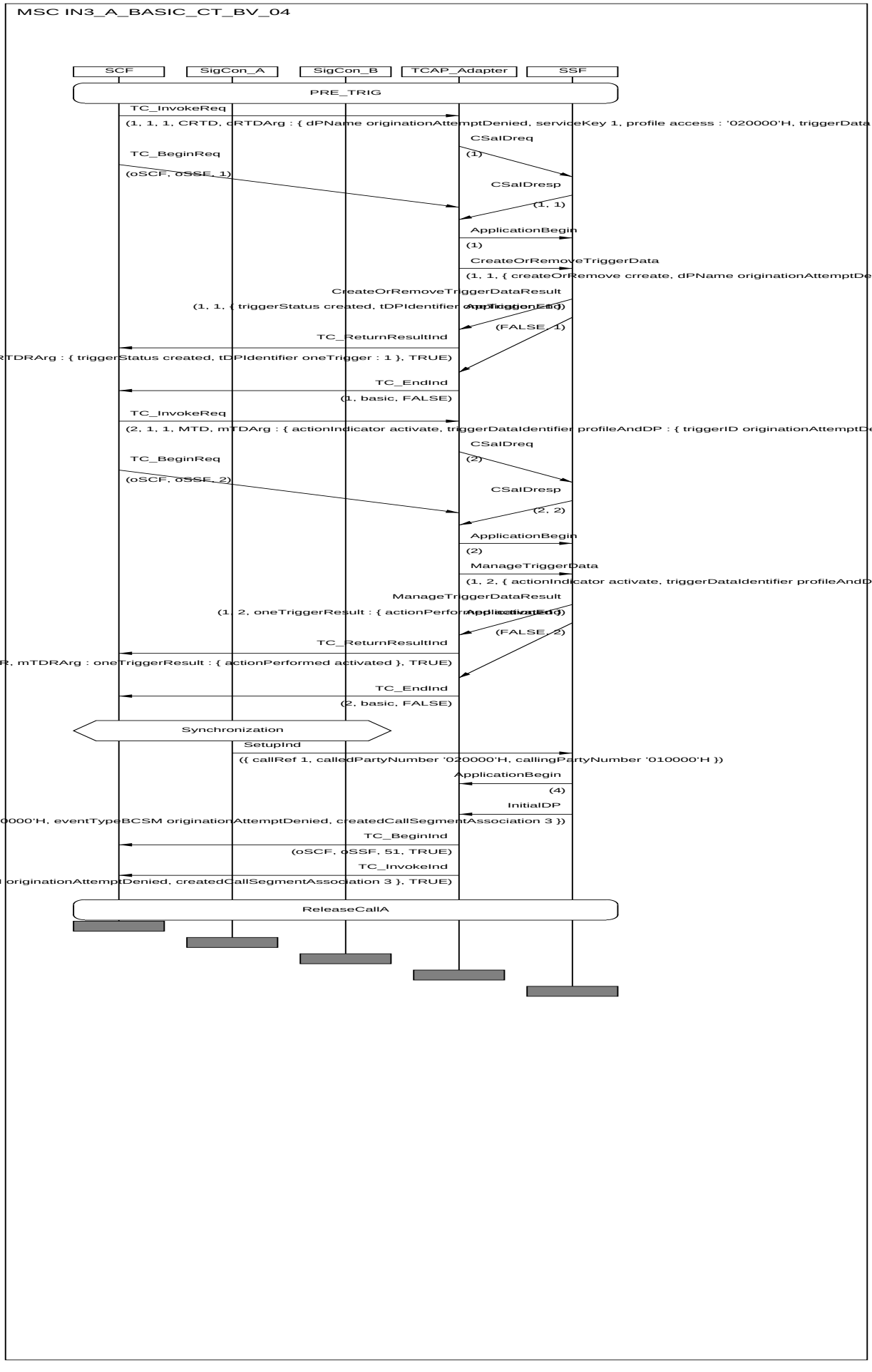
	IN3_A_BASIC_CT_BV_02
Work item no.:	ITEM_BASIC_183
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**oAbandon**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**oAbandon**", when the initiating user clears the call before the called user answers.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove **omitted**, **oAbandon**, triggerDPType **omitted**, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling **omitted**) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_1!ReleaseInd(Normal clearing) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oAbandon)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oAbandon)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oAbandon, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName oAbandon, triggerDPType tdp_r, serviceK
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oAbandon, profile access : '020000'H },
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oAban
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 1, cause '00'H })
ApplicationBegin
(3)
InitialDP
'020000'H, callingPartyNumber '010000'H, eventTypeBCSM oAbandon, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
umber '010000'H, eventTypeBCSM oAbandon, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

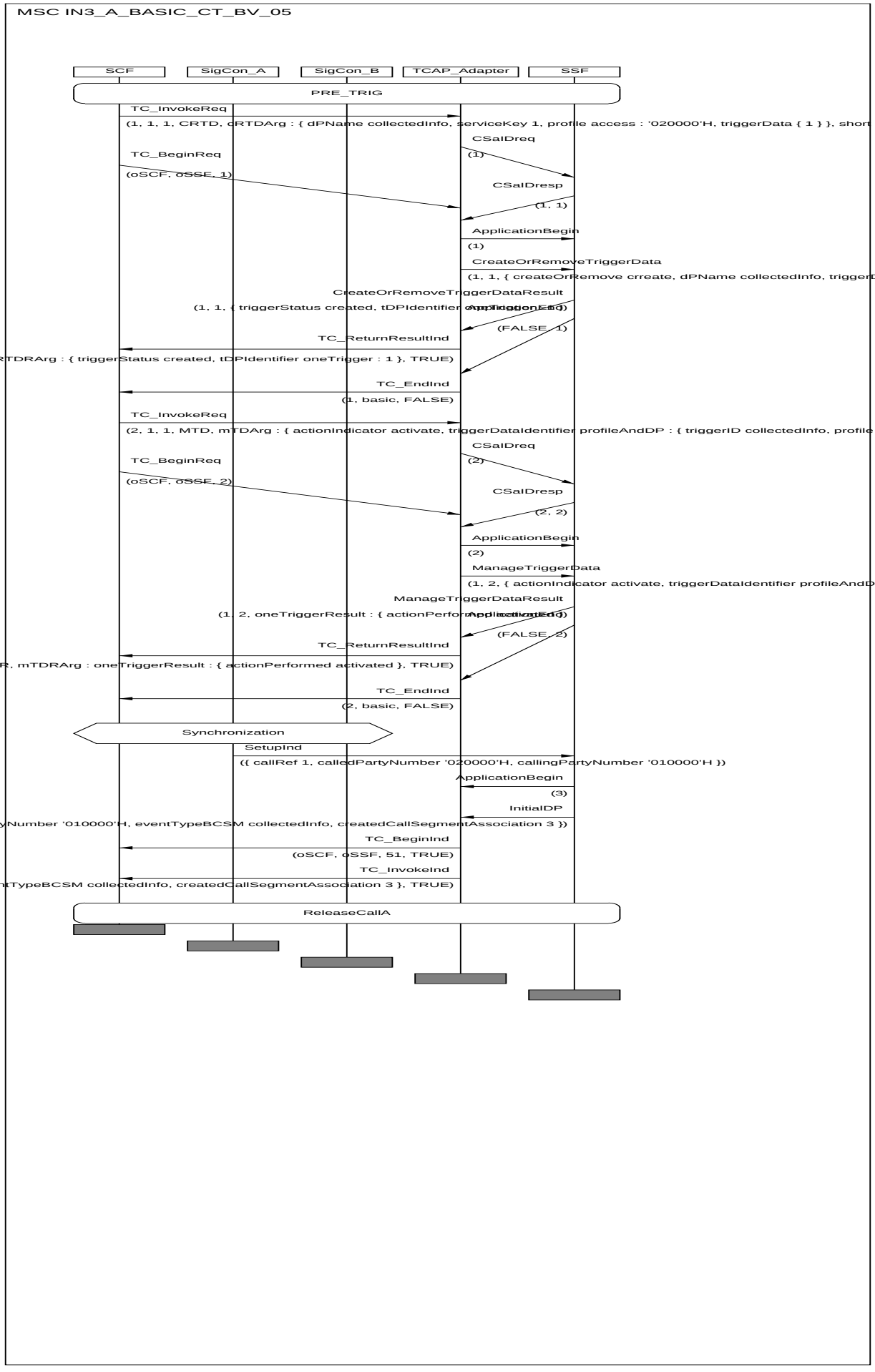
	IN3_A_BASIC_CT_BV_03
Work item no.:	ITEM_BASIC_184
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**origAttemptAuthorized**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**origAttemptAuthorized**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **origAttemptAuthorized**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '0
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(3)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

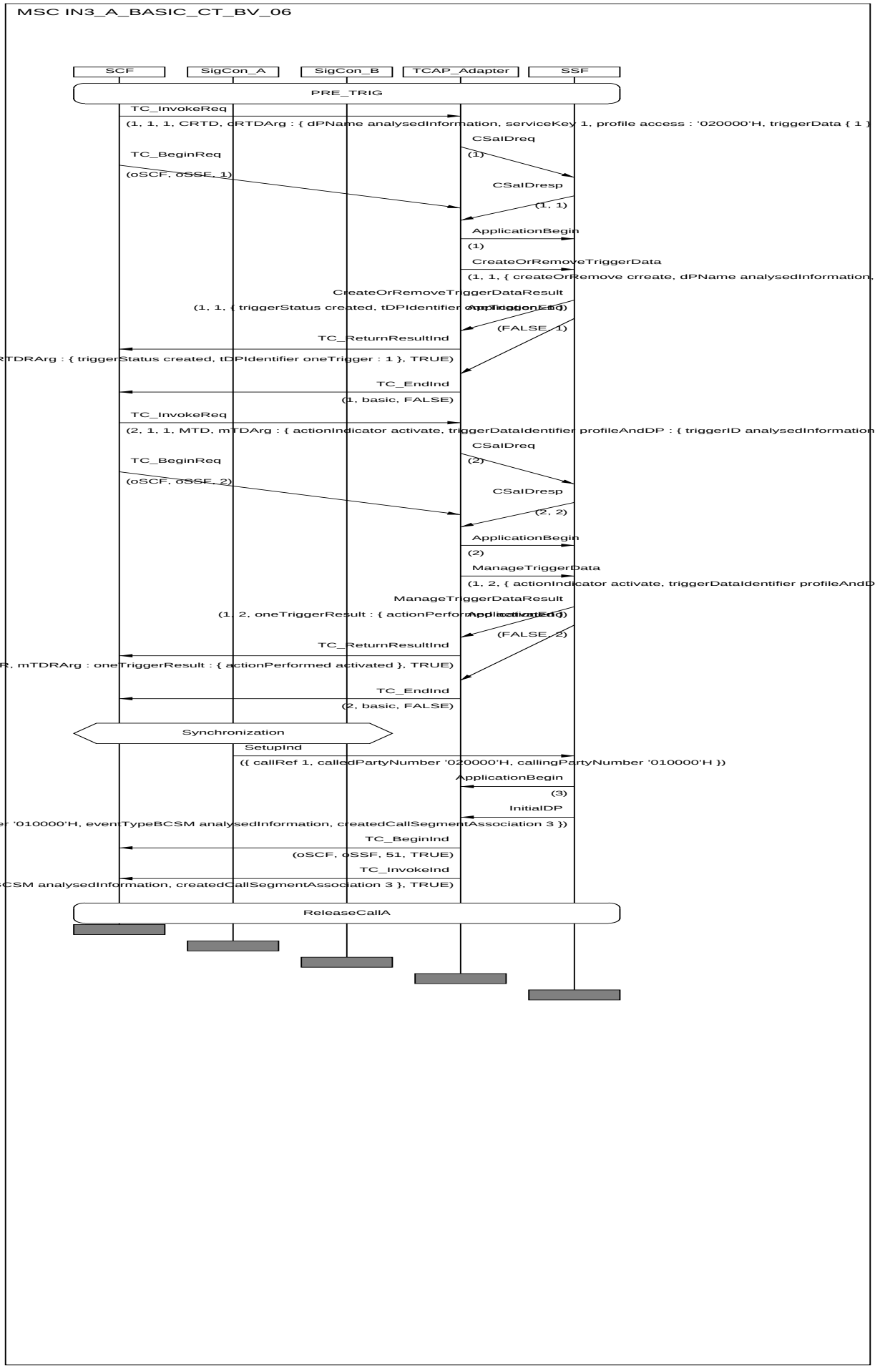
	IN3_A_BASIC_CT_BV_04
Work item no.:	ITEM_BASIC_185
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**originationAttemptDenied**", serviceKey = SERVICE_KEY2, profile = PROFILE_ID_2 and triggerData = TRIGGER_DATA_2, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, but not allowing the authorization to establish the call, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY2 and eventTypeBCSM = "**originationAttemptDenied**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **originationAttemptDenied**, triggerDPType omitted, SERVICE_KEY2, PROFILE_ID_2, TRIGGER_DATA_2, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_2), oneTrigger tDPIdentifier2) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-2) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY2, originationAttemptDenied)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY2, originationAttemptDenied)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName originationAttemptDenied, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName originationAttemptDenied, triggerDPType
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID originationAttemptDenied, profile access
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origina
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
ingPartyNumber '010000'H, eventTypeBCSM originationAttemptDenied, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
'H, eventTypeBCSM originationAttemptDenied, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

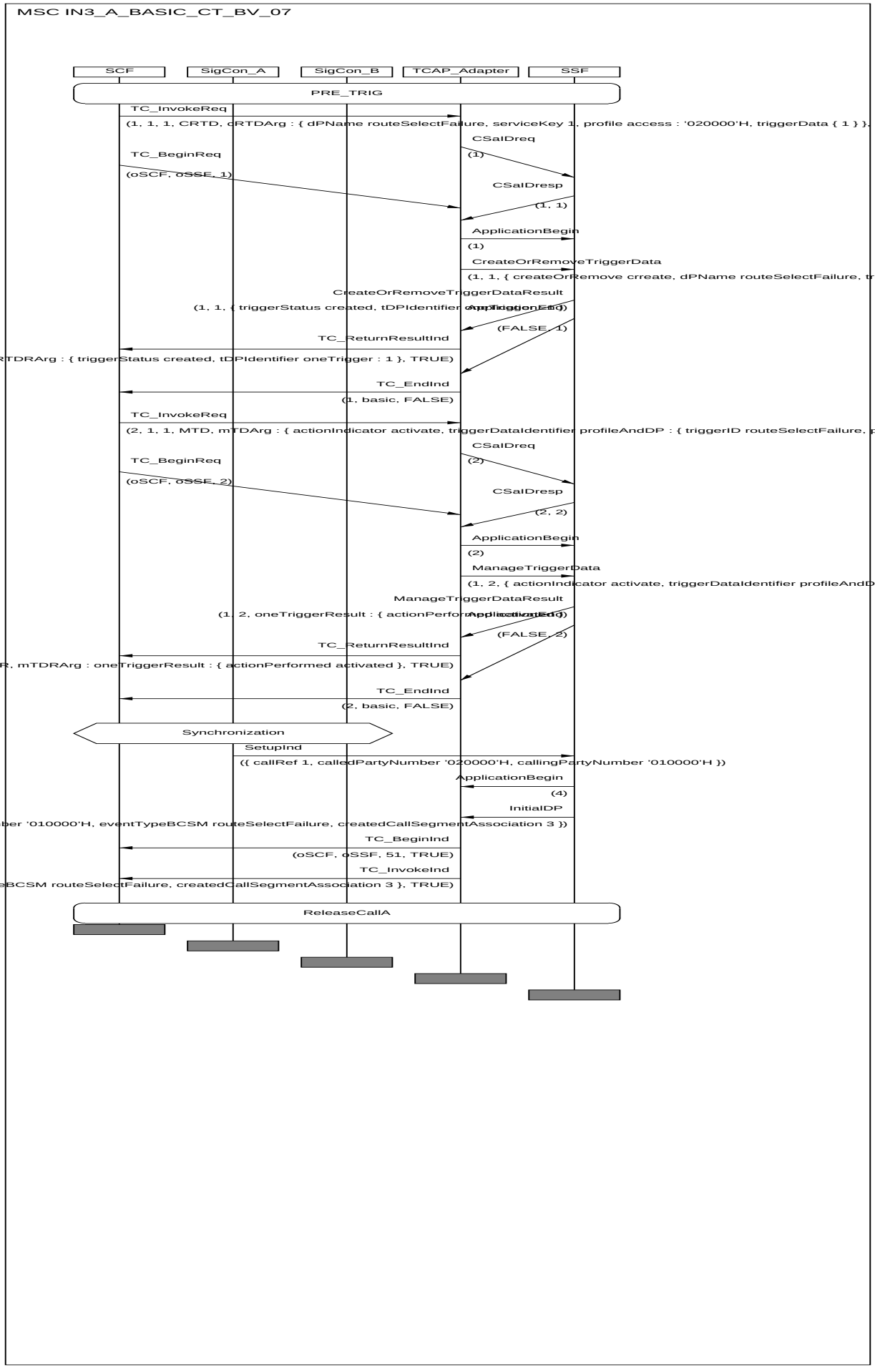
	IN3_A_BASIC_CT_BV_05
Work item no.:	ITEM_BASIC_186
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**collectedInfo**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**collectedInfo**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **collectedInfo**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, collectedInfo)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, collectedInfo)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_05
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName collectedInfo, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName collectedInfo, triggerDPType tdp_r, servic
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID collectedInfo, profile access : '020000'H }
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID collect
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(3)
InitialDP
0000'H, callingPartyNumber '010000'H, eventTypeBCSM collectedInfo, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ber '010000'H, eventTypeBCSM collectedInfo, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

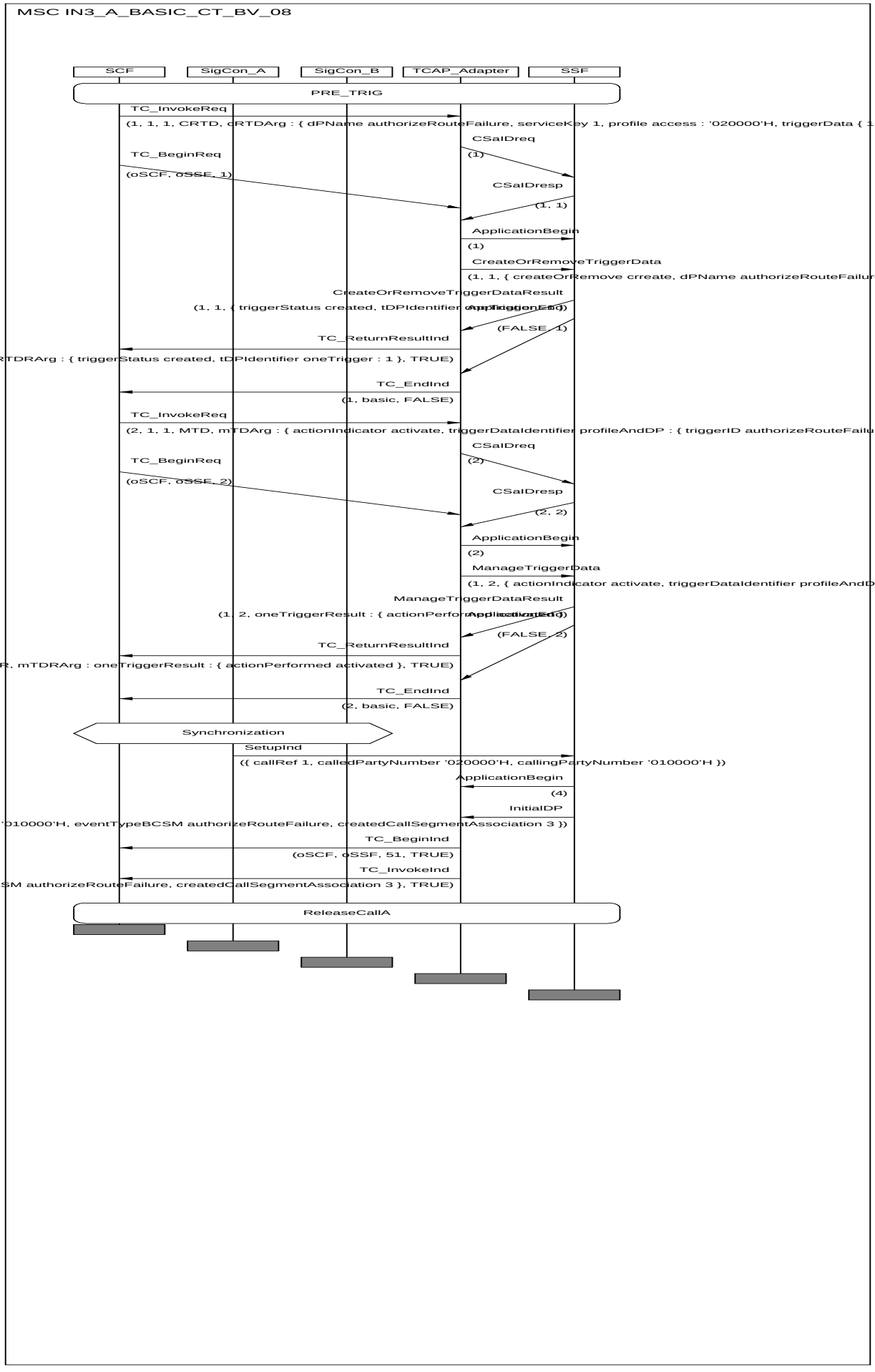
	IN3_A_BASIC_CT_BV_06
Work item no.:	ITEM_BASIC_187
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**analysedInformation**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**analysedInformation**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **analysedInformation**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, analysedInformation)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, analysedInformation)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_06
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName analysedInformation, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName analysedInformation, triggerDPType tdp_
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID analysedInformation, profile access : '020
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID analys
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated })
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(3)
InitialDP
, callingPartyNumber '010000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
0000'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

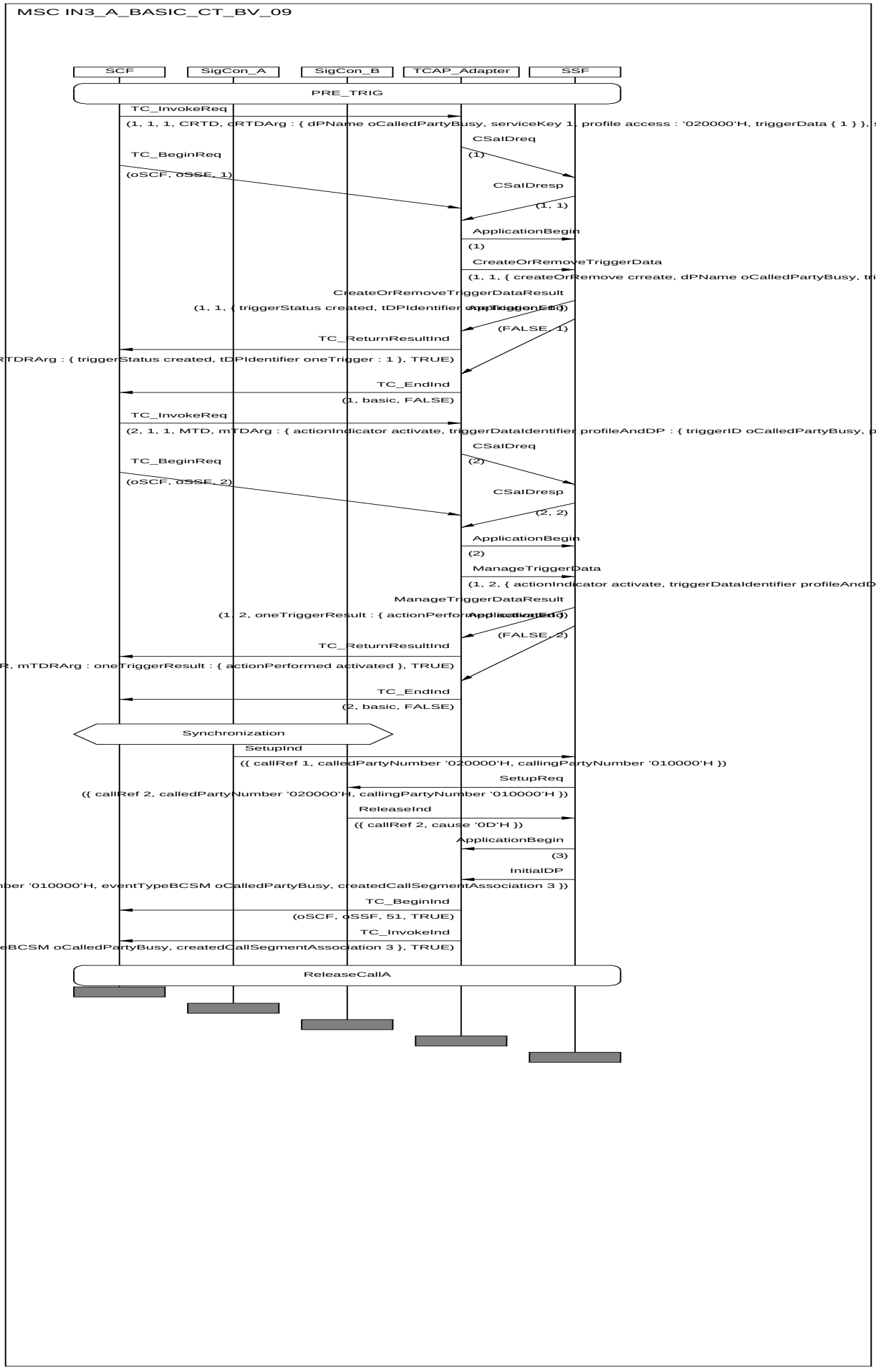
	IN3_A_BASIC_CT_BV_07
Work item no.:	ITEM_BASIC_188
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**routeSelectFailure**", serviceKey = SERVICE_KEY3, profile = PROFILE_ID_3 and triggerData = TRIGGER_DATA_3, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, but not allowing proper route selection to complete the call, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY3 and eventTypeBCSM = "**routeSelectFailure**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **routeSelectFailure**, triggerDPType omitted, SERVICE_KEY3, PROFILE_ID_3, TRIGGER_DATA_3, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier3) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_3), oneTrigger tDPIdentifier3) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-3) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY3, routeSelectFailure)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier3) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY3, routeSelectFailure)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_07
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName routeSelectFailure, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName routeSelectFailure, triggerDPType tdp_r,
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID routeSelectFailure, profile access : '0200
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID routeS
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
'H, callingPartyNumber '010000'H, eventTypeBCSM routeSelectFailure, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
10000'H, eventTypeBCSM routeSelectFailure, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

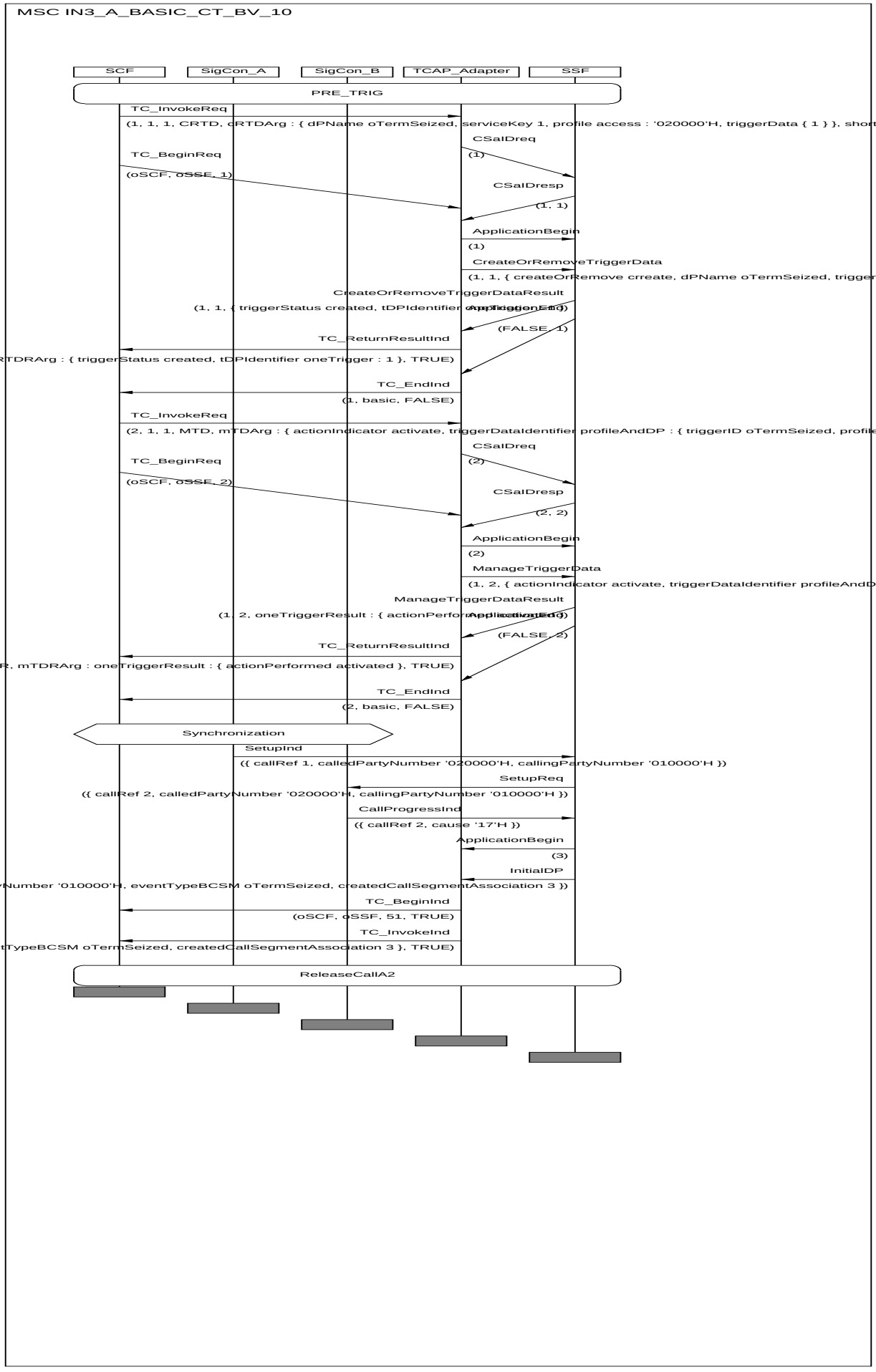
	IN3_A_BASIC_CT_BV_08
Work item no.:	ITEM_BASIC_189
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**authorizeRouteFailure**", serviceKey = SERVICE_KEY4, profile = PROFILE_ID_4 and triggerData = TRIGGER_DATA_4, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, but not allowing proper route authorization to complete the call, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY4 and eventTypeBCSM = "**authorizeRouteFailure**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **authorizeRouteFailure**, triggerDPType omitted, SERVICE_KEY4, PROFILE_ID_4, TRIGGER_DATA_4, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier4) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_4), oneTrigger tDPIdentifier4) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-4) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY4, authorizeRouteFailure)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier4) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY4, authorizeRouteFailure)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_08
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName authorizeRouteFailure, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName authorizeRouteFailure, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID authorizeRouteFailure, profile access : '0
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID author
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
callingPartyNumber '010000'H, eventTypeBCSM authorizeRouteFailure, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM authorizeRouteFailure, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

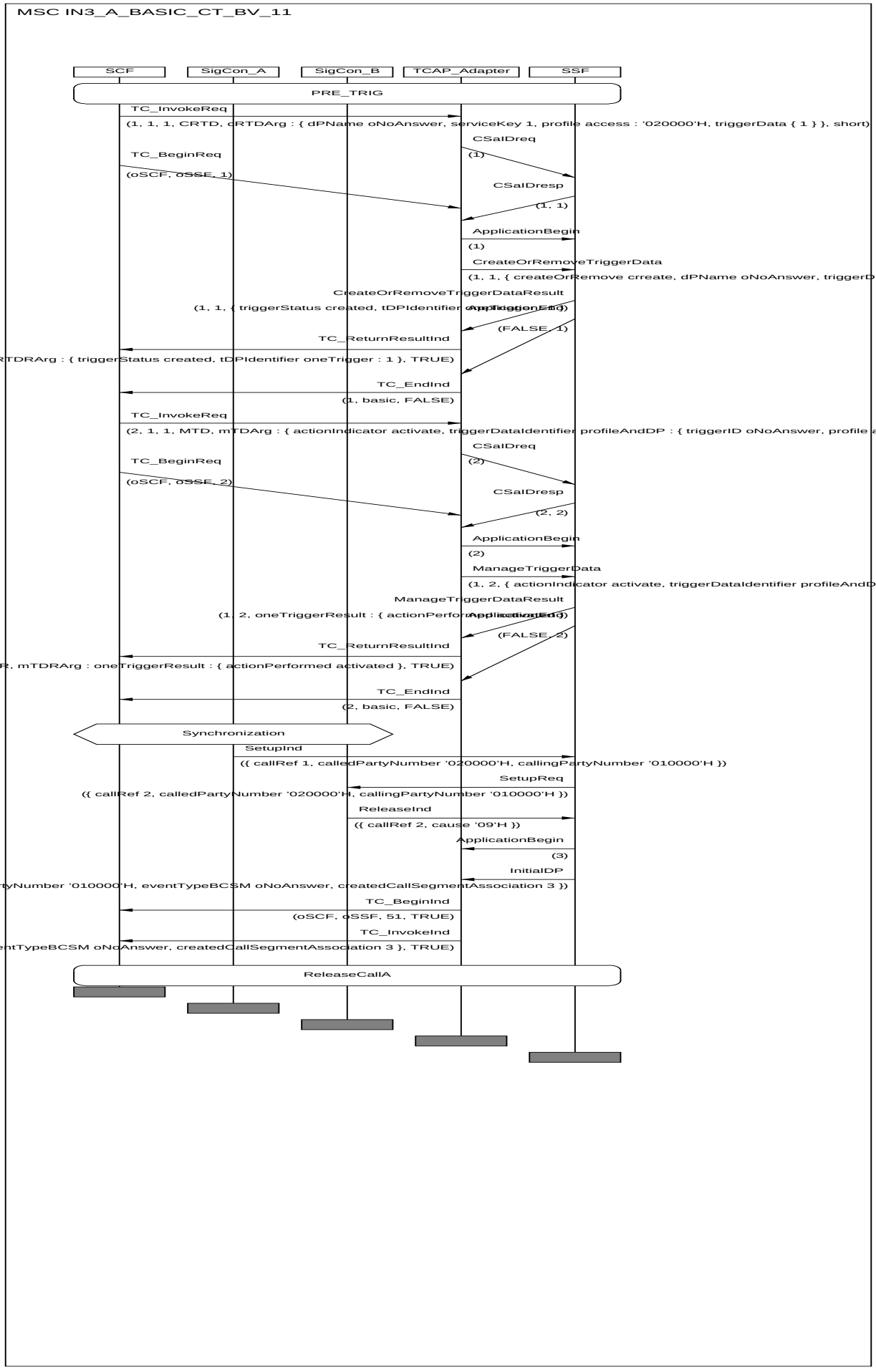
	IN3_A_BASIC_CT_BV_09
Work item no.:	ITEM_BASIC_190
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**oCalledPartyBusy**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having sent a SetupReq message to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**oCalledPartyBusy**", when the called user releases the call with a "busy" cause.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **oCalledPartyBusy**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!ReleaseReq(Busy cause) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oCalledPartyBusy)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oCalledPartyBusy)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_09
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oCalledPartyBusy, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName oCalledPartyBusy, triggerDPType tdp_r, s
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
ApplicationEnd
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oCalledPartyBusy, profile access : '0200
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oCalle
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
ApplicationEnd
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '0D'H })
ApplicationBegin
(3)
InitialDP
'H, callingPartyNumber '010000'H, eventTypeBCSM oCalledPartyBusy, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
10000'H, eventTypeBCSM oCalledPartyBusy, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

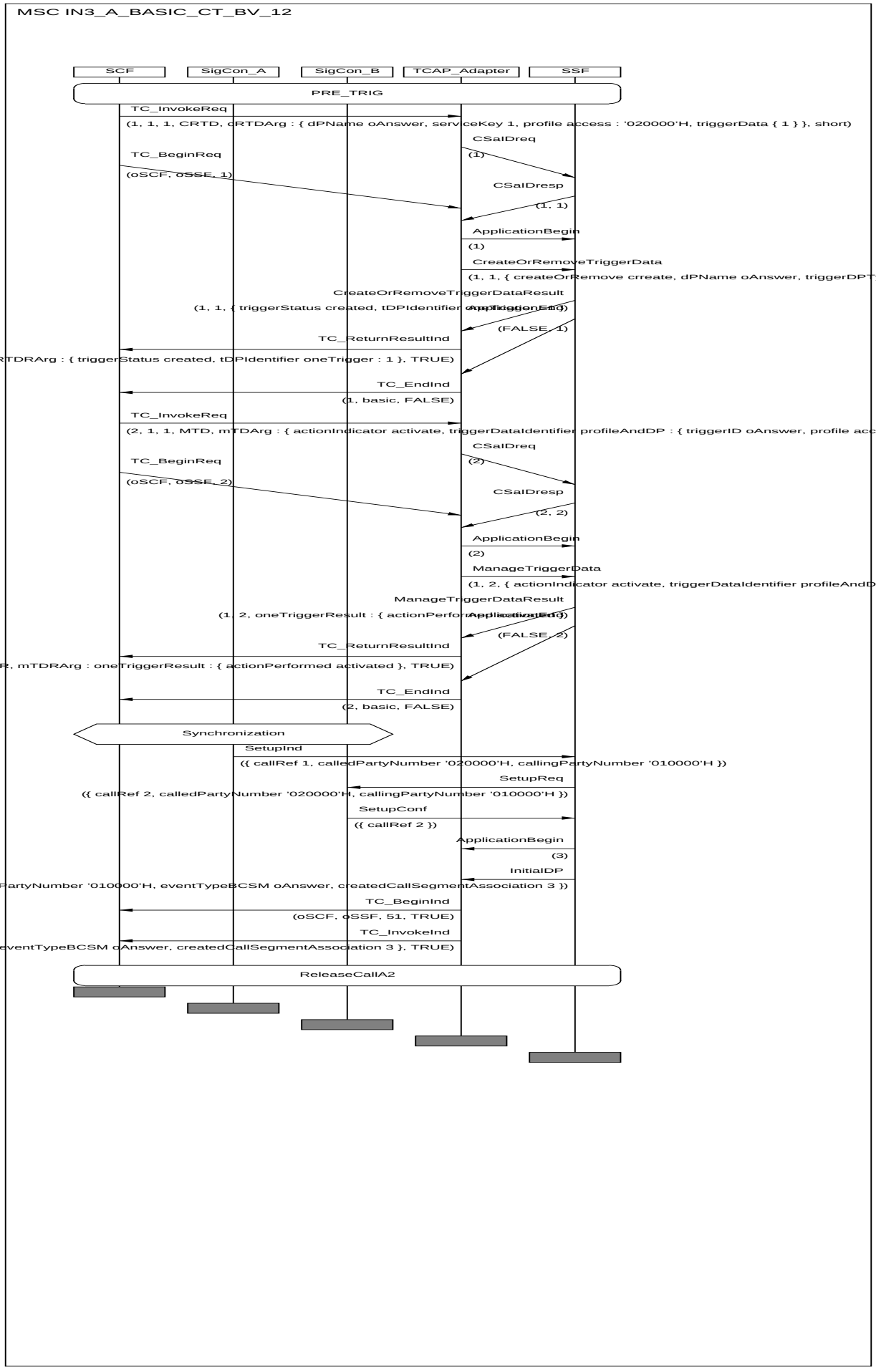
	IN3_A_BASIC_CT_BV_10
Work item no.:	ITEM_BASIC_191
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**oTermSeized**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having sent a SetupReq message to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**oTermSeized**", when the called user indicates "alerting".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **oTermSeized**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!CallProgressInd(cause "bPtyAlerted") L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oTermSeized)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oTermSeized)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_10
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oTermSeized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName oTermSeized, triggerDPType tdp_r, servic
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oTermSeized, profile access : '020000'H
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oTerm
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
CallProgressInd
({ callRef 2, cause '17'H })
ApplicationBegin
(3)
InitialDP
0000'H, callingPartyNumber '010000'H, eventTypeBCSM oTermSeized, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ber '010000'H, eventTypeBCSM oTermSeized, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

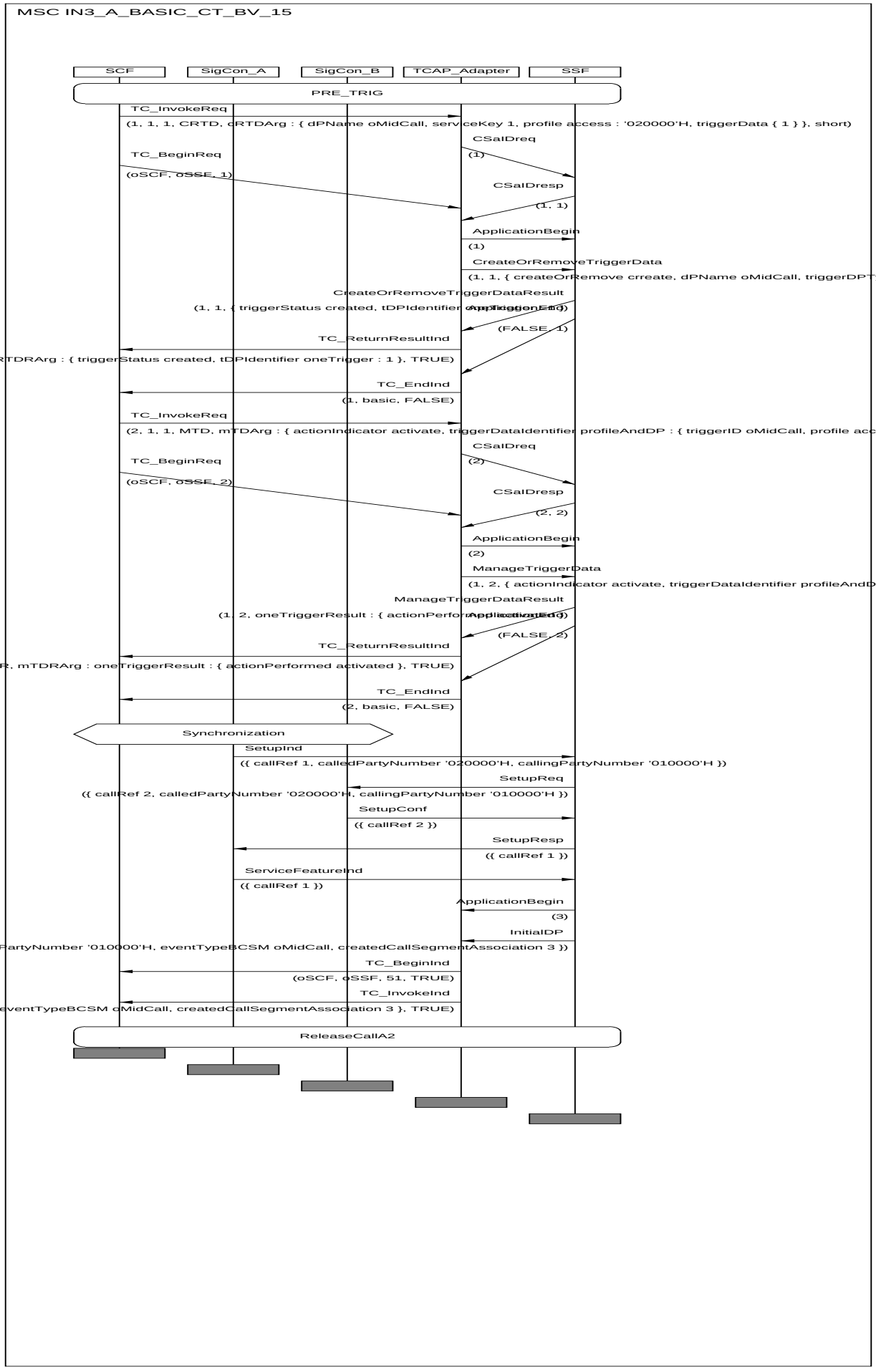
	IN3_A_BASIC_CT_BV_11
Work item no.:	ITEM_BASIC_192
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**oNoAnswer**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having sent a SetupReq message to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**oNoAnswer**", when the called user does not answer.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **oNoAnswer**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq Start timer(T-NO-ANSWER) CP1_2!ReleaseInd(cause "oNoAnswer") L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oNoAnswer)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oNoAnswer)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_11
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oNoAnswer, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName oNoAnswer, triggerDPType tdp_r, service
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oNoAnswer, profile access : '020000'H },
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oNoAn
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '09'H })
ApplicationBegin
(3)
InitialDP
20000'H, callingPartyNumber '010000'H, eventTypeBCSM oNoAnswer, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ber '010000'H, eventTypeBCSM oNoAnswer, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

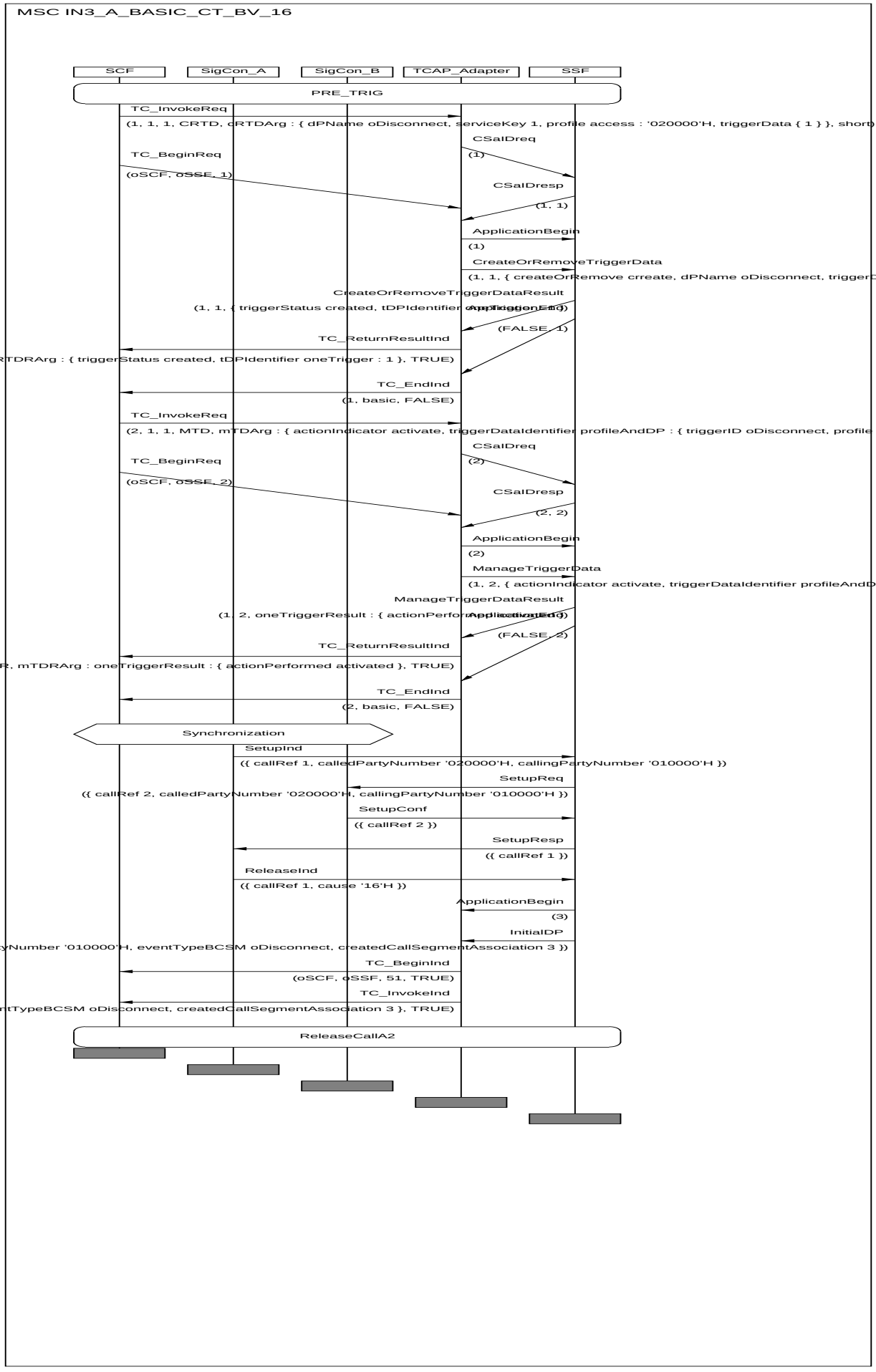
	IN3_A_BASIC_CT_BV_12
Work item no.:	ITEM_BASIC_193
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**oAnswer**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having sent a SetupReq message to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**oAnswer**", when the called user answers the call.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **oAnswer**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!SetUpConf L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oAnswer)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oAnswer)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_12
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oAnswer, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName oAnswer, triggerDPType tdp_r, serviceKe
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oAnswer, profile access : '020000'H }, tD
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oAnsw
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
ApplicationBegin
(3)
InitialDP
r '020000'H, callingPartyNumber '010000'H, eventTypeBCSM oAnswer, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
Number '010000'H, eventTypeBCSM oAnswer, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

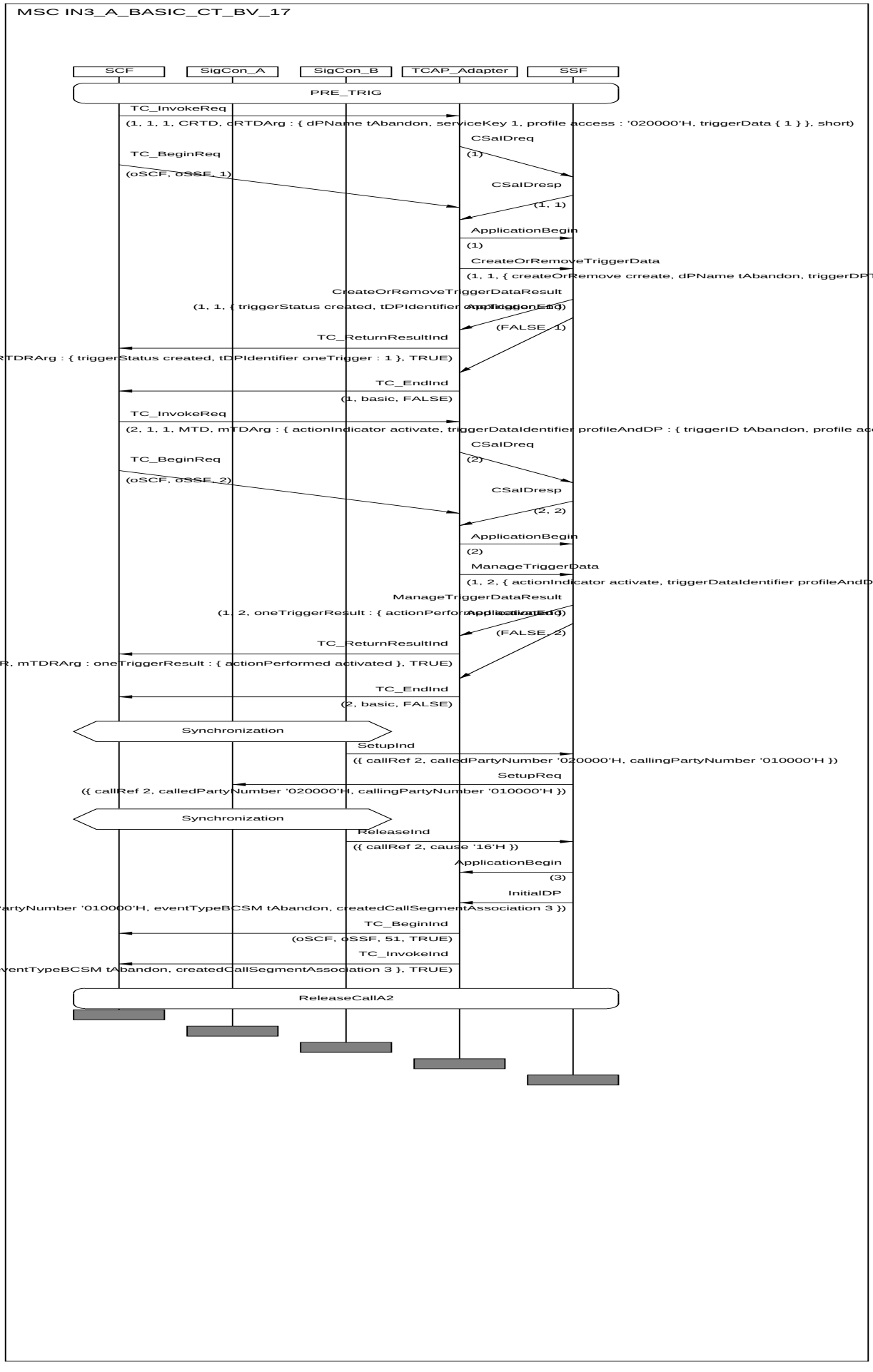
	IN3_A_BASIC_CT_BV_15
Work item no.:	ITEM_BASIC_196
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**oMidCall**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having successfully connected the call to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**oMidCall**", when the initiating user performs a ServiceFeature interaction.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **oMidCall**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_1!ServiceFeatureIndication L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oMidCall)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oMidCall)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_15
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oMidCall, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName oMidCall, triggerDPType tdp_r, serviceKe
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oMidCall, profile access : '020000'H }, tD
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oMidC
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated })
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ServiceFeatureInd
({ callRef 1 })
ApplicationBegin
(3)
InitialDP
r '020000'H, callingPartyNumber '010000'H, eventTypeBCSM oMidCall, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
Number '010000'H, eventTypeBCSM oMidCall, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

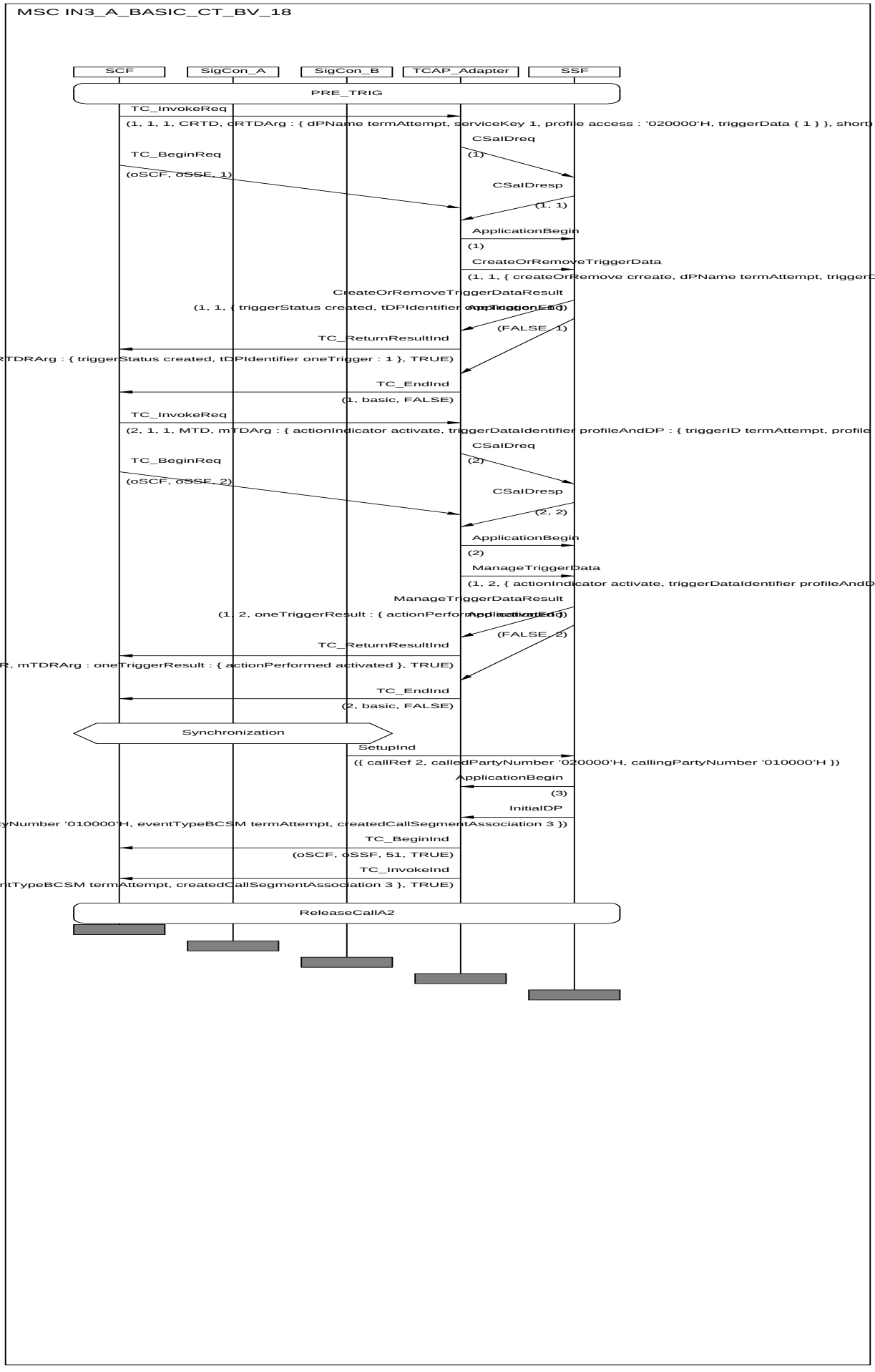
	IN3_A_BASIC_CT_BV_16
Work item no.:	ITEM_BASIC_197
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**oDisconnect**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having successfully connected the call to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**oDisconnect**", when the initiating user disconnects the call.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **oDisconnect**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_1!ReleaseInd(Normal cause) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oDisconnect)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, oDisconnect)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_16
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oDisconnect, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName oDisconnect, triggerDPType tdp_r, service
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oDisconnect, profile access : '020000'H }
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oDisco
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated })
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ReleaseInd
({ callRef 1, cause '16'H })
ApplicationBegin
(3)
InitialDP
20000'H, callingPartyNumber '010000'H, eventTypeBCSM oDisconnect, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ber '010000'H, eventTypeBCSM oDisconnect, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

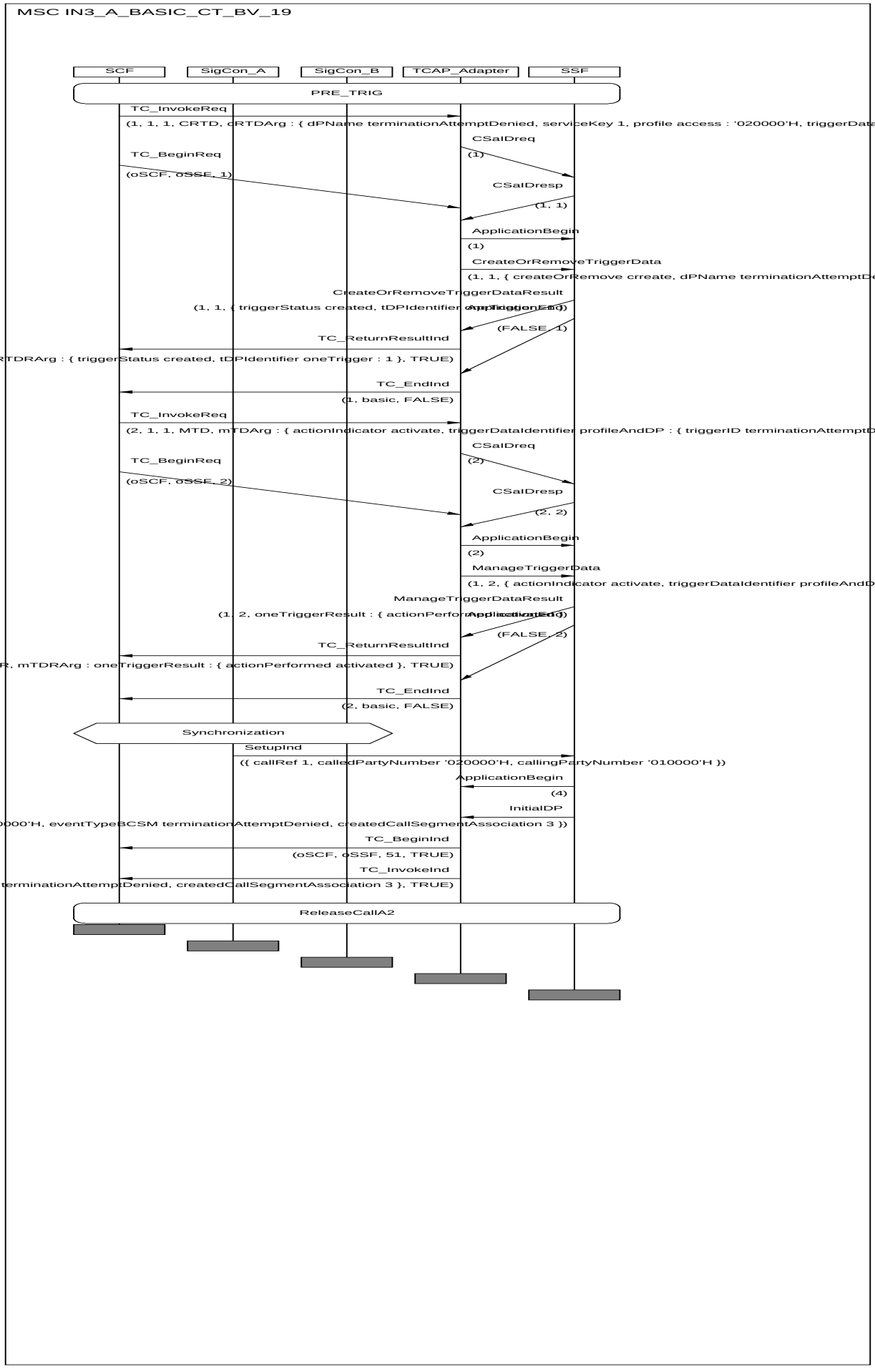
	IN3_A_BASIC_CT_BV_17
Work item no.:	ITEM_BASIC_198
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**tAbandon**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having sent a SetupReq message to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**tAbandon**", when the initiating user clears the call before the called user answers.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **tAbandon**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_1!ReleaseInd(Normal cause) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tAbandon)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tAbandon)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_17
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName tAbandon, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName tAbandon, triggerDPType tdp_r, serviceK
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tAbandon, profile access : '020000'H }, tD
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tAband
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
Synchronization
ReleaseInd
({ callRef 2, cause '16'H })
ApplicationBegin
(3)
InitialDP
'020000'H, callingPartyNumber '010000'H, eventTypeBCSM tAbandon, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
umber '010000'H, eventTypeBCSM tAbandon, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

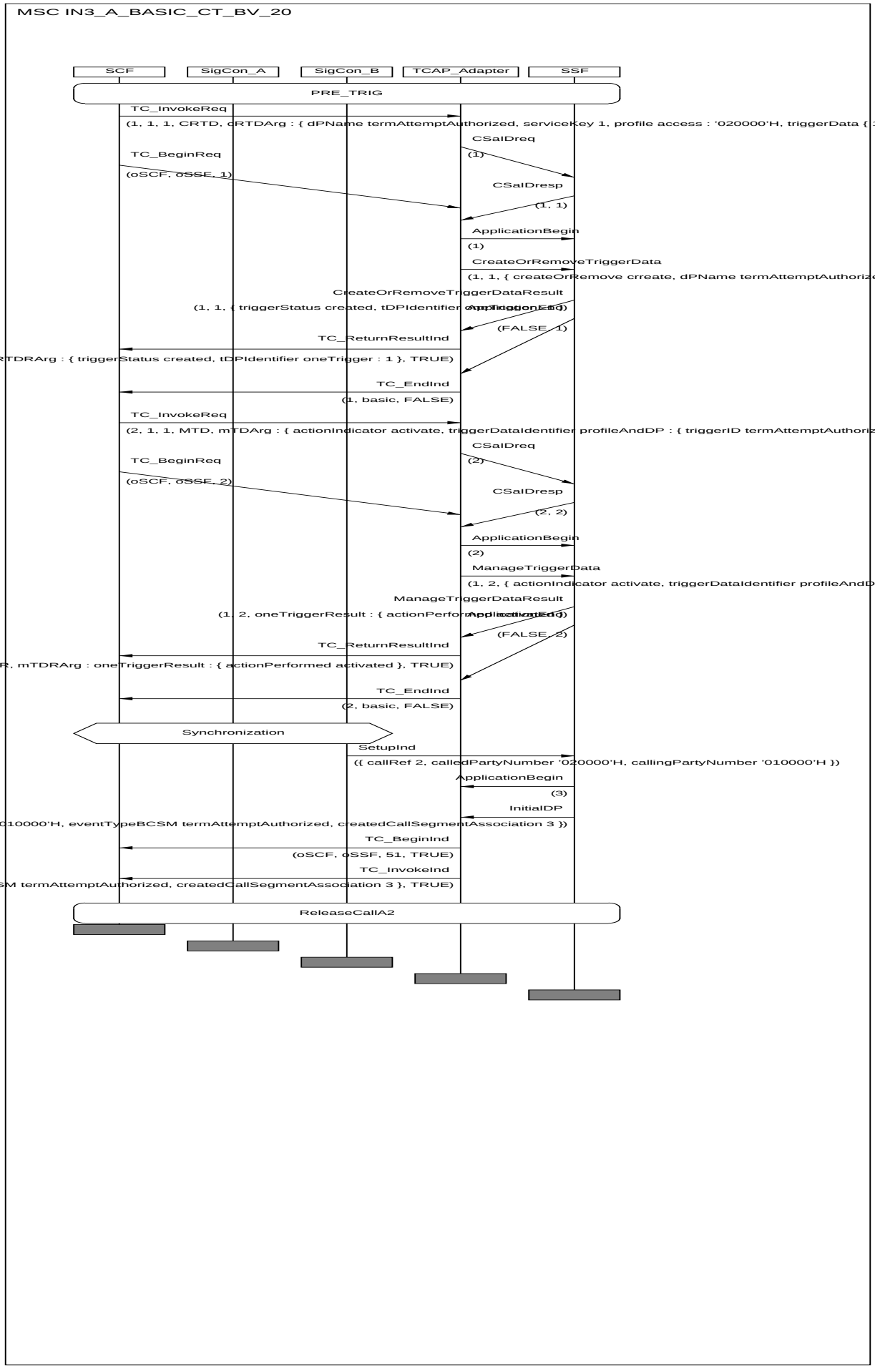
	IN3_A_BASIC_CT_BV_18
Work item no.:	ITEM_BASIC_199
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**termAttempt**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**termAttempt**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **termAttempt**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, termAttempt)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, termAttempt)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_18
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName termAttempt, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName termAttempt, triggerDPType tdp_r, service
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID termAttempt, profile access : '020000'H }
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID termA
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(3)
InitialDP
20000'H, callingPartyNumber '010000'H, eventTypeBCSM termAttempt, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ber '010000'H, eventTypeBCSM termAttempt, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

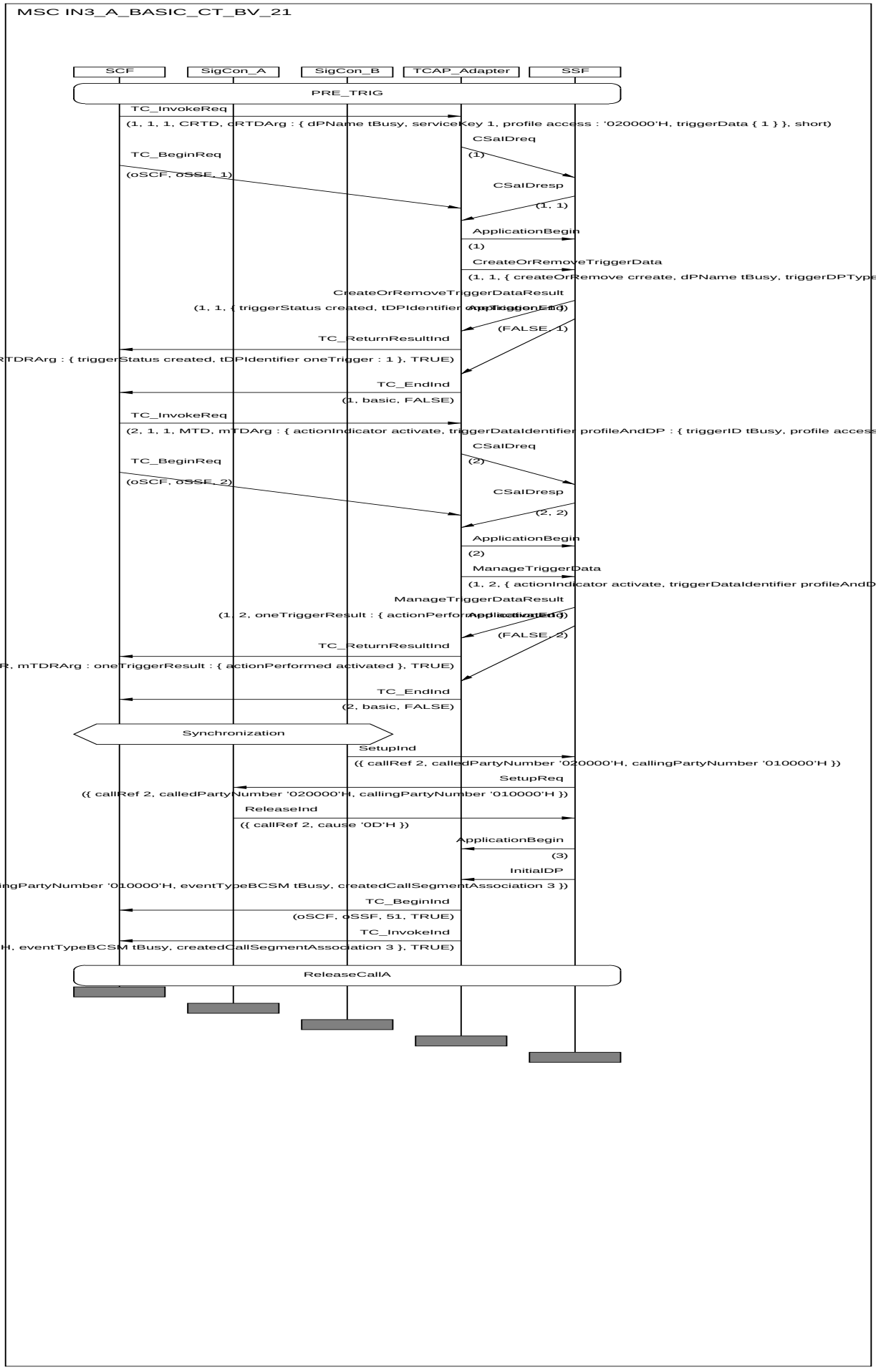
	IN3_A_BASIC_CT_BV_19
Work item no.:	ITEM_BASIC_200
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**terminationAttemptDenied**", serviceKey = SERVICE_KEY5, profile = PROFILE_ID_5 and triggerData = TRIGGER_DATA_5, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, but not allowing the authorization to establish the call at the terminating side, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY5 and eventTypeBCSM = "**terminationAttemptDenied**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **terminationAttemptDenied**, triggerDPType omitted, SERVICE_KEY5, PROFILE_ID_5, TRIGGER_DATA_5, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier5) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_5), oneTrigger tDPIdentifier5) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated) CP1_1!SetupInd(CALL-DATA-5) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY5, terminationAttemptDenied)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier5) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY5, terminationAttemptDenied)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_19
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName terminationAttemptDenied, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName terminationAttemptDenied, triggerDPType
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID terminationAttemptDenied, profile access
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID termin
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
ngPartyNumber '010000'H, eventTypeBCSM terminationAttemptDenied, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
, eventTypeBCSM terminationAttemptDenied, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

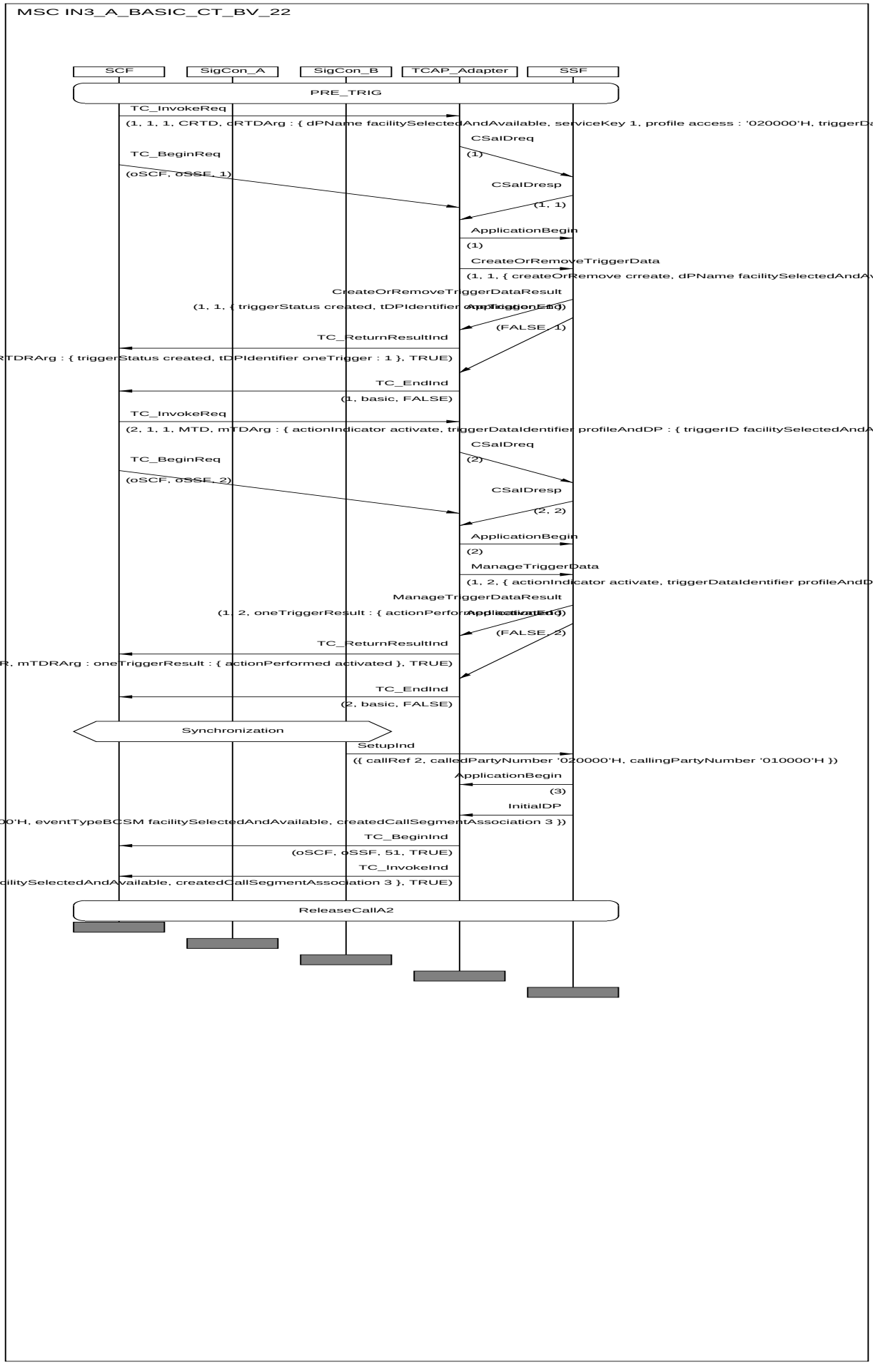
	IN3_A_BASIC_CT_BV_20
Work item no.:	ITEM_BASIC_201
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**termAttemptAuthorized**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**termAttemptAuthorized**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **termAttemptAuthorized**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, termAttemptAuthorized)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, termAttemptAuthorized)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_20
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName termAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName termAttemptAuthorized, triggerDPType td
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID termAttemptAuthorized, profile access :
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID termA
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(3)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM termAttemptAuthorized, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
0'H, eventTypeBCSM termAttemptAuthorized, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

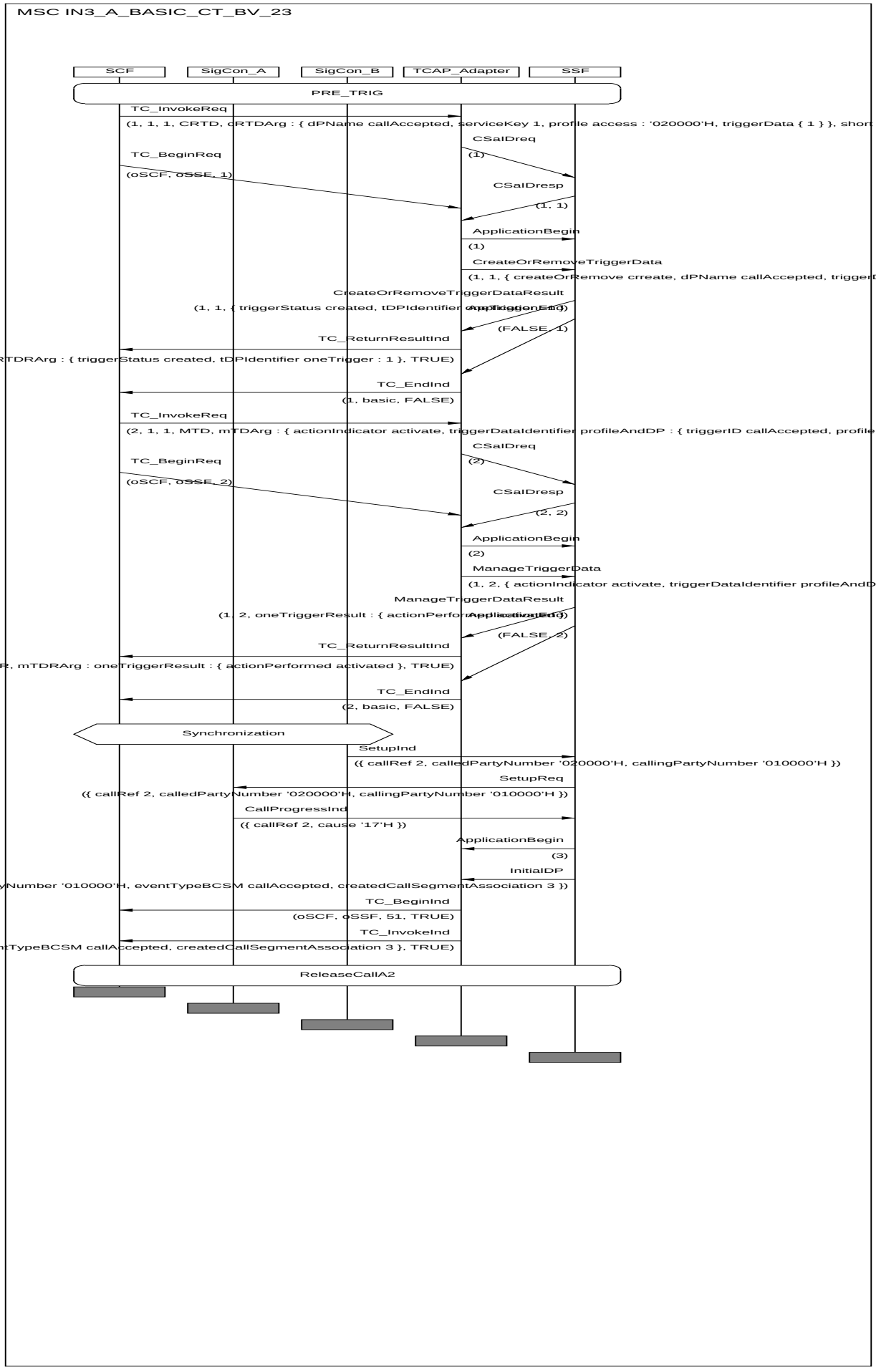
	IN3_A_BASIC_CT_BV_21
Work item no.:	ITEM_BASIC_202
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**tBusy**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having sent a SetupReq message to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**tBusy**", when the called user clears the call indicating a "busy" cause.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **tBusy**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated,)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!ReleaseInd(Busy cause) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tBusy)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tBusy)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_21
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName tBusy, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName tBusy, triggerDPType tdp_r, serviceKey 1
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tBusy, profile access : '020000'H }, tDPId
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tBusy,
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '0D'H })
ApplicationBegin
(3)
InitialDP
ber '020000'H, callingPartyNumber '010000'H, eventTypeBCSM tBusy, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
rtyNumber '010000'H, eventTypeBCSM tBusy, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

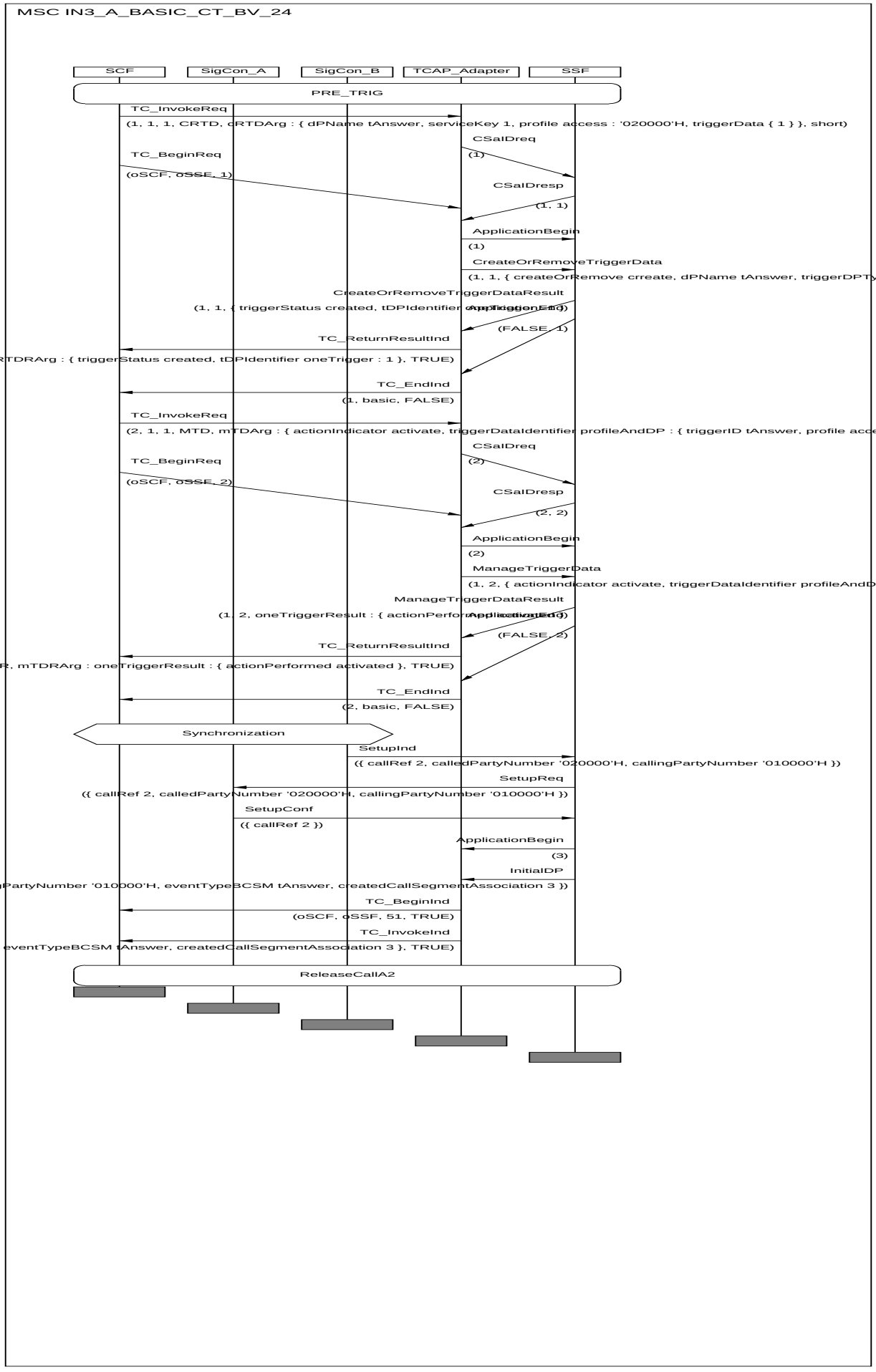
	IN3_A_BASIC_CT_BV_22
Work item no.:	ITEM_BASIC_203
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**facilitySelectedAndAvailable**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**facilitySelectedAndAvailable**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **facilitySelectedAndAvailable**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) . CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, facilitySelectedAndAvailable)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, facilitySelectedAndAvailable)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_22
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName facilitySelectedAndAvailable, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName facilitySelectedAndAvailable, triggerDPTy
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID facilitySelectedAndAvailable, profile acce
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID facility
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(3)
InitialDP
PartyNumber '010000'H, eventTypeBCSM facilitySelectedAndAvailable, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
eventTypeBCSM facilitySelectedAndAvailable, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

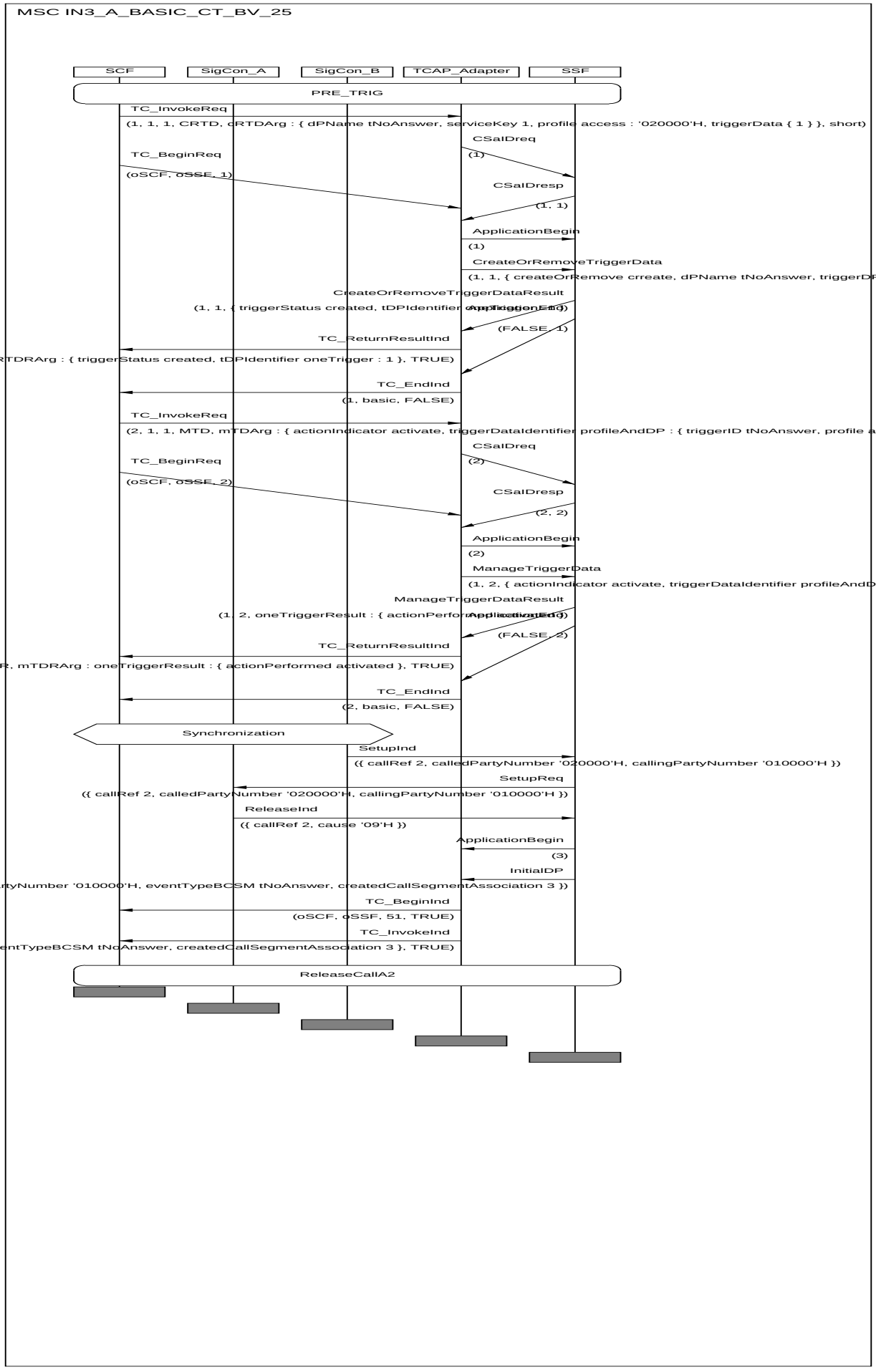
	IN3_A_BASIC_CT_BV_23
Work item no.:	ITEM_BASIC_204
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**callAccepted**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having sent a SetupReq message to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**callAccepted**", when the called user indicates "alerting".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **callAccepted**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!CallProgressInd(alerting) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, callAccepted)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, callAccepted)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_23
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName callAccepted, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName callAccepted, triggerDPType tdp_r, servic
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID callAccepted, profile access : '020000'H
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID callAcc
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
CallProgressInd
({ callRef 2, cause '17'H })
ApplicationBegin
(3)
InitialDP
0000'H, callingPartyNumber '010000'H, eventTypeBCSM callAccepted, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ber '010000'H, eventTypeBCSM callAccepted, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

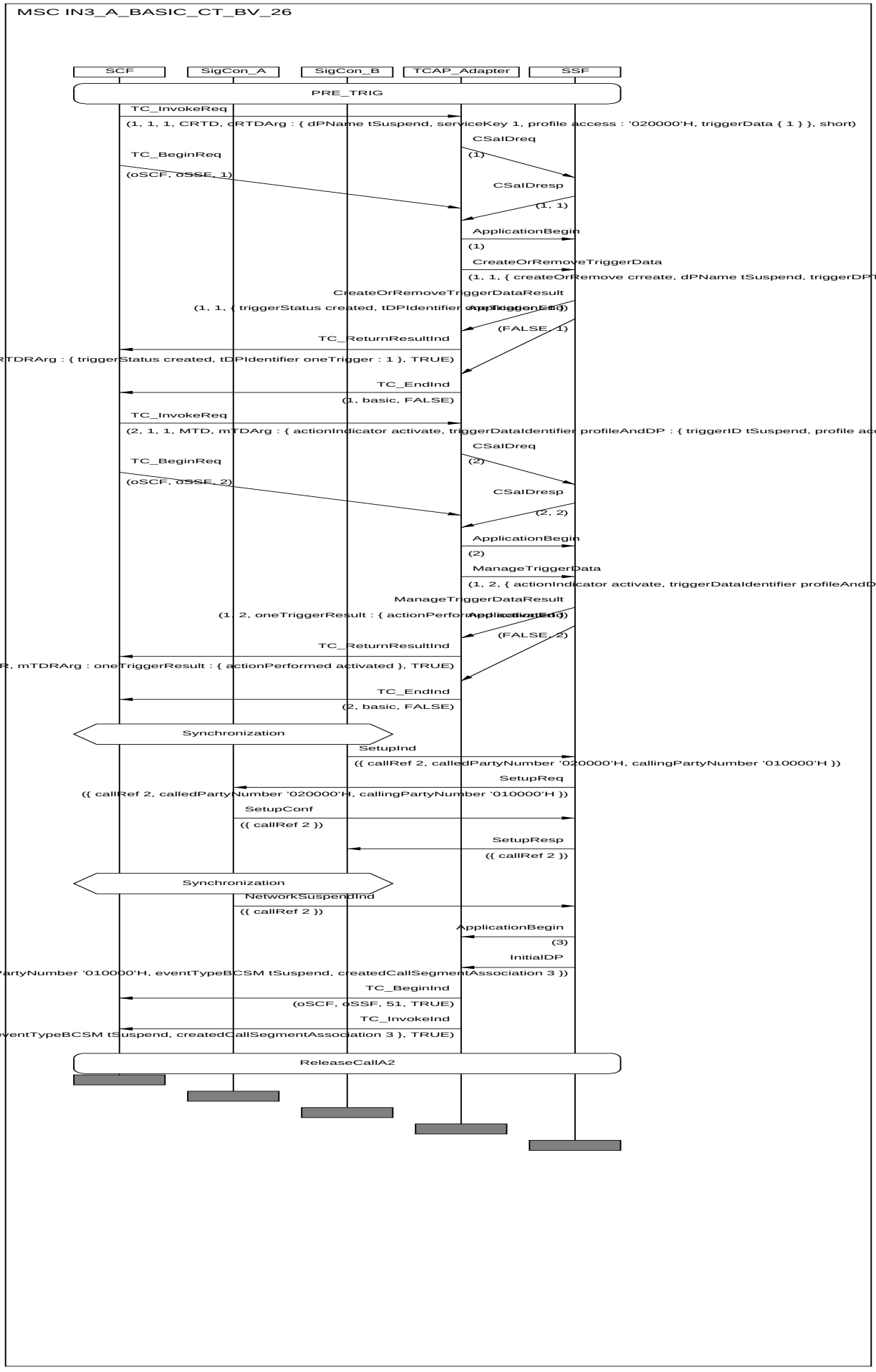
	IN3_A_BASIC_CT_BV_24
Work item no.:	ITEM_BASIC_205
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**tAnswer**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having sent a SetupReq message to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**tAnswer**", when the called user answers the call (SetupConfirm).
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **tAnswer**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!SetupConf L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tAnswer)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tAnswer)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_24
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName tAnswer, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName tAnswer, triggerDPType tdp_r, serviceKey
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tAnswer, profile access : '020000'H }, tDP
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tAnsw
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated })
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
ApplicationBegin
(3)
InitialDP
er '020000'H, callingPartyNumber '010000'H, eventTypeBCSM tAnswer, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
Number '010000'H, eventTypeBCSM tAnswer, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

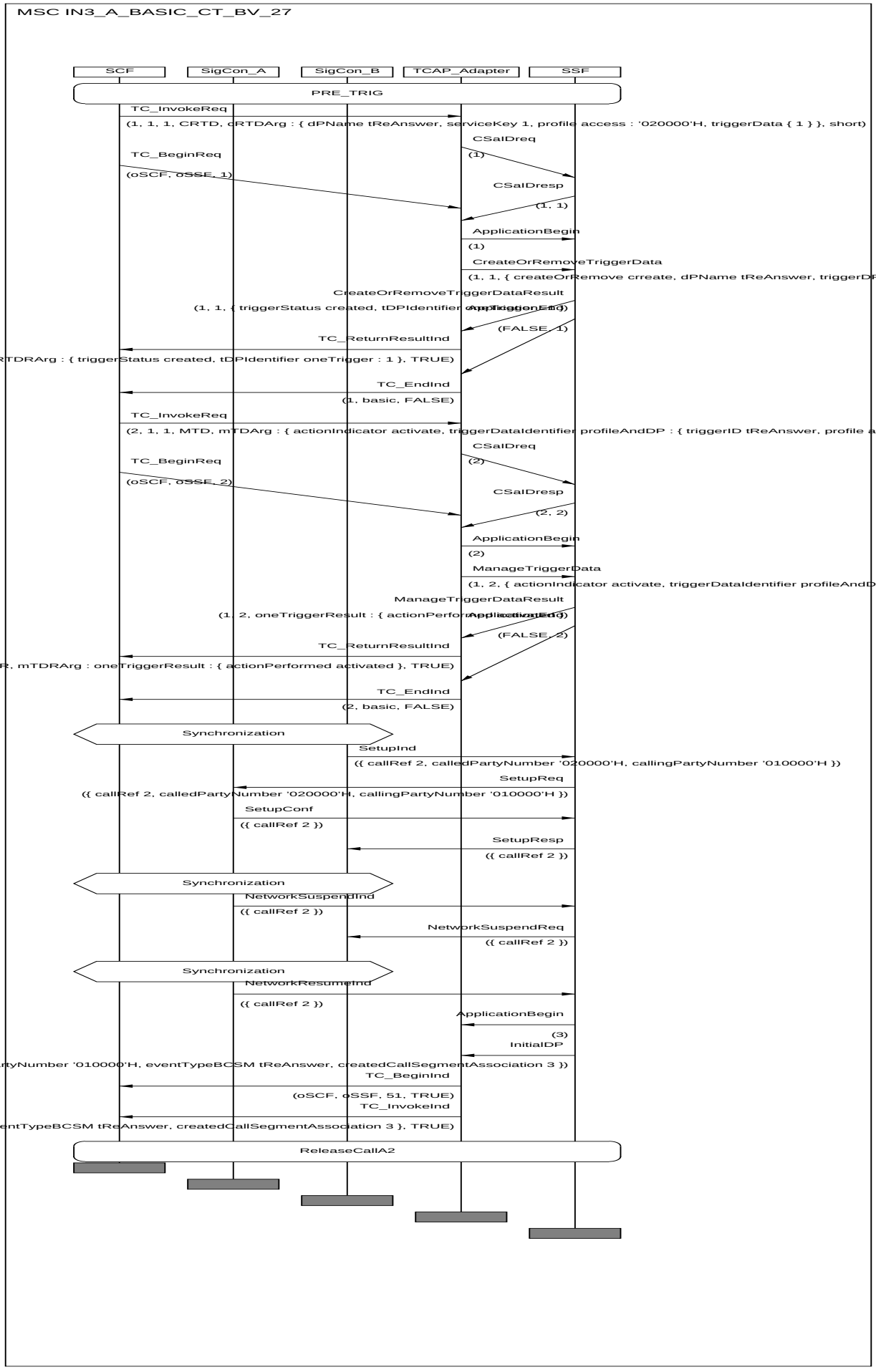
	IN3_A_BASIC_CT_BV_25
Work item no.:	ITEM_BASIC_206
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**tNoAnswer**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having sent a SetupReq message to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**tNoAnswer**", when the called user does not answer the call (no alerting and no SetupConfirm).
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **tNoAnswer**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq Start T-NO-ANSWER CP1_2!ReleaseInd(cause "oNoAnswer") L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tNoAnswer)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tNoAnswer)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_25
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName tNoAnswer, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName tNoAnswer, triggerDPType tdp_r, servicek
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tNoAnswer, profile access : '020000'H },
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tNoAn
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '09'H })
ApplicationBegin
(3)
InitialDP
020000'H, callingPartyNumber '010000'H, eventTypeBCSM tNoAnswer, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
mber '010000'H, eventTypeBCSM tNoAnswer, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

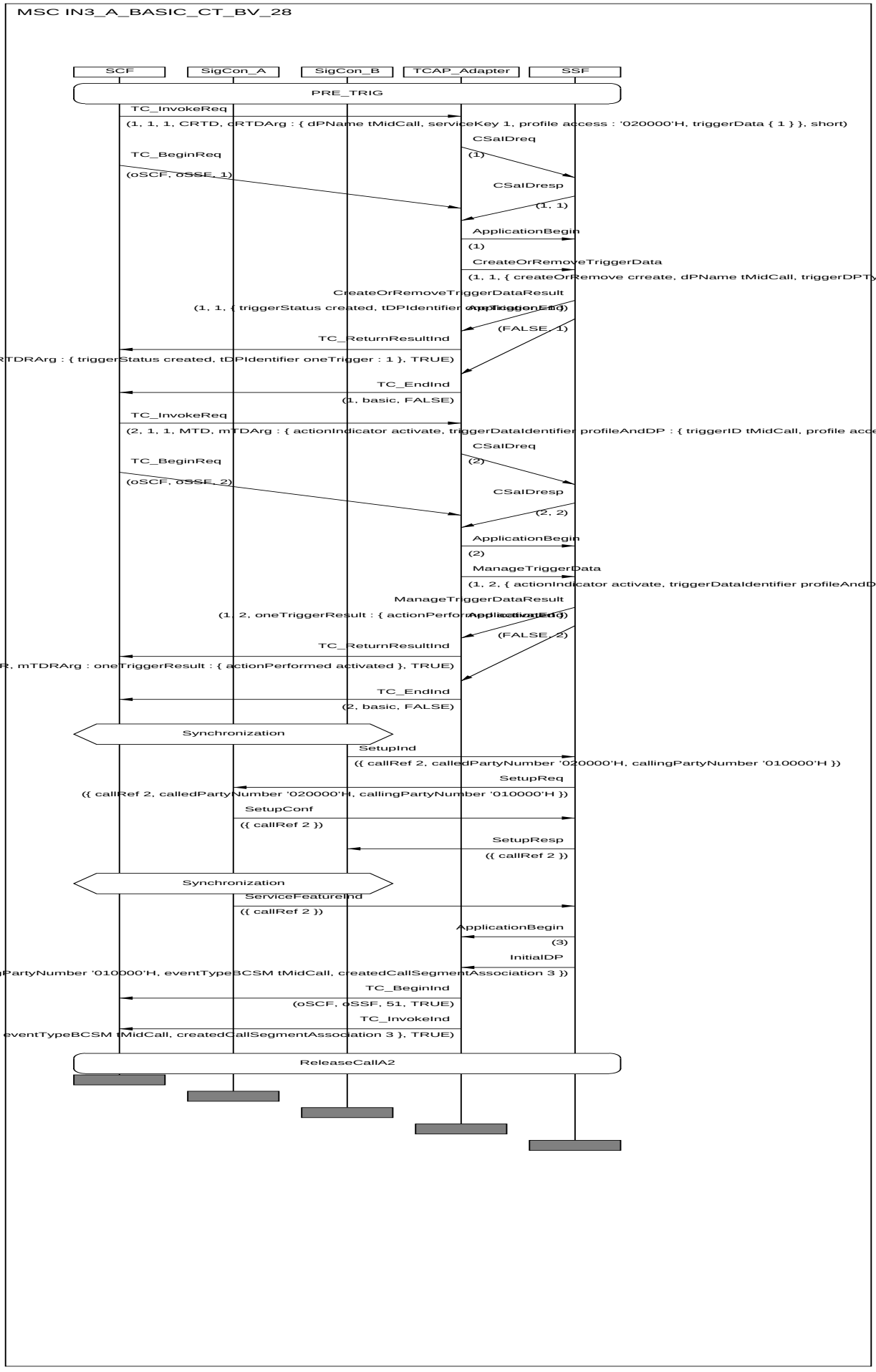
	IN3_A_BASIC_CT_BV_26
Work item no.:	ITEM_BASIC_207
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**tSuspend**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having connected the call to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**tSuspend**", when the called user suspends the call.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **tSuspend**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_2!NetworkSuspendInd L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tSuspend)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tSuspend)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_26
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName tSuspend, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName tSuspend, triggerDPType tdp_r, serviceKe
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tSuspend, profile access : '020000'H }, tD
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tSuspe
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated })
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 2 })
Synchronization
NetworkSuspendInd
({ callRef 2 })
ApplicationBegin
(3)
InitialDP
'020000'H, callingPartyNumber '010000'H, eventTypeBCSM tSuspend, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
umber '010000'H, eventTypeBCSM tSuspend, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

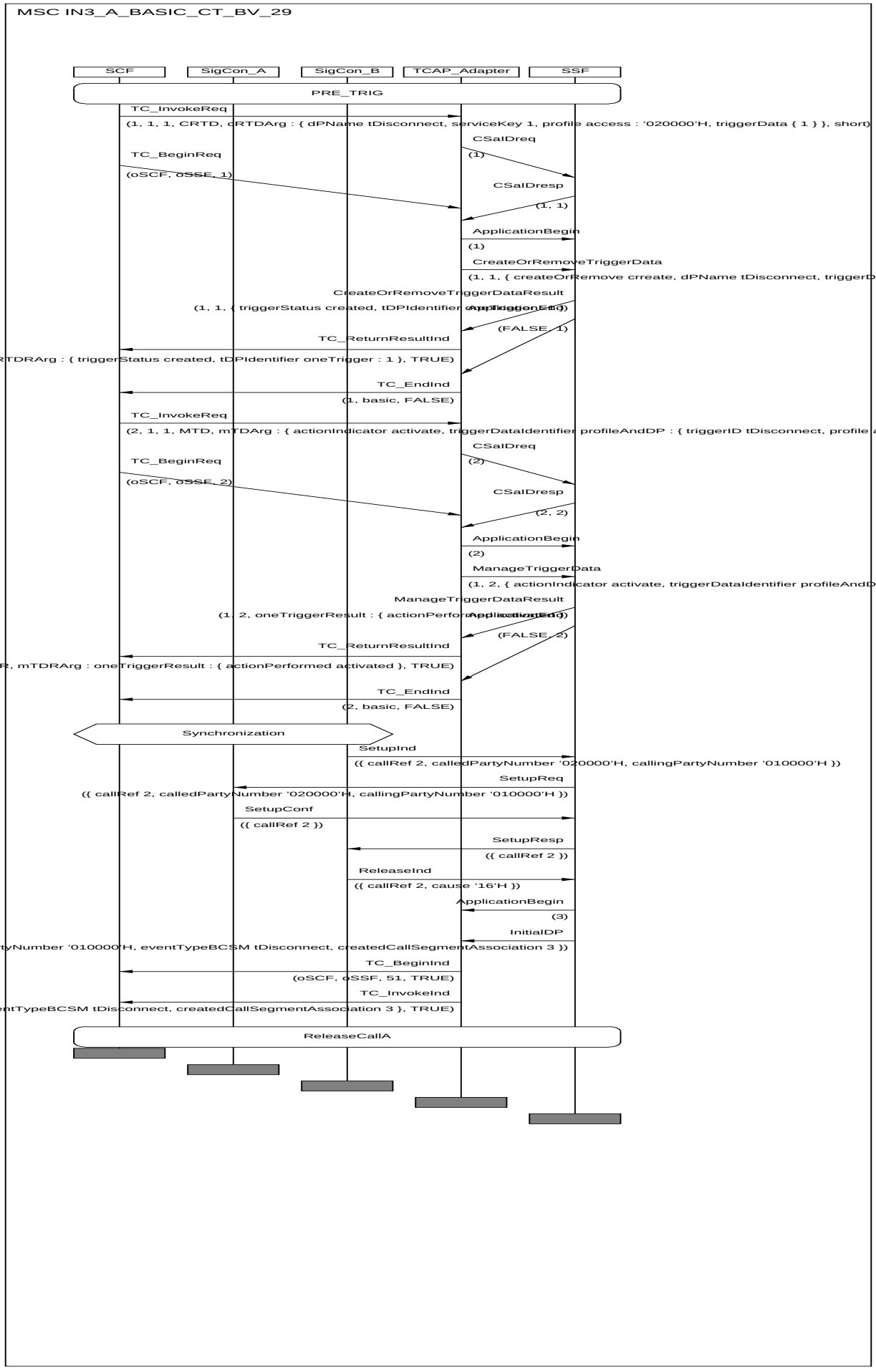
	IN3_A_BASIC_CT_BV_27
Work item no.:	ITEM_BASIC_208
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**tReAnswer**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having connected the call to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**tReAnswer**", when the called user resumes the call (after suspension).
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **tReAnswer**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_2!NetworkSuspendInd CP1_2!NetworkResumeInd L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tReAnswer)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tReAnswer)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_27
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName tReAnswer, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName tReAnswer, triggerDPType tdp_r, servicek
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tReAnswer, profile access : '020000'H },
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tReAn
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 2 })
Synchronization
NetworkSuspendInd
({ callRef 2 })
NetworkSuspendReq
({ callRef 2 })
Synchronization
NetworkResumeInd
({ callRef 2 })
ApplicationBegin
(3)
InitialDP
020000'H, callingPartyNumber '010000'H, eventTypeBCSM tReAnswer, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
mber '010000'H, eventTypeBCSM tReAnswer, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

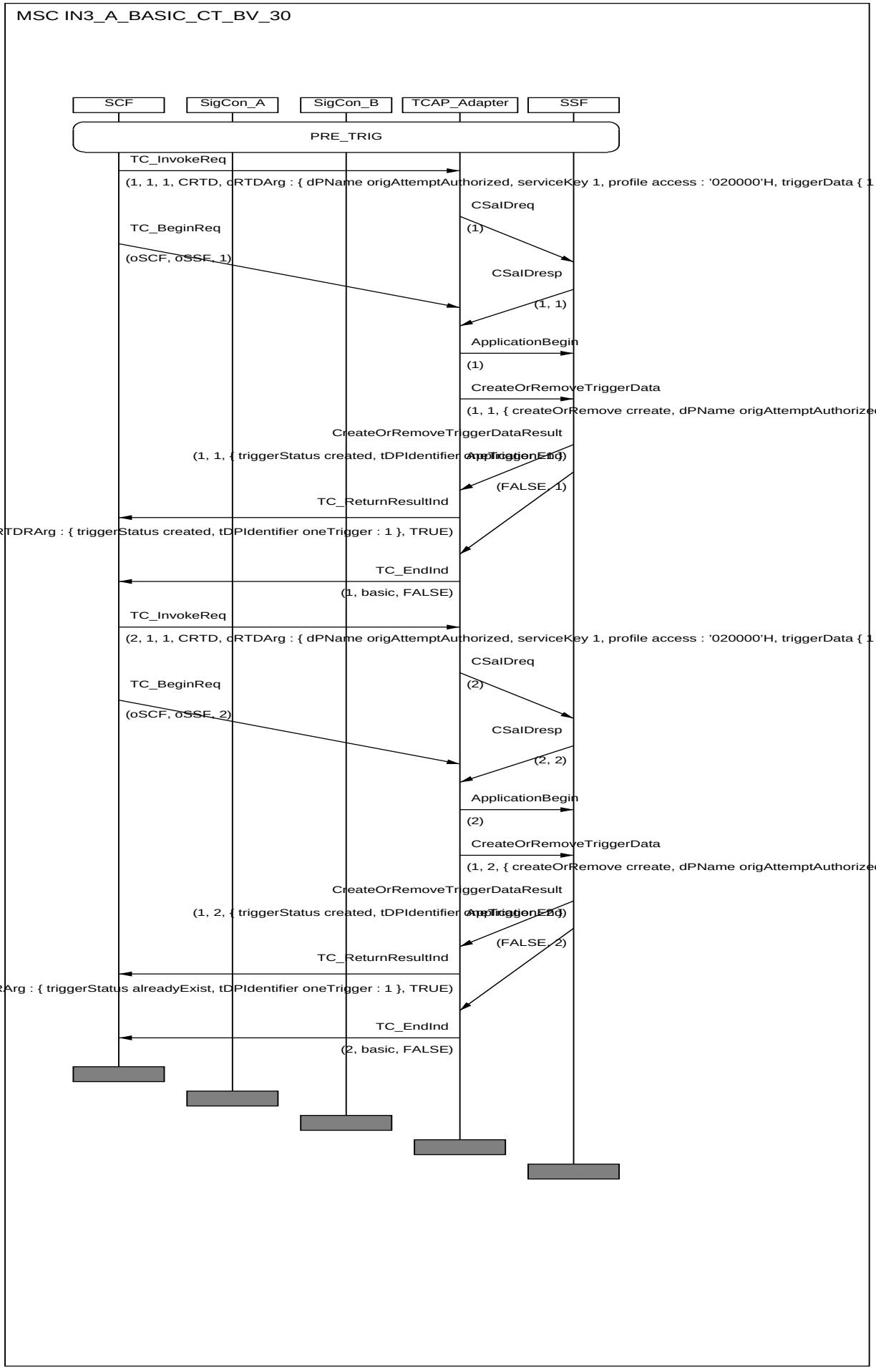
	IN3_A_BASIC_CT_BV_28
Work item no.:	ITEM_BASIC_209
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**tMidCall**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having connected the call to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**tMidCall**", when the called user performs a ServiceFeature interaction.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **tMidCall**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) . CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_2!ServiceFeatureActivation L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tMidCall)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tMidCall)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_28
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName tMidCall, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName tMidCall, triggerDPType tdp_r, serviceKey
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tMidCall, profile access : '020000'H }, tDP
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tMidCa
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated })
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 2 })
Synchronization
ServiceFeatureInd
({ callRef 2 })
ApplicationBegin
(3)
InitialDP
er '020000'H, callingPartyNumber '010000'H, eventTypeBCSM tMidCall, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
Number '010000'H, eventTypeBCSM tMidCall, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA2

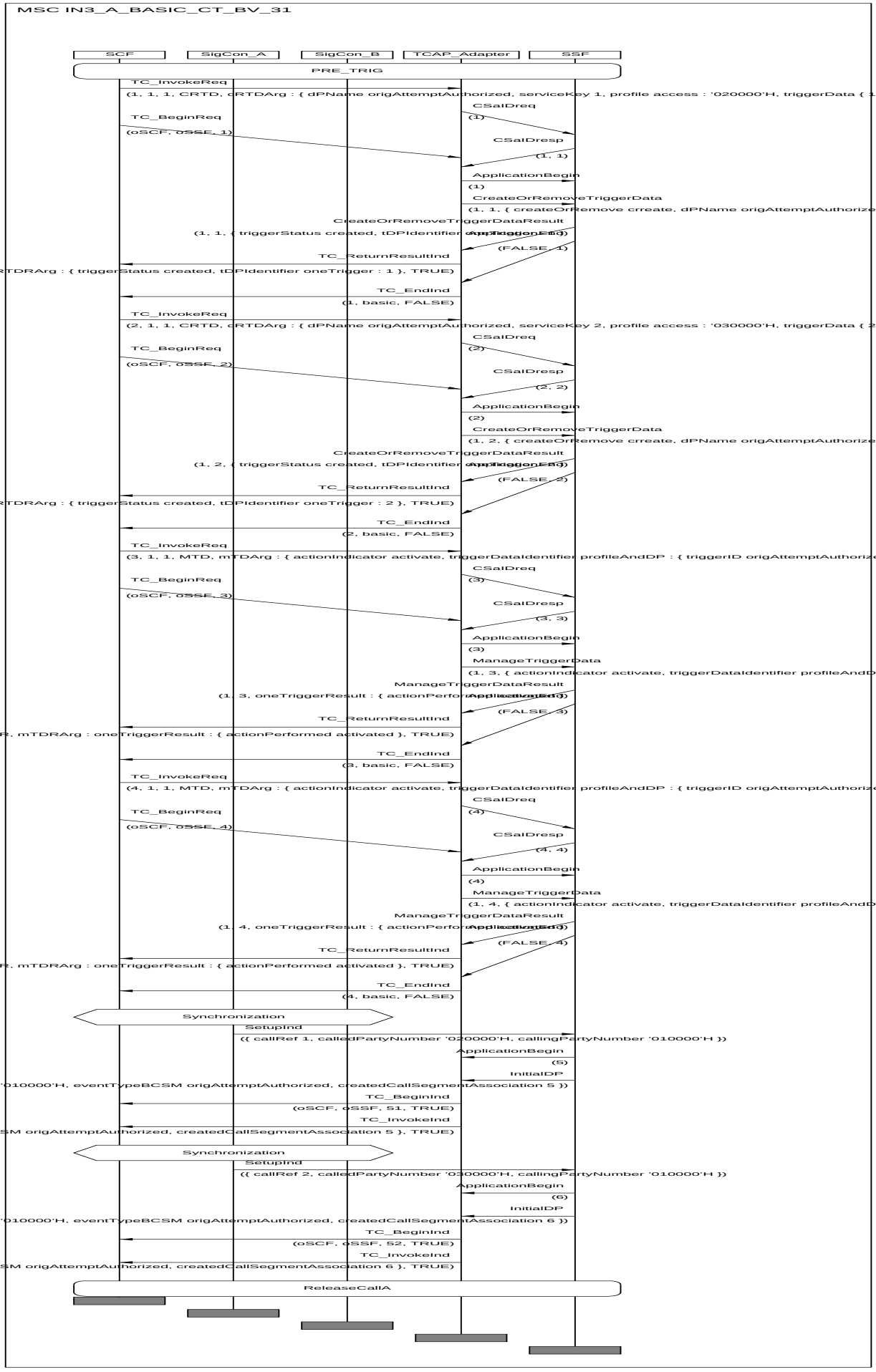
	IN3_A_BASIC_CT_BV_29
Work item no.:	ITEM_BASIC_210
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**tDisconnect**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set by the CreateOrRemoveTriggerData operation and having connected the call to the called user, sends to the SCF an InitialDP invoke component, containing parameters serviceKey = SERVICE_KEY1 and eventTypeBCSM = "**tDisconnect**", when the called user disconnects the call.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **tDisconnect**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp CP1_2!ReleaseInd(Normal cause) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tDisconnect)
Pass criteria	L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, tDisconnect)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_29
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName tDisconnect, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName tDisconnect, triggerDPType tdp_r, service
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tDisconnect, profile access : '020000'H },
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID tDisco
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerfo
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 2 })
ReleaseInd
({ callRef 2, cause '16'H })
ApplicationBegin
(3)
InitialDP
20000'H, callingPartyNumber '010000'H, eventTypeBCSM tDisconnect, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
ber '010000'H, eventTypeBCSM tDisconnect, createdCallSegmentAssociation 3 }, TRUE)
ReleaseCallA

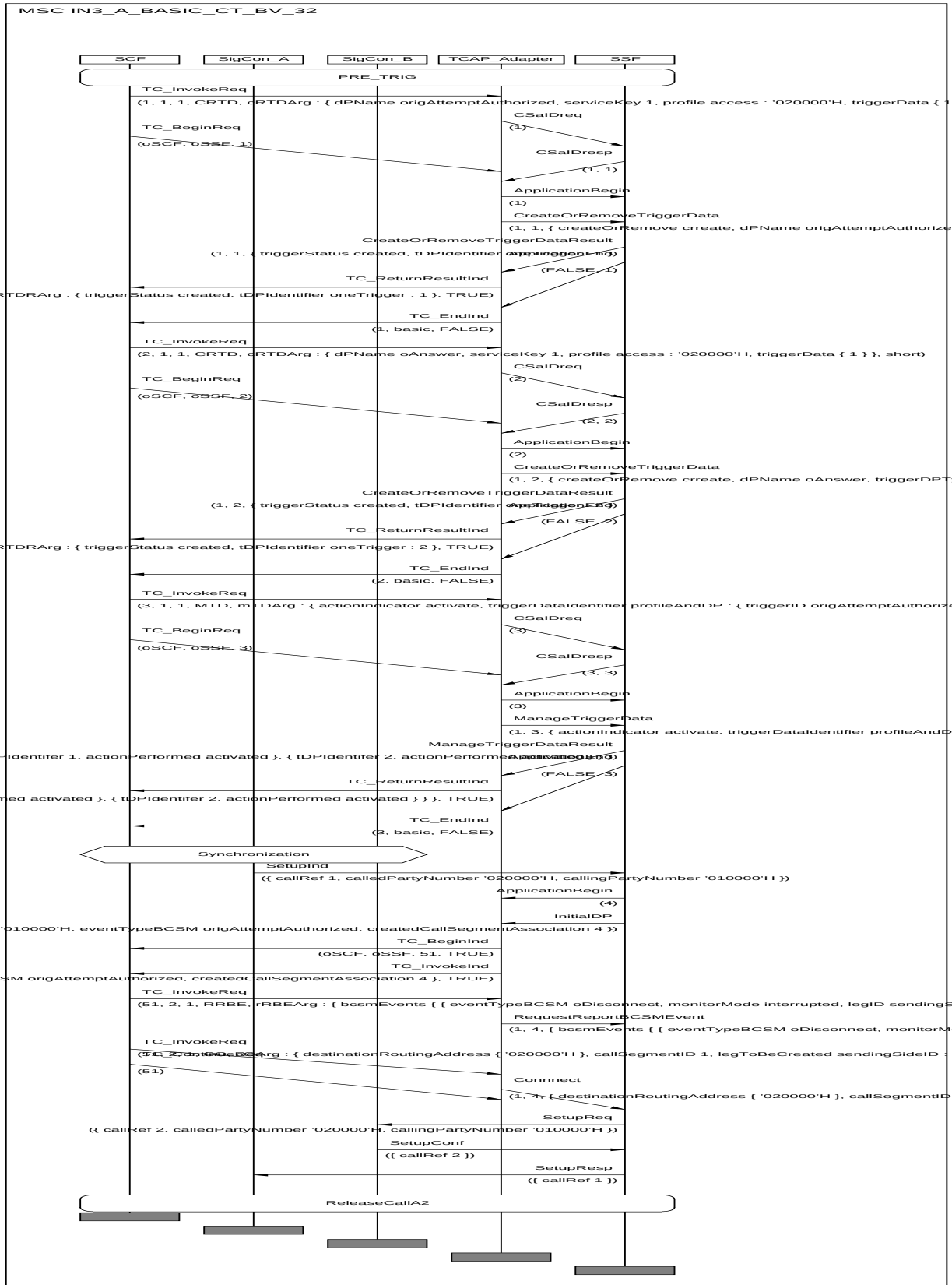
	IN3_A_BASIC_CT_BV_30
Work item no.:	ITEM_BASIC_211
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**origAttemptAuthorized**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier (referred to as tDPIdentifier1). Verify also that the SSF, when receiving again a CreateOrRemoveTriggerData invoke component, containing the same parameters dPName = "**origAttemptAuthorized**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "alreadyExist" and tDPIdentifier1.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(alreadyExist, tDPIdentifier1)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnResult(alreadyExists, tDPIdentifier1)
Postamble:	None


MSC IN3_A_BASIC_CT_BV_30
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
ApplyTriggerEDP
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
ApplyTriggerEDP
(FALSE, 2)
TC_ReturnResultInd
1, CRTD_R, cRTDRArg : { triggerStatus alreadyExist, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(2, basic, FALSE)

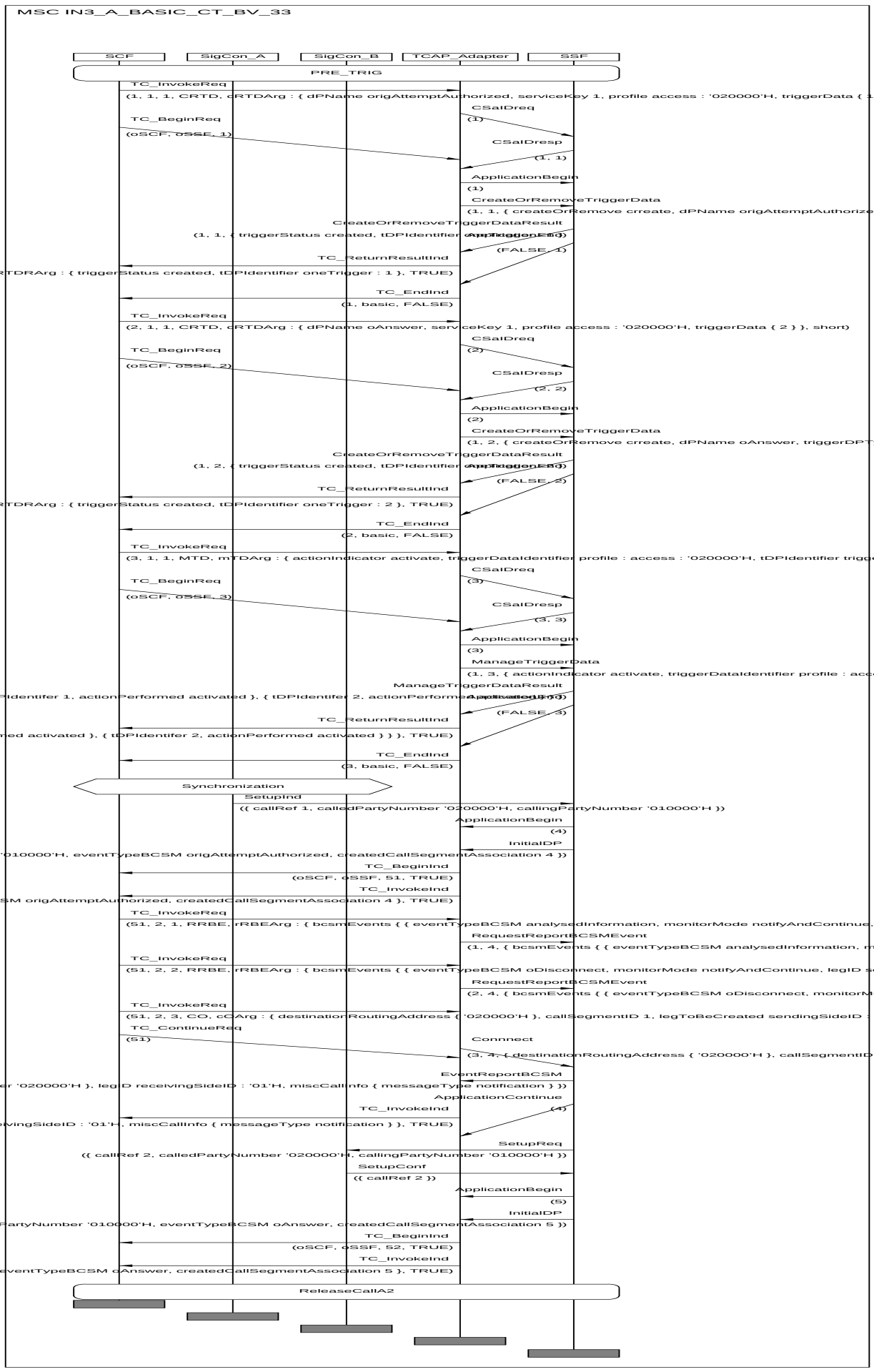
	IN3_A_BASIC_CT_BV_31
Work item no.:	ITEM_BASIC_212
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF can create two different Triggers for the same DP: perform two successive CreateOrRemoveTriggerData operation successfully, in each case using parameters createOrRemove = "create" and dPName = "**origAttemptAuthorized**", but with different valid values of parameters serviceKey, profile and triggerData, yielding tDPIdentifiers referred to as tDPIdentifier1 and tDPIdentifier2 respectively. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPIdentifier1, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier1) and eventTypeBCSM = "**origAttemptAuthorized**". Verify also that the SSF, when receiving a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPIdentifier2, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier2) and eventTypeBCSM = "**origAttemptAuthorized**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1A, PROFILE_ID_1A, TRIGGER_DATA_1A, defaultFaultHandling omitted) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPIdentifier1) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(activated)) L5!ManageTriggerData invoke(activate, profile(PROFILE_ID_1A), oneTrigger tDPIdentifier2) L5!TC-BEGIN L5?TC-END L5?ManageTriggerData returnResult(activated)) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) CP1_1!SetupInd(CALL-DATA-1A) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1A, origAttemptAuthorized)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1A, origAttemptAuthorized)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BV_31
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 2, profile access : '030000'H, triggerData { 2 } }, short
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus created, tDPIdentifier
(FALSE, 2)
TC_ReturnResultInd
2, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 2 }, TRUE)
TC_EndInd
(2, basic, FALSE)
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : 'c
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 3, oneTriggerResult : { actionPerfo
(FALSE, 3)
TC_ReturnResultInd
3, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(3, basic, FALSE)
TC_InvokeReq
(4, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : 'c
CSalDreq
(4)
TC_BeginReq
(oSCF, oSSF, 4)
CSalDresp
(4, 4)
ApplicationBegin
(4)
ManageTriggerData
(1, 4, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 4, oneTriggerResult : { actionPerfo
(FALSE, 4)
TC_ReturnResultInd
4, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(4, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(5)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 5 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 5 }, TRUE)
Synchronization
SetupInd
({ callRef 2, calledPartyNumber '030000'H, callingPartyNumber '010000'H })
ApplicationBegin
(6)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 6 })
TC_BeginInd
(oSCF, oSSF, 52, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 6 }, TRUE)
ReleaseCallA

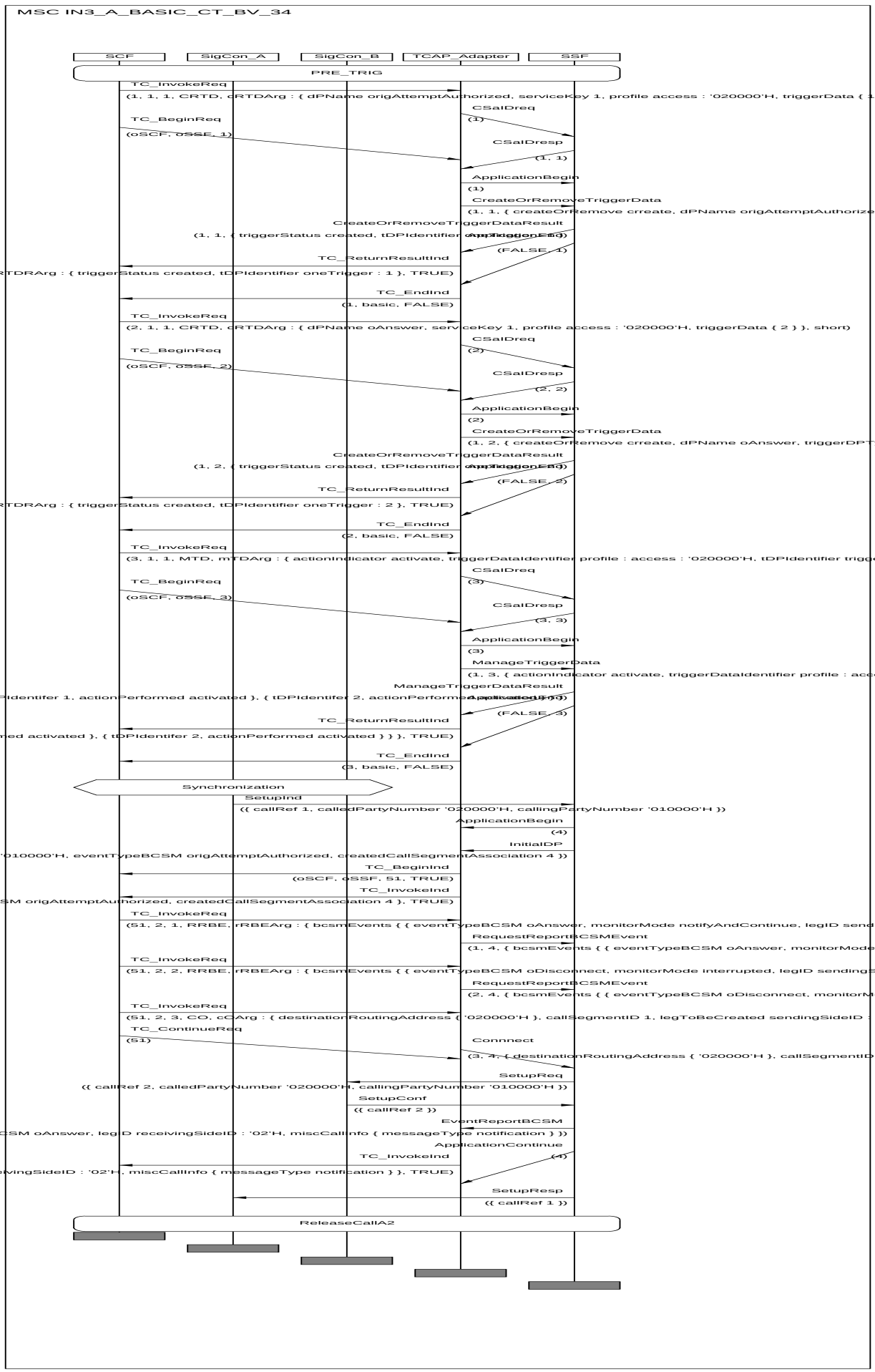
	IN3_A_BASIC_CT_BV_32
Work item no.:	ITEM_BASIC_213
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF can create two different Triggers for different DPs: perform two successive CreateOrRemoveTriggerData operation successfully, using parameter createOrRemove = "create", and dPName = "**origAttemptAuthorized**" respectively dPName = "**oAnswer**". Profile and triggerData should be applicable to the same call. The resulting tDPIdentifiers are referred to as tDPIdentifier1 and tDPIdentifier2 respectively. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPIdentifier1, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier1) and eventTypeBCSM = "**origAttemptAuthorized**". The relationship is maintained when the call continues, by arming DP oDisconnect as EDP-R. Verify that the SSF does not send an InitialDP invoke component when the called party answers the call (the TDP-R on oAnswer is ignored).
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, table 12 scenario 3.a (table 12/6.5.1.3), 11.17, 15.1.1.1.2.2
Selection Cond.	SinglePointOfControl
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, **oAnswer**, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), **triggers** ((tDPIdentifier1),(tDPIdentifier2)) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPIdentifier1), (activated, tDPIdentifier2))) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(1,interrupted,oDisconnect) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) No InitialDP after Connect
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_32
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { dPName oAnswer, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove crreate, dPName oAnswer, triggerDPType tdp_r, serviceKe
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus created, tDPIdentifier
(FALSE, 2)
TC_ReturnResultInd
2, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 2 }, TRUE)
TC_EndInd
(2, basic, FALSE)
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : 'c
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
sult : { results { { tDPIdentifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed
(FALSE, 3)
TC_ReturnResultInd
tifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }, TRUE)
TC_EndInd
(3, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(51)
Connnect
(1, 4, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ReleaseCallA2

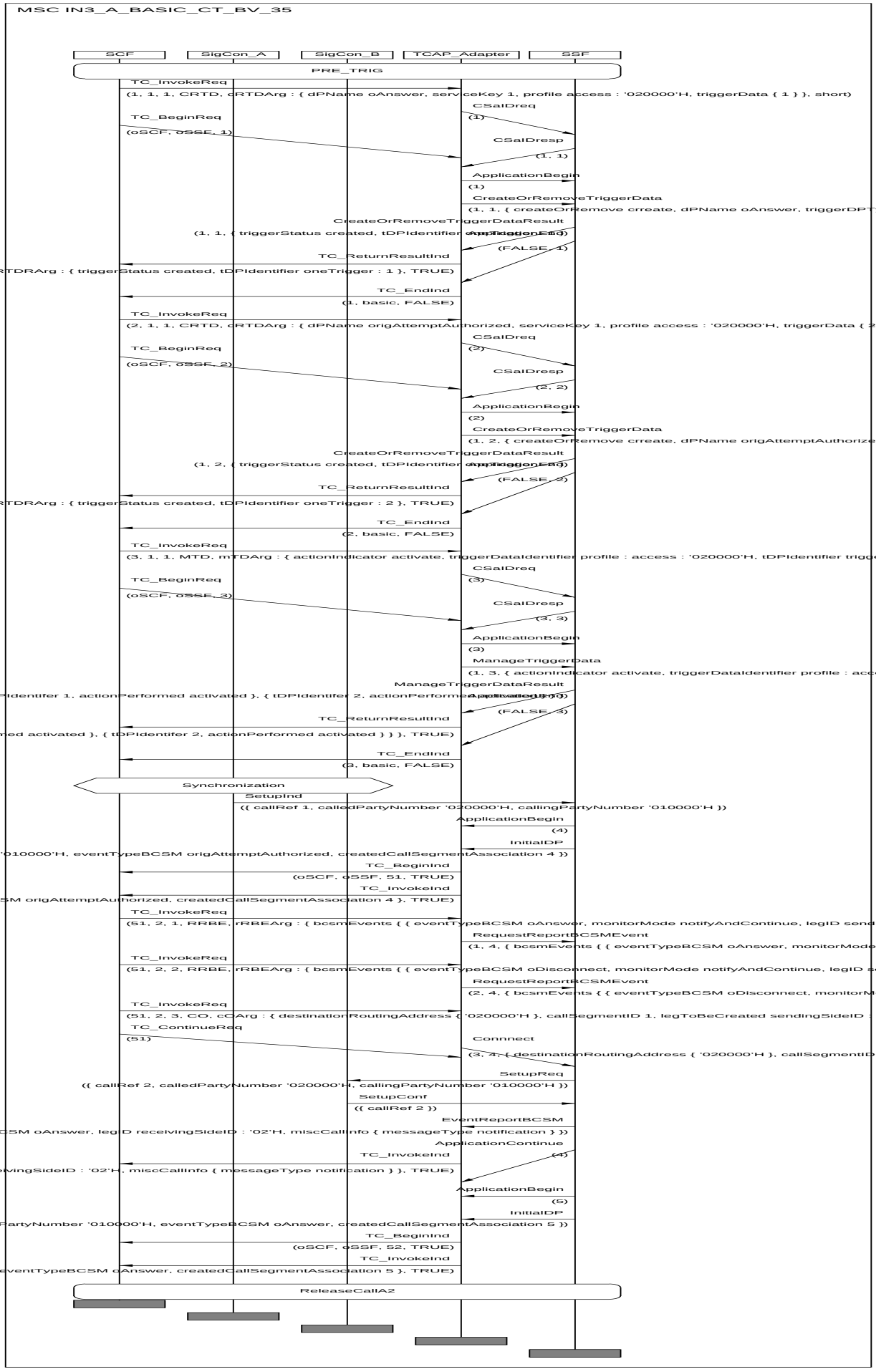
	IN3_A_BASIC_CT_BV_33
Work item no.:	ITEM_BASIC_214
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF can create two different Triggers for different DPs: perform two successive CreateOrRemoveTriggerData operation successfully, using parameter createOrRemove = "create", and dPName = "**origAttemptAuthorized**" respectively dPName = "**oAnswer**". Profile and triggerData should be applicable to the same call. The resulting tDPIdentifiers are referred to as tDPIdentifier1 and tDPIdentifier2 respectively. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPIdentifier1, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier1) and eventTypeBCSM = "**origAttemptAuthorized**". A relationship is established when the call continues, by arming DPs **oAnalyseInformation** and **oDisconnect** as EDP-N. Verify that the SSF sends an EventReportBCSM related to leg 1 and **oAnalyseInformation**, in the context of the first dialog. Verify also that the SSF sends an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier2) and eventTypeBCSM = "**oAnswer**", when the called party answers the call, establishing a new dialog.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, scenario 3.b and 6 (table 12/6.5.1.3), 11.17, 15.1.1.1.2.2
Selection Cond.	SinglePointOfControl
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAnswer, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), triggers ((tDPIdentifier1),(tDPIdentifier2)) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPIdentifier1), (activated, tDPIdentifier2))) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(1,notifyAndContinue,analysedInformation) L1!RequestReportBCSMEvent(1, notifyAndContinue,oDisconnect) L1!Connect(2,1) L1?EventReportBCSM(1,analysedInformation) CP1_2?SetUpReq CP1_2!SetUpConf L4?TC-BEGIN L4?InitialDP invoke(SERVICE_KEY1, oAnswer)
Pass criteria	L4?TC-BEGIN L4?InitialDP invoke(SERVICE_KEY1, oAnswer)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_33
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { dPName oAnswer, serviceKey 1, profile access : '020000'H, triggerData { 2 } }, short)
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove crreate, dPName oAnswer, triggerDPType tdp_r, serviceKe
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus created, tDPIdentifier
(FALSE, 2)
TC_ReturnResultInd
2, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 2 }, TRUE)
TC_EndInd
(2, basic, FALSE)
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDPIdentifier triggers : { { tDPIdentifier
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDP
ManageTriggerDataResult
sult : { results { { tDPIdentifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed
(FALSE, 3)
TC_ReturnResultInd
tifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }, TRUE)
TC_EndInd
(3, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM analysedInformation, monitorMode notifyAndContinue, legID sendingSideID
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM analysedInformation, monitorMode notifyAn
TC_InvokeReq
(51, 2, 2, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode notifyAndContinue, legID sendingSideID : '01'H
RequestReportBCSMEvent
(2, 4, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode notifyAndContin
TC_InvokeReq
(51, 2, 3, CC, cCArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
TC_ContinueReq
(51)
Connnect
(3, 4, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated
EventReportBCSM
: { calledPartynumber '020000'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } })
ApplicationContinue
(4)
TC_InvokeInd
0000'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } }, TRUE)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
ApplicationBegin
(5)
InitialDP
r '020000'H, callingPartyNumber '010000'H, eventTypeBCSM oAnswer, createdCallSegmentAssociation 5 })
TC_BeginInd
(oSCF, oSSF, 52, TRUE)
TC_InvokeInd
Number '010000'H, eventTypeBCSM oAnswer, createdCallSegmentAssociation 5 }, TRUE)
ReleaseCallA2

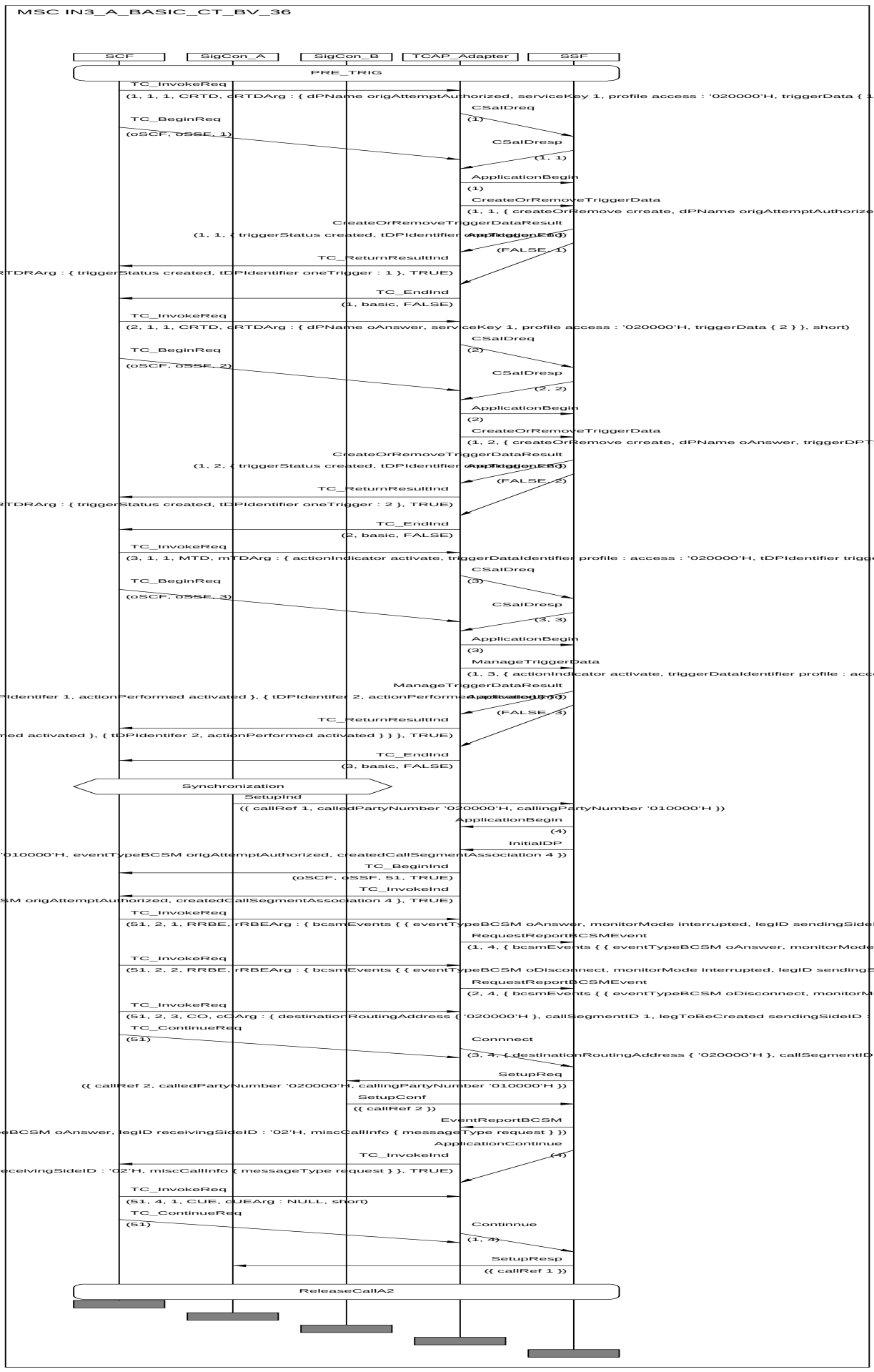
	IN3_A_BASIC_CT_BV_34
Work item no.:	ITEM_BASIC_215
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF can create two different Triggers for the different DPs: perform two successive CreateOrRemoveTriggerData operation successfully, using parameter createOrRemove = "create", and dPName = "**origAttemptAuthorized**" respectively dPName = "**oAnswer**". Profile and triggerData should be applicable to the same call. The resulting tDPIdentifiers are referred to as tDPIdentifier1 and tDPIdentifier2 respectively. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPIdentifier1, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier1) and eventTypeBCSM = "**origAttemptAuthorized**". A relationship is established when the call continues, by arming DPs **oAnswer** as EDP-N and **oDisconnect** as EDP-R. Verify that the SSF sends an EventReportBCSM related to leg 2 and **oAnswer**, in the context of the first dialog, when the called party answers the call. Verify also that the SSF does not send an (additional) InitialDP invoke component (TDP-R on oAnswer is ignored).
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, scenario 11.a (table 12/6.5.1.3), 11.17, 15.1.1.1.2.2
Selection Cond.	SinglePointOfControl
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAnswer, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), triggers ((tDPIdentifier1),(tDPIdentifier2)) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPIdentifier1), (activated, tDPIdentifier2))) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(2,notifyAndContinue,oAnswer) L1!RequestReportBCSMEvent(1,interrupted,oDisconnect) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf L1?EventReportBCSM(2,oAnswer) CP1_1?SetUpResp NOTE: EventReportBCSM(2,oAnswer) and SetUpResp may appear in any order.
Pass criteria	L1?EventReportBCSM(2,oAnswer) No InitialDP after EventReportBCSM
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_34
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
ApplicationEnd
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { dPName oAnswer, serviceKey 1, profile access : '020000'H, triggerData { 2 } }, short)
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove crreate, dPName oAnswer, triggerDPType tdp_r, serviceKe
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus created, tDPIdentifier oneTrigger : 2 }, TRUE)
ApplicationEnd
(FALSE, 2)
TC_ReturnResultInd
2, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 2 }, TRUE)
TC_EndInd
(2, basic, FALSE)
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDPIdentifier triggers : { { tDPIdentifier
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDP
ManageTriggerDataResult
sult : { results { { tDPIdentifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }, TRUE)
ApplicationEnd
(FALSE, 3)
TC_ReturnResultInd
tifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }, TRUE)
TC_EndInd
(3, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode notifyAndContinue, legID sendingSideID : '02'H } } }
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode notifyAndContinue,
TC_InvokeReq
(51, 2, 2, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(2, 4, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID
TC_InvokeReq
(51, 2, 3, CO, cCArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
TC_ContinueReq
(51)
Connnect
(3, 4, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
4, { eventTypeBCSM oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType notification } })
ApplicationContinue
(4)
TC_InvokeInd
oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType notification } }, TRUE)
SetupResp
({ callRef 1 })
ReleaseCallA2

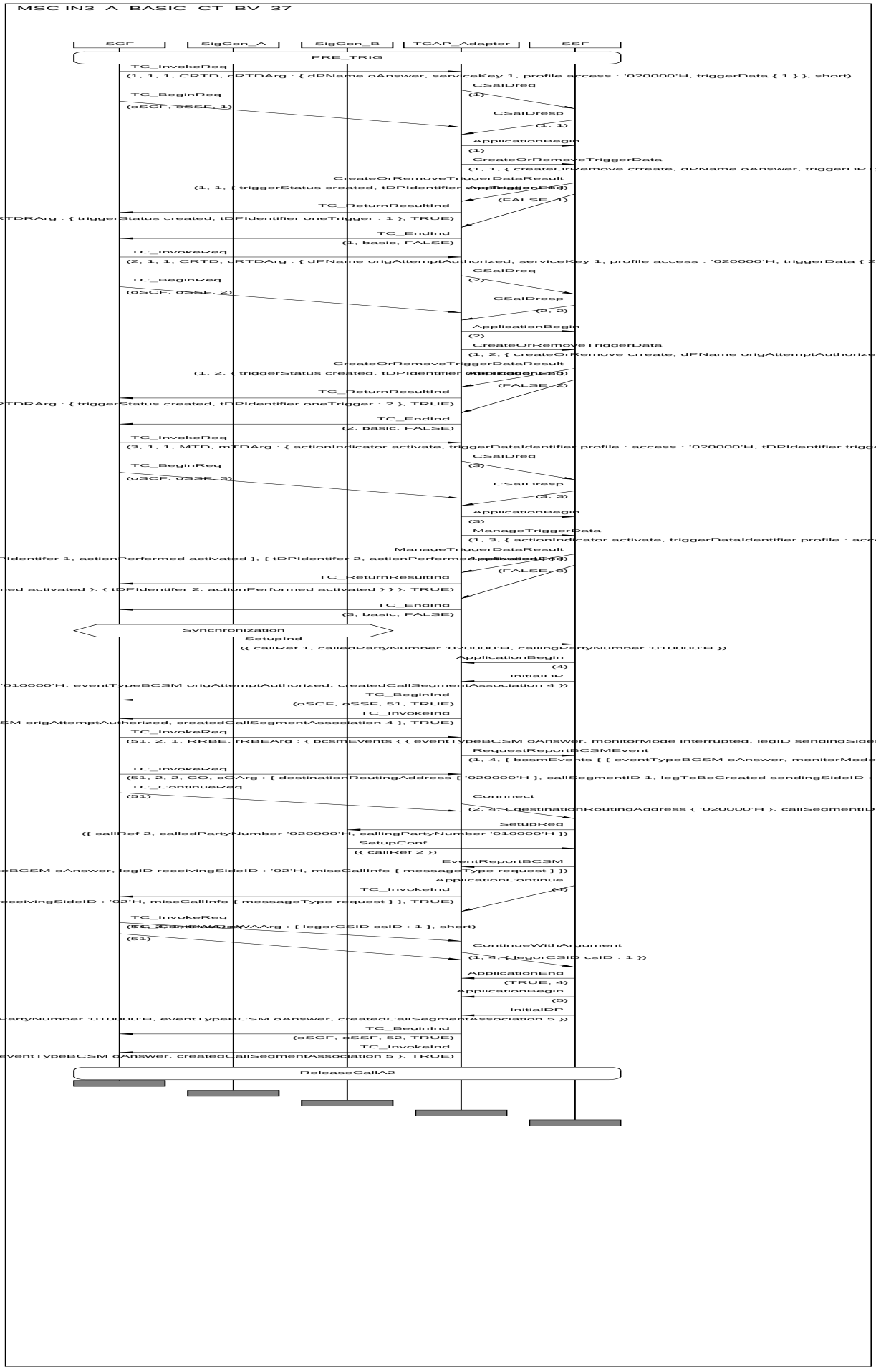
	IN3_A_BASIC_CT_BV_35
Work item no.:	ITEM_BASIC_216
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF can create two different Triggers for the different DPs: perform two successive CreateOrRemoveTriggerData operation successfully, using parameter createOrRemove = "create", and dPName = "**origAttemptAuthorized**" respectively dPName = "**oAnswer**". Profile and triggerData should be applicable to the same call. The resulting tDPIdentifiers are referred to as tDPIdentifier1 and tDPIdentifier2 respectively. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPIdentifier1, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier1) and eventTypeBCSM = "**origAttemptAuthorized**". A relationship is established when the call continues, by arming DPs **oAnswer** and **oDisconnect** as EDP-N. Verify that the SSF sends an EventReportBCSM related to leg 2 and **oAnswer**, in the context of the first dialog, when the called party answers the call. Verify also that the SSF afterwards sends an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier2) and eventTypeBCSM = "**oAnswer**", establishing a new dialog.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, scenario 11.b (table 12/6.5.1.3), 11.17, 15.1.1.1.2.2
Selection Cond.	SinglePointOfControl
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAnswer, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), triggers ((tDPIdentifier1),(tDPIdentifier2)) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPIdentifier1), (activated, tDPIdentifier2))) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(2, notifyAndContinue, oAnswer) L1!RequestReportBCSMEvent(1,notifyAndContinue,oDisconnect) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf L1?EventReportBCSM(2,oAnswer) L5?TC-BEGIN L5?InitialDP invoke(SERVICE_KEY1, oAnswer)
Pass criteria	L1?EventReportBCSM(2,oAnswer) L5?TC-BEGIN L5?InitialDP invoke(SERVICE_KEY1, oAnswer)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_35
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oAnswer, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName oAnswer, triggerDPType tdp_r, serviceKe
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 2 } }, short
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus created, tDPIdentifier
(FALSE, 2)
TC_ReturnResultInd
2, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 2 }, TRUE)
TC_EndInd
(2, basic, FALSE)
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDPIdentifier triggers : { { tDPIdentifier
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDP
ManageTriggerDataResult
sult : { results { { tDPIdentifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed
(FALSE, 3)
TC_ReturnResultInd
tifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }, TRUE)
TC_EndInd
(3, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode notifyAndContinue, legID sendingSideID : '02'H } } }
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode notifyAndContinue,
TC_InvokeReq
(51, 2, 2, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode notifyAndContinue, legID sendingSideID : '02'H
RequestReportBCSMEvent
(2, 4, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode notifyAndContin
TC_InvokeReq
(51, 2, 3, CO, cCArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
TC_ContinueReq
(51)
Connnect
(3, 4, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
4, { eventTypeBCSM oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType notification } })
ApplicationContinue
(4)
TC_InvokeInd
oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType notification } }, TRUE)
ApplicationBegin
(5)
InitialDP
r '020000'H, callingPartyNumber '010000'H, eventTypeBCSM oAnswer, createdCallSegmentAssociation 5 })
TC_BeginInd
(oSCF, oSSF, 52, TRUE)
TC_InvokeInd
Number '010000'H, eventTypeBCSM oAnswer, createdCallSegmentAssociation 5 }, TRUE)
ReleaseCallA2

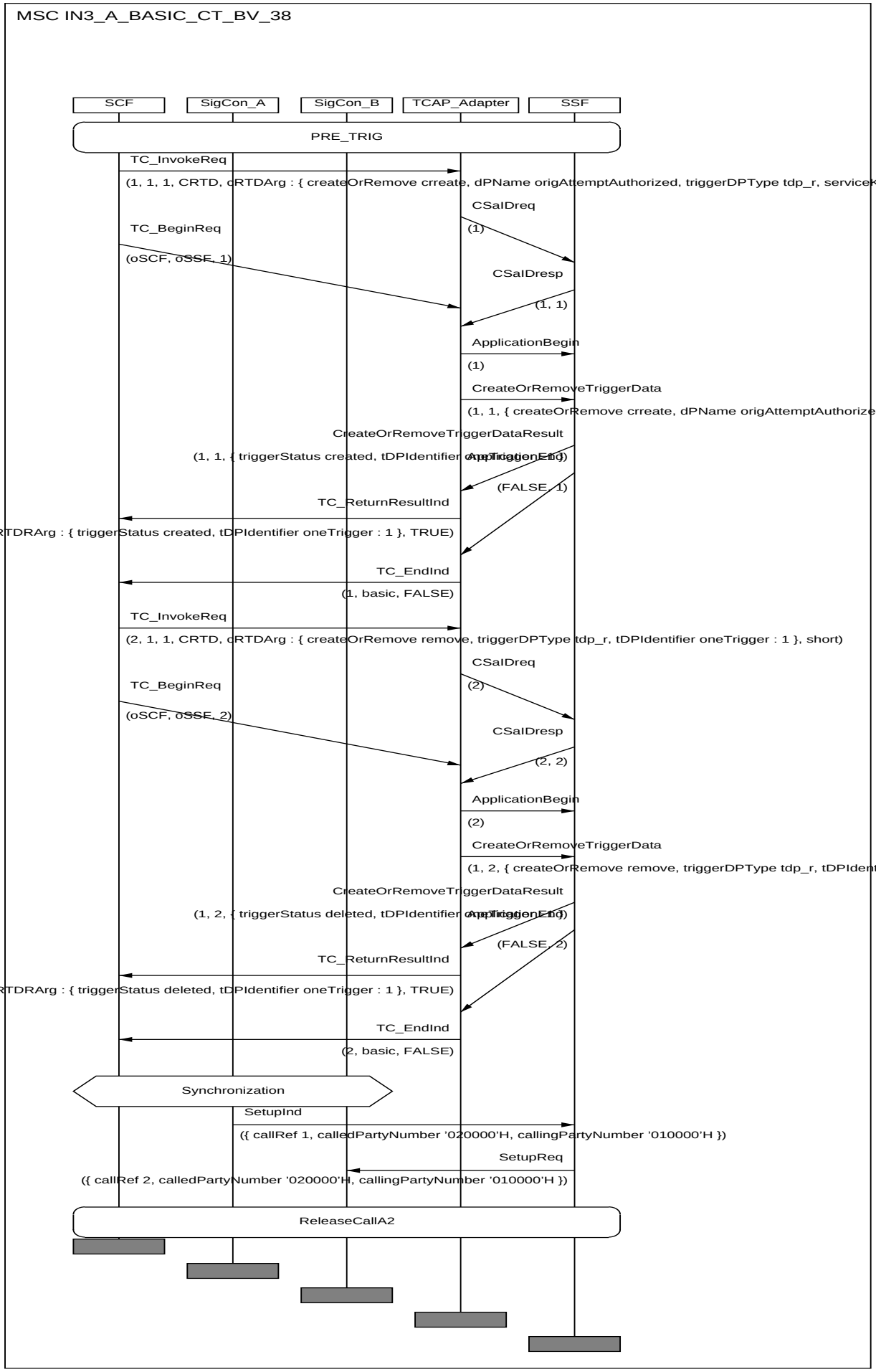
	IN3_A_BASIC_CT_BV_36
Work item no.:	ITEM_BASIC_217
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF can create two different Triggers for the different DPs: perform two successive CreateOrRemoveTriggerData operation successfully, using parameter createOrRemove = "create", and dPName = "**origAttemptAuthorized**" respectively dPName = "**oAnswer**". Profile and triggerData should be applicable to the same call. The resulting tDPIdentifiers are referred to as tDPIdentifier1 and tDPIdentifier2 respectively. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPIdentifier1, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier1) and eventTypeBCSM = "**origAttemptAuthorized**". A relationship is established when the call continues, by arming DPs **oAnswer** and **oDisconnect** as EDP-R. Verify that the SSF sends an EventReportBCSM related to leg 2 and **oAnswer**, in the context of the first dialog, when the called party answers the call. Verify also that the SSF after resumption does not send an additional InitialDP invoke component (the TDP-R on oAnswer is ignored, because the relationship persists).
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, scenario 13.a (table 12/6.5.1.3), 11.17, 15.1.1.1.2.2
Selection Cond.	SinglePointOfControl
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAnswer, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), triggers ((tDPIdentifier1),(tDPIdentifier2)) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPIdentifier1), (activated, tDPIdentifier2))) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(2,interrupted,oAnswer) L1!RequestReportBCSMEvent(1,interrupted,oDisconnect) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf L1?EventReportBCSM(2,oAnswer) L1!Continue CP1_1?SetUpResp
Pass criteria	L1?EventReportBCSM(2,oAnswer) No InitialDP after EventReportBCSM
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_36
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { dPName oAnswer, serviceKey 1, profile access : '020000'H, triggerData { 2 } }, short)
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove crreate, dPName oAnswer, triggerDPType tdp_r, serviceKe
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus created, tDPIdentifier
(FALSE, 2)
TC_ReturnResultInd
2, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 2 }, TRUE)
TC_EndInd
(2, basic, FALSE)
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDPIdentifier triggers : { { tDPIdentifier
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDP
ManageTriggerDataResult
sult : { results { { tDPIdentifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed
(FALSE, 3)
TC_ReturnResultInd
tifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }, TRUE)
TC_EndInd
(3, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, short
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID se
TC_InvokeReq
(51, 2, 2, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(2, 4, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(51, 2, 3, CO, cCArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
TC_ContinueReq
(51)
Connnect
(3, 4, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
4, { eventTypeBCSM oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
(4)
TC_InvokeInd
M oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
TC_InvokeReq
(51, 4, 1, CUE, cUEArg : NULL, short)
TC_ContinueReq
(51)
Continnue
(1, 4)
SetupResp
({ callRef 1 })
ReleaseCallA2

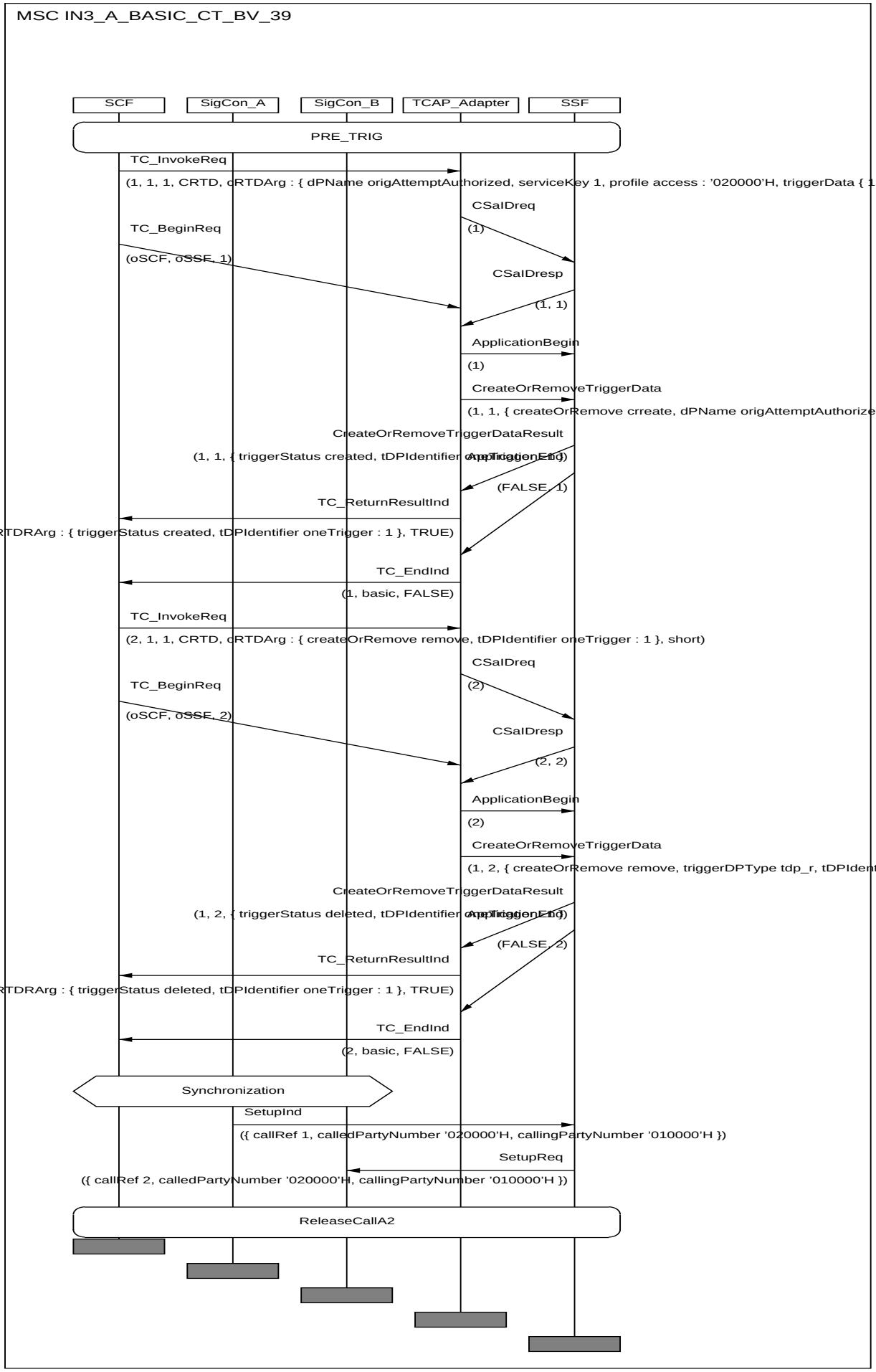
	IN3_A_BASIC_CT_BV_37
Work item no.:	ITEM_BASIC_218
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF can create two different Triggers for the different DPs: perform two successive CreateOrRemoveTriggerData operation successfully, using parameter createOrRemove = "create", and dPName = "**origAttemptAuthorized**" respectively dPName = "**oAnswer**". Profile and triggerData should be applicable to the same call. The resulting tDPIdentifiers are referred to as tDPIdentifier1 and tDPIdentifier2 respectively. Verify also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPIdentifier1, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier1) and eventTypeBCSM = "**origAttemptAuthorized**". A relationship is established when the call continues, by arming DPs **oAnswer** (only) as EDP-R. Verify that the SSF sends an EventReportBCSM related to leg 2 and **oAnswer**, in the context of the first dialog, when the called party answers the call. Verify also that the SSF afterwards sends an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier2) and eventTypeBCSM = "**oAnswer**", establishing a new dialog.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, scenario 13.a (table 12/6.5.1.3), 11.17, 15.1.1.1.2.2
Selection Cond.	SinglePointOfControl
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttemptAuthorized, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAnswer, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier2) L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), triggers ((tDPIdentifier1),(tDPIdentifier2)) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPIdentifier1), (activated, tDPIdentifier2))) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(2,interrupted,oAnswer) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf L1?EventReportBCSM(2,oAnswer) L1!ContinueWithArgument(CSID = 1) L5?TC-BEGIN L5?InitialDP invoke(SERVICE_KEY1, oAnswer)
Pass criteria	L1?EventReportBCSM(2,oAnswer) L5?TC-BEGIN L5?InitialDP invoke(SERVICE_KEY1, oAnswer)
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_37
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oAnswer, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove create, dPName oAnswer, triggerDPType tdp_r, serviceKe
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 2 } }, short
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove create, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus created, tDPIdentifier
(FALSE, 2)
TC_ReturnResultInd
2, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 2 }, TRUE)
TC_EndInd
(2, basic, FALSE)
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDPIdentifier triggers : { { tDPIdentifier
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDP
ManageTriggerDataResult
sult : { results { { tDPIdentifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }
(FALSE, 3)
TC_ReturnResultInd
tifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }, TRUE)
TC_EndInd
(3, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
InitialDP
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeInd
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, short
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID se
TC_InvokeReq
(51, 2, 2, CO, cCArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
TC_ContinueReq
(51)
Connnect
(2, 4, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
4, { eventTypeBCSM oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
(4)
TC_InvokeInd
M oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
TC_InvokeReq
(51, 2, ContinueWithArgumentArg : { legorCSID csID : 1 }, short)
(51)
ContinueWithArgument
(1, 4, { legorCSID csID : 1 })
ApplicationEnd
(TRUE, 4)
ApplicationBegin
(5)
InitialDP
r '020000'H, callingPartyNumber '010000'H, eventTypeBCSM oAnswer, createdCallSegmentAssociation 5 })
TC_BeginInd
(oSCF, oSSF, 52, TRUE)
Number '010000'H, eventTypeBCSM oAnswer, createdCallSegmentAssociation 5 }, TRUE)
TC_InvokeInd
ReleaseCallA2

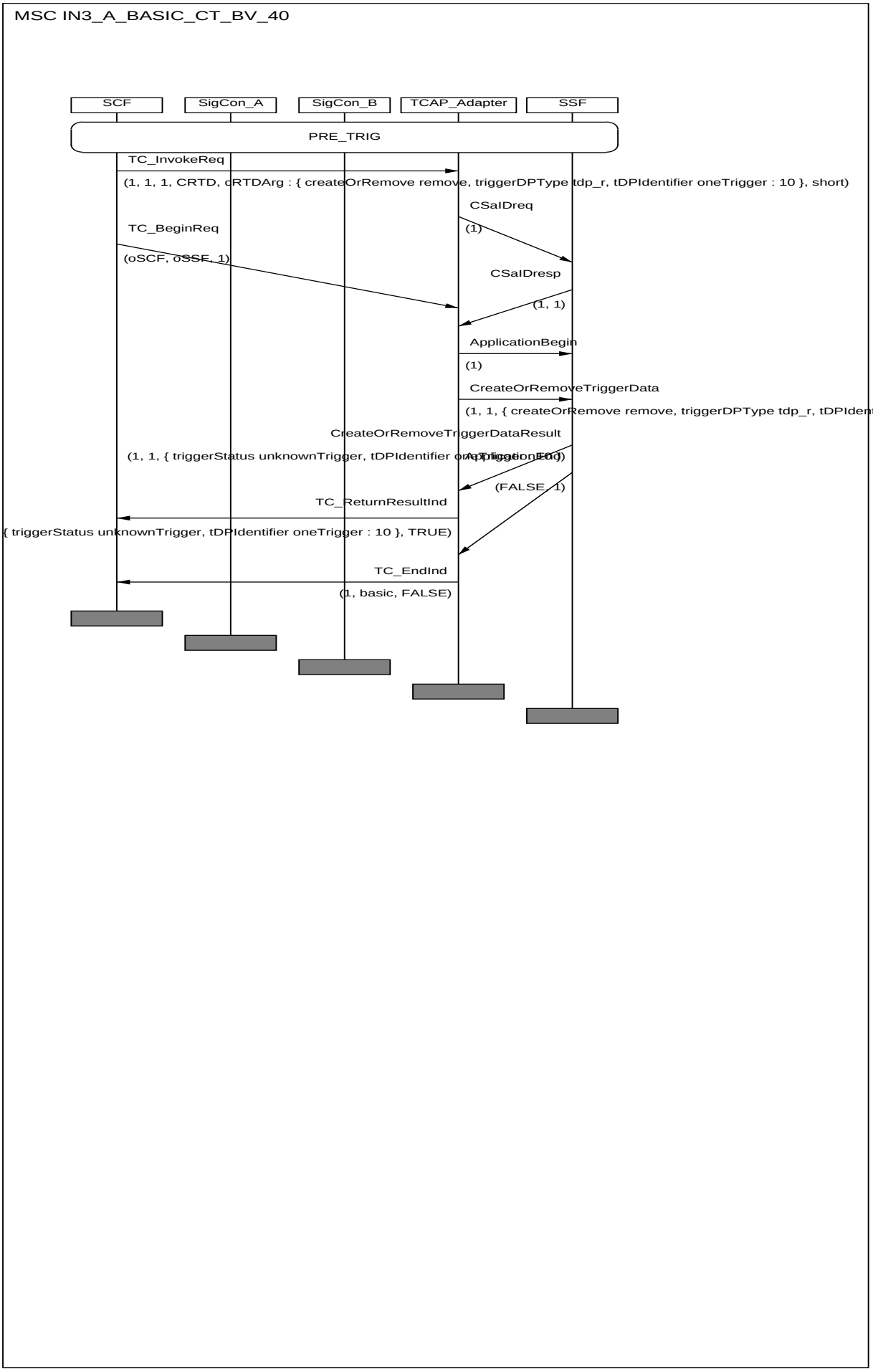
	IN3_A_BASIC_CT_BV_38
Work item no.:	ITEM_BASIC_219
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters createOrRemove = "create", dPName = "**origAttempt**", triggerDPType = "tdp-r", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1, triggerData = TRIGGER_DATA_1 and defaultFaultHandling = "resumeCallProcessing", sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier (referred to as tDPIdentifier1). Verify also that the SSF, after having created these trigger data, and having received a CreateOrRemoveTriggerData invoke component containing parameters createOrRemove = "remove", dPName = "**origAttempt**", triggerDPType = "tdp-r", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1, triggerData = TRIGGER_DATA_1, defaultFaultHandling = "resumeCallProcessing" and tDPIdentifier = tDPIdentifier1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "deleted" and tDPIdentifier1. Check also that the SSF, when receiving after trigger activation a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set and removed by the previous CreateOrRemoveTriggerData operations, does **not** send to the SCF an InitialDP invoke component.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(create, origAttempt, tdp-r, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, resumeCallProcessing) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!CreateOrRemoveTriggerData invoke(**remove**, tdp-r, tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(deleted, tDPIdentifier1) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq
Pass criteria	L3?CreateOrRemoveTriggerData returnResult(deleted, tDPIdentifier1) CP1_2?SetUpReq
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_38
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp_r, serviceKey 1, profile access
CSaIDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSaIDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
ApplicationEnd
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { createOrRemove remove, triggerDPType tdp_r, tDPIdentifier oneTrigger : 1 }, short)
CSaIDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSaIDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove remove, triggerDPType tdp_r, tDPIdentifier oneTrigger : 1 }
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus deleted, tDPIdentifier oneTrigger : 1 }, TRUE)
ApplicationEnd
(FALSE, 2)
TC_ReturnResultInd
2, 1, CRTD_R, cRTDRArg : { triggerStatus deleted, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseCallA2

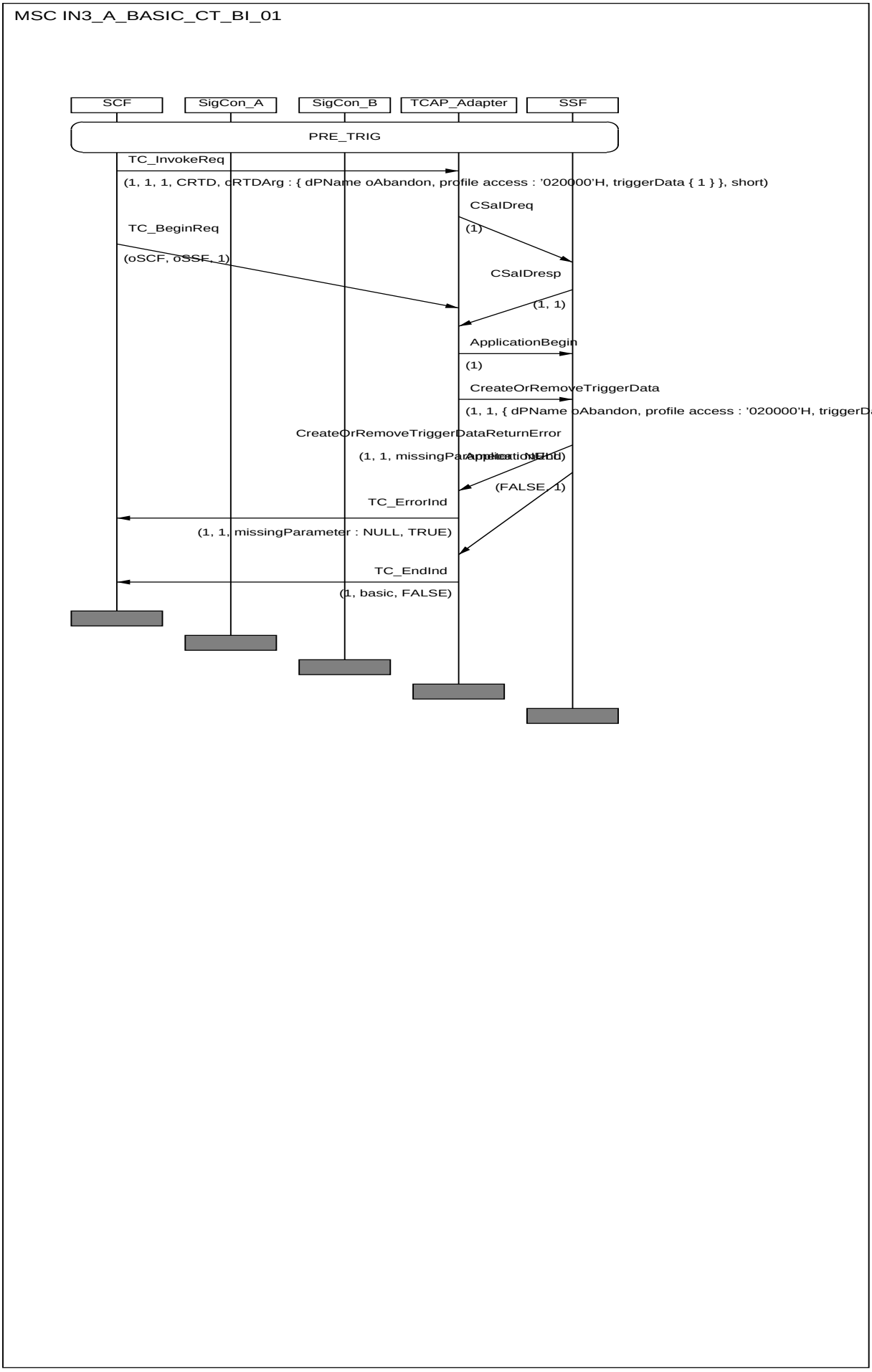
	IN3_A_BASIC_CT_BV_39
Work item no.:	ITEM_BASIC_220
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**origAttempt**", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1, and triggerData = TRIGGER_DATA_1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "created" and a tDPIdentifier (referred to as tDPIdentifier1). Verify also that the SSF, after having created these trigger data, and having received a CreateOrRemoveTriggerData invoke component containing parameters createOrRemove = "remove" and tDPIdentifier = tDPIdentifier1, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "deleted" and tDPIdentifier1. Check also that the SSF, when receiving a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data set and removed by the previous CreateOrRemoveTriggerData operations, does **not** send to the SCF an InitialDP invoke component.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, origAttempt, triggerDPType omitted, SERVICE_KEY1, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnResult(created, tDPIdentifier1) L3!CreateOrRemoveTriggerData invoke(remove, triggerDPType omitted, tDPIdentifier1) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(deleted, tDPIdentifier1) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetUpReq
Pass criteria	L3?CreateOrRemoveTriggerData returnResult(deleted, tDPIdentifier1) CP1_2?SetUpReq
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_CT_BV_39
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName origAttemptAuthorized, serviceKey 1, profile access : '020000'H, triggerData { 1 } }, short
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove crreate, dPName origAttemptAuthorized, triggerDPType tdp
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
ApplicationEnd
(FALSE, 1)
TC_ReturnResultInd
1, 1, CRTD_R, cRTDRArg : { triggerStatus created, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(1, basic, FALSE)
TC_InvokeReq
(2, 1, 1, CRTD, cRTDArg : { createOrRemove remove, tDPIdentifier oneTrigger : 1 }, short)
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
CreateOrRemoveTriggerData
(1, 2, { createOrRemove remove, triggerDPType tdp_r, tDPIdentifier oneTrigger : 1 }
CreateOrRemoveTriggerDataResult
(1, 2, { triggerStatus deleted, tDPIdentifier oneTrigger : 1 }, TRUE)
ApplicationEnd
(FALSE, 2)
TC_ReturnResultInd
2, 1, CRTD_R, cRTDRArg : { triggerStatus deleted, tDPIdentifier oneTrigger : 1 }, TRUE)
TC_EndInd
(2, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseCallA2

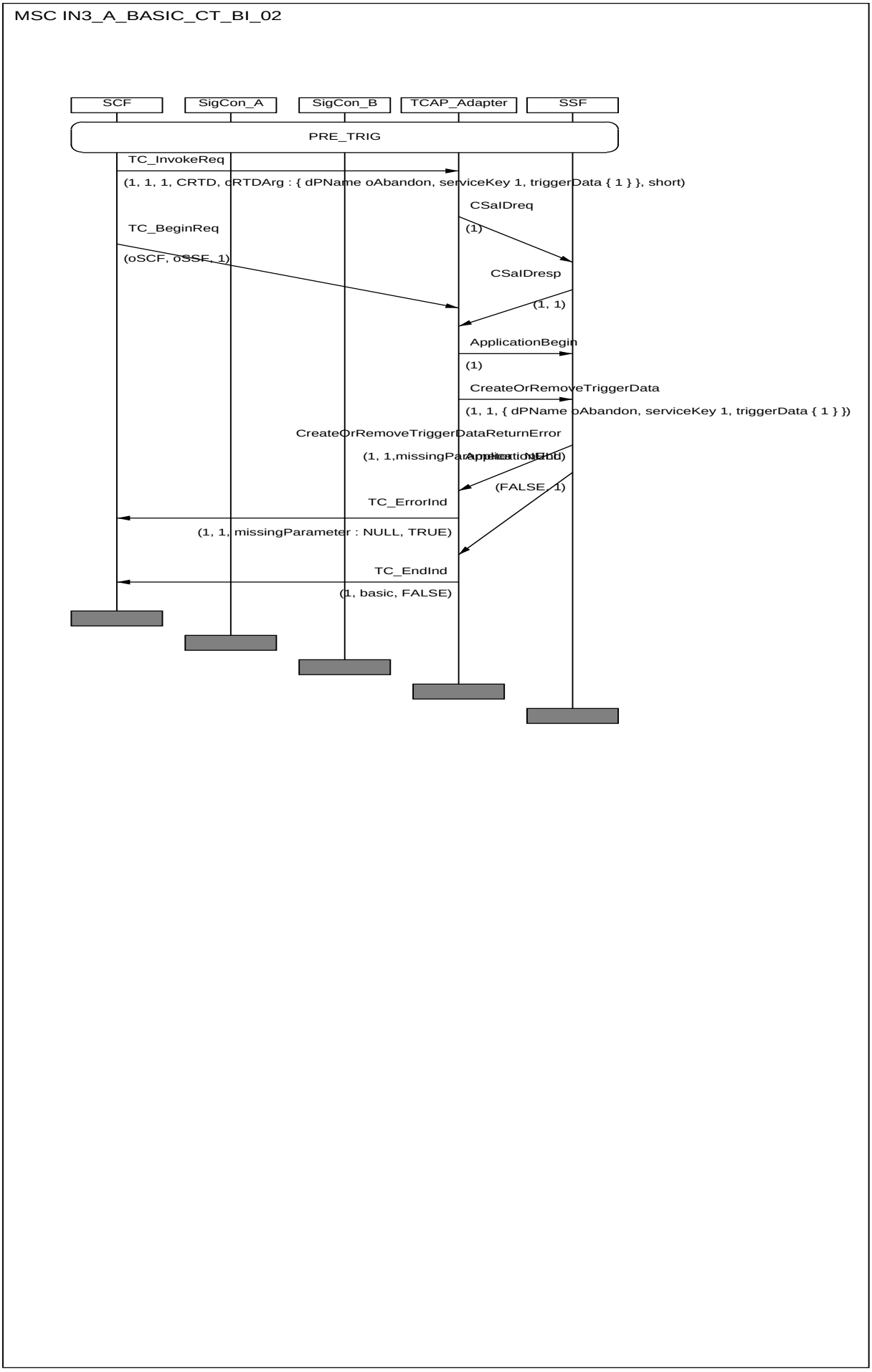
	IN3_A_BASIC_CT_BV_40
Work item no.:	ITEM_BASIC_221
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters createOrRemove = "remove", dPName = "**origAttempt**", triggerDPType = "tdp-r", serviceKey = SERVICE_KEY1, profile = PROFILE_ID_1, triggerData = TRIGGER_DATA_1, defaultFaultHandling = "resumeCallProcessing" and tDPIdentifier is an unknown tDPIdentifier, sends a CreateOrRemoveTriggerData returnResult component containing parameters triggerStatus = "unknownTrigger" and the unknown tDPIdentifier.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L3!CreateOrRemoveTriggerData invoke(remove, tdp-r, **unknown** tDPIdentifier) L3!TC-BEGIN L3?TC-END L3?CreateOrRemoveTriggerData returnResult(unknownTrigger, **unknown** tDPIdentifier)
Pass criteria	L3?TC-END L3?CreateOrRemoveTriggerData returnResult(unknownTrigger, unknown tDPIdentifier)
Postamble:	None


MSC IN3_A_BASIC_CT_BV_40
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { createOrRemove remove, triggerDPType tdp_r, tDPIdentifier oneTrigger : 10 }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { createOrRemove remove, triggerDPType tdp_r, tDPIdentifier oneTrigger : 10
CreateOrRemoveTriggerDataResult
(1, 1, { triggerStatus unknownTrigger, tDPIdentifier oneTrigger : 10 })
ApplicationEnd
(FALSE, 1)
TC_ReturnResultInd
TD_R, cRTDRArg : { triggerStatus unknownTrigger, tDPIdentifier oneTrigger : 10 }, TRUE)
TC_EndInd
(1, basic, FALSE)

	IN3_A_BASIC_CT_BI_01
Work item no.:	ITEM_BASIC_222
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters dPName = "**oAbandon**", profile = PROFILE_ID_1 and triggerData = TRIGGER_DATA_1, mandatory parameter serviceKey being omitted, sends a CreateOrRemoveTriggerData returnError component with error value "missingParameter".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAbandon, triggerDPType omitted, serviceKey omitted, PROFILE_ID_1, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnError(missingParameter)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnError(missingParameter)
Postamble:	None

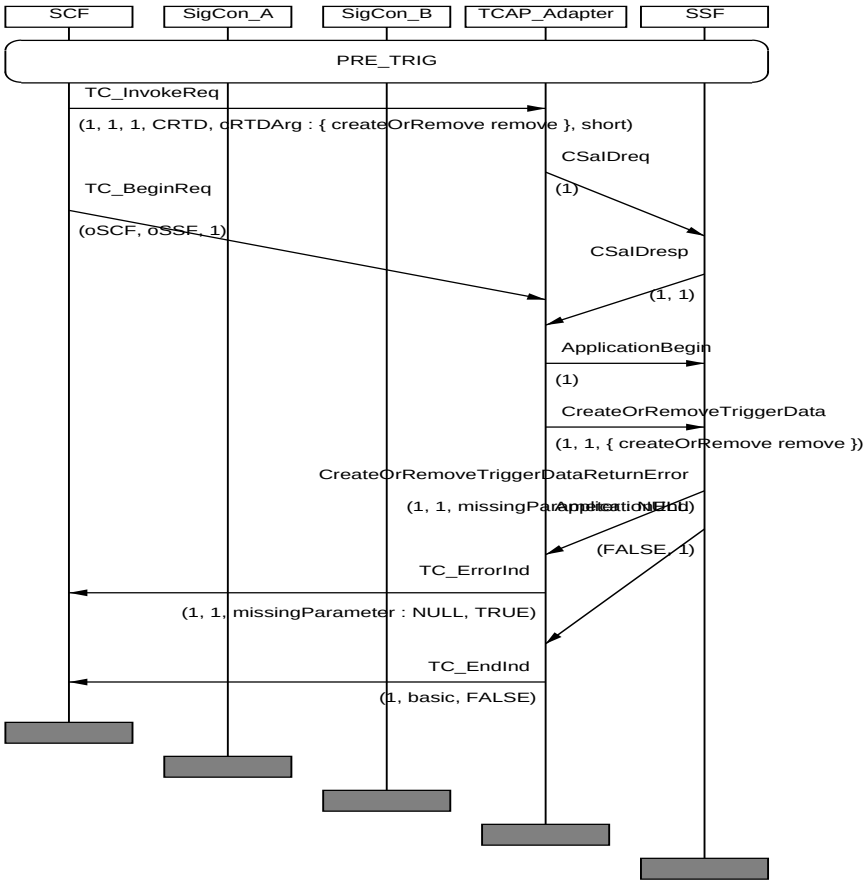

MSC IN3_A_BASIC_CT_BI_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oAbandon, profile access : '020000'H, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { dPName oAbandon, profile access : '020000'H, triggerData { 1 } }
CreateOrRemoveTriggerDataReturnError
(1, 1, missingParameter : NULL)
(FALSE, 1)
TC_ErrorInd
(1, 1, missingParameter : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)

	IN3_A_BASIC_CT_BI_02
Work item no.:	ITEM_BASIC_223
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters createOrRemove = "create", dPName = "**oAbandon**", serviceKey = SERVICE_KEY_1 and triggerData = TRIGGER_DATA_1, mandatory parameter profile being omitted, sends a CreateOrRemoveTriggerData returnError component with error value "missingParameter".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(createOrRemove omitted, oAbandon, triggerDPType omitted, SERVICE_KEY_1, profile omitted, TRIGGER_DATA_1, defaultFaultHandling omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnError(missingParameter)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnError(missingParameter)
Postamble:	None

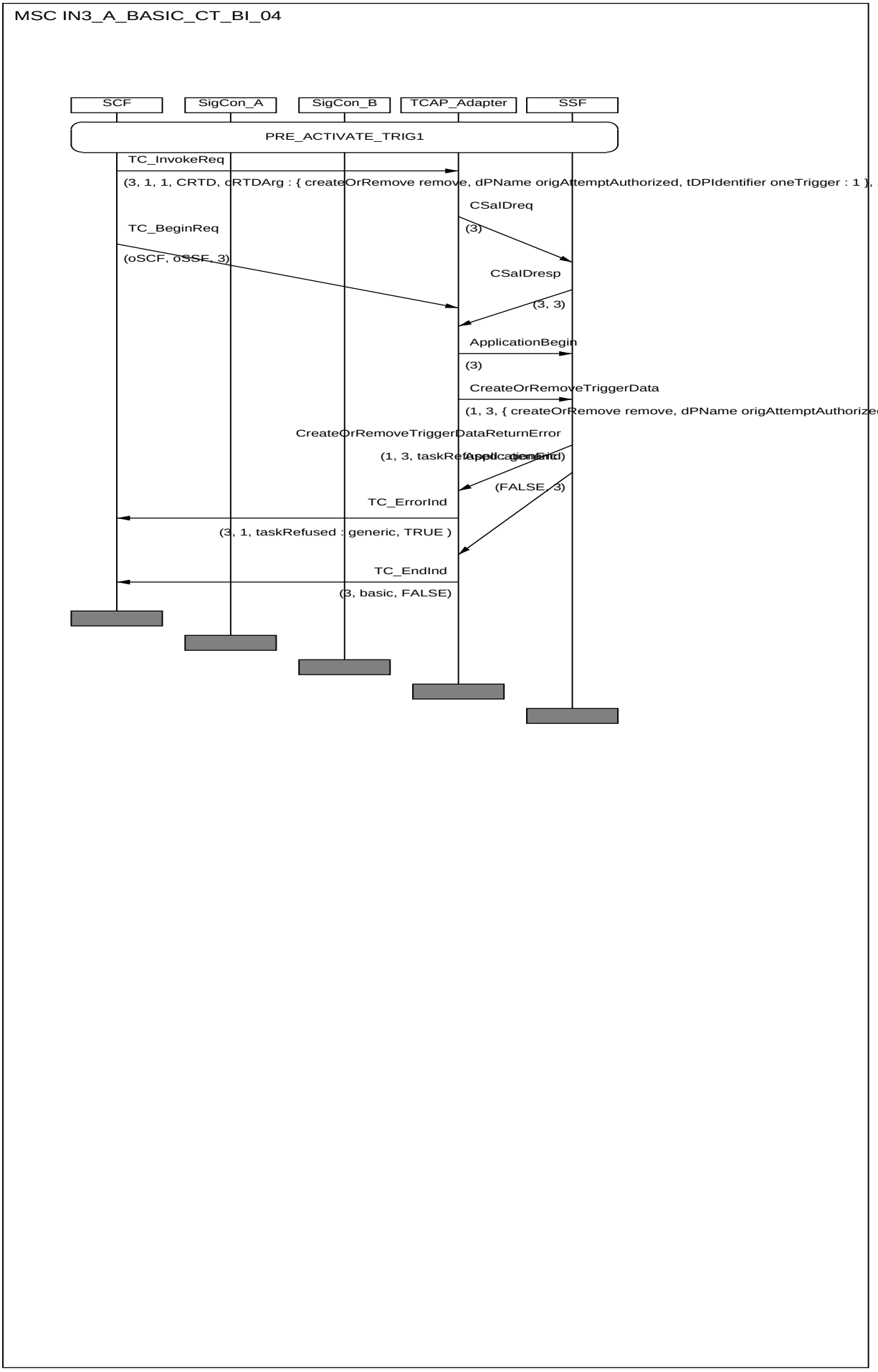

MSC IN3_A_BASIC_CT_BI_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_TRIG
TC_InvokeReq
(1, 1, 1, CRTD, cRTDArg : { dPName oAbandon, serviceKey 1, triggerData { 1 } }, short)
CSalDreq
(1)
TC_BeginReq
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
CreateOrRemoveTriggerData
(1, 1, { dPName oAbandon, serviceKey 1, triggerData { 1 } })
CreateOrRemoveTriggerDataReturnError
(FALSE, 1)
TC_ErrorInd
(1, 1, missingParameter : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)

	IN3_A_BASIC_CT_BI_03
Work item no.:	ITEM_BASIC_224
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters createOrRemove = "remove", mandatory parameter tDPIdentifier being omitted, sends a CreateOrRemoveTriggerData returnError component with error value "missingParameter".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_TRIG
Test description	L2!CreateOrRemoveTriggerData invoke(remove, dPName omitted, triggerDPType omitted, serviceKey omitted, profile omitted, triggerData omitted, defaultFaultHandling omitted, tDPIdentifier omitted) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnError(missingParameter)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnError(missingParameter)
Postamble:	None

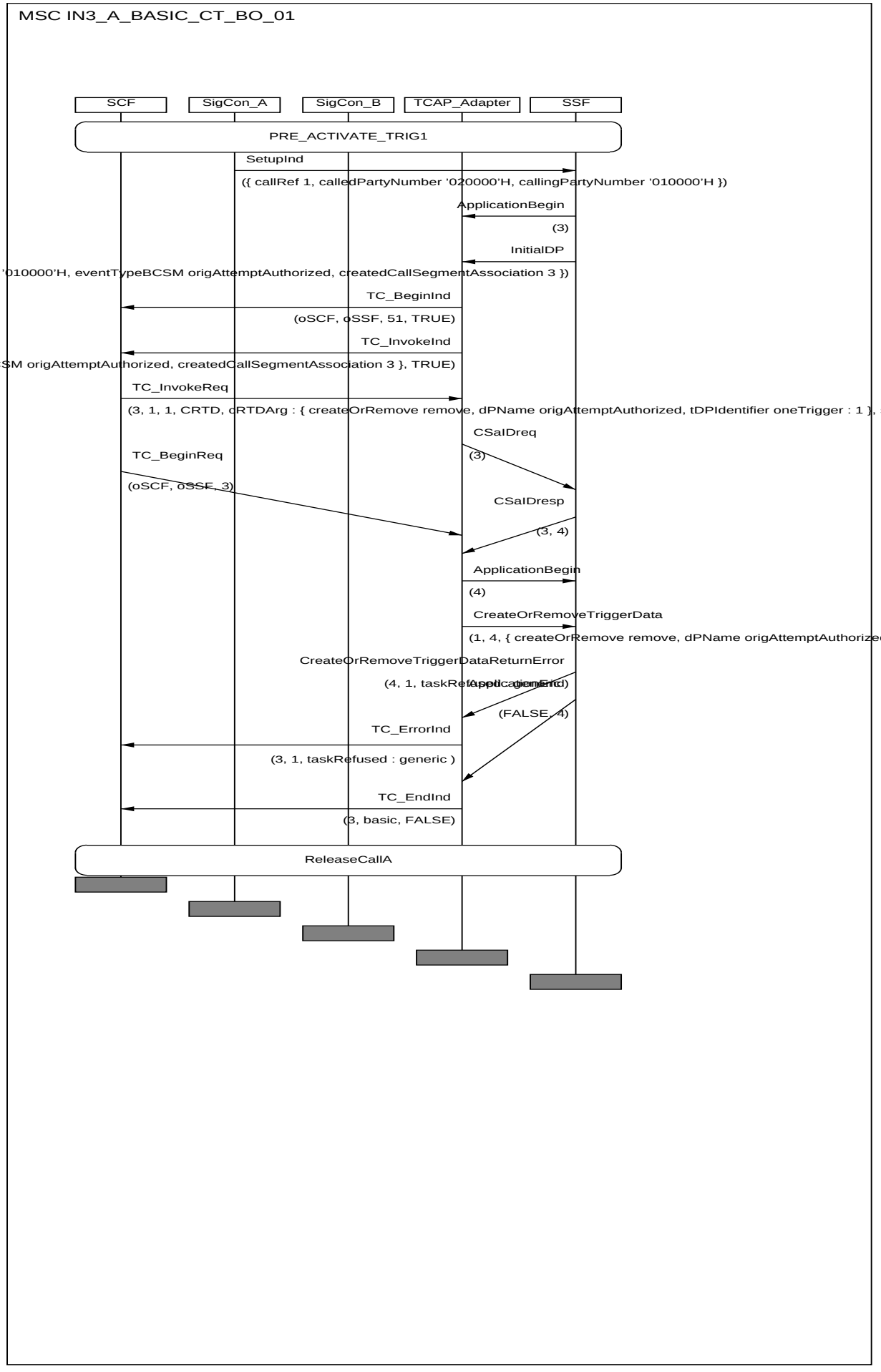
MSC IN3_A_BASIC_CT_BI_03



	IN3_A_BASIC_CT_BI_04
Work item no.:	ITEM_BASIC_225
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component containing parameters createOrRemove = "remove", and parameter tDPIdentifier identifies an activated trigger, sends a CreateOrRemoveTriggerData returnError component with error value "taskRefused".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG1
Test description	L2!CreateOrRemoveTriggerData invoke(remove, origAttempt, triggerDPType omitted, serviceKey omitted, profileId omitted, triggerData omitted, defaultFaultHandling omitted, tDPIdentifier1) L2!TC-BEGIN L2?TC-END L2?CreateOrRemoveTriggerData returnError(taskRefused)
Pass criteria	L2?TC-END L2?CreateOrRemoveTriggerData returnError(taskRefused)
Postamble:	None


MSC IN3_A_BASIC_CT_BI_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG1
TC_InvokeReq
(3, 1, 1, CRTD, cRTDArg : { createOrRemove remove, dPName origAttemptAuthorized, tDPIdentifier oneTrigger : 1 }, short
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
CreateOrRemoveTriggerData
(1, 3, { createOrRemove remove, dPName origAttemptAuthorized, tDPIdentifier oneT
CreateOrRemoveTriggerDataReturnError
(1, 3, taskRefused : generic)
(FALSE, 3)
TC_ErrorInd
(3, 1, taskRefused : generic, TRUE)
TC_EndInd
(3, basic, FALSE)

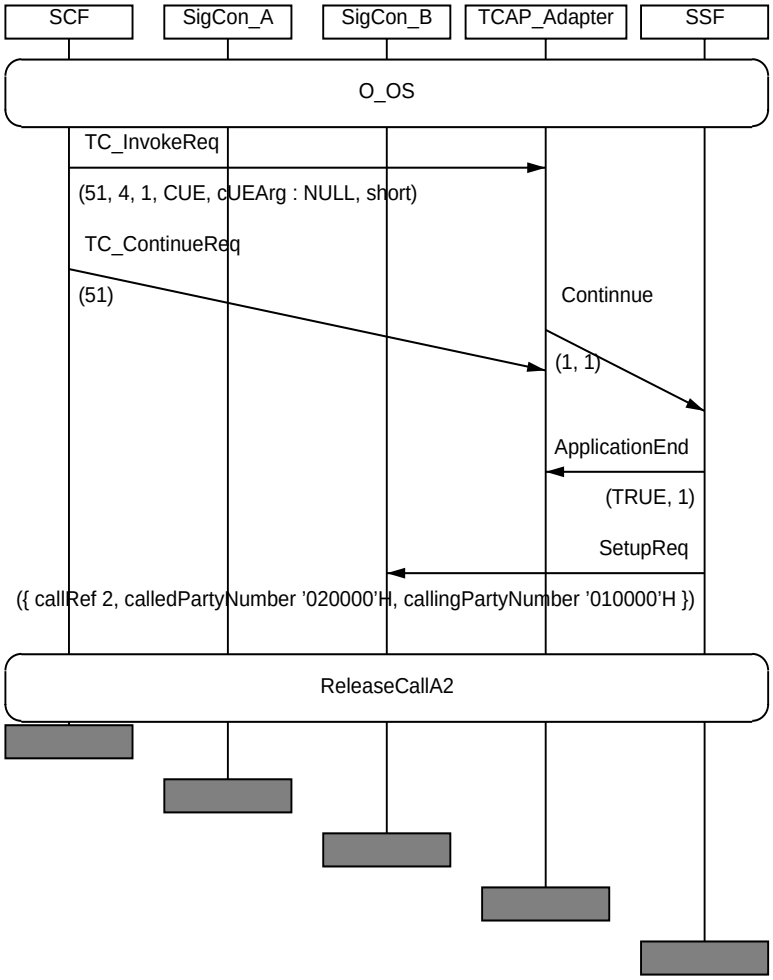
	IN3_A_BASIC_CT_BO_01
Work item no.:	ITEM_BASIC_226
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a CreateOrRemoveTriggerData invoke component in the context of a call, sends a CreateOrRemoveTriggerData returnError component with error value "taskRefused" or "unexpectedComponentSequence".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG1
Test description	CP1_1!SetupInd(CALL-DATA-1) L1?InitialDP L1!CreateOrRemoveTriggerData invoke(remove, origAttempt, triggerDPType omitted, serviceKey omitted, profileId omitted, triggerData omitted, defaultFaultHandling omitted, tDPIdentifier1) L1?CreateOrRemoveTriggerData returnError("taskRefused" or "unexpectedComponentSequence")
Pass criteria	L1?CreateOrRemoveTriggerData returnError("taskRefused" or "unexpectedComponentSequence")
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_CT_BO_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG1
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(3)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 3 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 3 }, TRUE)
TC_InvokeReq
(3, 1, 1, CRTD, cRTDArg : { createOrRemove remove, dPName origAttemptAuthorized, tDPIdentifier oneTrigger : 1 }, short
CSalDreq
TC_BeginReq
(3)
(oSCF, oSSF, 3)
CSalDresp
(3, 4)
ApplicationBegin
(4)
CreateOrRemoveTriggerData
(1, 4, { createOrRemove remove, dPName origAttemptAuthorized, tDPIdentifier oneT
CreateOrRemoveTriggerDataReturnError
(4, 1, taskRefused : generic)
(FALSE, 4)
TC_ErrorInd
(3, 1, taskRefused : generic)
TC_EndInd
(3, basic, FALSE)
ReleaseCallA

6.6.10 Continue (CU) procedure

	IN3_A_BASIC_CU_BV_01
Work item no.:	ITEM_BASIC_57
IN2 Ref(tmp)	IN2_A_BASIC_CU_CA_01
Purpose:	Verify that the SSF continues call processing, i.e. issues a SetupReq at SigConB, when it receives a **Continue** invoke component in the **Waiting for Instructions** state.
Requirements refs	6.5.1.2.3, 8.2, 8.2.1.2, 8.2.2, 11.14.1, 11.14.3.1
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:	ReleaseCallA2

MSC IN3_A_BASIC_CU_BV_01



6.6.11 ContinueWithArgument (CWA) procedure

CWA is tested directly and implicitly within the CPH procedures in EN 301 933-2 [5].

6.6.12 FurnishChargingInformation (FC) procedure

	IN3_A_BASIC_FC_BV_01
Work item no.:	ITEM_BASIC_332
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "currentTariffCurrency" but no "tariffSwitchCurrency", callAttemptChargeCurrency and callSetupChargeCurrency being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffCurrency) and call connection and release being performed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the call duration and the "currentTariffCurrency" registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency (currencyFactorScale, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = OMIT, iNRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value TR*currencyFactorScale
Postamble:	None

	IN3_A_BASIC_FC_BV_02
Work item no.:	ITEM_BASIC_333
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "currentTariffCurrency" and a "tariffSwitchCurrency", callAttemptChargeCurrency and callSetupChargeCurrency being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffCurrency) and call connection and release being performed, switchOverTime being trespassed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the call phases durations before/after switchOverTime and the "currentTariffCurrency" and the "nextTariffCurrency" registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release TS := Time from start of charging to tariffSwitchOverTime Timing conditions: TS<TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency (currencyFactorScale, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = (nextFactorScale,switchOverTime))), tariffFromSuccExchange = OMIT, iNRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value: TS*currencyFactorScale + (TR-TS)*nextFactorScale
Postamble:	None

	IN3_A_BASIC_FC_BV_03
Work item no.:	ITEM_BASIC_334
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "currentTariffCurrency" but no "tariffSwitchCurrency", callAttemptChargeCurrency and callSetupChargeCurrency being omitted. Verify also that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF another FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "tariffSwitchCurrency" but no "currentTariffCurrency", callAttemptChargeCurrency and callSetupChargeCurrency being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffCurrency) and call connection and release being performed, switchOverTime being trespassed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the call phases durations before/after switchOverTime and the "currentTariffCurrency" and the "nextTariffCurrency" registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release TS := Time from start of charging to tariffSwitchOverTime Timing conditions: TS<TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency (currencyFactorScale, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency = OMIT, tariffSwitchCurrency = (nextFactorScale,switchOverTime))), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI2, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	No FurnishChargingInformation returnError received (2 times) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value: TS*currencyFactorScale + (TR-TS)*nextFactorScale
Postamble:	None

	IN3_A_BASIC_FC_BV_04
Work item no.:	ITEM_BASIC_335
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "currentTariffCurrency" but no "tariffSwitchCurrency", callAttemptChargeCurrency being specified and callSetupChargeCurrency being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffCurrency) and call connection being performed unsuccessfully, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the callAttemptChargeCurrency registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency (currencyFactorScale, callAttemptChargeCurrency = callAttemptChargeFactorScale, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!ReleaseInd CP1_1?ReleaseReq L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) NOTE: CP1_1?ReleaseReq and L1?ApplyChargingReport may be received in any order.
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value callAttemptChargeFactorScale
Postamble:	None

	IN3_A_BASIC_FC_BV_05
Work item no.:	ITEM_BASIC_336
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "currentTariffCurrency" but no "tariffSwitchCurrency", callSetupChargeCurrency being specified and callAttemptChargeCurrency being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffCurrency) and call connection and release being performed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the callSetupChargeCurrency registered in the FurnishChargingInformation operation plus the amount resulting from the call duration and the "currentTariffCurrency" registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency (currencyFactorScale, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = callSetupChargeFactorScale), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value TR*currencyFactorScale + callSetupChargeFactorScale
Postamble:	None

	IN3_A_BASIC_FC_BV_06
Work item no.:	ITEM_BASIC_337
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "currentTariffCurrency" but no "tariffSwitchCurrency", callAttemptChargeCurrency and callSetupChargeCurrency being omitted. Verify also that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF another FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFAddOnCharge. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffCurrency) and call connection and release being performed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the addOnCharge registered in the second FurnishChargingInformation operation plus the amount resulting from the call duration and the "currentTariffCurrency" registered in the first FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency (currencyFactorScale, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFAddOnCharge(addOnChargeCurrencyFactorScale), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI2, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value TR*currencyFactorScale + addOnChargeCurrencyFactorScale
Postamble:	None

	IN3_A_BASIC_FC_BV_07
Work item no.:	ITEM_BASIC_338
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "currentTariffCurrency" but no "tariffSwitchCurrency", callAttemptChargeCurrency and callSetupChargeCurrency being omitted, and containing parameter tariffFromSuccExchange indicating value "notTakenIntoAccount". Check that after ApplyCharging being requested (supervisionMethod = maximumTariffCurrency) and call connection and release being performed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the call duration and the "currentTariffCurrency" registered in the FurnishChargingInformation operation. Check also that an ApplicationTransport message (APM) received from the succeeding exchange and containing a chargingTariff, is not taken into account, i.e. it has no effect on the "usedCost".
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency (currencyFactorScale, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = notTakenIntoAccount, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,currentCurrencyFactorScale)) CP1_2?APM(ChargingAcknowledgementInformation(not-accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value TR*currencyFactorScale
Postamble:	None

	IN3_A_BASIC_FC_BV_08
Work item no.:	ITEM_BASIC_339
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter tariffFromSuccExchange indicating value "takeIntoAccountNoAOC". Request ApplyCharging (supervisionMethod = maximumTariffCurrency) and initiate call connection to the succeeding exchange. Check that the SSF sends a SetupReq signal to SigConB and responds to an ApplicationTransport message (APM) received from the succeeding exchange and containing chargingTariffInformation with an ApplicationTransport message (APM) containing chargingAcknowledementInformation("accepted"). Verify also that, after connection establishment has been completed and release has been initiated, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the call duration and the "currentTariffCurrency" received in the ApplicationTransportMessage. Check also that no ApplicationTransport message (APM) containing chargingTariffInformation is sent by the SSF to SigConA.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,currentCurrencyFactorScale)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) no APM is sent by the SSF to SigConA
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value TR*currentCurrencyFactorScale no APM is sent by the SSF to SigConA
Postamble:	None

	IN3_A_BASIC_FC_BV_09
Work item no.:	ITEM_BASIC_340
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "currentTariffCurrency" but no "tariffSwitchCurrency", callAttemptChargeCurrency and callSetupChargeCurrency being omitted, and containing parameter tariffFromSuccExchange indicating value "takeIntoAccountNoAOC". Request ApplyCharging (supervisionMethod = maximumTariffCurrency) and initiate call connection to the succeeding exchange. Check that the SSF, after connection establishment has been completed, responds to an ApplicationTransport message (APM) received from the succeeding exchange and containing addOnchargingInformation, with an ApplicationTransport message (APM) containing chargingAcknowledementInformation("accepted"). Verify also that, after connection release has been initiated, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the call duration and the "currentTariffCurrency" received in the FurnishChargingInformation operation, plus the cost associated with the addOnChargingInformation received from the succeeding exchange. Check also that no ApplicationTransport message (APM) containing addOnChargingInformation is sent by the SSF to SigConA.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency (currencyFactorScale, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = takeIntoAccountNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(addOnchargingInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,addOnChargeCurrencyFactorScale)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) no APM is sent by the SSF to SigConA
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value TR*currencyFactorScale + addOnChargeCurrencyFactorScale no APM is sent by the SSF to SigConA
Postamble:	None

	IN3_A_BASIC_FC_BV_10
Work item no.:	ITEM_BASIC_341
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter tariffFromSuccExchange indicating value "takeIntoAccountTranslateIntoAOC". Request ApplyCharging (supervisionMethod = maximumTariffCurrency) and initiate call connection to the succeeding exchange. Check that the SSF sends a SetupReq signal to SigConB and responds to an ApplicationTransport message (APM) received from the succeeding exchange and containing chargingTariffInformation with an ApplicationTransport message (APM) containing chargingAcknowledementInformation("accepted"). Verify also that, after connection establishment has been completed and release has been initiated, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the call duration and the "currentTariffCurrency" received in the ApplicationTransportMessage. Check also that an ApplicationTransport message (APM) containing chargingTariffInformation is sent by the SSF to SigConA.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,currentCurrencyFactorScale)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_1?APM(ChargingTariffInformation(advice-of-charge,immediate-tariff-change=?,start-of-tariffing=?,currentCurrencyFactorScale)) CP1_1!APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value TR*currentCurrencyFactorScale CP1_1?APM(ChargingTariffInformation(advice-of-charge,immediate-tariff-change=?, start-of-tariffing=?,currentCurrencyFactorScale))
Postamble:	None

	IN3_A_BASIC_FC_BV_11
Work item no.:	ITEM_BASIC_342
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargeCurrency) indicating a "currentTariffCurrency" but no "tariffSwitchCurrency", callAttemptChargeCurrency and callSetupChargeCurrency being omitted, and containing parameter tariffFromSuccExchange indicating value "takeIntoAccountTranslateIntoAOC". Request ApplyCharging (supervisionMethod = maximumTariffCurrency) and initiate call connection to the succeeding exchange. Check that the SSF, after connection establishment has been completed, responds to an ApplicationTransport message (APM) received from the succeeding exchange and containing addOnChargingInformation, with an ApplicationTransport message (APM) containing chargingAcknowledementInformation("accepted"). Verify also that, after connection release has been initiated, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedCost" according to the call duration and the "currentTariffCurrency" received in the FurnishChargingInformation operation, plus the cost associated with the addOnChargingInformation received from the succeeding exchange. Check also that an ApplicationTransport message (APM) containing addOnChargingInformation is sent by the SSF to SigConA.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	CurrencyTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffCurrency (FC) and sub-parameters of maximumTariffCurrency(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffCurrency (currentTariffCurrency (currencyFactorScale, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffCurrency, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(addOnchargingInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,addOnChargeCurrencyFactorScale)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_1?APM(addOnchargingInformation(advice-of-charge,immediate-tariff-change=?,start-of-tariffing=?,addOnChargeCurrencyFactorScale)) CP1_!APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) CP1_1?APM(addOnchargingInformation(advice-of-charge,immediate-tariff-change=?,start-of-tariffing=?,addOnChargeCurrencyFactorScale))
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedCost) usedCost is within ±5 % of calculatory value TR*currencyFactorScale + addOnChargeCurrencyFactorScale CP1_1?APM(addOnchargingInformation(advice-of-charge,immediate-tariff-change=?,start-of-tariffing=?,addOnChargeCurrencyFactorScale))
Postamble:	None

	IN3_A_BASIC_FC_BV_12
Work item no.:	ITEM_BASIC_343
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "currentTariffPulse" but no "tariffSwitchPulse", callAttemptChargePulse and callSetupChargePulse being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffPulse) and call connection and release being performed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the call duration and the "currentTariffPulse" registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = OMIT), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value TR*currentPulseUnits
Postamble:	None

	IN3_A_BASIC_FC_BV_13
Work item no.:	ITEM_BASIC_344
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "currentTariffPulse" and a "tariffSwitchPulse", callAttemptChargePulse and callSetupChargePulse being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffPulse) and call connection and release being performed, switchOverTime being trespassed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the call phases durations before/after switchOverTime and the "currentTariffPulse" and the "nextTariffPulse" registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release TS := Time from start of charging to tariffSwitchOverTime Timing conditions: TS<TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = OMIT), tariffSwitchPulse = (nextPulseUnits,switchOverTime))), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value: TS*currentPulseUnits + (TR-TS)*nextPulseUnits
Postamble:	None

	IN3_A_BASIC_FC_BV_14
Work item no.:	ITEM_BASIC_345
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "currentTariffPulse" but no "tariffSwitchPulse", callAttemptChargePulse and callSetupChargePulse being omitted. Verify also that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF another FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "tariffSwitchPulse" but no "currentTariffPulse", callAttemptChargePulse and callSetupChargePulse being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffPulse) and call connection and release being performed, switchOverTime being trespassed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the call phases durations before/after switchOverTime and the "currentTariffPulse" and the "nextTariffPulse" registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release TS := Time from start of charging to tariffSwitchOverTime Timing conditions: TS<TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = OMIT), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse = OMIT, tariffSwitchPulse = (nextPulseUnits,switchOverTime))), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI2, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	No FurnishChargingInformation returnError received (2 times) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value: TS*currentPulseUnits + (TR-TS)*nextPulseUnits
Postamble:	None

	IN3_A_BASIC_FC_BV_15
Work item no.:	ITEM_BASIC_346
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "currentTariffPulse" but no "tariffSwitchPulse", callAttemptChargePulse being specified and callSetupChargePulse being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffPulse) and call connection being performed unsuccessfully, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the callAttemptChargePulse registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = callAttemptChargePulse, callSetupChargePulse = OMIT), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!ReleaseInd CP1_1?ReleaseReq L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) NOTE: CP1_1?ReleaseReq and L1?ApplyChargingReport may be received in any order.
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value callAttemptChargePulse
Postamble:	None

	IN3_A_BASIC_FC_BV_16
Work item no.:	ITEM_BASIC_347
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "currentTariffPulse" but no "tariffSwitchPulse", callSetupChargePulse being specified and callAttemptChargePulse being omitted. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffPulse) and call connection and release being performed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the callSetupChargePulse registered in the FurnishChargingInformation operation plus the amount resulting from the call duration and the "currentTariffPulse" registered in the FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = callSetupChargePulse), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value TR*currentPulseUnits + callSetupChargePulse
Postamble:	None

	IN3_A_BASIC_FC_BV_17
Work item no.:	ITEM_BASIC_348
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "currentTariffPulse" but no "tariffSwitchPulse", callAttemptChargePulse and callSetupChargePulse being omitted. Verify also that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF another FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFAddOnCharge. Check that after ApplyCharging being requested (supervisionMethod = maximumTariffPulse) and call connection and release being performed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the addOnCharge registered in the second FurnishChargingInformation operation plus the amount resulting from the call duration and the "currentTariffPulse" registered in the first FurnishChargingInformation operation.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = OMIT), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFAddOnCharge(addOnChargePulse), tariffFromSuccExchange = OMIT, iINRecordIndicators = FCI_IN_CRI2, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value TR*currentPulseUnits + addOnChargePulse
Postamble:	None

	IN3_A_BASIC_FC_BV_18
Work item no.:	ITEM_BASIC_349
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "currentTariffPulse" but no "tariffSwitchPulse", callAttemptChargePulse and callSetupChargePulse being omitted, and containing parameter tariffFromSuccExchange indicating value "notTakenIntoAccount". Check that after ApplyCharging being requested (supervisionMethod = maximumTariffPulse) and call connection and release being performed, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the call duration and the "currentTariffPulse" registered in the FurnishChargingInformation operation. Check also that an ApplicationTransport message (APM) received from the succeeding exchange and containing a chargingTariff, is not taken into account, i.e. it has no effect on the "usedPulses".
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = OMIT), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = notTakenIntoAccount, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,pulseUnits)) CP1_2?APM(ChargingAcknowledgementInformation(not-accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value TR*currentPulseUnits
Postamble:	None

	IN3_A_BASIC_FC_BV_19
Work item no.:	ITEM_BASIC_350
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter tariffFromSuccExchange indicating value "takeIntoAccountNoAOC". Request ApplyCharging (supervisionMethod = maximumTariffPulse) and initiate call connection to the succeeding exchange. Check that the SSF sends a SetupReq signal to SigConB and responds to an ApplicationTransport message (APM) received from the succeeding exchange and containing chargingTariffInformation with an ApplicationTransport message (APM) containing chargingAcknowledementInformation("accepted"). Verify also that, after connection establishment has been completed and release has been initiated, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the call duration and the "currentTariffPulse" received in the ApplicationTransportMessage. Check also that no ApplicationTransport message (APM) containing chargingTariffInformation is sent by the SSF to SigConA.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,currentPulseUnits)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) no APM is sent by the SSF to SigConA
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value TR*currentPulseUnits no APM is sent by the SSF to SigConA
Postamble:	None

	IN3_A_BASIC_FC_BV_20
Work item no.:	ITEM_BASIC_351
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "currentTariffPulse" but no "tariffSwitchPulse", callAttemptChargePulse and callSetupChargePulse being omitted, and containing parameter tariffFromSuccExchange indicating value "takeIntoAccountNoAOC". Request ApplyCharging (supervisionMethod = maximumTariffPulse) and initiate call connection to the succeeding exchange. Check that the SSF, after connection establishment has been completed, responds to an ApplicationTransport message (APM) received from the succeeding exchange and containing addOnchargingInformation, with an ApplicationTransport message (APM) containing chargingAcknowledementInformation("accepted"). Verify also that, after connection release has been initiated, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the call duration and the "currentTariffPulse" received in the FurnishChargingInformation operation, plus the cost associated with the addOnChargingInformation received from the succeeding exchange. Check also that no ApplicationTransport message (APM) containing addOnChargingInformation is sent by the SSF to SigConA.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = OMIT), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = takeIntoAccountNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(addOnchargingInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,addOnChargePulseUnits)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) no APM is sent by the SSF to SigConA
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value TR*currentPulseUnits + addOnChargePulseUnits no APM is sent by the SSF to SigConA
Postamble:	None

	IN3_A_BASIC_FC_BV_21
Work item no.:	ITEM_BASIC_352
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter tariffFromSuccExchange indicating value "takeIntoAccountTranslateIntoAOC". Request ApplyCharging (supervisionMethod = maximumTariffPulse) and initiate call connection to the succeeding exchange. Check that the SSF sends a SetupReq signal to SigConB and responds to an ApplicationTransport message (APM) received from the succeeding exchange and containing chargingTariffInformation with an ApplicationTransport message (APM) containing chargingAcknowledementInformation("accepted"). Verify also that, after connection establishment has been completed and release has been initiated, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the call duration and the "currentTariffPulse" received in the ApplicationTransportMessage. Check also that an ApplicationTransport message (APM) containing chargingTariffInformation is sent by the SSF to SigConA.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,currentPulseUnits)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_1?APM(ChargingTariffInformation(advice-of-charge,immediate-tariff-change=?,start-of-tariffing=?,currentPulseUnits)) CP1_1!APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses)
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value TR*currentPulseUnits CP1_1?APM(ChargingTariffInformation(advice-of-charge,immediate-tariff-change=?,start-of-tariffing=?,currentPulseUnits))
Postamble:	None

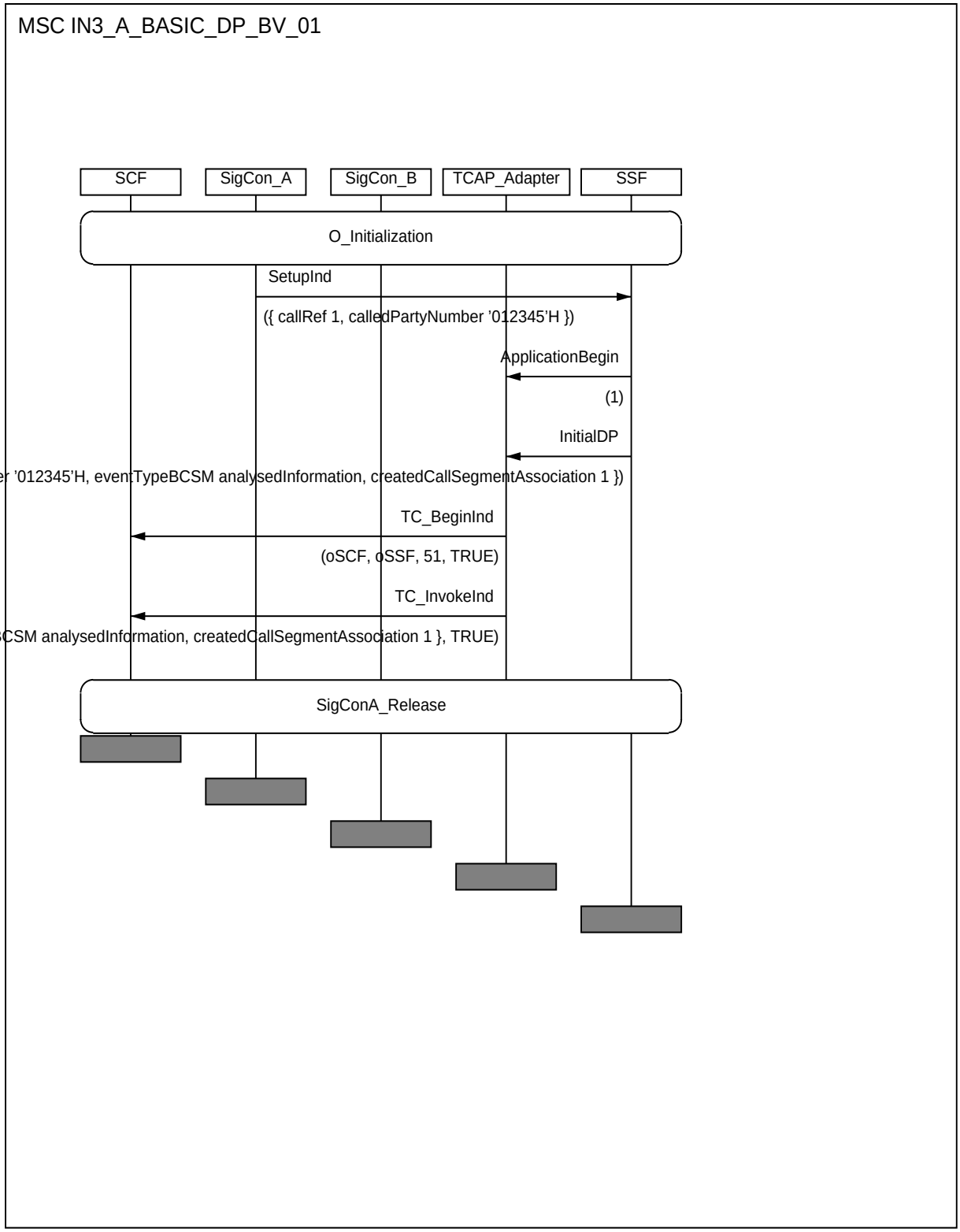
	IN3_A_BASIC_FC_BV_22
Work item no.:	ITEM_BASIC_353
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF does not send a FurnishChargingInformation returnError component, when having received from the SCF a FurnishChargingInformation invoke component containing parameter chargeMessageFromSCF:sCFChargingTariff (single CommunicationChargePulse) indicating a "currentTariffPulse" but no "tariffSwitchPulse", callAttemptChargePulse and callSetupChargePulse being omitted, and containing parameter tariffFromSuccExchange indicating value "takeIntoAccountTranslateIntoAOC". Request ApplyCharging (supervisionMethod = maximumTariffPulse) and initiate call connection to the succeeding exchange. Check that the SSF, after connection establishment has been completed, responds to an ApplicationTransport message (APM) received from the succeeding exchange and containing addOnChargingInformation, with an ApplicationTransport message (APM) containing chargingAcknowledementInformation("accepted"). Verify also that, after connection release has been initiated, the SSF sends an ApplyChargingReport invoke component, containing supervisionResult "usedPulses" according to the call duration and the "currentTariffPulse" received in the FurnishChargingInformation operation, plus the cost associated with the addOnChargingInformation received from the succeeding exchange. Check also that an ApplicationTransport message (APM) containing addOnChargingInformation is sent by the SSF to SigConA.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16 7
Selection Cond.	PulseTariff
Preamble:	O_OSA_FCI
Test description	TM := maximum Time calculated from currentTariffPulse (FC) and sub-parameters of maximumTariffPulse(AC) TR := Time from start of charging to user-initiated release Timing conditions: TR<TM L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = OMIT), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) Wait a while to assure that no returnError is received L1!ApplyCharging invoke(supervisionMethod = maximumTariffPulse, warningBeforeLimitReached = OMIT, releaseWhenLimitReached = OMIT, reportCondition = OMIT) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(addOnchargingInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,addOnChargePulseUnits)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_1?APM(addOnchargingInformation(advice-of-charge,immediate-tariff-change=?,start-of-tariffing=?,addOnChargePulseUnits)) CP1_!APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) Start TR CP1_1?SetupResp ?Timeout TR CP1_2!ReleaseInd(Normal clearing) CP1_1!ReleaseInd(Normal clearing) L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) CP1_1?APM(addOnchargingInformation(advice-of-charge,immediate-tariff-change=?,start-of-tariffing=?,addOnChargePulseUnits))
Pass criteria	No FurnishChargingInformation returnError received L1?ApplyChargingReport invoke(reportConditionInformation = finalReportCallReleased, supervisionResult = usedPulses) usedPulses is within ±5 % of calculatory value TR*currentPulseUnits + addOnChargePulseUnits CP1_1?APM(addOnchargingInformation(advice-of-charge,immediate-tariff-change=?,start-of-tariffing=?,addOnChargePulseUnits))
Postamble:	None

	IN3_A_BASIC_FC_BI_01
Work item no.:	ITEM_BASIC_354
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF sends an FurnishChargingInformation returnError component with errorCode "missingParameter", after having received an FurnishChargingInformation invoke component without mandatory parameter iINRecordIndicators.
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	
Preamble:	O_OSA_FCI
Test description	[BASIC_TARIFF_METHOD = "Currency"] L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffCurrency (currentChargeCurrency, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = OMIT, partyToCharge = FCI_PARTY_TO_CHARGE) [BASIC_TARIFF_METHOD = "Pulse"] L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = OMIT), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = OMIT, partyToCharge = FCI_PARTY_TO_CHARGE)
Pass criteria	L1?FurnishChargingInformation returnError(missingParameter)
Postamble:	ReleaseCallA

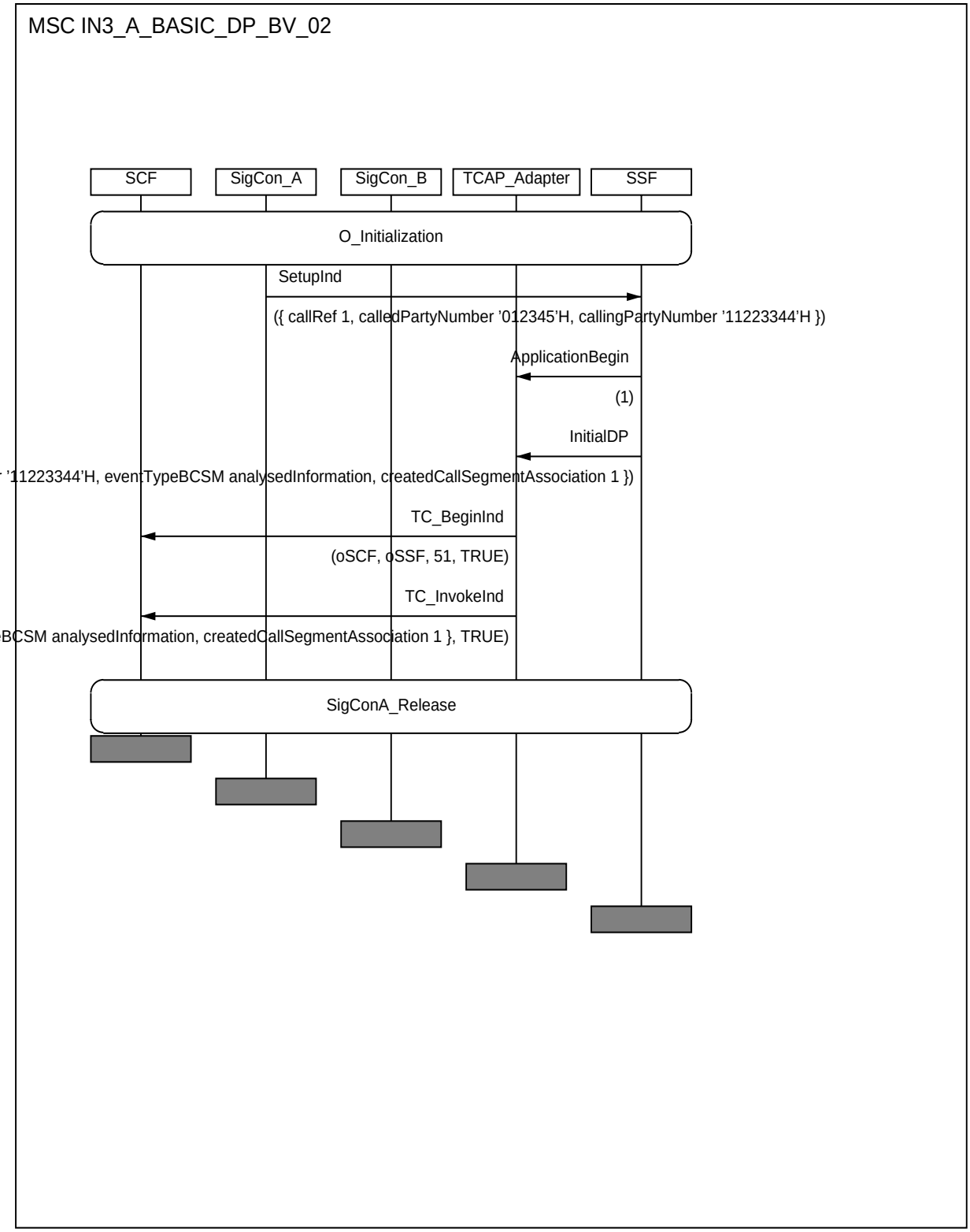
	IN3_A_BASIC_FC_BO_01
Work item no.:	ITEM_BASIC_355
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF sends an FurnishChargingInformation returnError component with errorCode "taskRefused" or "unexpectedComponentSequence", after having received an FurnishChargingInformation invoke component in state "Idle".
Requirements refs	6.6.3.3.2.1, 8.2.2 (tables 19, 20 and 21), 8.2.2.1, 8.2.2.2, 8.2.2.5, 11.3, 11.4, 11.25, 12.68, 16
Selection Cond.	
Preamble:	None
Test description	[BASIC_TARIFF_METHOD = "Currency"] L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffCurrency (currentChargeCurrency, callAttemptChargeCurrency = OMIT, callSetupChargeCurrency = OMIT), tariffSwitchCurrency = OMIT)), tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!TCAP-BEGIN [BASIC_TARIFF_METHOD = "Pulse"] L1!FurnishChargingInformation invoke(chargeMessageFromSCF:sCFChargingTariff (tariffPulse (currentTariffPulse (currentPulseUnits, callAttemptChargePulse = OMIT, callSetupChargePulse = OMIT), tariffSwitchPulse = OMIT)), tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!TCAP-BEGIN
Pass criteria	L1?FurnishChargingInformation returnError("taskRefused" or "unexpectedComponentSequence")
Postamble:	None

6.6.13 InitialDP (DP) procedure

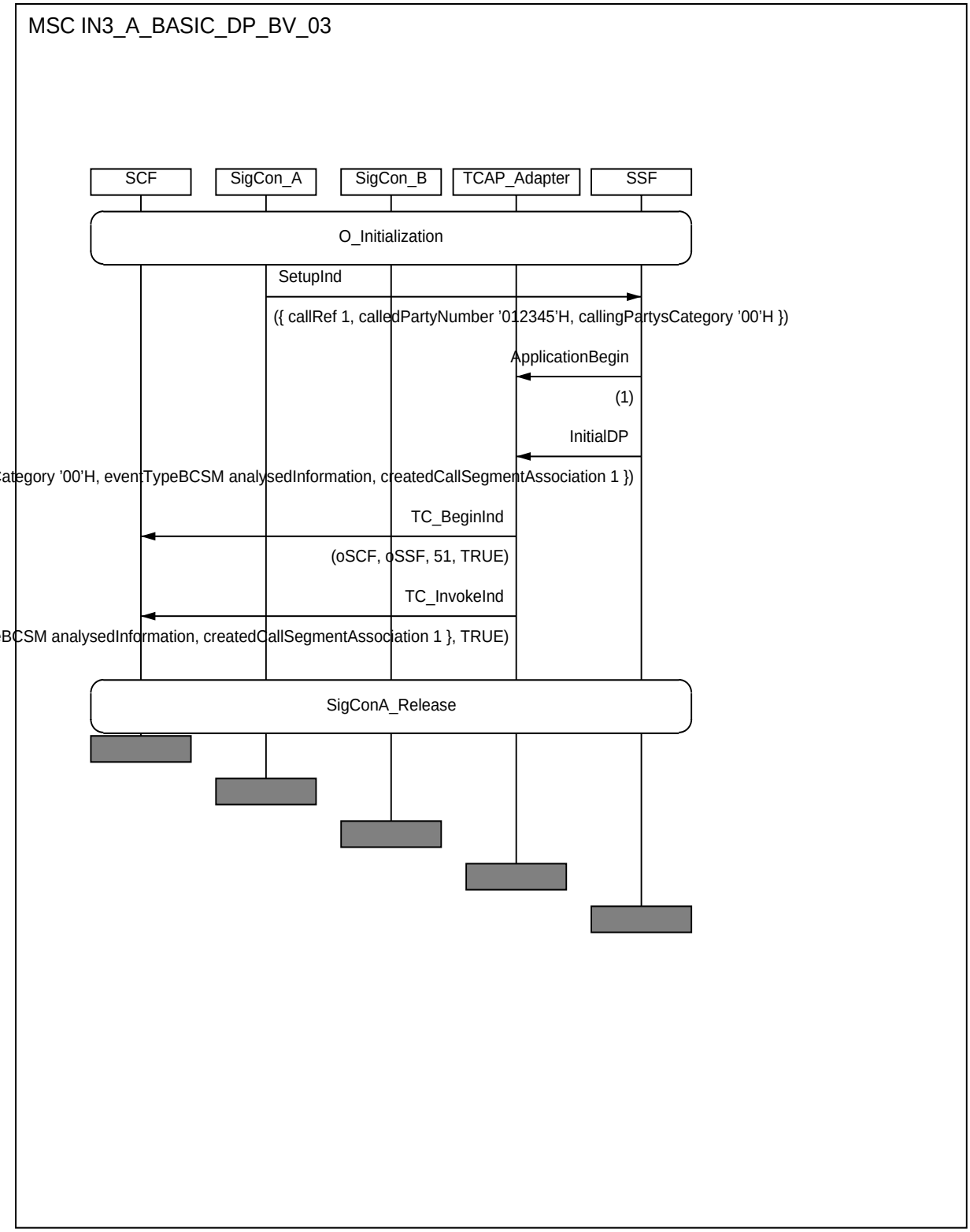
	IN3_A_BASIC_DP_BV_01
Work item no.:	ITEM_BASIC_61
IN2 Ref(tmp)	IN2_A_BASIC_DP_CA_01
Purpose:	Verify that the SSF receiving a **SetupInd** containing at least the parameter **calledPartyNumber**, sends to the SCF an **InitialDP** invoke component containing the parameter **calledPartyNumber**.
Requirements refs	8.2.1.1, 8.2.2, 8.2.2.1, 11.26.1, 11.26.1.1.1, 11.26.2.1
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


MSC IN3_A_BASIC_DP_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
SetupInd
({ callRef 1, calledPartyNumber '012345'H })
ApplicationBegin
(1)
InitialDP
1, calledPartyNumber '012345'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
2345'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }, TRUE)
SigConA_Release

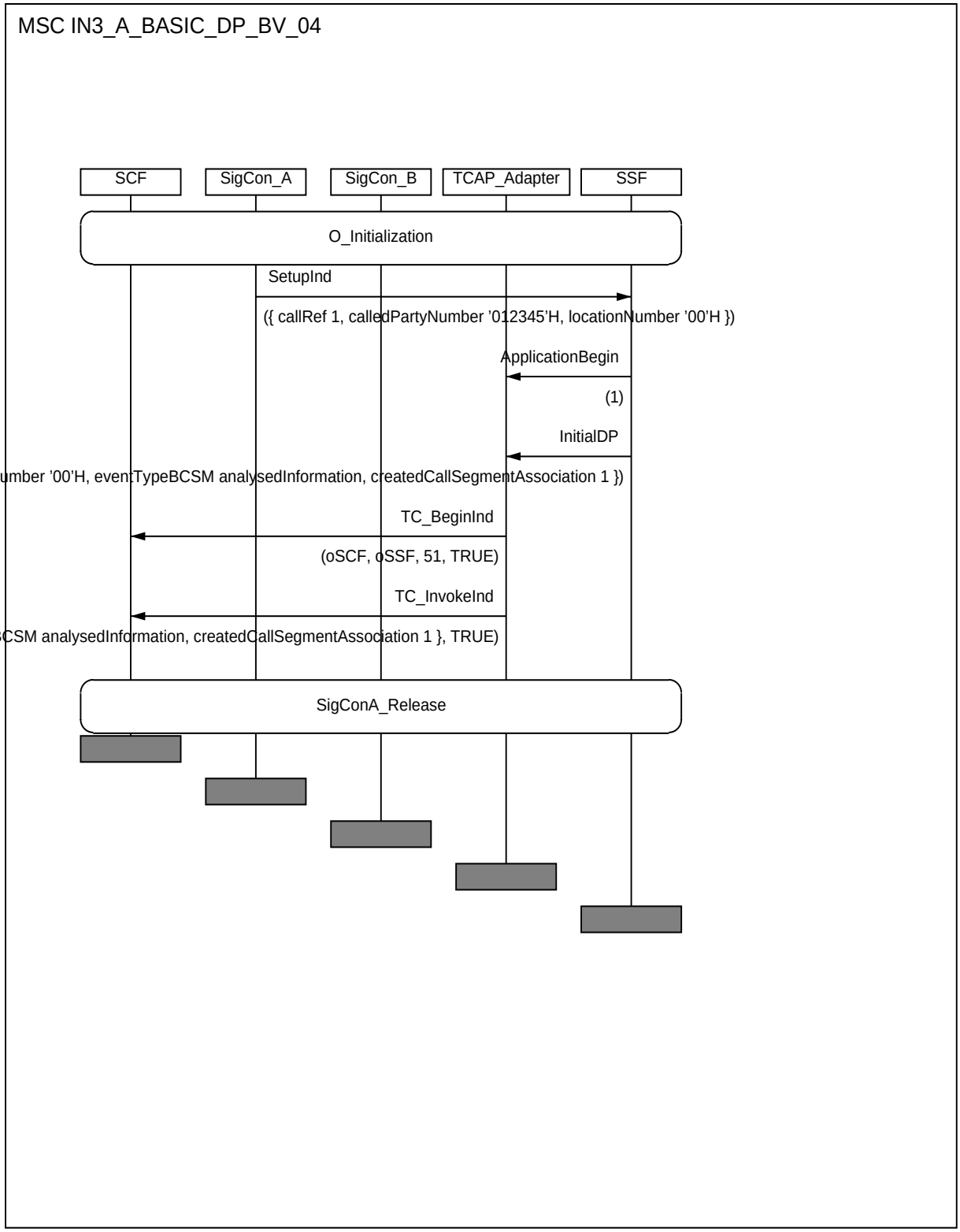
	IN3_A_BASIC_DP_BV_02
Work item no.:	ITEM_BASIC_62
IN2 Ref(tmp)	IN2_A_BASIC_DP_CA_02
Purpose:	Verify that the SSF receiving a **SetupInd** containing at least the parameter **callingPartyNumber**, sends to the SCF an **InitialDP** invoke component containing the parameter **callingPartyNumber**.
Requirements refs	8.2.1.1, 8.2.2, 8.2.2.1, 11.26.1, 11.26.1.1.1, 11.26.2.1
Selection Cond.	
Preamble:	None
Test description	SigConA sends to SSF a **SetupInd** containing at least the parameter: - callingPartyNumber
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to the calling party: - callingPartyNumber
Postamble:	SigConA_Release


MSC IN3_A_BASIC_DP_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
SetupInd
({ callRef 1, calledPartyNumber ’012345’H, callingPartyNumber ’11223344’H })
ApplicationBegin
(1)
InitialDP
allingPartyNumber ’11223344’H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
3344’H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }, TRUE)
SigConA_Release

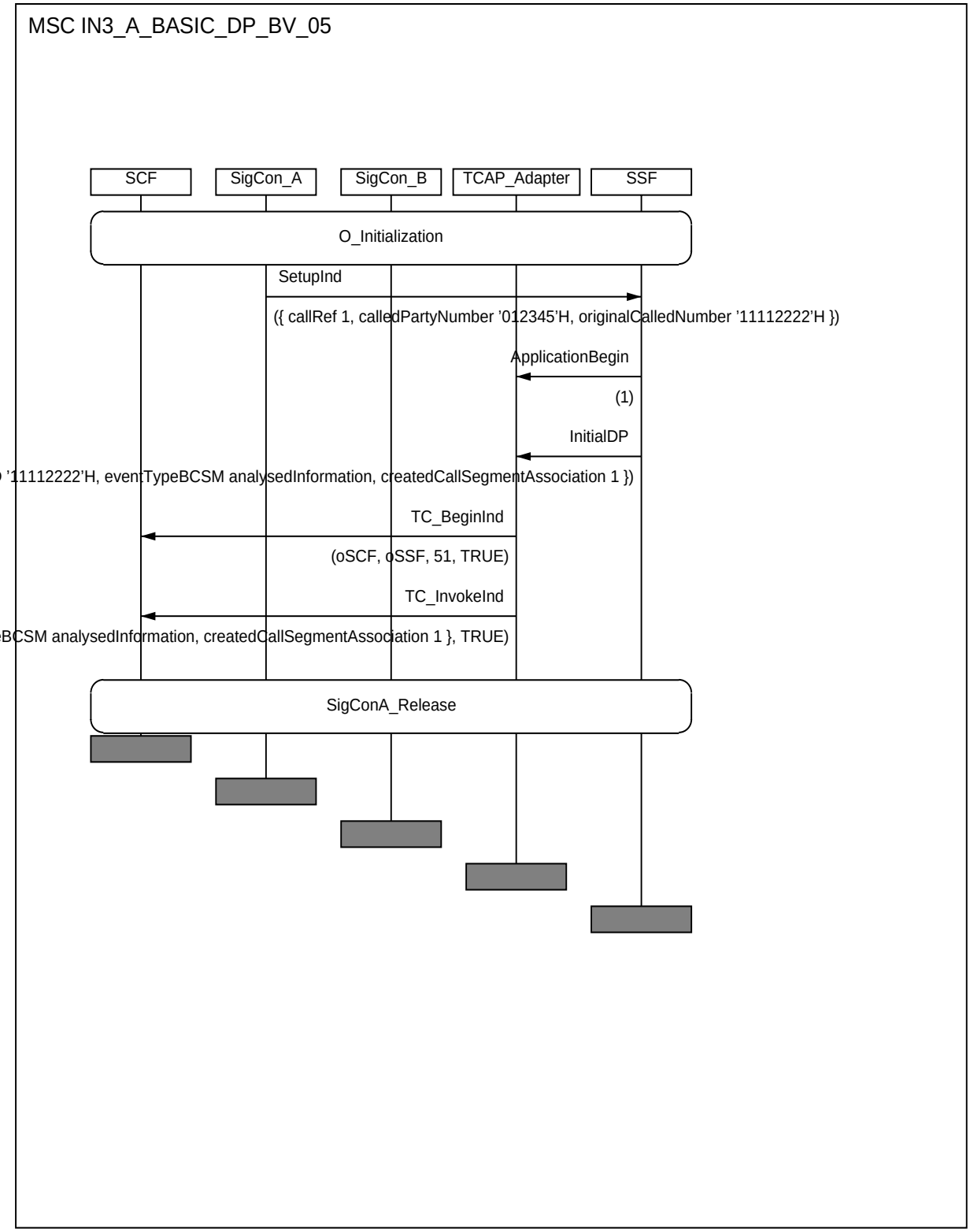
	IN3_A_BASIC_DP_BV_03
Work item no.:	ITEM_BASIC_63
IN2 Ref(tmp)	IN2_A_BASIC_DP_BV_01
Purpose:	Verify that the SSF receiving a **SetupInd** containing at least the parameter **callingPartysCategory**, sends to the SCF an **InitialDP** invoke component containing the parameter **callingPartysCategory**.
Requirements refs	8.2.1.1, 8.2.2, 8.2.2.1, 11.26.1, 11.26.1.1.1, 11.26.2.1
Selection Cond.	
Preamble:	None
Test description	SigConA sends to SSF a **SetupInd** containing at least the parameter: - callingPartysCategory
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to the calling party category: - callingPartysCategory
Postamble:	SigConA_Release


MSC IN3_A_BASIC_DP_BV_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
SetupInd
({ callRef 1, calledPartyNumber ’012345’H, callingPartysCategory ’00’H })
ApplicationBegin
(1)
InitialDP
5’H, callingPartysCategory ’00’H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
y ’00’H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }, TRUE)
SigConA_Release

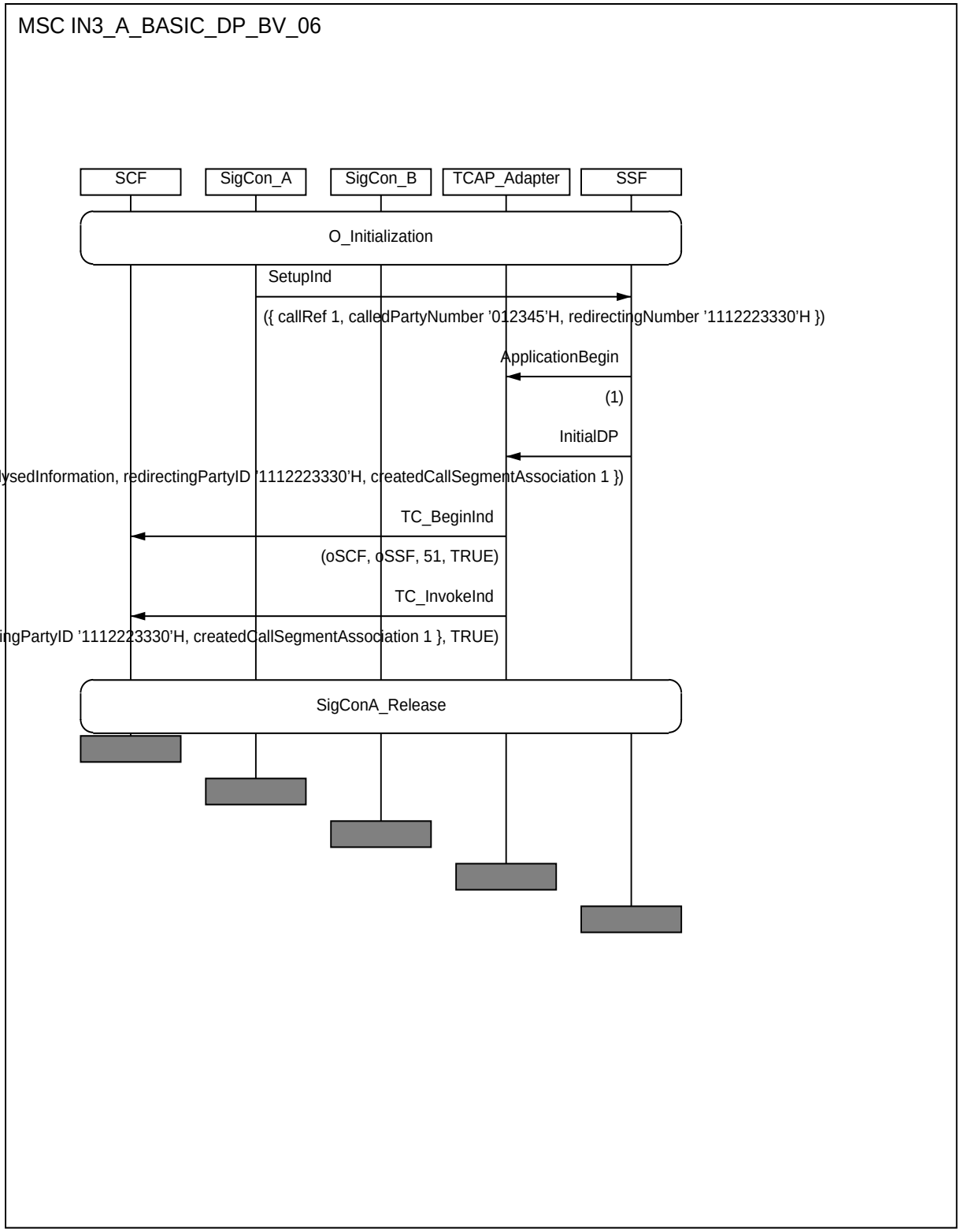
	IN3_A_BASIC_DP_BV_04
Work item no.:	ITEM_BASIC_64
IN2 Ref(tmp)	IN2_A_BASIC_DP_BV_02
Purpose:	Verify that the SSF receiving a **SetupInd** containing at least the parameter **locationNumber**, sends to the SCF an **InitialDP** invoke component containing the parameter **locationNumber**.
Requirements refs	8.2.1.1, 8.2.2, 8.2.2.1, 11.26.1, 11.26.1.1.1, 11.26.2.1
Selection Cond.	
Preamble:	None
Test description	SigConA sends to SSF a **SetupInd** containing at least the parameter: - locationNumber
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to the location information: - locationNumber
Postamble:	SigConA_Release


MSC IN3_A_BASIC_DP_BV_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
SetupInd
({ callRef 1, calledPartyNumber ’012345’H, locationNumber ’00’H })
ApplicationBegin
(1)
InitialDP
012345’H, locationNumber ’00’H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
r ’00’H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }, TRUE)
SigConA_Release

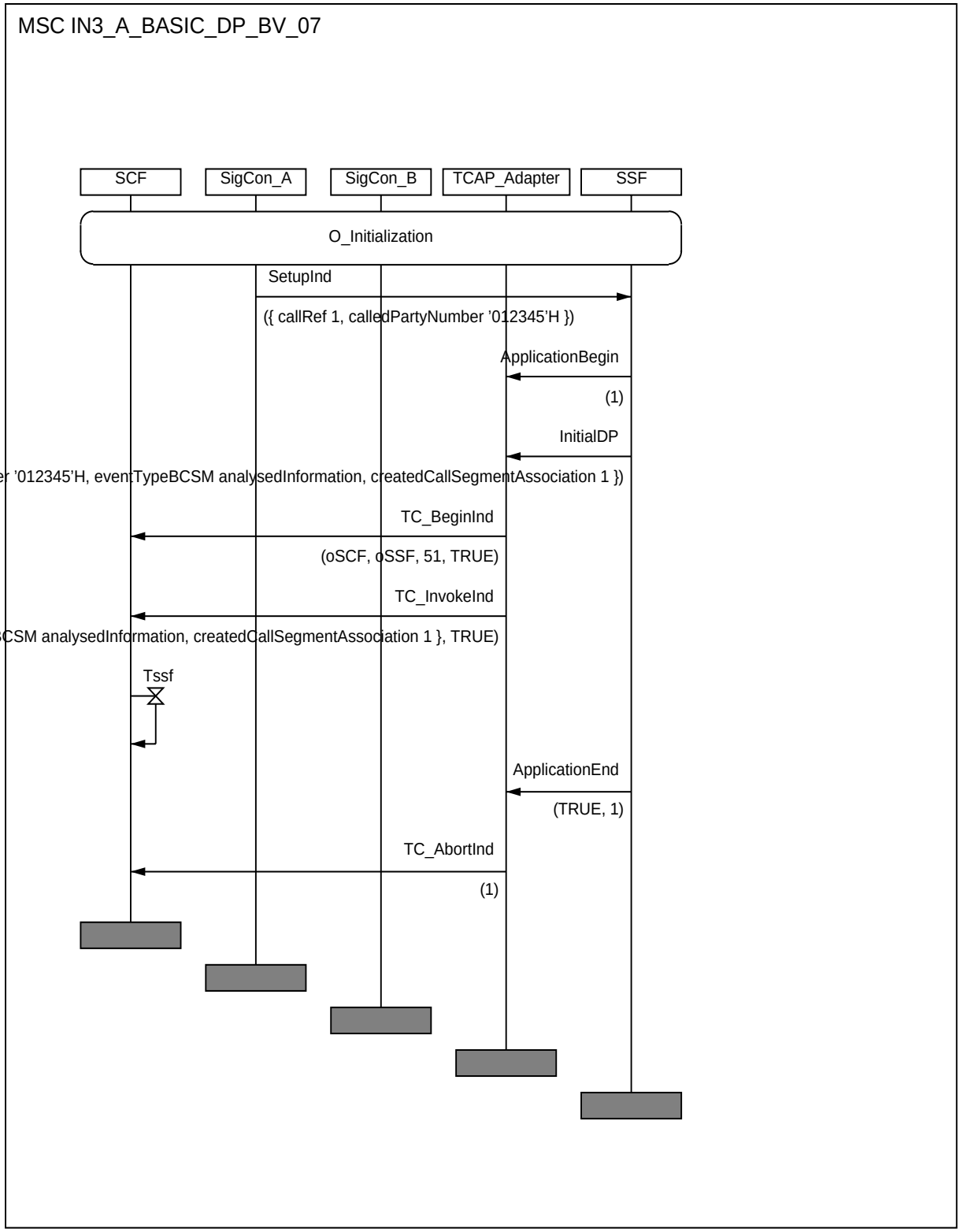
	IN3_A_BASIC_DP_BV_05
Work item no.:	ITEM_BASIC_65
IN2 Ref(tmp)	IN2_A_BASIC_DP_BV_03
Purpose:	Verify that the SSF receiving a **SetupInd** containing at least the parameter **originalCalledPartyID**, sends to the SCF an **InitialDP** invoke component containing the parameter **originalCalledPartyID**.
Requirements refs	8.2.1.1, 8.2.2, 8.2.2.1, 11.26.1, 11.26.1.1.1, 11.26.2.1
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


MSC IN3_A_BASIC_DP_BV_05
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
SetupInd
({ callRef 1, calledPartyNumber '012345'H, originalCalledNumber '11112222'H })
ApplicationBegin
(1)
InitialDP
iginalCalledPartyID '11112222'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
2222'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }, TRUE)
SigConA_Release

	IN3_A_BASIC_DP_BV_06
Work item no.:	ITEM_BASIC_66
IN2 Ref(tmp)	IN2_A_BASIC_DP_BV_04
Purpose:	Verify that the SSF receiving a **SetupInd** containing at least the parameter **redirectingPartyID**, sends to the SCF an **InitialDP** invoke component containing the parameter **redirectingPartyID**.
Requirements refs	8.2.1.1, 8.2.2, 8.2.2.1, 11.26.1, 11.26.1.1.1, 11.26.2.1
Selection Cond.	
Preamble:	None
Test description	SigConA sends to SSF a **SetupInd** containing at least the parameter: - redirectingPartyID
Pass criteria	Check that SSF sends to SCF an **InitialDP** invoke containing the parameter related to redirecting party number: - redirectingPartyID
Postamble:	SigConA_Release

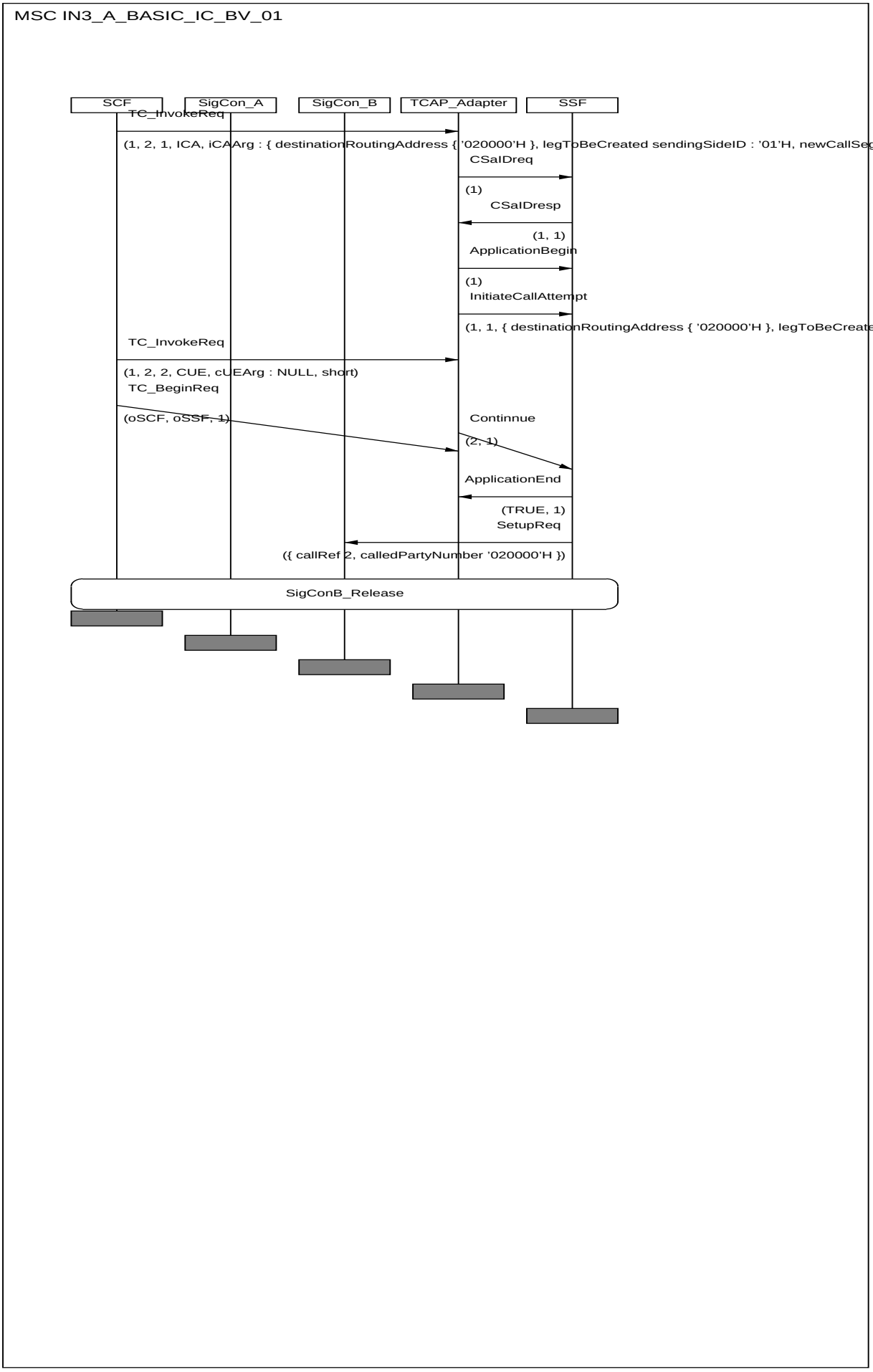

MSC IN3_A_BASIC_DP_BV_06
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
SetupInd
({ callRef 1, calledPartyNumber '012345'H, redirectingNumber '1112223330'H })
ApplicationBegin
(1)
InitialDP
entTypeBCSM analysedInformation, redirectingPartyID '1112223330'H, createdCallSegmentAssociation 1 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
Information, redirectingPartyID '1112223330'H, createdCallSegmentAssociation 1 }, TRUE)
SigConA_Release

	IN3_A_BASIC_DP_BV_07
Work item no.:	ITEM_BASIC_67
IN2 Ref(tmp)	IN2_A_BASIC_DP_BV_05
Purpose:	Verify that the SSF having sent to the SCF an **InitialDP** invoke component on receipt of a **SetupInd** from SigConA, detects that timer Tssf expires, and aborts the dialogue.
Requirements refs	8.2.1.1, 8.2.2, 8.2.2.1, 11.26.1, 11.26.1.1.1, 11.26.2.1
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 a TC_U_ABORT
Postamble:	None

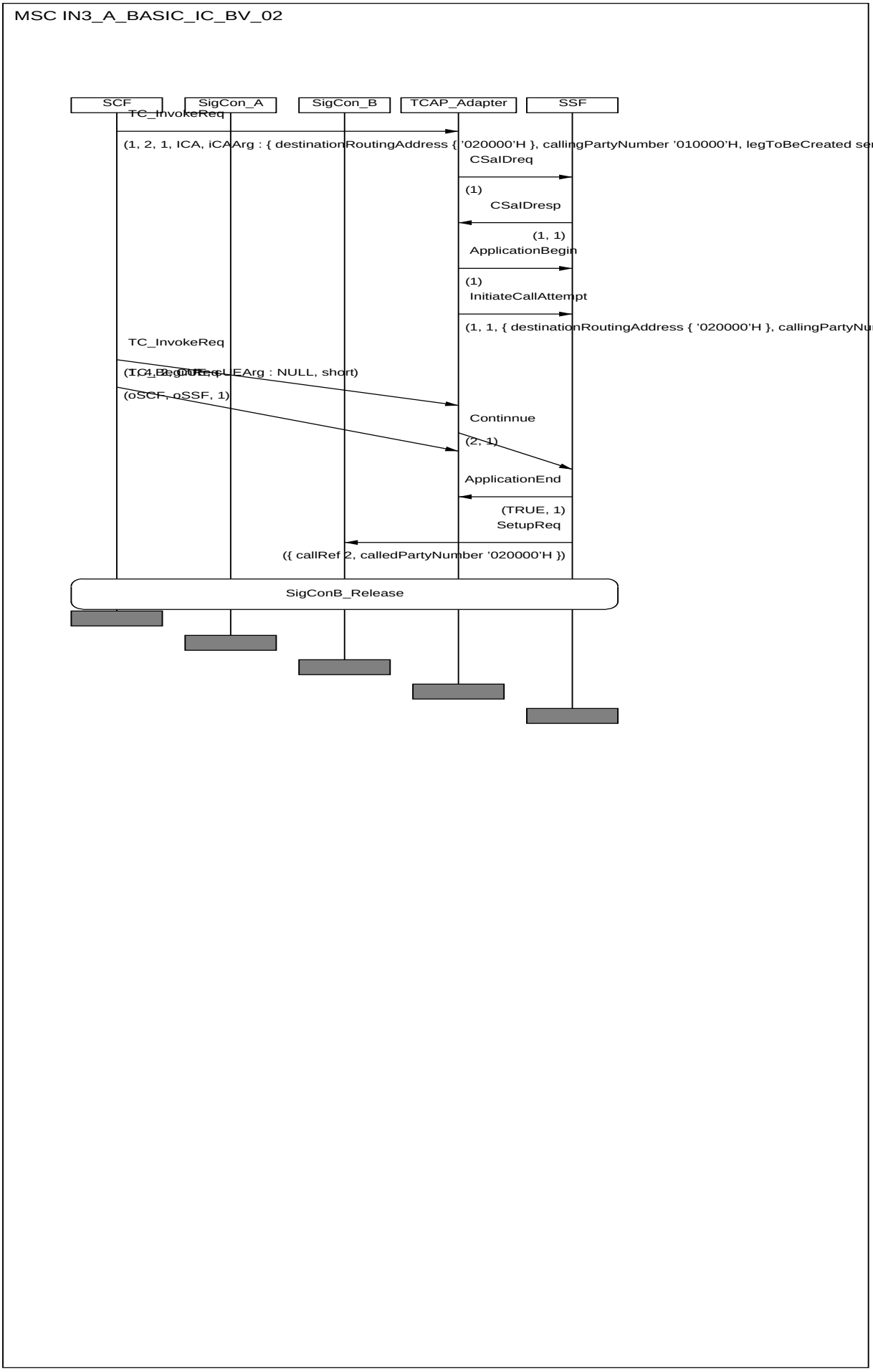

MSC IN3_A_BASIC_DP_BV_07
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_Initialization
SetupInd
({ callRef 1, calledPartyNumber '012345'H })
ApplicationBegin
(1)
InitialDP
1, calledPartyNumber '012345'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
2345'H, eventTypeBCSM analysedInformation, createdCallSegmentAssociation 1 }, TRUE)
Tssf
ApplicationEnd
(TRUE, 1)
TC_AbortInd
(1)

6.6.14 InitiateCallAttempt (IC) procedure

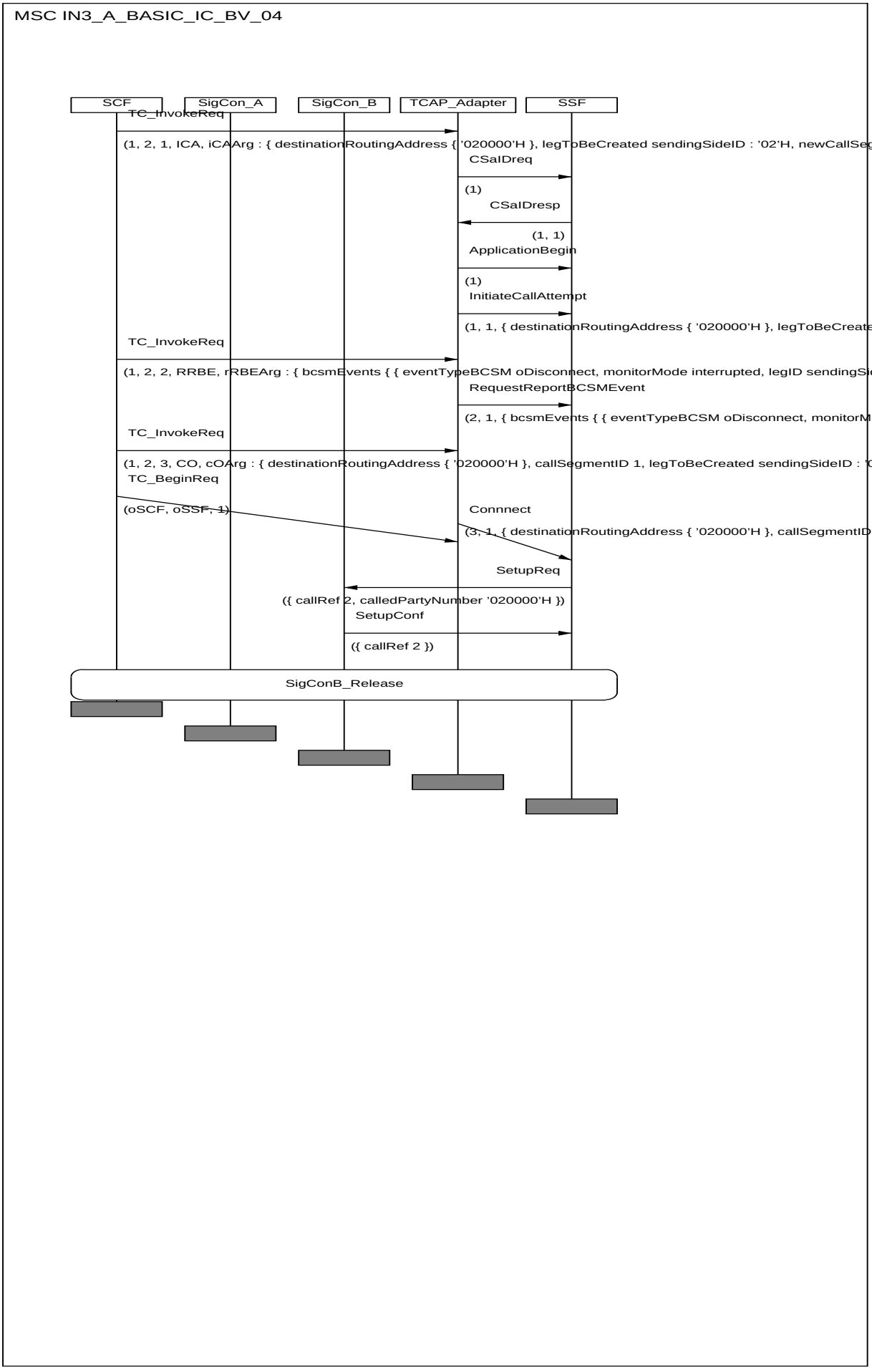
	IN3_A_BASIC_IC_BV_01
Work item no.:	ITEM_BASIC_70
IN2 Ref(tmp)	IN2_A_BASIC_IC_CA_01
Purpose:	Verify that the SSF sends a SetupReq to the SigConA addressed by the **destinationRoutingAddress**, after receiving from the SCF an **InitiateCallAttempt** invoke component with mandatory parameter **destinationRoutingAddress**, followed by a **Continue** invoke component.
Requirements refs	6.5.1.2.3, 8.2, 8.2.1, 8.2.1.1, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 11.14, 11.27.1, 11.27.1.1.1, 11.27.3.1
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 SigConB according to the **InitiateCallAttempt,** that is with calledPartyNumber parameter equal to destinationRoutingAddress.
Postamble:	SigConB_Release


MSC IN3_A_BASIC_IC_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
TC_InvokeReq
(1, 2, 1, ICA, iCAArg : { destinationRoutingAddress { '020000'H }, legToBeCreated sendingSideID : '01'H, newCallSegment 1 }, short
CSalDreq
(1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
InitiateCallAttempt
(1, 1, { destinationRoutingAddress { '020000'H }, legToBeCreated sendingSideID : '0
TC_InvokeReq
(1, 2, 2, CUE, cUEArg : NULL, short)
TC_BeginReq
(oSCF, oSSF, 1)
Continnue
(2, 1)
ApplicationEnd
(TRUE, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H })
SigConB_Release

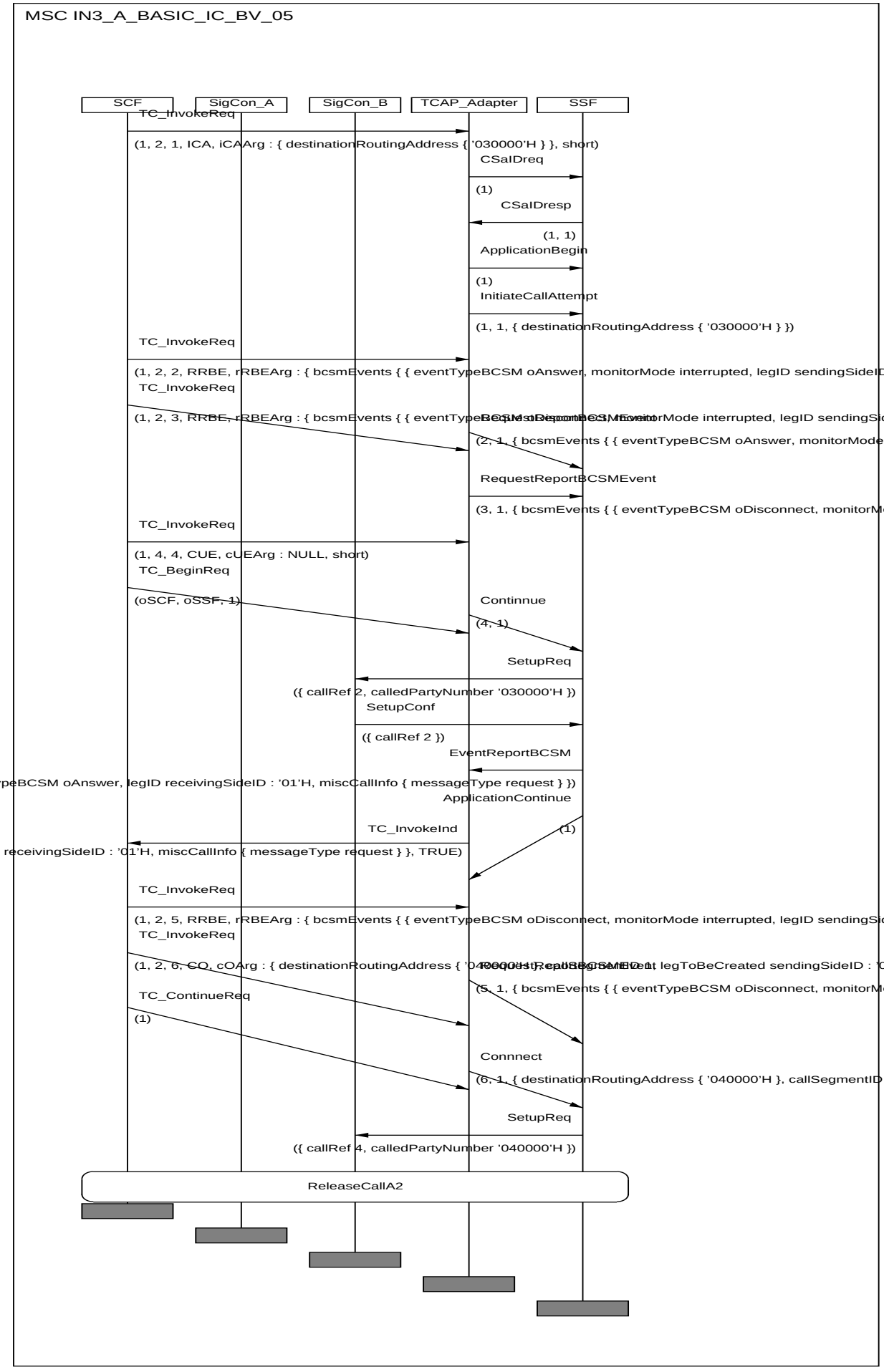
	IN3_A_BASIC_IC_BV_02
Work item no.:	ITEM_BASIC_71
IN2 Ref(tmp)	IN2_A_BASIC_IC_BV_01
Purpose:	Verify that the SSF sends a SetupReq to the SigConA addressed by the **destinationRoutingAddress**, after receiving from the SCF an **InitiateCallAttempt** invoke component containing mandatory parameter **destinationRoutingAddress** and optional parameter **callingPartyNumber**, followed by a **Continue** invoke component.
Requirements refs	6.5.1.2.3, 8.2, 8.2.1, 8.2.1.1, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 11.14, 11.27.1, 11.27.1.1.1, 11.27.3.1
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 SigConB according to the **InitiateCallAttempt,** that is with calledPartyNumber parameter equal to destinationRoutingAddress and callingPartyNumber parameter equal to the original callingPartyNumber.
Postamble:	SigConB_Release


MSC IN3_A_BASIC_IC_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
TC_InvokeReq
(1, 2, 1, ICA, iCAArg : { destinationRoutingAddress { '020000'H }, callingPartyNumber '010000'H, legToBeCreated sendingSideID : '01'H,
CSaIDreq
(1)
CSaIDresp
(1, 1)
ApplicationBegin
(1)
InitiateCallAttempt
(1, 1, { destinationRoutingAddress { '020000'H }, callingPartyNumber '010000'H, legT
TC_InvokeReq
(1, 2, 1, ERB, eRBUEArg : NULL, short)
(oSCF, oSSF, 1)
Continnue
(2, 1)
ApplicationEnd
(TRUE, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H })
SigConB_Release

	IN3_A_BASIC_IC_BV_04
Work item no.:	ITEM_BASIC_369
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received an **InitiateCallAttempt** invoke component containing parameters legID (value 2) and destinationRoutingAddress (valid value), sends a SetupReq to SigCon B after having received a Connect invoke component related to the leg referred to in the **InitiateCallAttempt** invoke.
Requirements refs	6.5.1.2.3, 8.2, 8.2.1, 8.2.1.1, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 11.14, 11.27
Selection Cond.	
Preamble:	none
Test description	L1!InitiateCallAttempt(legID=2,csID=1, valid destinationRoutingAddress) L1!RequestReportBCSMEvent(2,interrupted,oDisconnect) L1!Connect(legID=2,csID=1) CP1_2?SetupReq CP1_2!SetupConf
Pass criteria	CP1_2?SetupReq
Postamble:	SigConB_Release


MSC IN3_A_BASIC_IC_BV_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
TC_InvokeReq
(1, 2, 1, ICA, iCAArg : { destinationRoutingAddress { '020000'H }, legToBeCreated sendingSideID : '02'H, newCallSegment 1 }, short
CSalDreq
(1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
InitiateCallAttempt
(1, 1, { destinationRoutingAddress { '020000'H }, legToBeCreated sendingSideID : '0
TC_InvokeReq
(1, 2, 2, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '02'H } } }, sho
RequestReportBCSMEvent
(2, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID
TC_InvokeReq
(1, 2, 3, CO, cOArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
TC_BeginReq
(oSCF, oSSF, 1)
Connnect
(3, 1, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '020000'H })
SetupConf
({ callRef 2 })
SigConB_Release

	IN3_A_BASIC_IC_BV_05
Work item no.:	ITEM_BASIC_370
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received an **InitiateCallAttempt** invoke component containing only parameter destinationRoutingAddress (valid value), and having performed the connection to SigConA, waiting for Instructions at DP oAnswer, sends a SetupReq to SigCon B after having received a Connect invoke component related to a new leg (leg 2).
Requirements refs	6.5.1.2.3, 8.2, 8.2.1, 8.2.1.1, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 11.14, 11.27
Selection Cond.	
Preamble:	None
Test description	L1!InitiateCallAttempt(legID=OMIT,csID=OMIT, valid destinationRoutingAddress) L1!RequestReportBCSMEvent(1,interrupted,oAnswer) L1!RequestReportBCSMEvent(1,interrupted,oDisconnect) L1!Continue CP1_1?SetupReq CP1_1!SetupConf L1?EventReportBCSM(1,oAnswer) L1!Connect(legID=2,csID=1, new destinationRoutingAddress) CP1_2?SetupReq
Pass criteria	CP1_1?SetupReq CP1_2?SetupReq
Postamble:	ReleaseCallA2


MSC IN3_A_BASIC_IC_BV_05
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
TC_InvokeReq
(1, 2, 1, ICA, iCAArg : { destinationRoutingAddress { '030000'H } }, short)
CSaIDreq
(1)
CSaIDresp
(1, 1)
ApplicationBegin
(1)
InitiateCallAttempt
(1, 1, { destinationRoutingAddress { '030000'H } })
TC_InvokeReq
(1, 2, 2, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID sendingSideID : '01'H } } }, short
TC_InvokeReq
(1, 2, 3, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sho
(2, 1, { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID se
RequestReportBCSMEvent
(3, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(1, 4, 4, CUE, cUEArg : NULL, short)
TC_BeginReq
(oSCF, oSSF, 1)
Continnue
(4, 1)
SetupReq
({ callRef 2, calledPartyNumber '030000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
1, { eventTypeBCSM oAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
M oAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)
TC_InvokeReq
(1, 2, 5, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '02'H }, { event
TC_InvokeReq
(1, 2, 6, CQ, cQArg : { destinationRoutingAddress { '040000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
(5, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_ContinueReq
(1)
Connnect
(6, 1, { destinationRoutingAddress { '040000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 4, calledPartyNumber '040000'H })
ReleaseCallA2

6.6.15 ManageTriggerData (MT) procedure

In the ManageTriggerData TPs triggers have to be identified which have been created and activated or deactivated in a preamble. To identify a particular trigger, terms like **tDPId1** are used, which are assumed to be **INTEGER** values, i.e. one of the indicated and compared values can be:

a) a value of field "oneTrigger" (TDPIdentifier type), or

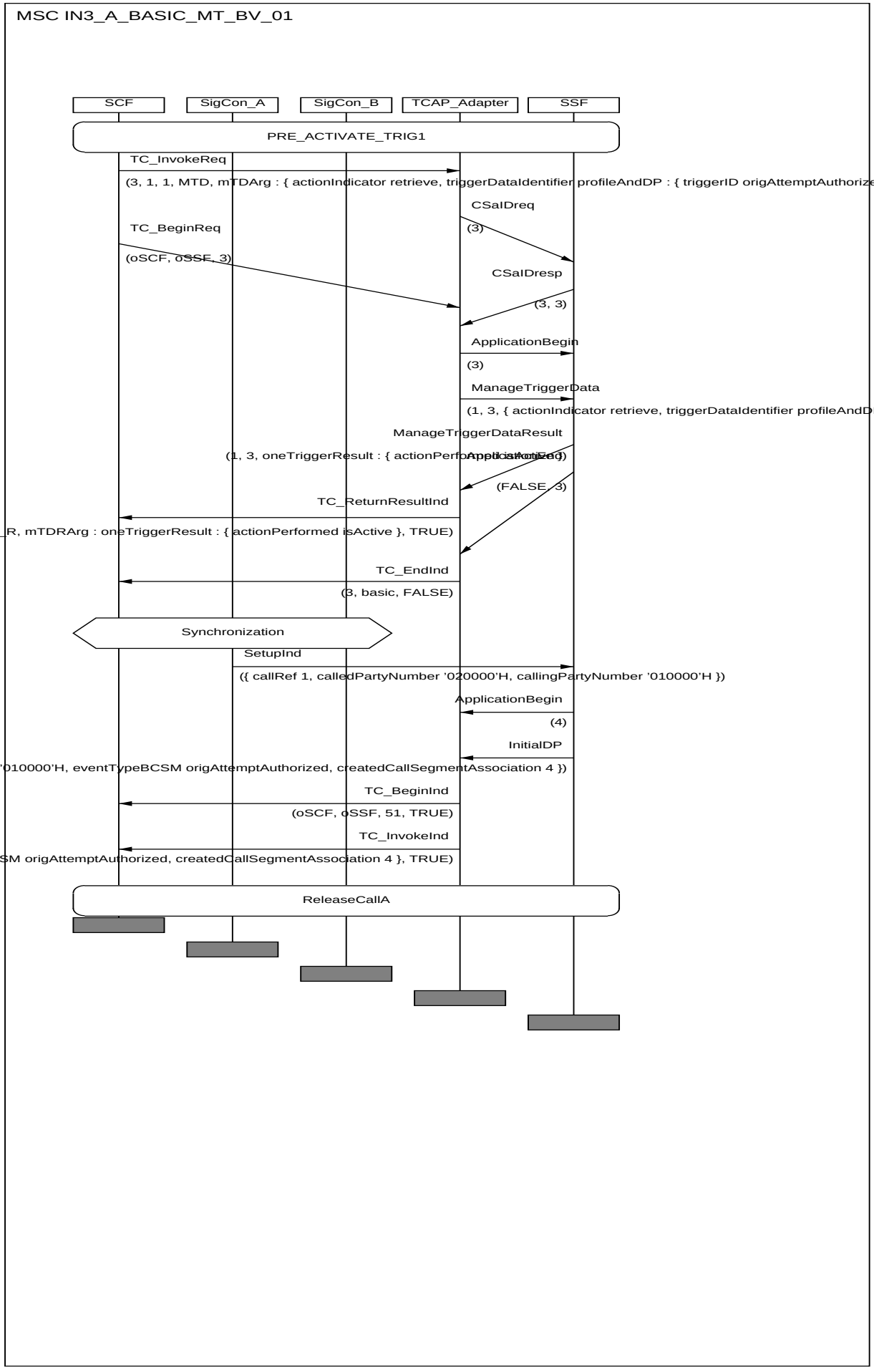
b) a value of field "tDPIdentifier" (Trigger type), or

c) a value of field "tDPIdentifier" (TriggerResult type),

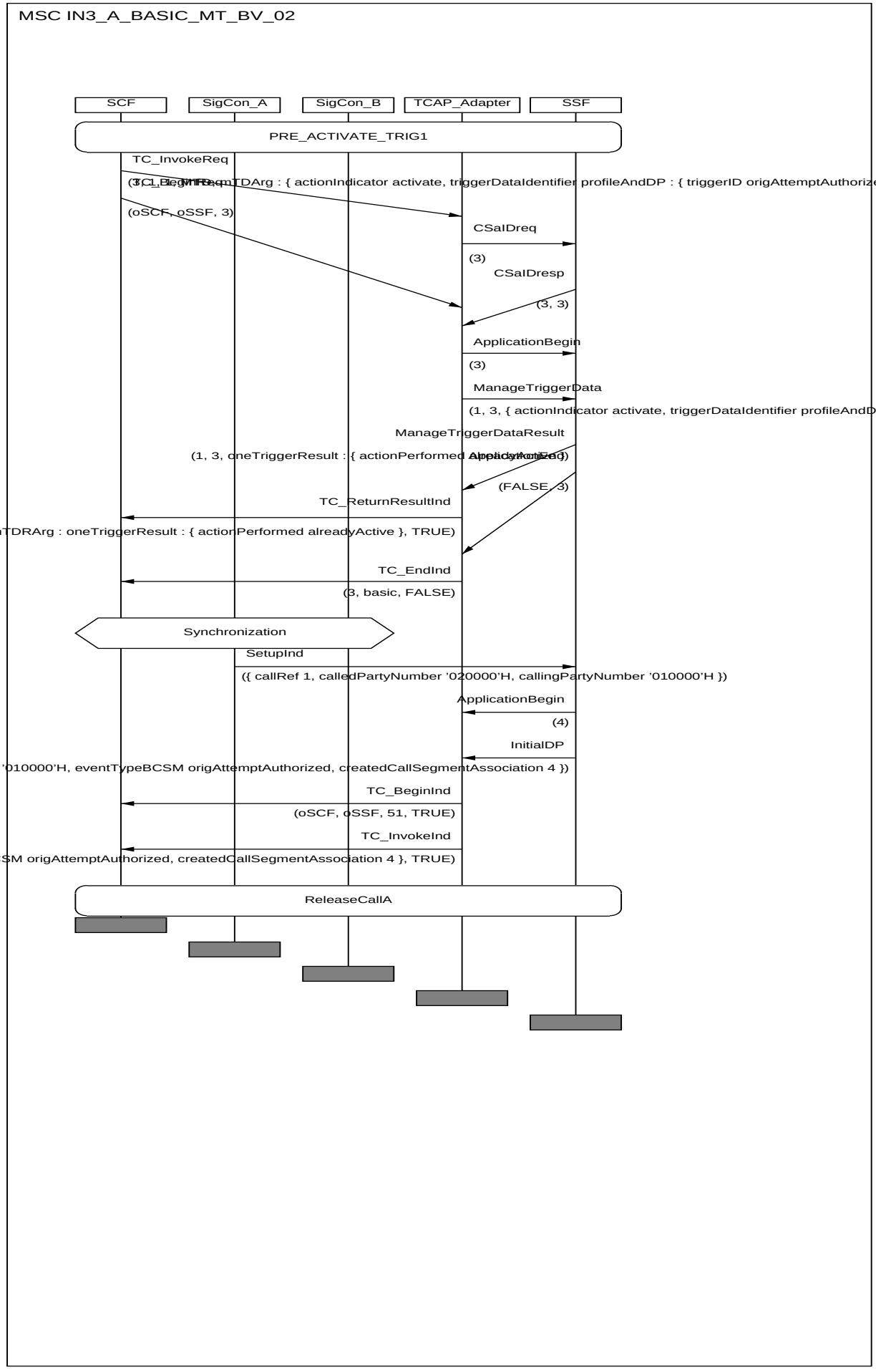
but not direct values of field "tDPIdentifier" appearing the arguments and result arguments of the CreateOrRemoveTriggerData and ManageTriggerData operations.

NOTE: Regarding the situation where the ManageTriggerData argument leaves only one trigger to be reported in the returnResult component: there is no requirement that the "oneTrigger" alternative of TDPIdentifier **must** be used (although this is clearly the most economical way). Therefore the "oneTrigger" and the "triggers" alternatives are accepted in these cases.

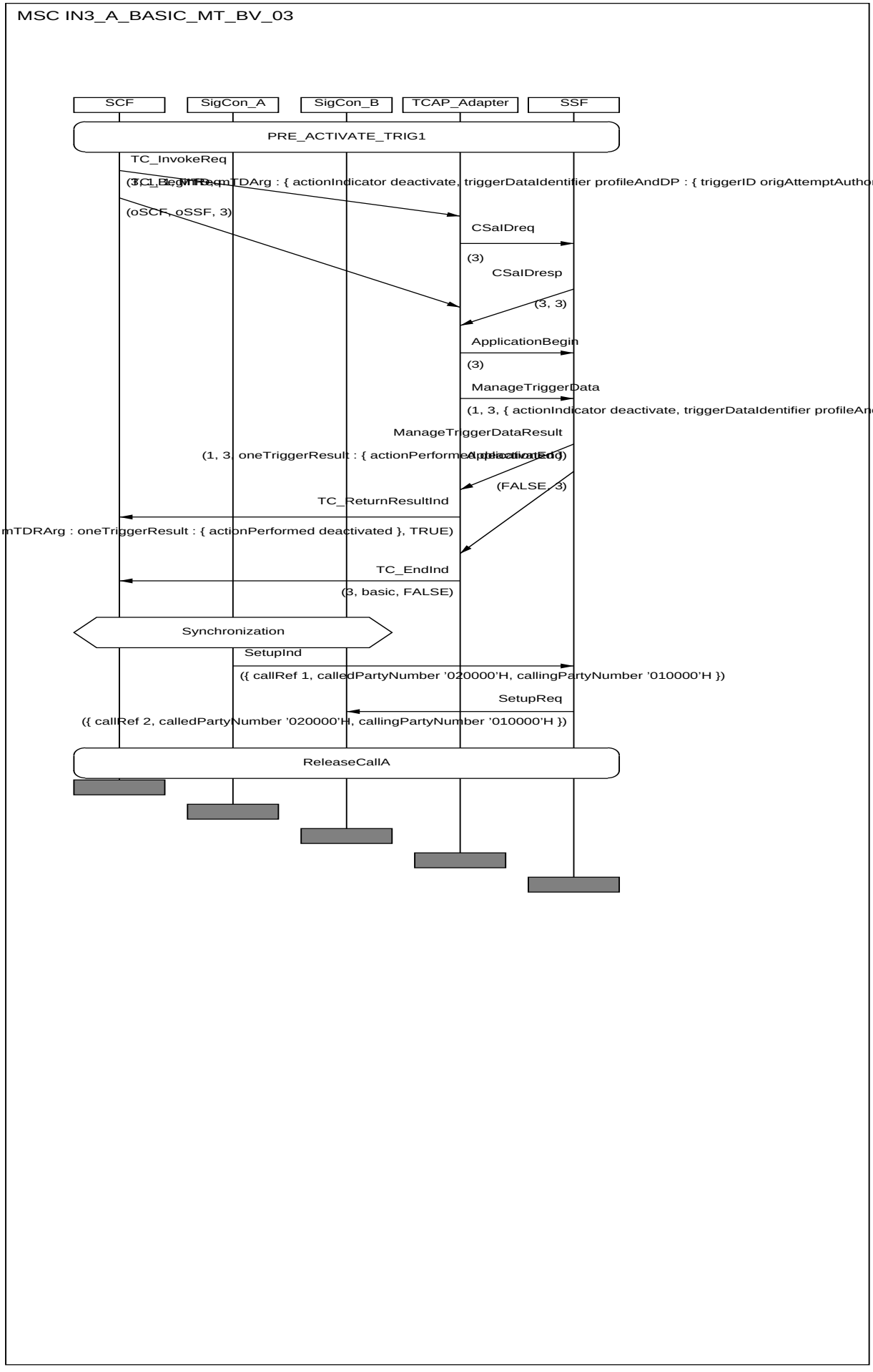
	IN3_A_BASIC_MT_BV_01
Work item no.:	ITEM_BASIC_227
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "isActive", when having received a ManageTriggerData invoke component with parameters actionIndicator = "retrieve", triggerDataIdentifier = profile, identifying **PROFILE_ID_1** and tDPIdentifier = **tDPId1**. Verify also that the SSF, when receiving a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPId1, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPId1) and eventTypeBCSM = "**origAttemptAuthorized**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG1
Test description	L4!ManageTriggerData invoke(retrieve, profile (PROFILE_ID_1), oneTrigger tDPId1) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(isActive, tDPId1) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized)
Pass criteria	L4?ManageTriggerData returnResult(isActive, tDPId1) L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_MT_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG1
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator retrieve, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '0
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator retrieve, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 3, oneTriggerResult : { actionPerformed isActive }, TRUE)
(FALSE, 3)
TC_ReturnResultInd
3, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed isActive }, TRUE)
TC_EndInd
(3, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
ReleaseCallA

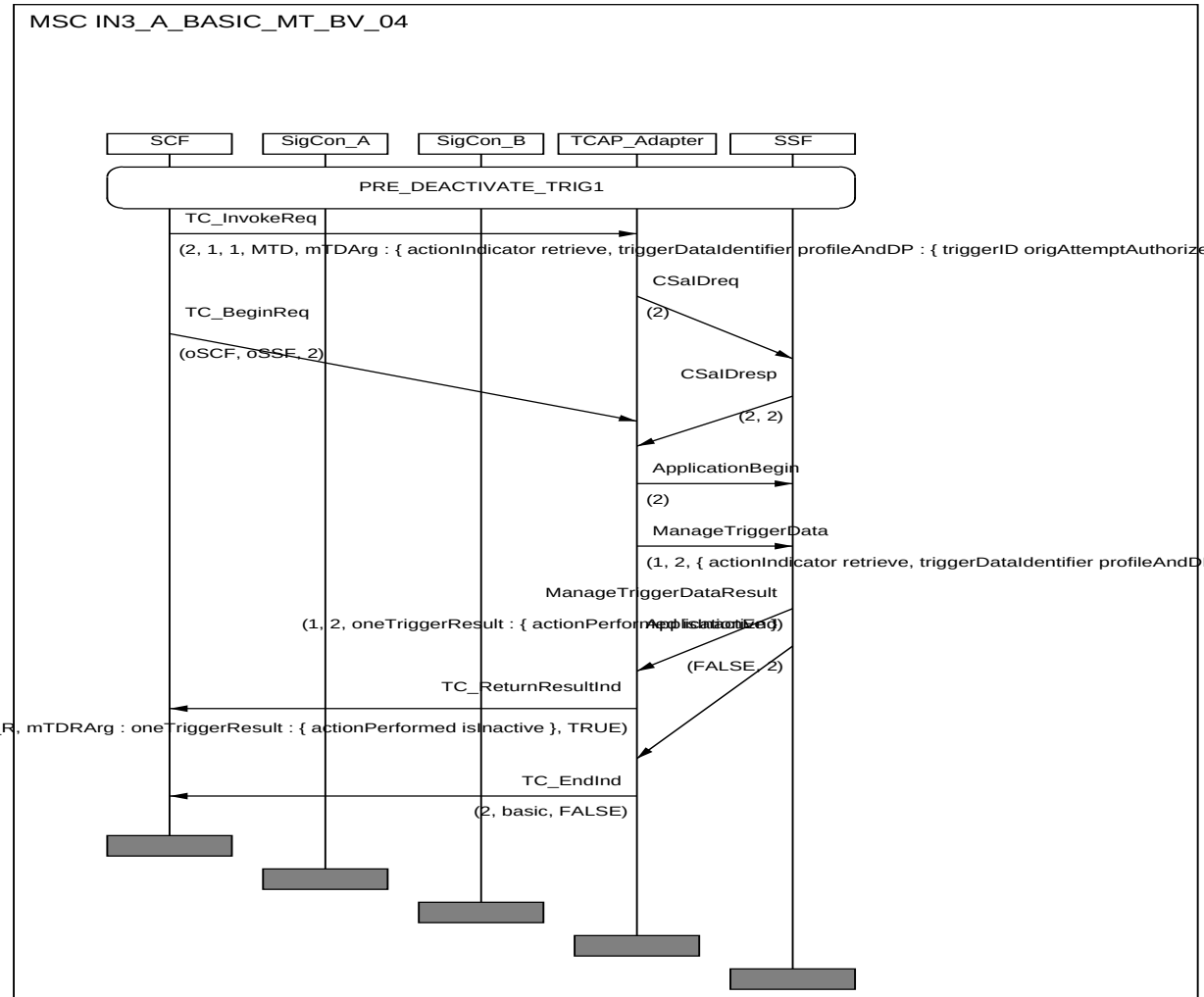
	IN3_A_BASIC_MT_BV_02
Work item no.:	ITEM_BASIC_228
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "alreadyActive", when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1** and tDPIdentifier = **tDPId1**. Verify also that the SSF, when receiving a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPId1, sends to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPId1) and eventTypeBCSM = "**origAttemptAuthorized**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG1
Test description	L4!ManageTriggerData invoke(activate, profile (PROFILE_ID_1), oneTrigger tDPId1) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(alreadyActive, tDPId1) CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized)
Pass criteria	L4?ManageTriggerData returnResult(alreadyActive, tDPId1) L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized)
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_MT_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG1
TC_InvokeReq
(3, 1, MTD_R, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '0
(oSCF, oSSF, 3)
CSalDreq
(3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 3, oneTriggerResult : { actionPerformed alreadyActive }, TRUE)
(FALSE, 3)
TC_ReturnResultInd
3, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed alreadyActive }, TRUE)
TC_EndInd
(3, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
ReleaseCallA

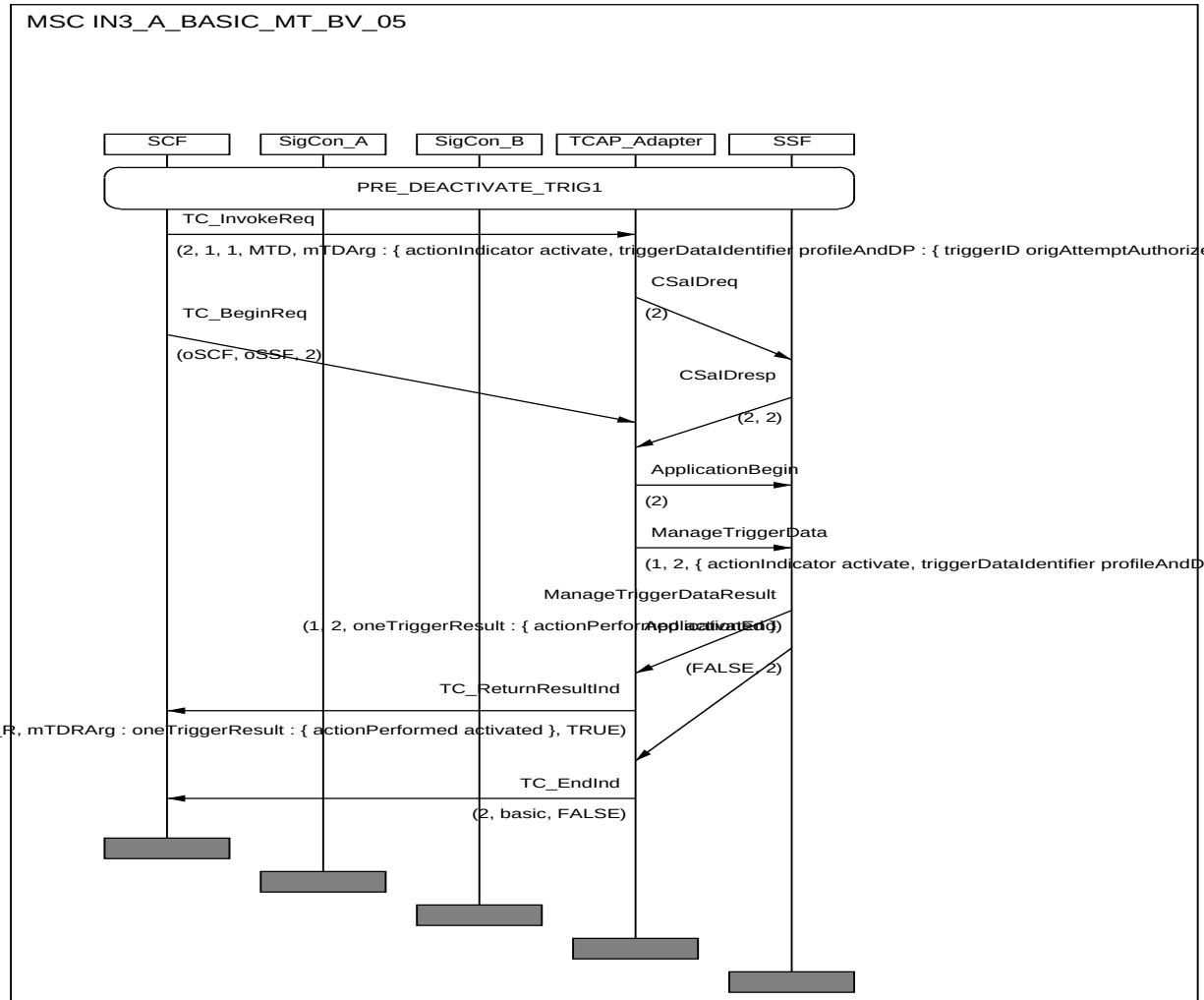
	IN3_A_BASIC_MT_BV_03
Work item no.:	ITEM_BASIC_229
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "deactivated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "deactivate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1** and tDPIdentifier = **tDPId1**. Verify also that the SSF, when receiving a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPId1, does not send to the SCF an InitialDP invoke component containing parameters serviceKey (identifying the serviceKey of tDPId1) and eventTypeBCSM = "**origAttemptAuthorized**".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG1
Test description	L4!ManageTriggerData invoke(deactivate, profile (PROFILE_ID_1), oneTrigger tDPId1) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(deactivated, tDPId1) CP1_1!SetupInd(CALL-DATA-1) CP1_2?SetupReq
Pass criteria	L4?ManageTriggerData returnResult(deactivated, tDPId1) No InitialDP received after SetupInd
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_MT_BV_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG1
TC_InvokeReq
(3, 1, MTDReq, mTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
(oSCF, oSSF, 3)
CSaIDreq
(3)
CSaIDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID orig
ManageTriggerDataResult
(1, 3, oneTriggerResult : { actionPerformed deactivated })
(FALSE, 3)
TC_ReturnResultInd
3, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed deactivated }, TRUE)
TC_EndInd
(3, basic, FALSE)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseCallA

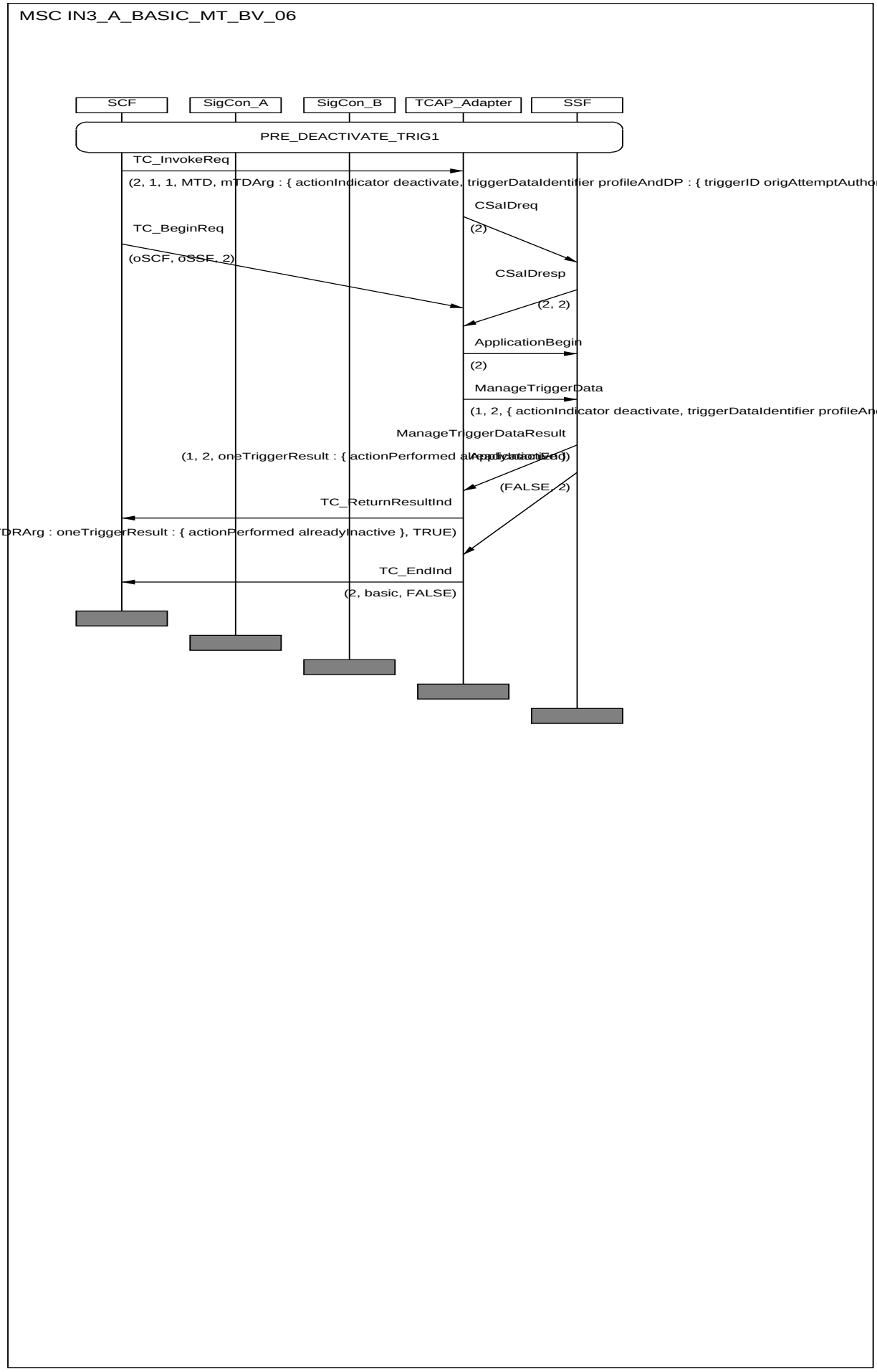
	IN3_A_BASIC_MT_BV_04
Work item no.:	ITEM_BASIC_230
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "isInactive", when having received a ManageTriggerData invoke component with parameters actionIndicator = "retrieve", triggerDataIdentifier = profile, identifying **PROFILE_ID_1** and tDPIdentifier = **tDPId1**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG1
Test description	L3!ManageTriggerData invoke(retrieve, profile (PROFILE_ID_1), oneTrigger tDPId1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(isInactive, tDPId1)
Pass criteria	L3?ManageTriggerData returnResult(isInactive, tDPId1)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG1
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator retrieve, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '0
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator retrieve, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed isInactive }, TRUE)
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed isInactive }, TRUE)
TC_EndInd
(2, basic, FALSE)

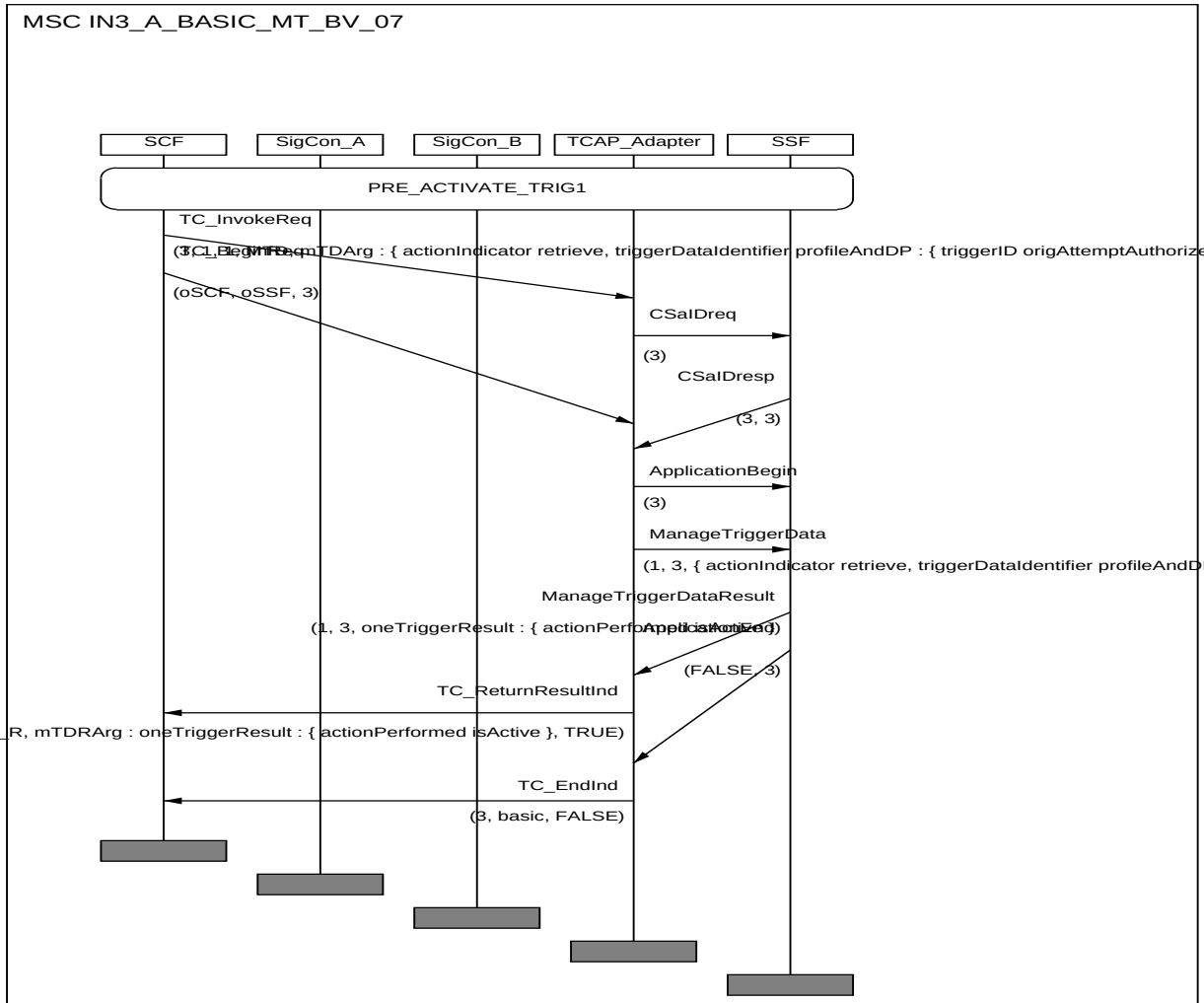
	IN3_A_BASIC_MT_BV_05
Work item no.:	ITEM_BASIC_231
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "activated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1** and tDPIdentifier = **tDPId1**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG1
Test description	L3!ManageTriggerData invoke(activate, profile (PROFILE_ID_1), oneTrigger tDPId1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)
Pass criteria	L3?ManageTriggerData returnResult(activated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_05
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG1
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '0
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated }
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)

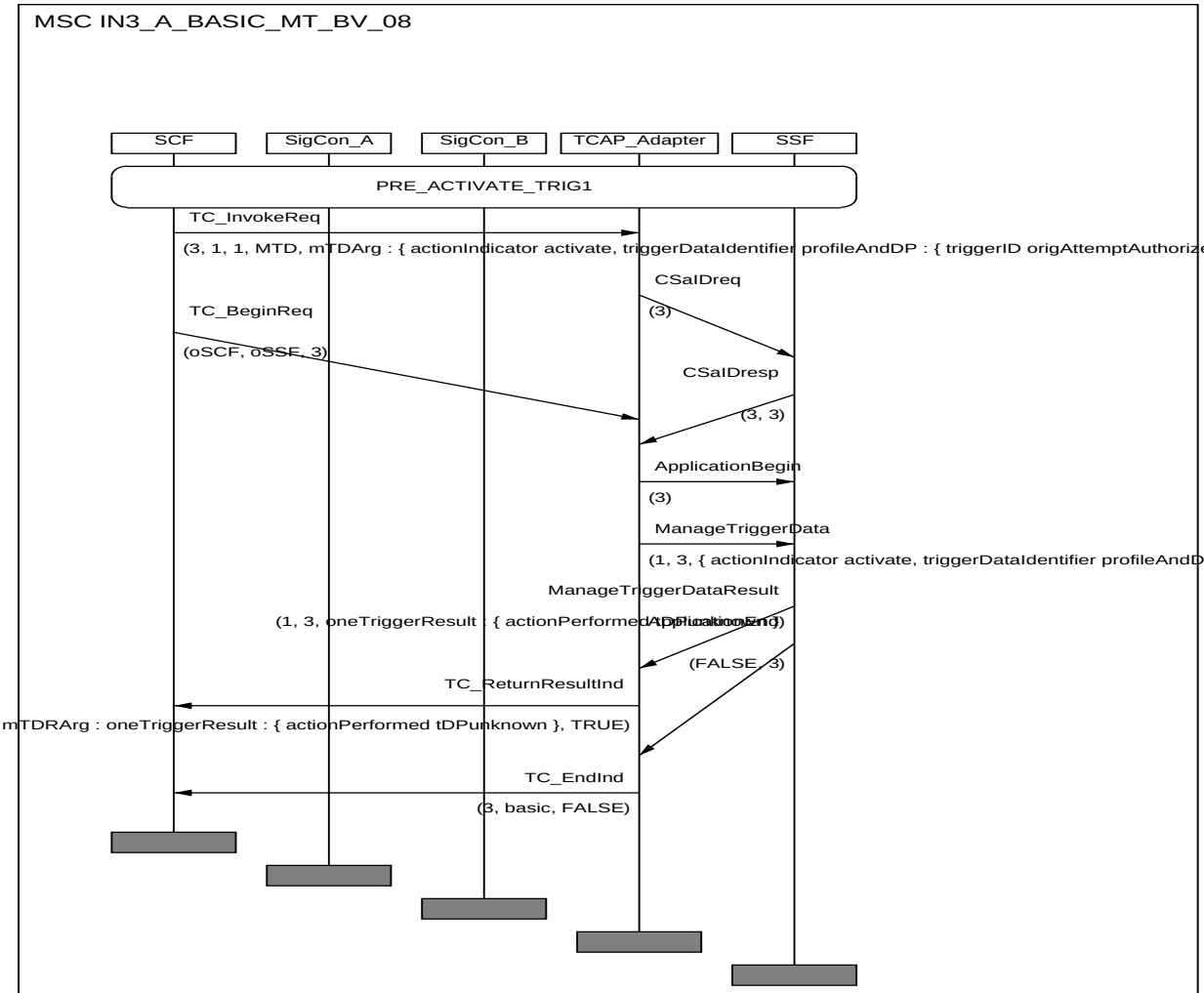
	IN3_A_BASIC_MT_BV_06
Work item no.:	ITEM_BASIC_232
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "alreadyInactive", when having received a ManageTriggerData invoke component with parameters actionIndicator = "deactivate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1** and tDPIdentifier = **tDPId1**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG1
Test description	L3!ManageTriggerData invoke(deactivate, profile (PROFILE_ID_1), oneTrigger tDPId1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(alreadyInactive)
Pass criteria	L3?ManageTriggerData returnResult(alreadyInactive)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_06
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG1
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID orig
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed alreadyInactive }, TRUE)
TC_EndInd
(2, basic, FALSE)

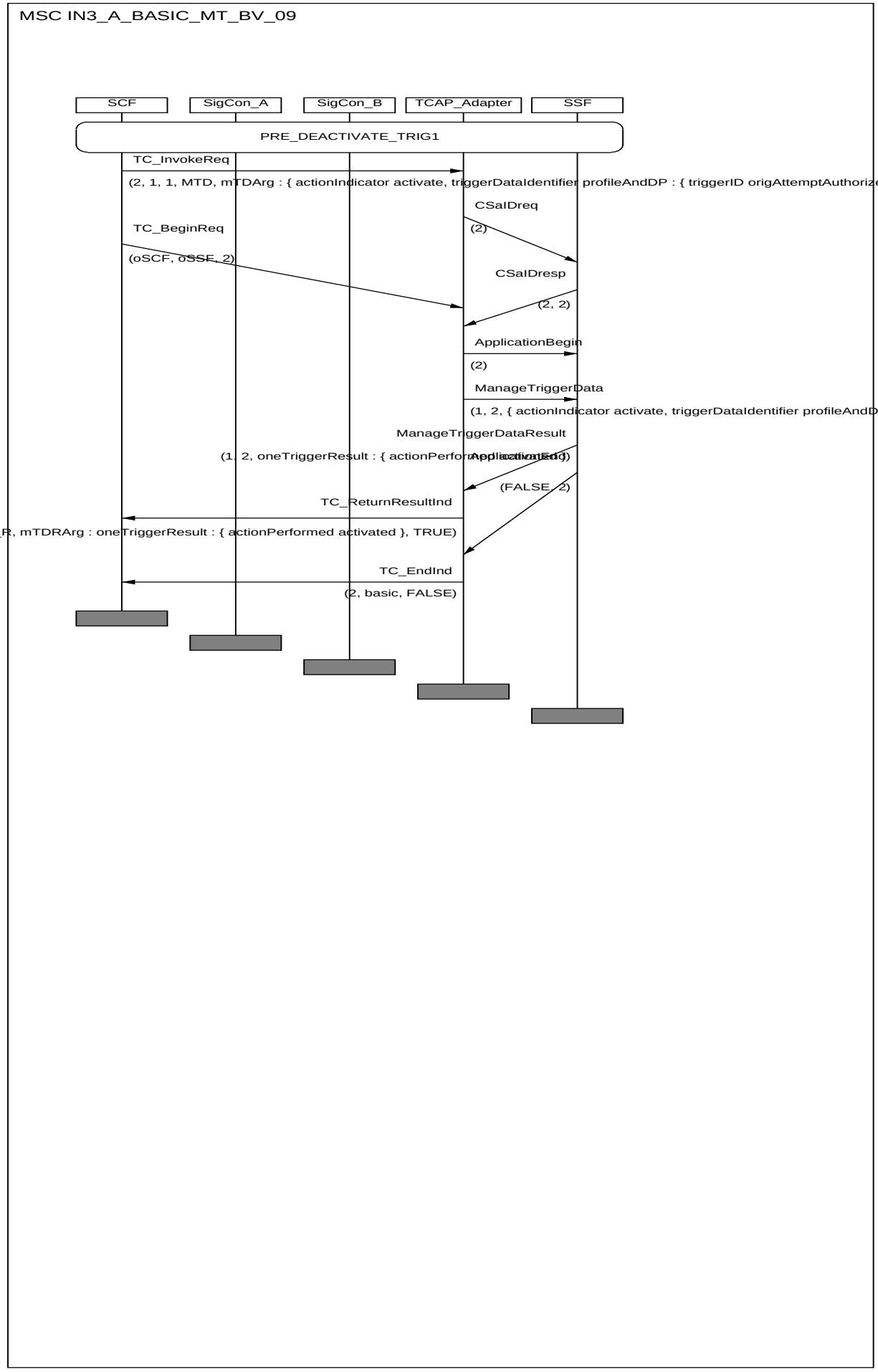
	IN3_A_BASIC_MT_BV_07
Work item no.:	ITEM_BASIC_233
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "isActive", when having received a ManageTriggerData invoke component with parameters actionIndicator = "retrieve", triggerDataIdentifier = profile, identifying **PROFILE_ID_1** and tDPIdentifier is omitted.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG1
Test description	L4!ManageTriggerData invoke(retrieve, profile (PROFILE_ID_1), tDPIdentifier **omitted**) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(isActive)
Pass criteria	L4?ManageTriggerData returnResult(isActive)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_07
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG1
TC_InvokeReq
(3, 1, MTD_R, mTDArg : { actionIndicator retrieve, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : ''H
(oSCF, oSSF, 3)
CSalDreq
(3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator retrieve, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 3, oneTriggerResult : { actionPerformed isActive })
(FALSE, 3)
TC_ReturnResultInd
3, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed isActive }, TRUE)
TC_EndInd
(3, basic, FALSE)

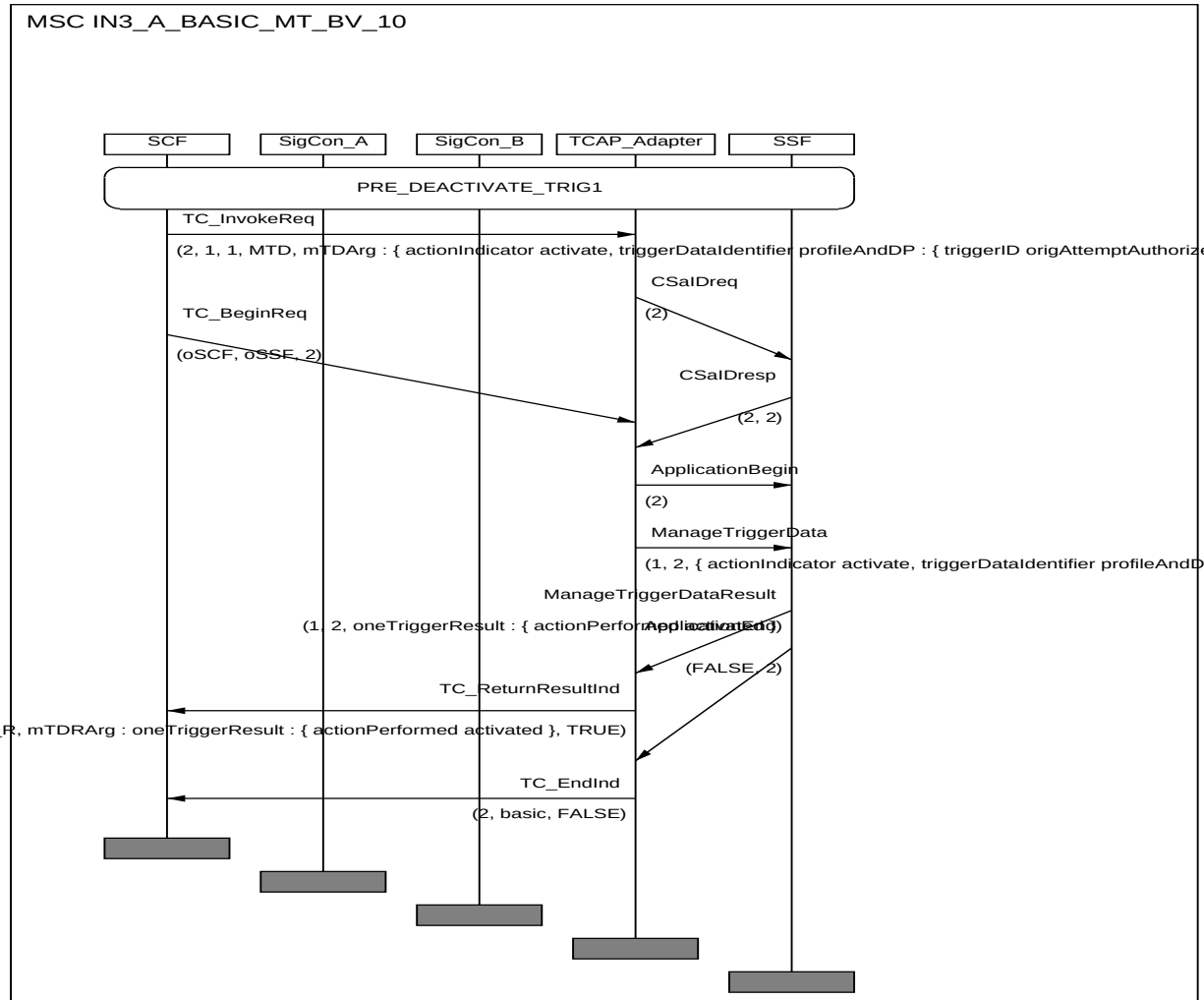
	IN3_A_BASIC_MT_BV_08
Work item no.:	ITEM_BASIC_234
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating the unknown tDPIdentifier and actionPerformed = "tDPunknown", when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1** and tDPIdentifier is **present** and **not equal** to tDPId1.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG1
Test description	L4!ManageTriggerData invoke(activate, profile (PROFILE_ID_1), **unknown** tDPIdentifier) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(tDPunknown)
Pass criteria	L4?ManageTriggerData returnResult(tDPunknown)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_08
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG1
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '0
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 3, oneTriggerResult : { actionPerformed tDPunknown }, TRUE)
(FALSE, 3)
TC_ReturnResultInd
3, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed tDPunknown }, TRUE)
TC_EndInd
(3, basic, FALSE)

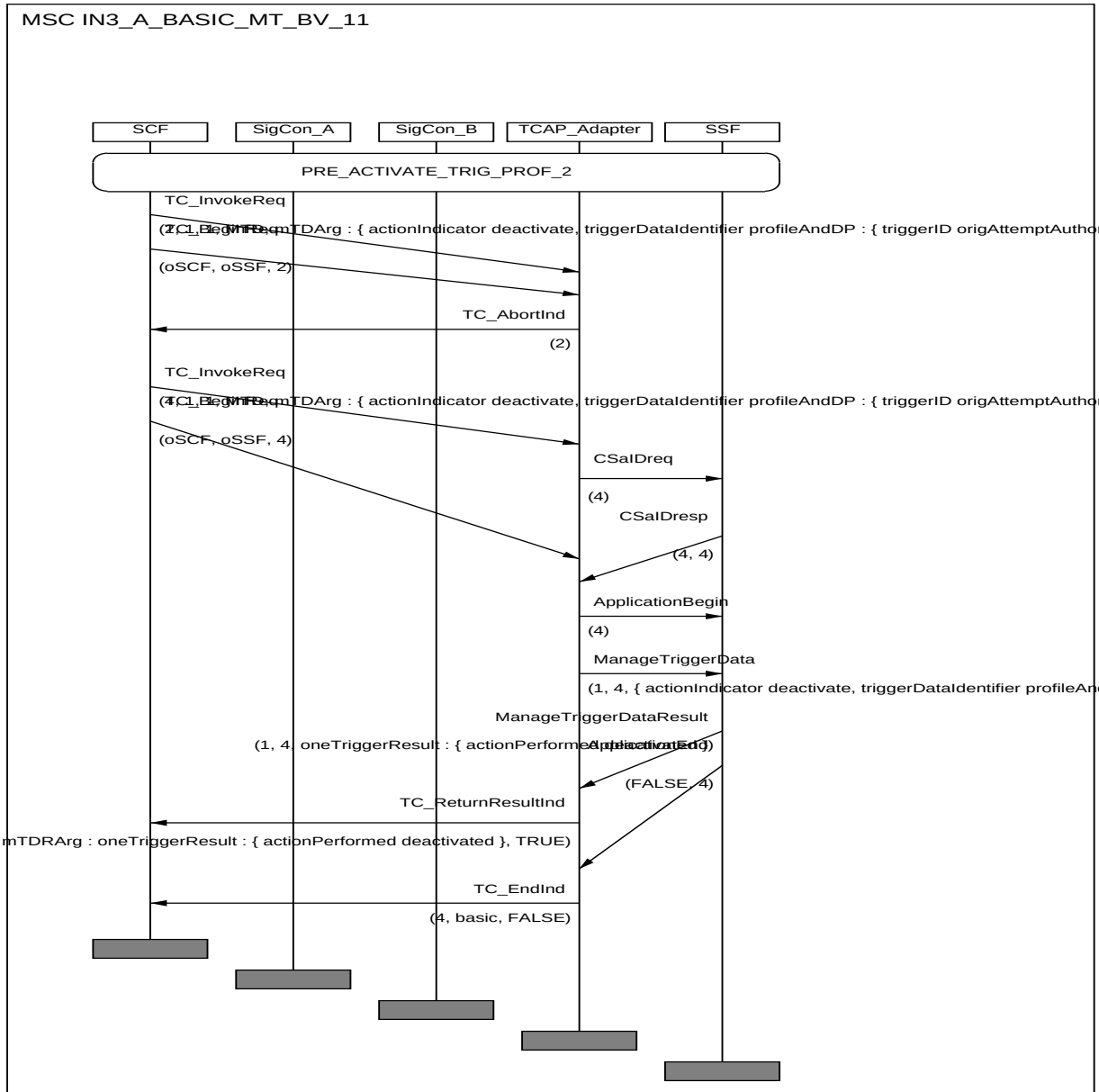
	IN3_A_BASIC_MT_BV_09
Work item no.:	ITEM_BASIC_235
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "activated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profileAndDP, identifying **oOrigAttemptAuthorized** and **PROFILE_ID_1,** and tDPIdentifier = **tDPId1**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG1
Test description	L3!ManageTriggerData invoke(activate, profileAndDP(OrigAttemptAuthorized, PROFILE_ID_1), oneTrigger tDPId1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)
Pass criteria	L3?ManageTriggerData returnResult(activated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_09
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG1
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '0
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated }, TRUE)
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)

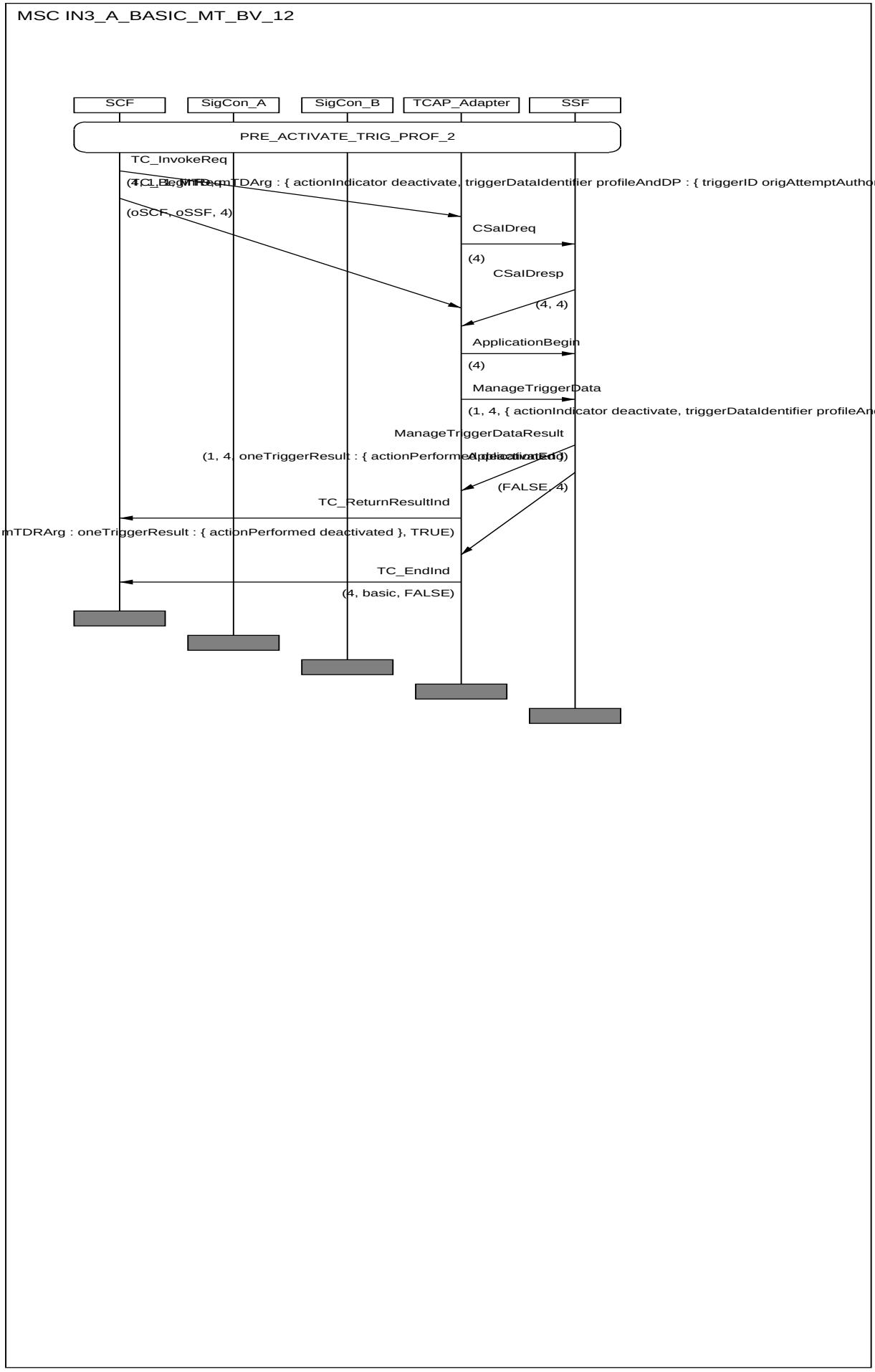
	IN3_A_BASIC_MT_BV_10
Work item no.:	ITEM_BASIC_236
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "activated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profileAndDP, identifying **oOrigAttemptAuthorized** and **PROFILE_ID_1** and tDPIdentifier is omitted.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG1
Test description	L3!ManageTriggerData invoke(activate, profileAndDP(OrigAttemptAuthorized, PROFILE_ID_1), tDPIdentifier **omitted**) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnResult(activated)
Pass criteria	L3?ManageTriggerData returnResult(activated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_10
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG1
TC_InvokeReq
(2, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '0
CSalDreq
(2)
TC_BeginReq
(oSCF, oSSF, 2)
CSalDresp
(2, 2)
ApplicationBegin
(2)
ManageTriggerData
(1, 2, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 2, oneTriggerResult : { actionPerformed activated }, TRUE)
(FALSE, 2)
TC_ReturnResultInd
2, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(2, basic, FALSE)

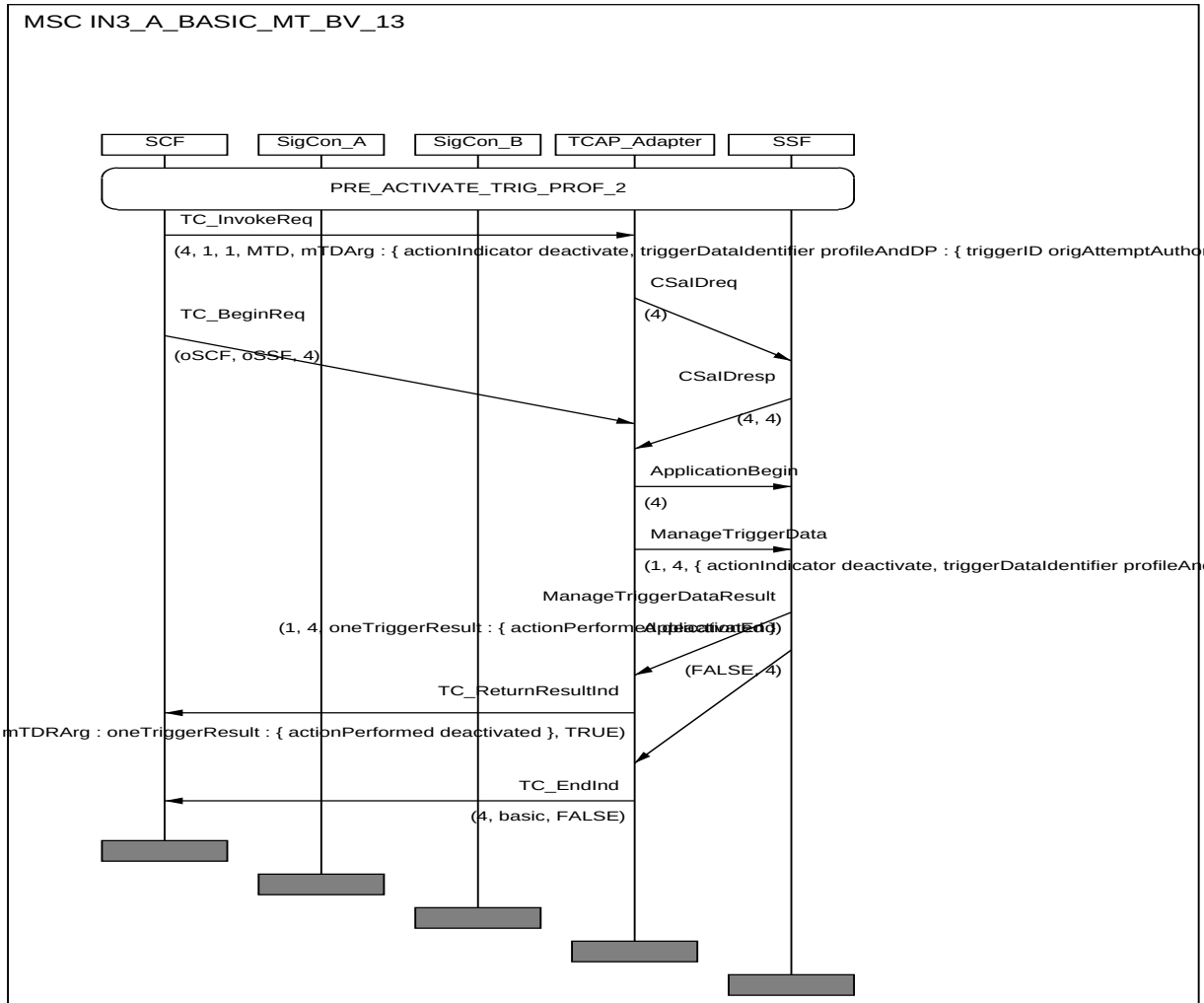
	IN3_A_BASIC_MT_BV_11
Work item no.:	ITEM_BASIC_237
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated two Triggers, both on DP **oOrigAttemptAuthorized** and for different profiles identified by **PROFILE_ID_1** and **PROFILE_ID_1A** respectively, with specific tDPIdentifiers (Integer) = **tDPId1** and **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "deactivated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "deactivate", triggerDataIdentifier = profileAndDP, identifying **oOrigAttemptAuthorized** and **PROFILE_ID_1,** and tDPIdentifier = **tDPId1**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_PROF_2
Test description	L5!ManageTriggerData invoke(deactivate, profileAndDP(OrigAttemptAuthorized, PROFILE_ID_1), oneTrigger tDPId1) L5!TC-BEGIN L5?TC-END L5?ManageTriggerData returnResult(deactivated)
Pass criteria	L5?ManageTriggerData returnResult(deactivated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_11
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_PROF_2
TC_InvokeReq
(2, 1, MTDReq, mTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
(oSCF, oSSF, 2)
TC_AbortInd
(2)
TC_InvokeReq
(4, 1, MTDReq, mTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
(oSCF, oSSF, 4)
CSalDreq
(4)
CSalDresp
(4, 4)
ApplicationBegin
(4)
ManageTriggerData
(1, 4, { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID orig
ManageTriggerDataResult
(1, 4, oneTriggerResult : { actionPerformed deactivated }, TRUE)
(FALSE, 4)
TC_ReturnResultInd
4, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed deactivated }, TRUE)
TC_EndInd
(4, basic, FALSE)

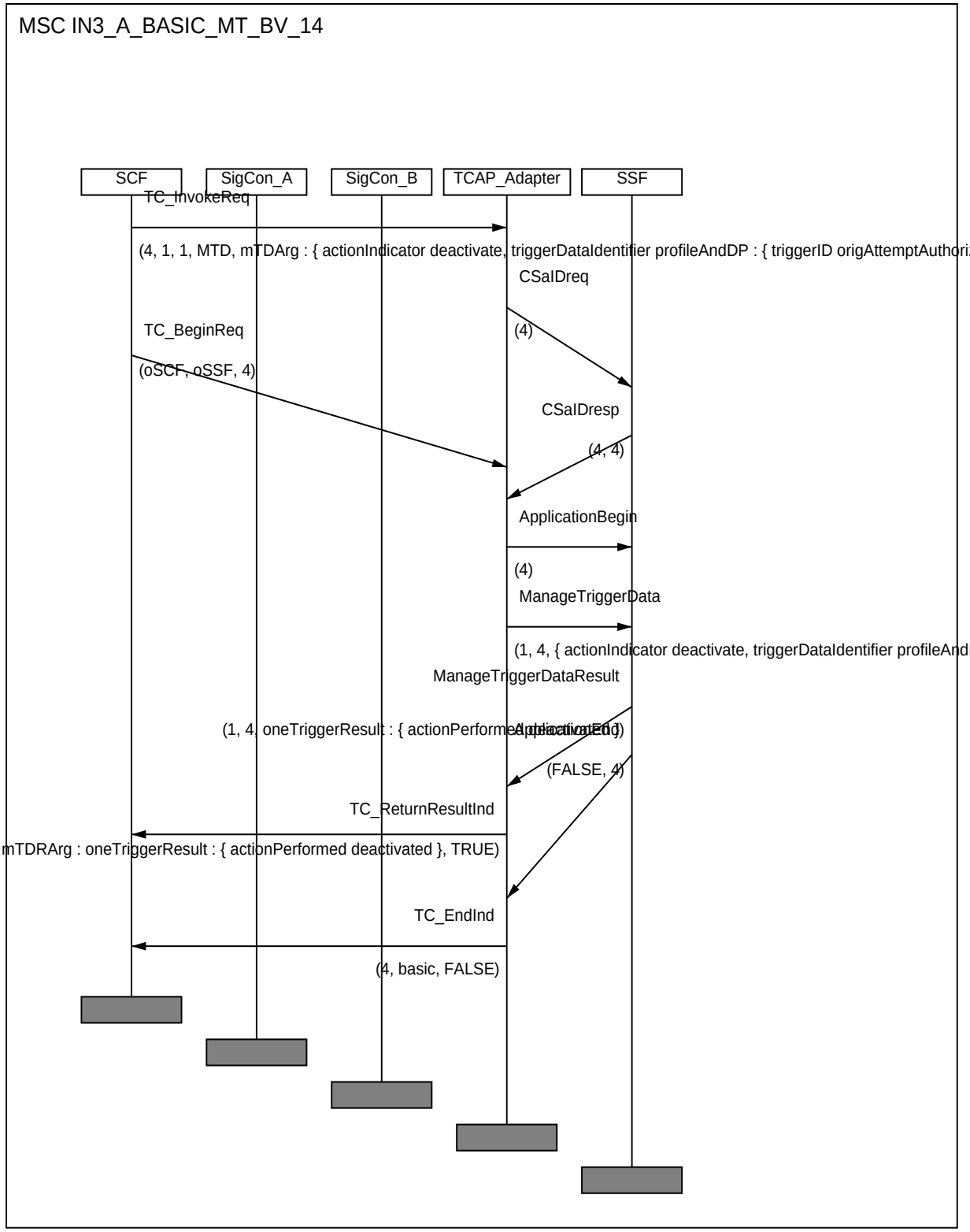
	IN3_A_BASIC_MT_BV_12
Work item no.:	ITEM_BASIC_238
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated two Triggers, both on DP **oOrigAttemptAuthorized** and for different profiles identified by **PROFILE_ID_1** and **PROFILE_ID_1A** respectively, with specific tDPIdentifiers (Integer) = **tDPId1** and **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId2** and actionPerformed = "deactivated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "deactivate", triggerDataIdentifier = profileAndDP, identifying **oOrigAttemptAuthorized** and **PROFILE_ID_1A,** and tDPIdentifier = **tDPId2**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_PROF_2
Test description	L5!ManageTriggerData invoke(deactivate, profileAndDP(OrigAttemptAuthorized, PROFILE_ID_1A), oneTrigger **tDPId2**) L5!TC-BEGIN L5?TC-END L5?ManageTriggerData returnResult(deactivated)
Pass criteria	L5?ManageTriggerData returnResult(deactivated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_12
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_PROF_2
TC_InvokeReq
(4, 1, MTD_R, mTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
(oSCF, oSSF, 4)
CSalDreq
(4)
CSalDresp
(4, 4)
ApplicationBegin
(4)
ManageTriggerData
(1, 4, { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID orig
ManageTriggerDataResult
(1, 4, oneTriggerResult : { actionPerformed deactivated })
(FALSE, 4)
TC_ReturnResultInd
4, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed deactivated }, TRUE)
TC_EndInd
(4, basic, FALSE)

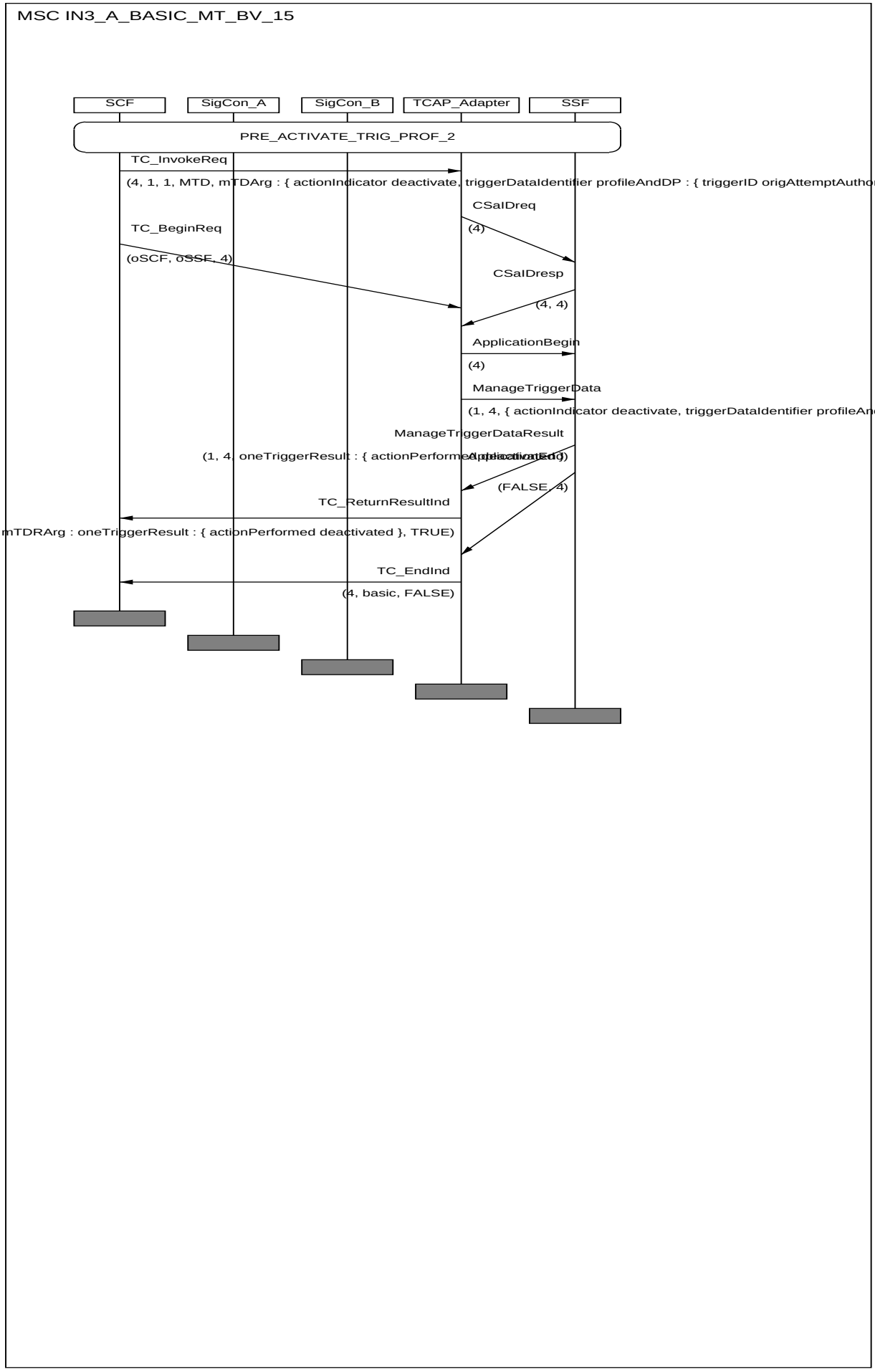
	IN3_A_BASIC_MT_BV_13
Work item no.:	ITEM_BASIC_239
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated two Triggers, both on DP **oOrigAttemptAuthorized** and for different profiles identified by **PROFILE_ID_1** and **PROFILE_ID_1A** respectively, with specific tDPIdentifiers (Integer) = **tDPId1** and **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId2** and actionPerformed = "deactivated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "deactivate", triggerDataIdentifier = profileAndDP, identifying **oOrigAttemptAuthorized** and **PROFILE_ID_1A,** and tDPIdentifier is omitted.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_PROF_2
Test description	L5!ManageTriggerData invoke(deactivate, profileAndDP(OrigAttemptAuthorized, PROFILE_ID_1A), tDPIdentifier **omitted**) L5!TC-BEGIN L5?TC-END L5?ManageTriggerData returnResult(deactivated)
Pass criteria	L5?ManageTriggerData returnResult(deactivated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_13
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_PROF_2
TC_InvokeReq
(4, 1, 1, MTD, mTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
CSalDreq
(4)
TC_BeginReq
(oSCF, oSSF, 4)
CSalDresp
(4, 4)
ApplicationBegin
(4)
ManageTriggerData
(1, 4, { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID orig
ManageTriggerDataResult
(1, 4, oneTriggerResult : { actionPerformed deactivated
(FALSE, 4)
TC_ReturnResultInd
4, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed deactivated }, TRUE)
TC_EndInd
(4, basic, FALSE)

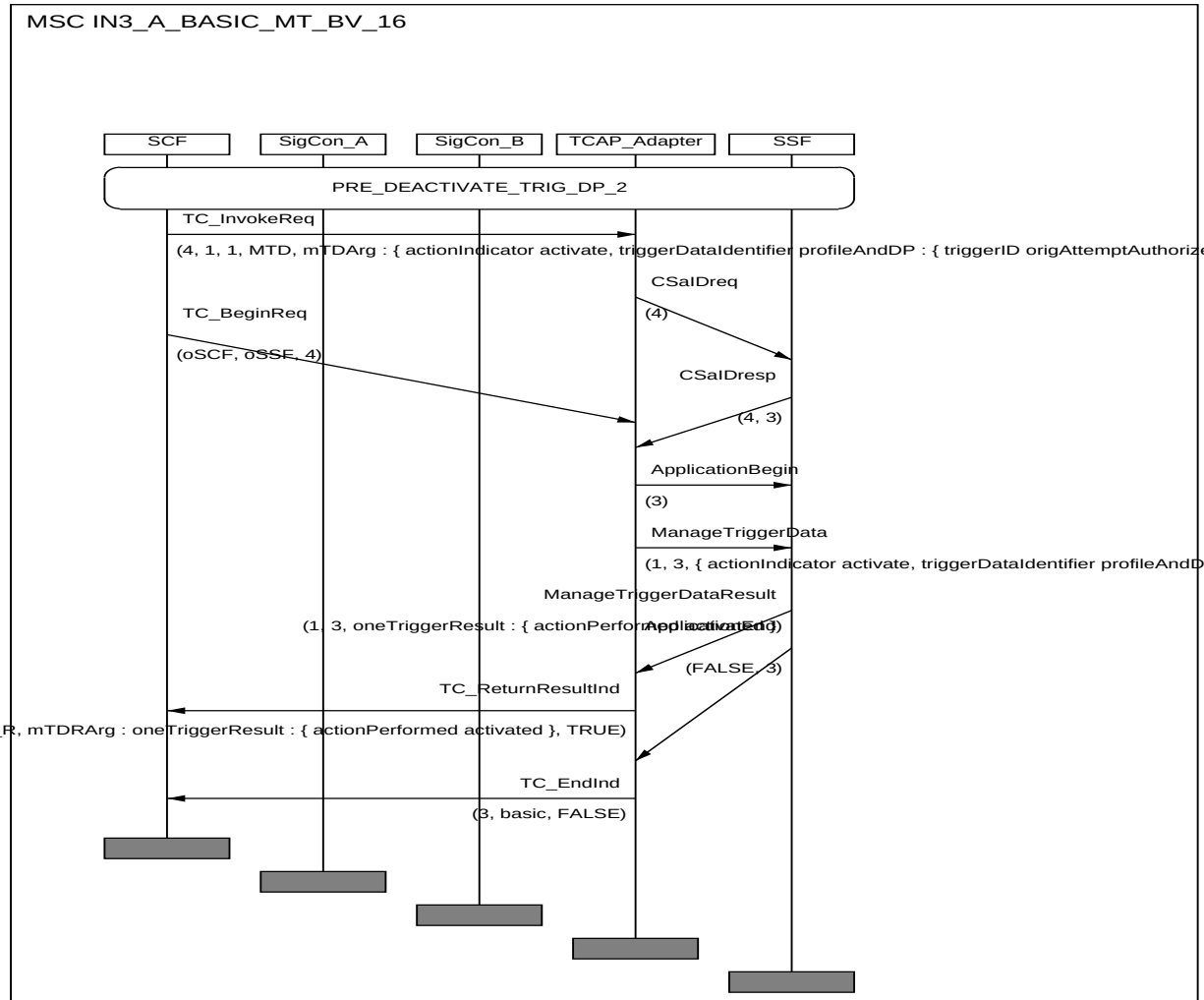
	IN3_A_BASIC_MT_BV_14
Work item no.:	ITEM_BASIC_240
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated two Triggers, both on DP **oOrigAttemptAuthorized** and for different profiles identified by **PROFILE_ID_1** and **PROFILE_ID_1A** respectively, with specific tDPIdentifiers (Integer) = **tDPId1** and **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "deactivated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "deactivate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1,** and tDPIdentifier = **tDPId1**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_PROF_2
Test description	L5!ManageTriggerData invoke(deactivate, profile(PROFILE_ID_1), oneTrigger tDPId1) L5!TC-BEGIN L5?TC-END L5?ManageTriggerData returnResult(deactivated)
Pass criteria	L5?ManageTriggerData returnResult(deactivated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_14
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
TC_InvokeReq
(4, 1, 1, MTD, mTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
CSaIDreq
TC_BeginReq
(4)
(oSCF, oSSF, 4)
CSaIDresp
(4, 4)
ApplicationBegin
(4)
ManageTriggerData
(1, 4, { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID orig
ManageTriggerDataResult
(1, 4, oneTriggerResult : { actionPerformed deactivated }
(FALSE, 4)
TC_ReturnResultInd
4, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed deactivated }, TRUE)
TC_EndInd
(4, basic, FALSE)

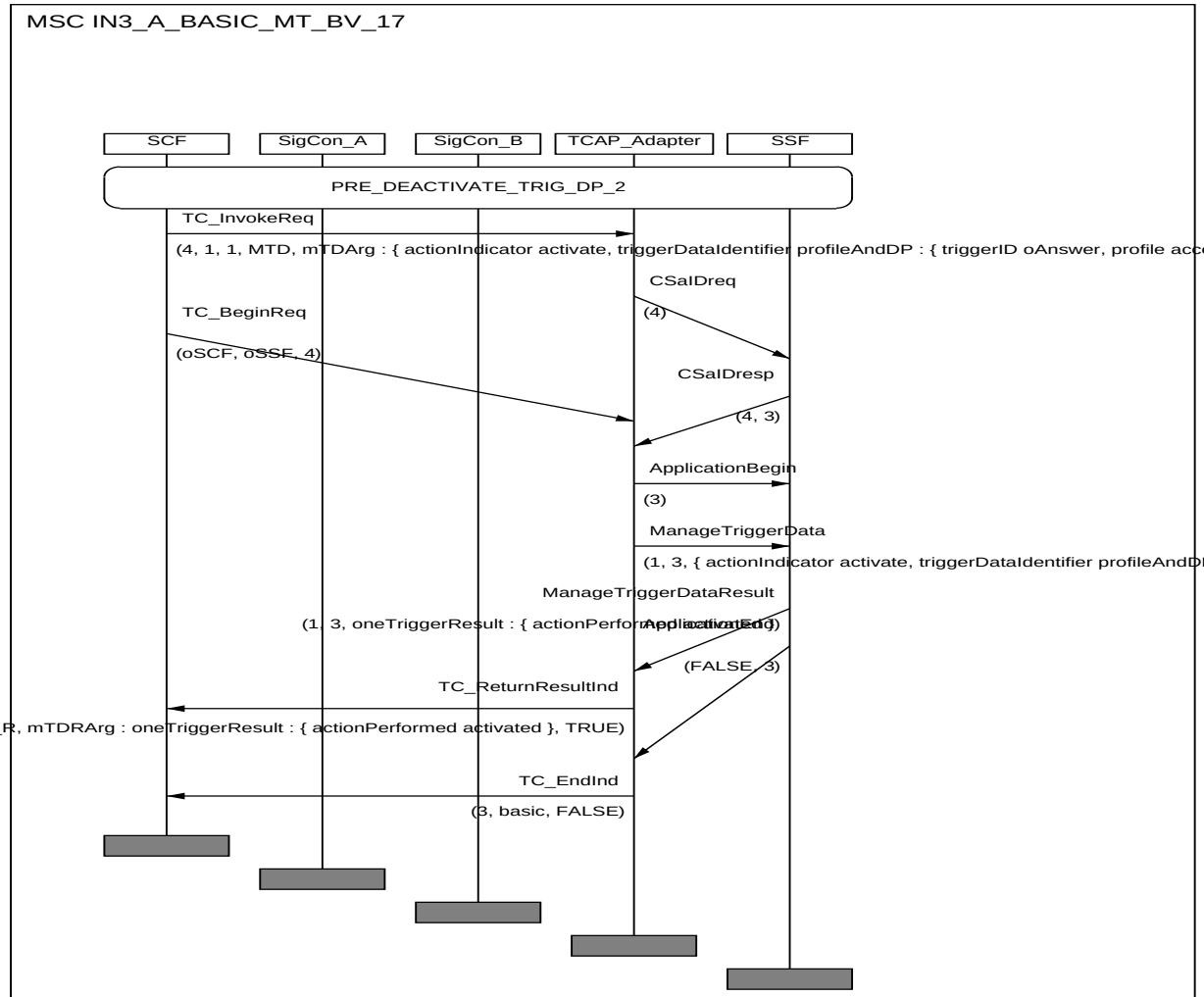
	IN3_A_BASIC_MT_BV_15
Work item no.:	ITEM_BASIC_241
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated two Triggers, both on DP **oOrigAttemptAuthorized** and for different profiles identified by **PROFILE_ID_1** and **PROFILE_ID_1A** respectively, with specific tDPIdentifiers (Integer) **tDPId1** and **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "deactivated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "deactivate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1,** and tDPIdentifier is omitted.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_PROF_2
Test description	L5!ManageTriggerData invoke(deactivate, profile(PROFILE_ID_1), tDPIdentifier **omitted**) L5!TC-BEGIN L5?TC-END L5?ManageTriggerData returnResult(deactivated)
Pass criteria	L5?ManageTriggerData returnResult(deactivated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_15
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_PROF_2
TC_InvokeReq
(4, 1, 1, MTD, mTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
CSalDreq
(4)
TC_BeginReq
(oSCF, oSSF, 4)
CSalDresp
(4, 4)
ApplicationBegin
(4)
ManageTriggerData
(1, 4, { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID orig
ManageTriggerDataResult
(1, 4, oneTriggerResult : { actionPerformed deactivated
(FALSE, 4)
TC_ReturnResultInd
4, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed deactivated }, TRUE)
TC_EndInd
(4, basic, FALSE)

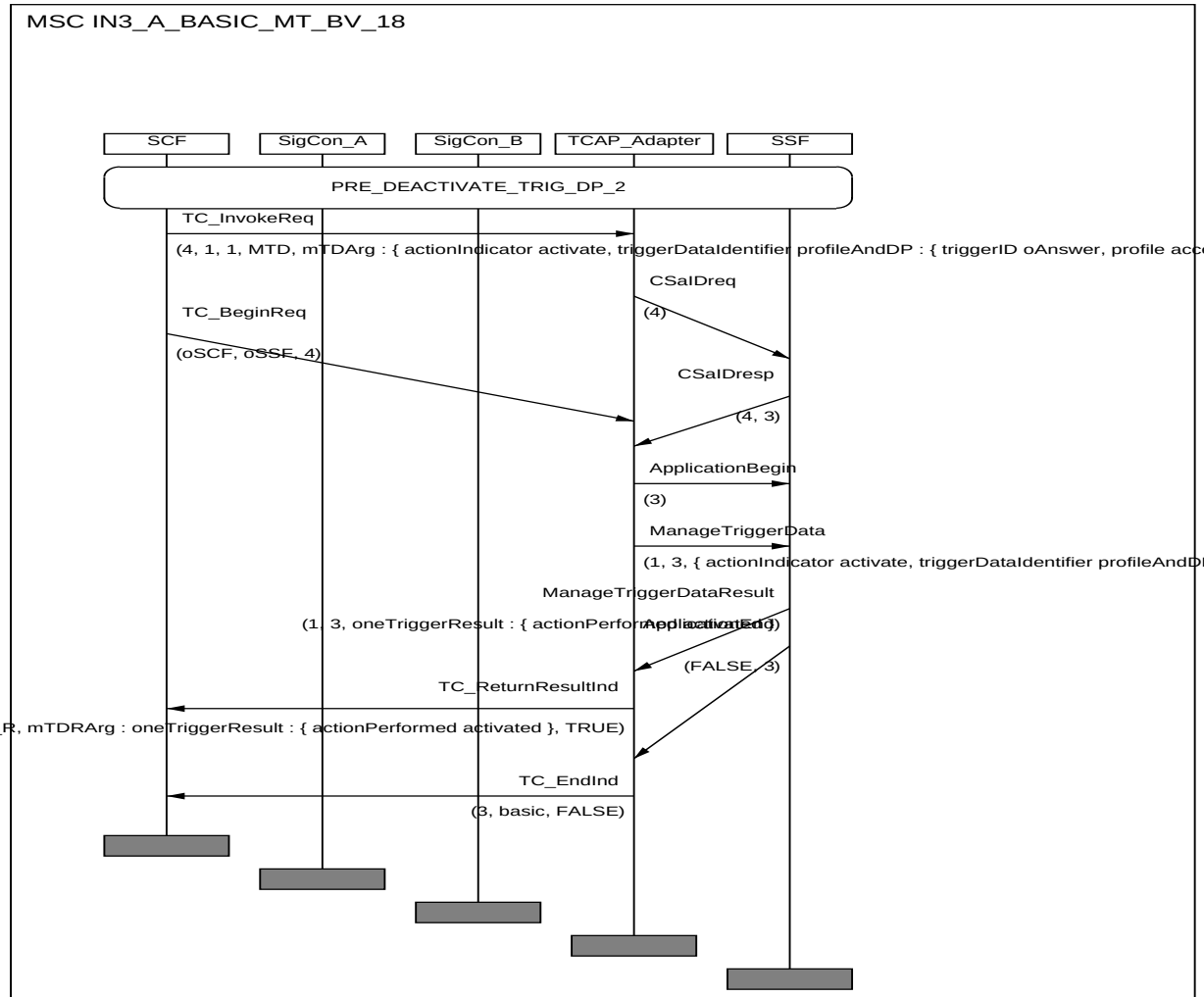
	IN3_A_BASIC_MT_BV_16
Work item no.:	ITEM_BASIC_242
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated two Triggers, one on DP **oOrigAttemptAuthorized** and one on DP **oAnswer**, both for the profile identified by **PROFILE_ID_1**, with specific tDPIdentifiers (Integer) **tDPId1** respectively **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId1** and actionPerformed = "activated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1,** and tDPIdentifier = **tDPId1**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_**DE**ACTIVATE_TRIG_DP_2
Test description	L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), oneTrigger tDPId1) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(activated)
Pass criteria	L4?ManageTriggerData returnResult(activated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_16
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG_DP_2
TC_InvokeReq
(4, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '0
CSalDreq
(4)
TC_BeginReq
(oSCF, oSSF, 4)
CSalDresp
(4, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID origAtt
ManageTriggerDataResult
(1, 3, oneTriggerResult : { actionPerformed activated })
(FALSE, 3)
TC_ReturnResultInd
3, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(3, basic, FALSE)

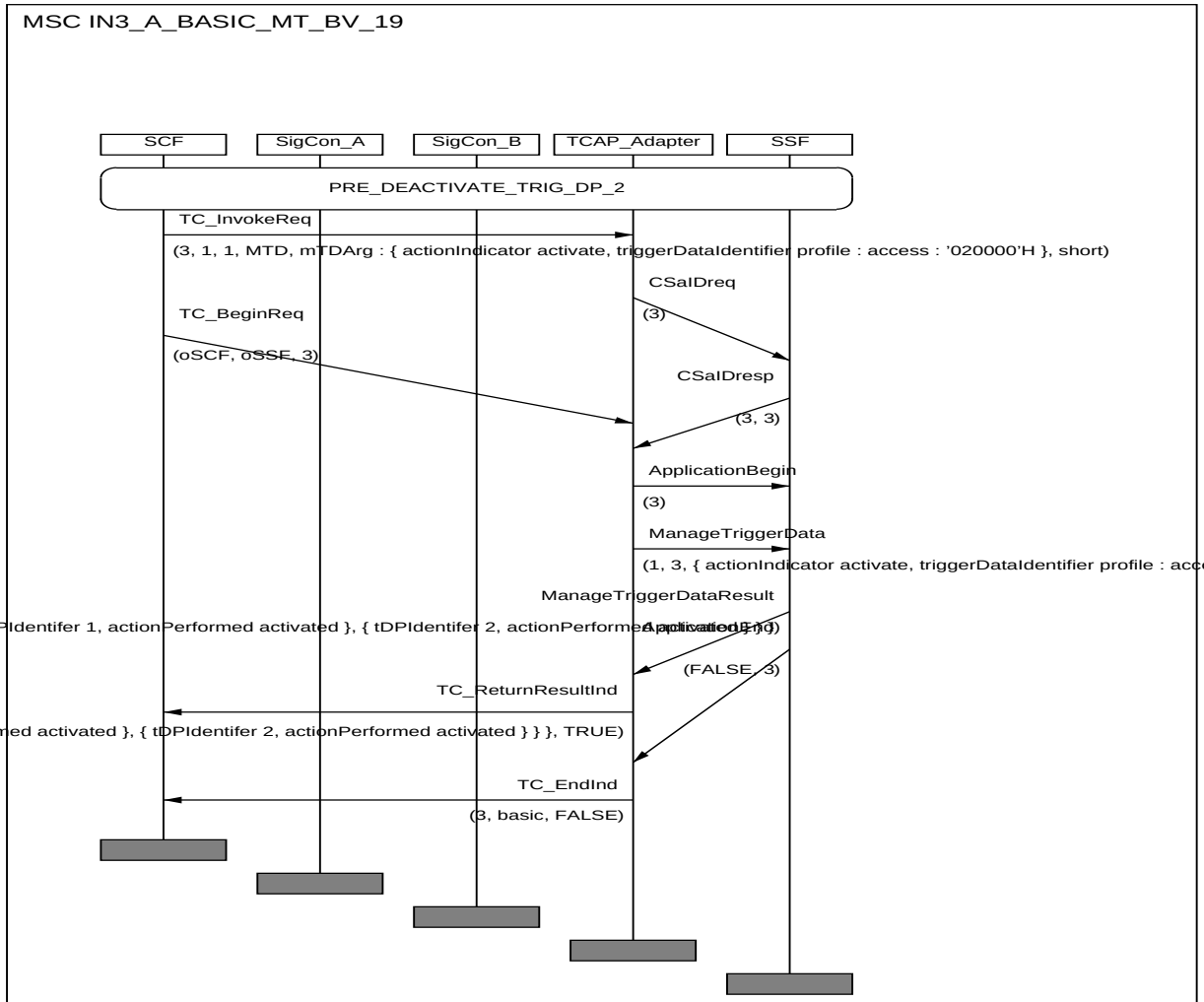
	IN3_A_BASIC_MT_BV_17
Work item no.:	ITEM_BASIC_243
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated two Triggers, one on DP **oOrigAttemptAuthorized** and one on DP **oAnswer**, both for the profile identified by **PROFILE_ID_1**, with specific tDPIdentifiers (Integer) **tDPId1** respectively **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId2** and actionPerformed = "activated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profileAndDP, identifying **oAnswer** and **PROFILE_ID_1,** and tDPIdentifier = **tDPId2**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG_DP_2
Test description	L4!ManageTriggerData invoke(activate, profileAndDP(**oAnswer**, PROFILE_ID_1), oneTrigger **tDPId2**) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(activated)
Pass criteria	L4?ManageTriggerData returnResult(activated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_17
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG_DP_2
TC_InvokeReq
(4, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oAnswer, profile access : '020000'H }, tD
CSalDreq
(4)
TC_BeginReq
(oSCF, oSSF, 4)
CSalDresp
(4, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oAnsw
ManageTriggerDataResult
(1, 3, oneTriggerResult : { actionPerformed activated }, TRUE)
(FALSE, 3)
TC_ReturnResultInd
3, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(3, basic, FALSE)

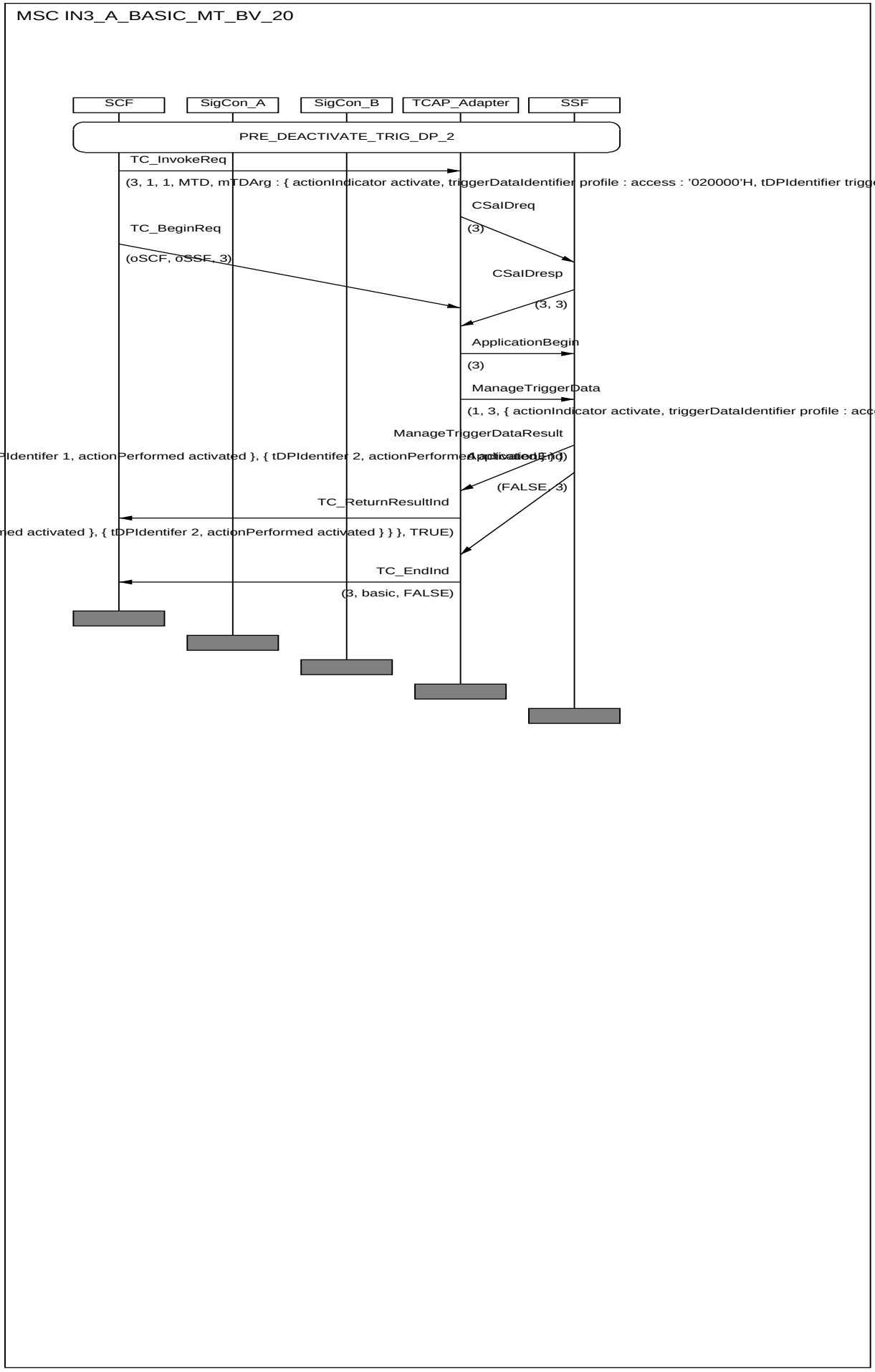
	IN3_A_BASIC_MT_BV_18
Work item no.:	ITEM_BASIC_244
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated two Triggers, one on DP **oOrigAttemptAuthorized** and one on DP **oAnswer**, both for the profile identified by **PROFILE_ID_1**, with specific tDPIdentifiers (Integer) **tDPId1** respectively **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **tDPId2** and actionPerformed = "activated", when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profileAndDP, identifying **oAnswer** and **PROFILE_ID_1,** and tDPIdentifier is omitted.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG_DP_2
Test description	L4!ManageTriggerData invoke(activate, profileAndDP(oAnswer, PROFILE_ID_1), tDPIdentifier **omitted**) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(activated)
Pass criteria	L4?ManageTriggerData returnResult(activated)
Postamble:	None


MSC IN3_A_BASIC_MT_BV_18
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG_DP_2
TC_InvokeReq
(4, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oAnswer, profile access : '020000'H } }, s
CSalDreq
(4)
TC_BeginReq
(oSCF, oSSF, 4)
CSalDresp
(4, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profileAndDP : { triggerID oAnsw
ManageTriggerDataResult
(1, 3, oneTriggerResult : { actionPerformed activated }, TRUE)
(FALSE, 3)
TC_ReturnResultInd
3, 1, MTD_R, mTDRArg : oneTriggerResult : { actionPerformed activated }, TRUE)
TC_EndInd
(3, basic, FALSE)

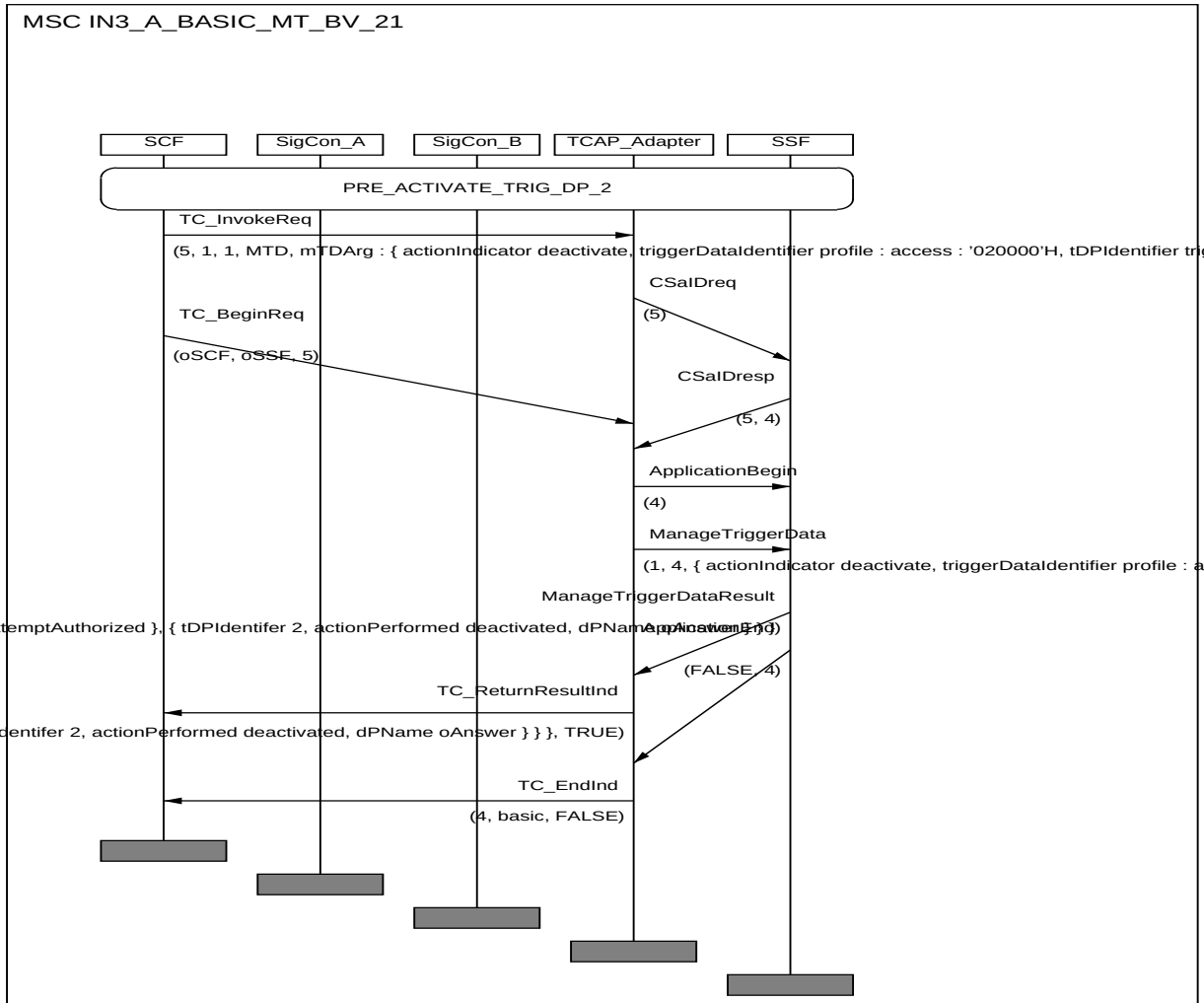
	IN3_A_BASIC_MT_BV_19
Work item no.:	ITEM_BASIC_245
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated two Triggers, one on DP **oOrigAttemptAuthorized** and one on DP **oAnswer**, both for the profile identified by **PROFILE_ID_1**, with specific tDPIdentifiers (Integer) **tDPId1** respectively **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **severalTriggerResult** containing (at least) two (SEQUENCE-OF) components, both indicating actionPerformed = "activated", and indicating **tDPId1** or **tDPId2** respectively, when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1,** and tDPIdentifier is omitted.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG_DP_2
Test description	L4!ManageTriggerData invoke(activate, **profile**(PROFILE_ID_1), tDPIdentifier omitted) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPId1), (activated, tDPId2)))
Pass criteria	L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPId1), (activated, tDPId2)))
Postamble:	None


MSC IN3_A_BASIC_MT_BV_19
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG_DP_2
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H }, short)
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H }
ManageTriggerDataResult
sult : { results { { tDPIdentifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } }
(FALSE, 3)
TC_ReturnResultInd
tifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }, TRUE)
TC_EndInd
(3, basic, FALSE)

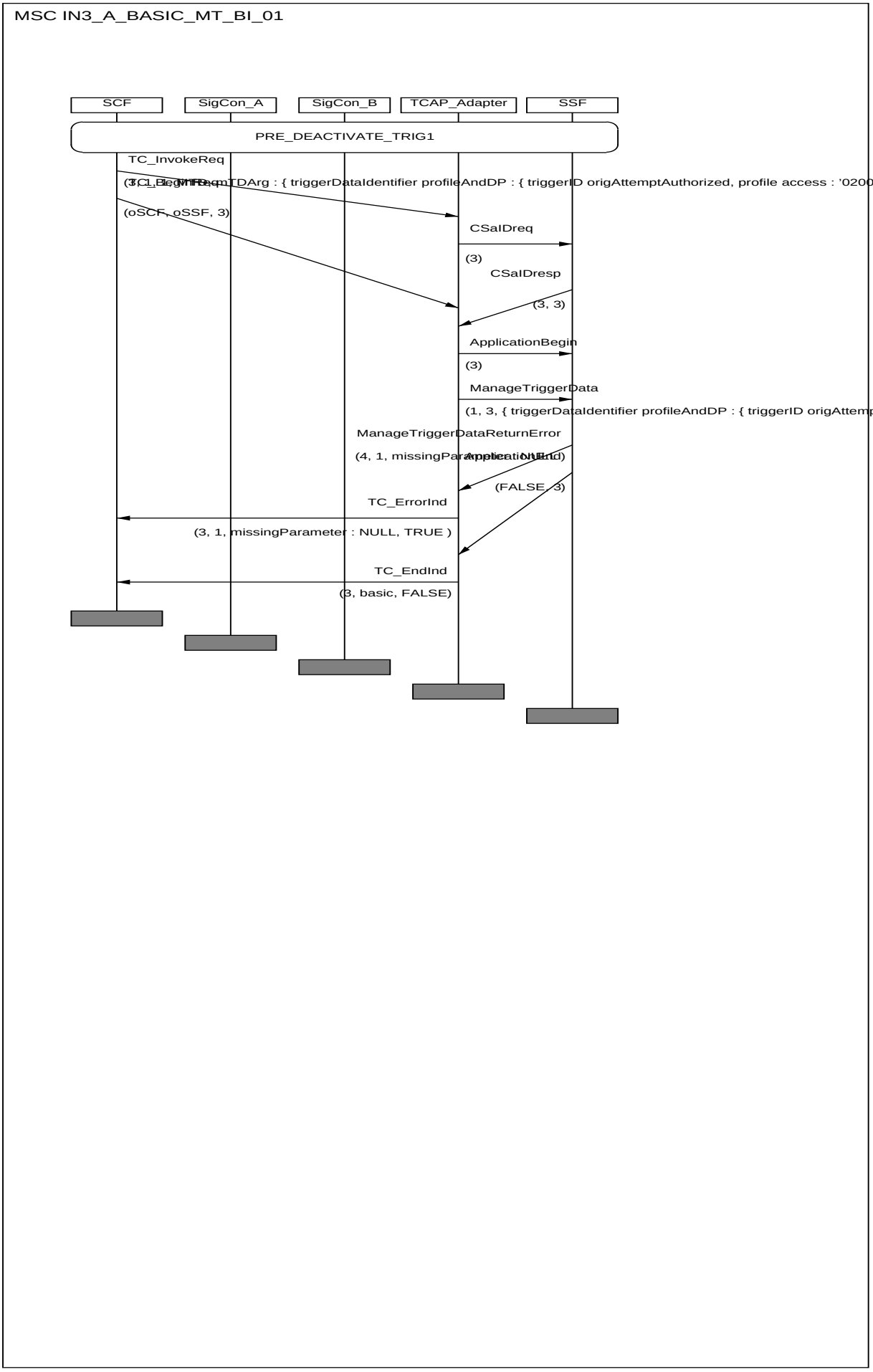
	IN3_A_BASIC_MT_BV_20
Work item no.:	ITEM_BASIC_246
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated two Triggers, one on DP **oOrigAttemptAuthorized** and one on DP **oAnswer**, both for the profile identified by **PROFILE_ID_1**, with specific tDPIdentifiers (Integer) **tDPId1** respectively **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **severalTriggerResult** containing (at least) two (SEQUENCE-OF) components, both indicating actionPerformed = "activated", and indicating **tDPId1** or **tDPId2** respectively, when having received a ManageTriggerData invoke component with parameters actionIndicator = "activate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1,** and tDPIdentifier = **triggers** with two (SEQUENCE-OF) components, where one Trigger identifies **tDPId1** and the other Trigger identifies **tDPId2**, dpName being omitted in each case.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG_DP_2
Test description	L4!ManageTriggerData invoke(activate, profile(PROFILE_ID_1), **triggers** ((tDPId1)**,**(tDPId2)) L4!TC-BEGIN L4?TC-END L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPId1), (activated, tDPId2)))
Pass criteria	L4?ManageTriggerData returnResult(severalTriggerResult ((activated, tDPId1), (activated, tDPId2)))
Postamble:	None


MSC IN3_A_BASIC_MT_BV_20
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG_DP_2
TC_InvokeReq
(3, 1, 1, MTD, mTDArg : { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDPIdentifier triggers : { { tDPIdentifier
CSalDreq
(3)
TC_BeginReq
(oSCF, oSSF, 3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, triggerDataIdentifier profile : access : '020000'H, tDP
ManageTriggerDataResult
sult : { results { { tDPIdentifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }
(FALSE, 3)
TC_ReturnResultInd
tifer 1, actionPerformed activated }, { tDPIdentifer 2, actionPerformed activated } } }, TRUE)
TC_EndInd
(3, basic, FALSE)

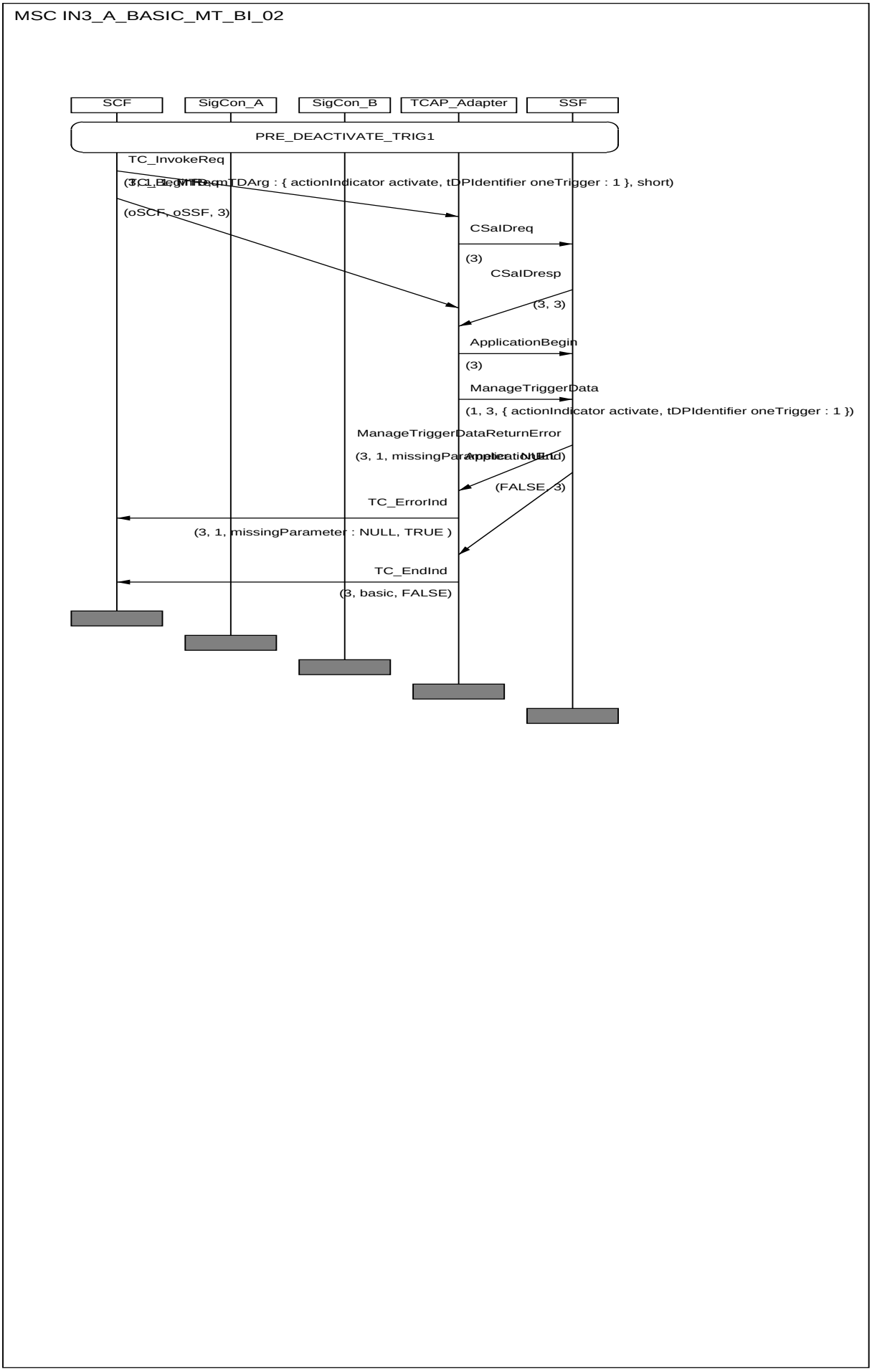
	IN3_A_BASIC_MT_BV_21
Work item no.:	ITEM_BASIC_247
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated two Triggers, one on DP **oOrigAttemptAuthorized** and one on DP **oAnswer**, both for the profile identified by **PROFILE_ID_1**, with specific tDPIdentifiers (Integer) **tDPId1** respectively **tDPId2** assigned, sends to the SCF a ManageTriggerData returnResult component indicating **severalTriggerResult** containing (at least) two (SEQUENCE-OF) components, both indicating actionPerformed = "deactivated", and indicating **tDPId1** or **tDPId2** respectively, when having received a ManageTriggerData invoke component with parameters actionIndicator = **"deactivate"**,triggerDataIdentifier = profile, identifying **PROFILE_ID_1,** and tDPIdentifier = **triggers** with two (SEQUENCE-OF) components, where one Trigger identifies **tDPId1** and the other Trigger identifies **tDPId2**, dpName being **oOrigAttemptAuthorized** and **oAnswer** respectively.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_DP_2
Test description	L5!ManageTriggerData invoke(deactivate, profile(PROFILE_ID_1), **triggers** ((OrigAttemptAuthorized, tDPId1), (oAnswer, tDPId2)) L5!TC-BEGIN L5?TC-END L5?ManageTriggerData returnResult(severalTriggerResult ((deactivated, tDPId1), (deactivated, tDPId2)))reporting also the respective dpName
Pass criteria	L5?ManageTriggerData returnResult(severalTriggerResult ((deactivated, tDPId1), (deactivated, tDPId2)))
Postamble:	None


MSC IN3_A_BASIC_MT_BV_21
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_DP_2
TC_InvokeReq
(5, 1, 1, MTD, mTDArg : { actionIndicator deactivate, triggerDataIdentifier profile : access : '020000'H, tDPIdentifier triggers : { { tDPIdentifi
CSalDreq
(5)
TC_BeginReq
(oSCF, oSSF, 5)
CSalDresp
(5, 4)
ApplicationBegin
(4)
ManageTriggerData
(1, 4, { actionIndicator deactivate, triggerDataIdentifier profile : access : '020000'H, tD
ManageTriggerDataResult
ted, dPName origAttemptAuthorized }, { tDPIdentifer 2, actionPerformed deactivated, dPName
(FALSE, 4)
TC_ReturnResultInd
tAuthorized }, { tDPIdentifer 2, actionPerformed deactivated, dPName oAnswer } } }, TRUE)
TC_EndInd
(4, basic, FALSE)

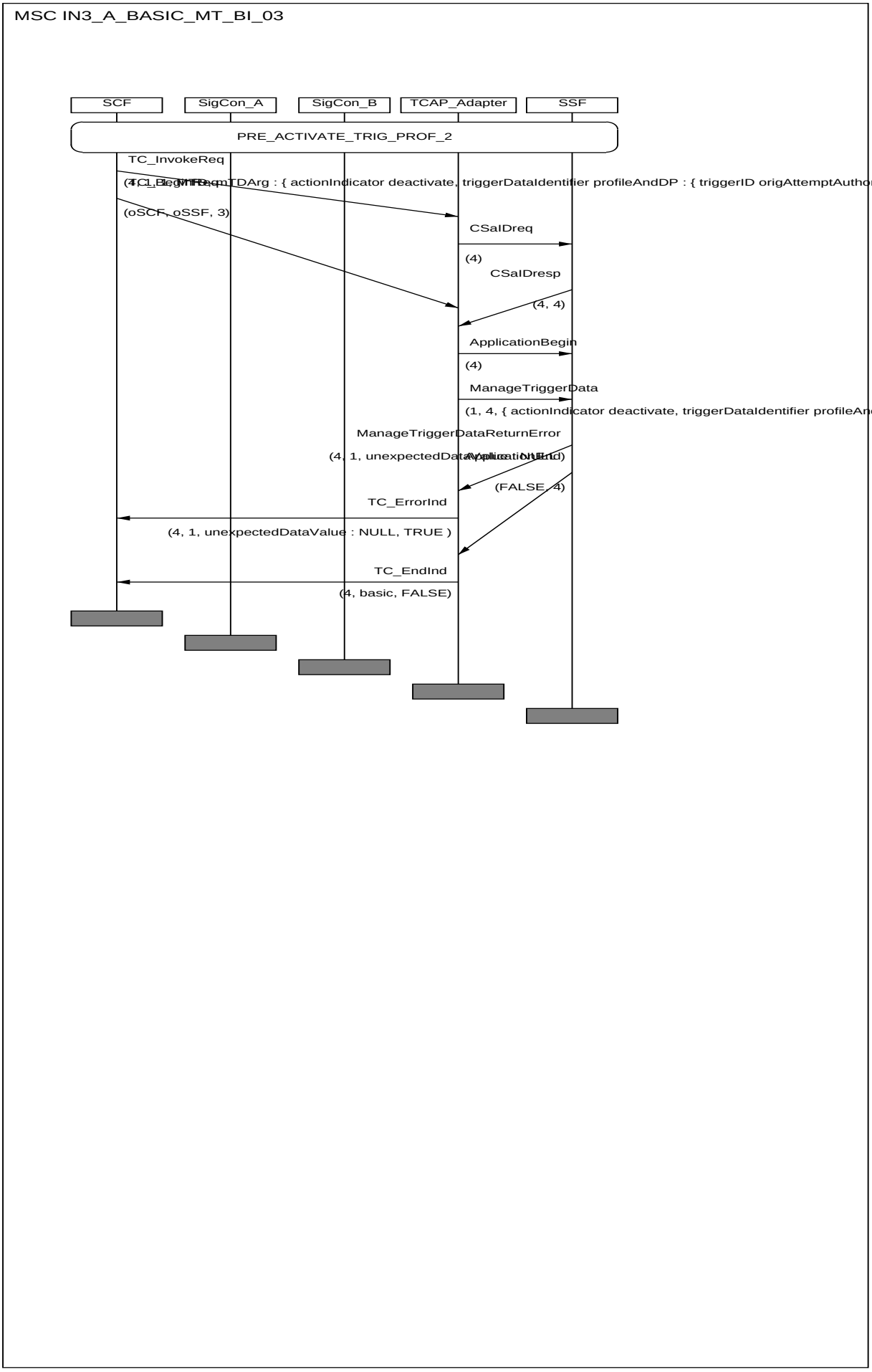
	IN3_A_BASIC_MT_BI_01
Work item no.:	ITEM_BASIC_248
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnError component with error value "missingParameter", when having received a ManageTriggerData invoke component where mandatory parameter actionIndicator is omitted.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG1
Test description	L3!ManageTriggerData invoke(actionIndicator **omitted**, profile (PROFILE_ID_1), oneTrigger tDPId1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnError(missingParameter)
Pass criteria	L3?ManageTriggerData returnError(missingParameter)
Postamble:	None


MSC IN3_A_BASIC_MT_BI_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG1
TC_InvokeReq
(3, 1, ManageTDArg : { triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access : '020000'H }, tDPIdentifier
(oSCF, oSSF, 3)
CSalDreq
(3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile a
ManageTriggerDataReturnError
(4, 1, missingParameter : NULL)
(FALSE, 3)
TC_ErrorInd
(3, 1, missingParameter : NULL, TRUE)
TC_EndInd
(3, basic, FALSE)

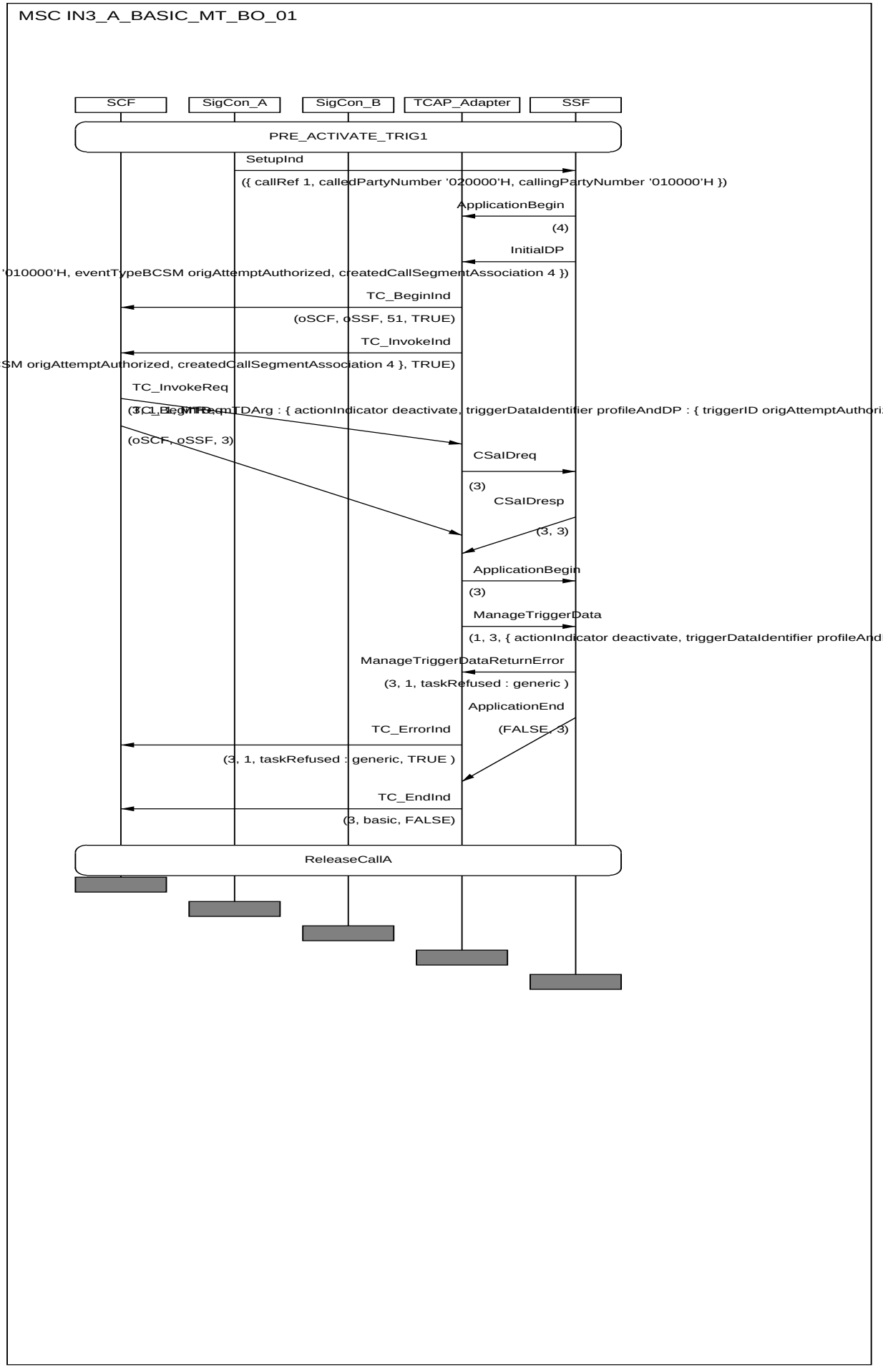
	IN3_A_BASIC_MT_BI_02
Work item no.:	ITEM_BASIC_249
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and deactivated a single Trigger, on DP **oOrigAttemptAuthorized** and for a profile identified by **PROFILE_ID_1**, with specific tDPIdentifier (Integer) = **tDPId1** assigned, sends to the SCF a ManageTriggerData returnError component with error value "missingParameter", when having received a ManageTriggerData invoke component where mandatory parameter triggerDataIdentifier is omitted.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_DEACTIVATE_TRIG1
Test description	L3!ManageTriggerData invoke(activate, triggerDataIdentifier **omitted**, oneTrigger tDPId1) L3!TC-BEGIN L3?TC-END L3?ManageTriggerData returnError(missingParameter)
Pass criteria	L3?ManageTriggerData returnError(missingParameter)
Postamble:	None


MSC IN3_A_BASIC_MT_BI_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_DEACTIVATE_TRIG1
TC_InvokeReq
(3, 1, ManageTriggerDataArg : { actionIndicator activate, tDPIdentifier oneTrigger : 1 }, short)
(oSCF, oSSF, 3)
CSaIDreq
(3)
CSaIDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator activate, tDPIdentifier oneTrigger : 1 })
ManageTriggerDataReturnError
(3, 1, missingParameter : NULL, TRUE)
(FALSE, 3)
TC_ErrorInd
(3, 1, missingParameter : NULL, TRUE)
TC_EndInd
(3, basic, FALSE)

	IN3_A_BASIC_MT_BI_03
Work item no.:	ITEM_BASIC_250
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated two Triggers, both on DP **oOrigAttemptAuthorized** and for different profiles identified by **PROFILE_ID_1** and **PROFILE_ID_1A** respectively, with specific tDPIdentifiers (Integer) = **tDPId1** and **tDPId2** assigned, sends to the SCF a ManageTriggerData returnError component with error value "unexpectedDataValue" or "unexpectedParameter", when having received a ManageTriggerData invoke component with parameters actionIndicator = "deactivate", triggerDataIdentifier = profile, identifying **PROFILE_ID_1,** and tDPIdentifier = **tDPId2**.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.28, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_PROF_2
Test description	L5!ManageTriggerData invoke(deactivate, profile(PROFILE_ID_1), oneTrigger tDPId2) L5!TC-BEGIN L5?TC-END L5?ManageTriggerData returnError("unexpectedDataValue" or "unexpectedParameter")
Pass criteria	L5?ManageTriggerData returnError("unexpectedDataValue" or "unexpectedParameter")
Postamble:	None


MSC IN3_A_BASIC_MT_BI_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_PROF_2
TC_InvokeReq
(4, 1, ManageTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
(oSCF, oSSF, 3)
CSalDreq
(4)
CSalDresp
(4, 4)
ApplicationBegin
(4)
ManageTriggerData
(1, 4, { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID orig
ManageTriggerDataReturnError
(4, 1, unexpectedDataValue : NULL)
(FALSE, 4)
TC_ErrorInd
(4, 1, unexpectedDataValue : NULL, TRUE)
TC_EndInd
(4, basic, FALSE)

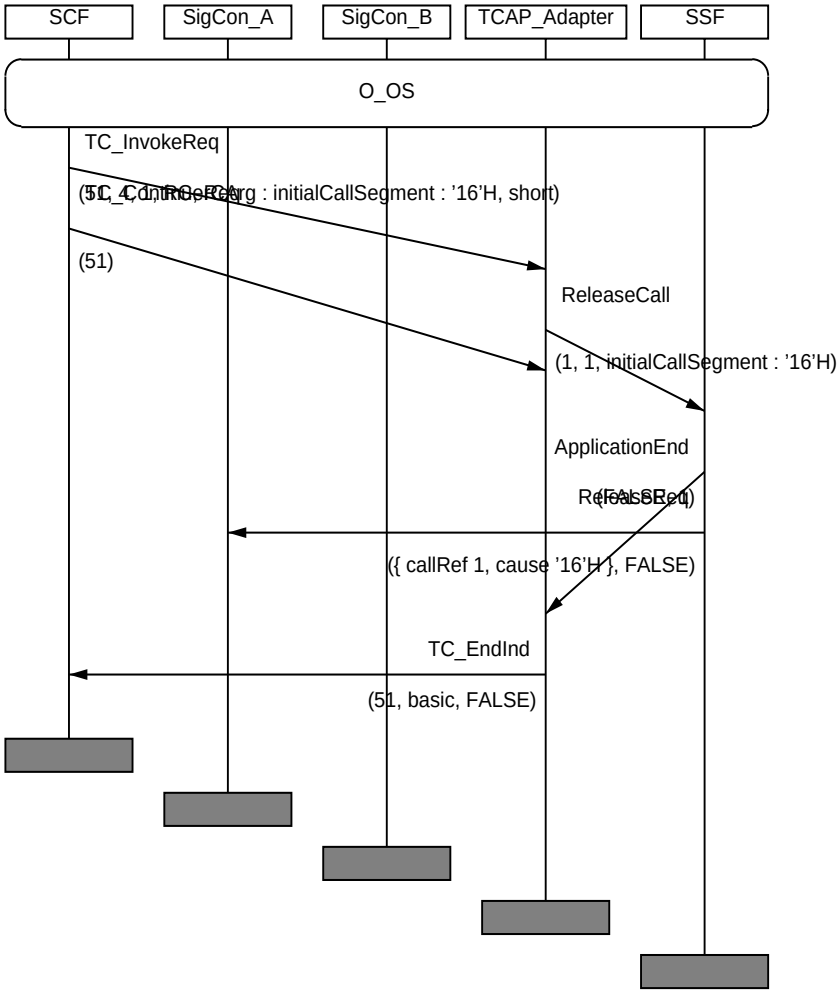
	IN3_A_BASIC_MT_BO_01
Work item no.:	ITEM_BASIC_251
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received a ManageTriggerData invoke component in the context of a call, sends a CreateOrRemoveTriggerData returnError component with error value "taskRefused" or "unexpectedComponentSequence".
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.17, 15.1.1.1.2.2
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG1
Test description	CP1_1!SetupInd(CALL-DATA-1) L1?InitialDP L1!ManageTriggerData invoke(deactivate, profile(PROFILE_ID_1), oneTrigger tDPId1) L1?ManageTriggerData returnError("taskRefused" or "unexpectedComponentSequence")
Pass criteria	L1?ManageTriggerData returnError("taskRefused" or "unexpectedComponentSequence")
Postamble:	TrigReleaseA


MSC IN3_A_BASIC_MT_BO_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG1
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(3, 1, ManageTDArg : { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID origAttemptAuthorized, profile access
(oSCF, oSSF, 3)
CSalDreq
(3)
CSalDresp
(3, 3)
ApplicationBegin
(3)
ManageTriggerData
(1, 3, { actionIndicator deactivate, triggerDataIdentifier profileAndDP : { triggerID orig
ManageTriggerDataReturnError
(3, 1, taskRefused : generic)
ApplicationEnd
TC_ErrorInd
(FALSE, 3)
(3, 1, taskRefused : generic, TRUE)
TC_EndInd
(3, basic, FALSE)
ReleaseCallA

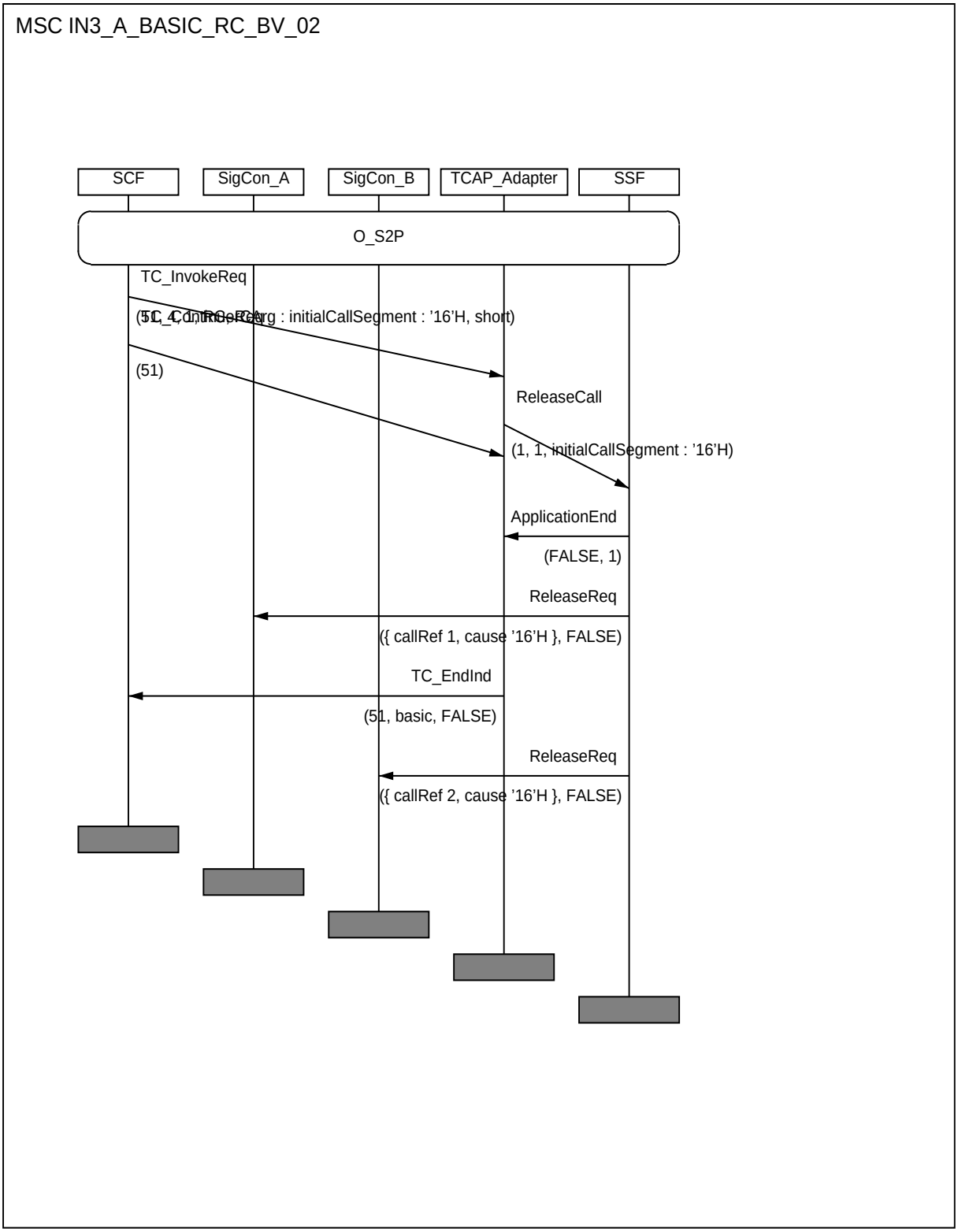
6.6.16 ReleaseCall (RC) procedure

	IN3_A_BASIC_RC_BV_01
Work item no.:	ITEM_BASIC_75
IN2 Ref(tmp)	IN2_A_BASIC_RC_CA_01
Purpose:	Verify that the SSF releases the call to SigConA after receiving a **ReleaseCall** invoke component from the SCF indicating initialCallSegment (cause), when only one party (SigConA) is involved in the connection.
Requirements refs	6.5.1.2.3/3.3, 8.2, 8.2.1, 8.2.1.2, 8.2.2, 8.2.2.2, 8.2.2.5, 8.3.2, 8.4.2, 11.32.1, 11.32.1.1.1, 11.32.3.1
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

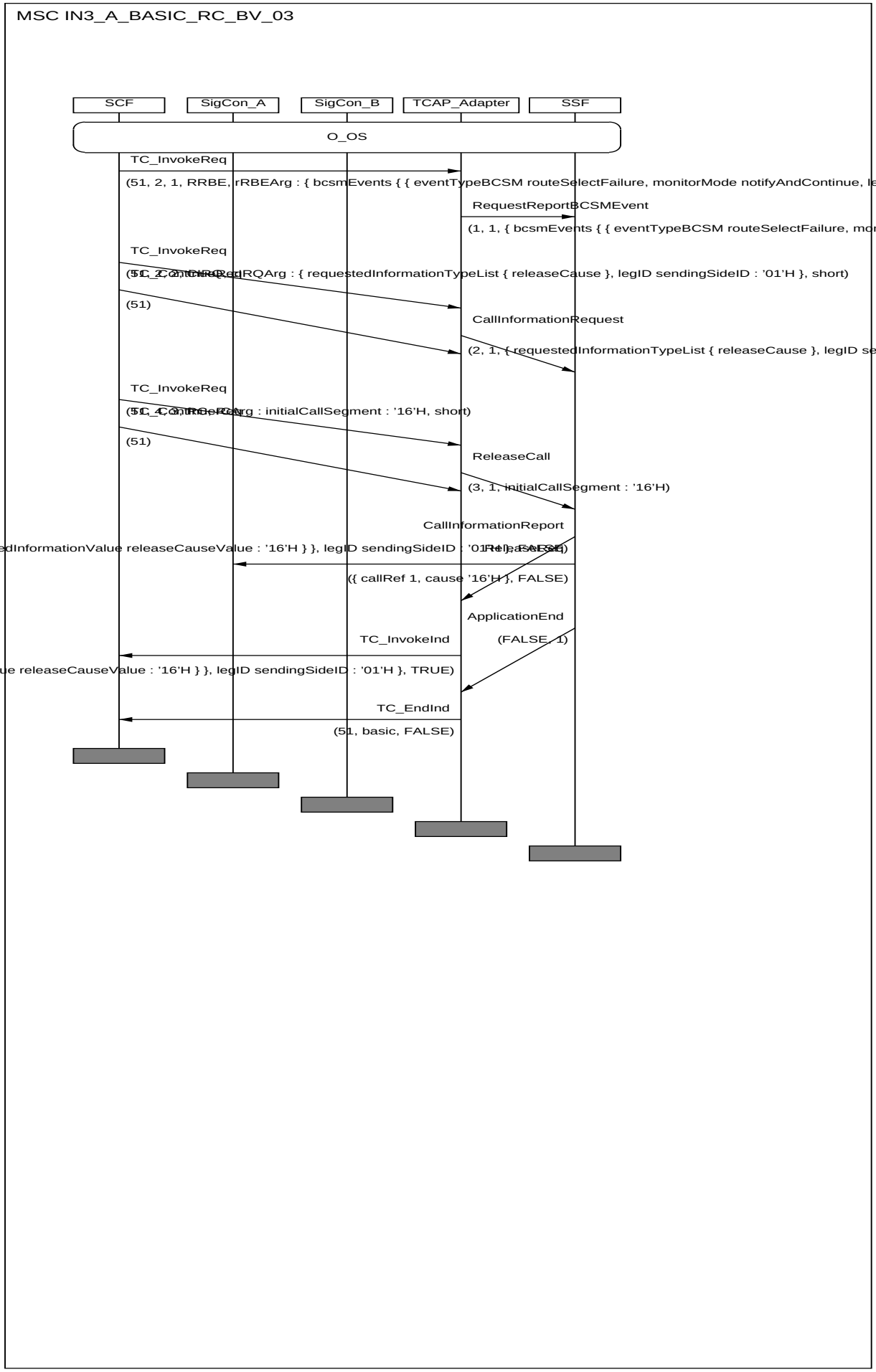
MSC IN3_A_BASIC_RC_BV_01



	IN3_A_BASIC_RC_BV_02
Work item no.:	ITEM_BASIC_76
IN2 Ref(tmp)	IN2_A_BASIC_RC_BV_01
Purpose:	Verify that the SSF releases the call to **SigConA** and to **SigConB** after receiving a **ReleaseCall** invoke component from the SCF indicating initialCallSegment (cause), when only two parties (SigConA and SigConB) are involved in the connection.
Requirements refs	6.5.1.2.3/3.3, 8.2, 8.2.1, 8.2.1.2, 8.2.2, 8.2.2.2, 8.2.2.5, 8.3.2, 8.4.2, 11.32.1, 11.32.1.1.1, 11.32.3.1
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

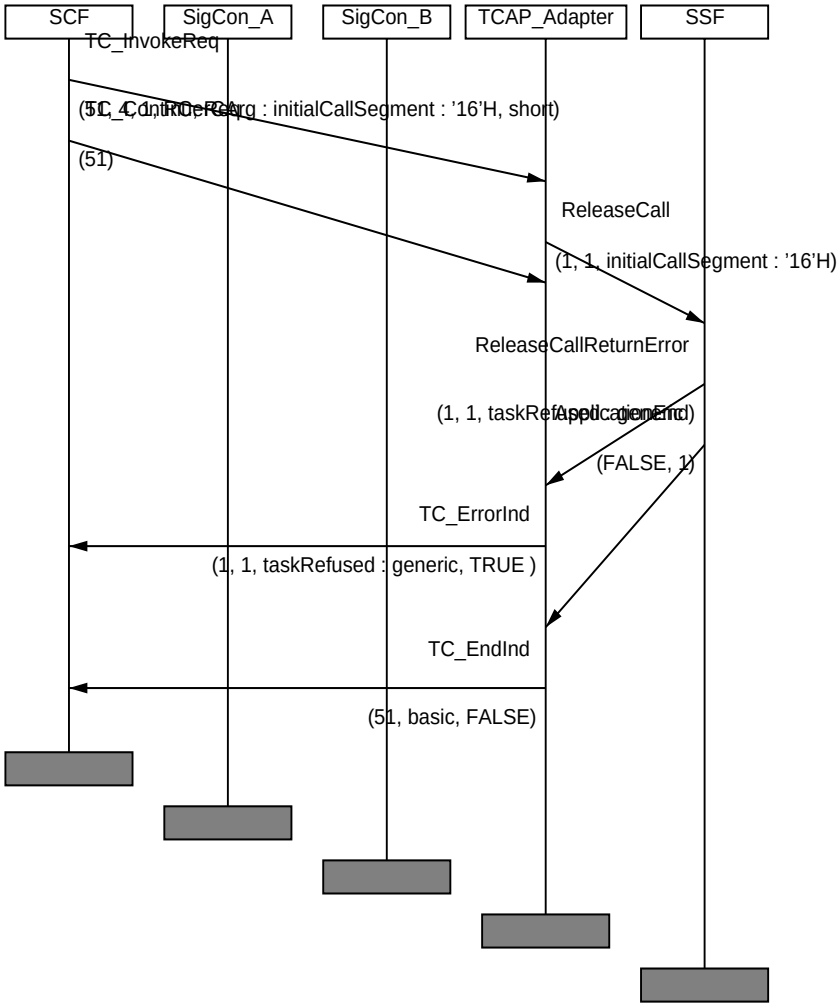

MSC IN3_A_BASIC_RC_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_S2P
TC_InvokeReq
(51, 1, initialCallSegment : '16'H, short)
(51)
ReleaseCall
(1, 1, initialCallSegment : '16'H)
ApplicationEnd
(FALSE, 1)
ReleaseReq
({ callRef 1, cause '16'H }, FALSE)
TC_EndInd
(51, basic, FALSE)
ReleaseReq
({ callRef 2, cause '16'H }, FALSE)

	IN3_A_BASIC_RC_BV_03
Work item no.:	ITEM_BASIC_77
IN2 Ref(tmp)	IN2_A_BASIC_RC_BV_04
Purpose:	Verify that the SSF after having received an **ReleaseCall** invoke with the **initialCallSegment** (cause) parameter, sends the outstanding **CallInformationReport** if the requestedInformationType (releaseCause) was included in the request for this report, does **not** send an **EventReportBCSM** if the Route_Select_Failure DP has been armed by the SCF, and releases the call.
Requirements refs	6.5.1.2.3/3.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1, 8.2.1.2, 8.2.2, 8.2.2.1, 8.2.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.7.1, 11.7.1.1, 11.7.2.1, 11.8.1, 11.8.1.1.1, 11.8.3.1, 11.32.1, 11.32.1.1.1, 11.32.3.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
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 IN3_A_BASIC_RC_BV_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode notifyAndContinue, legID sendingSideID :
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode notifyAndC
TC_InvokeReq
(51, 2, CIRReq, cIRReqRQArg : { requestedInformationTypeList { releaseCause }, legID sendingSideID : '01'H }, short)
(51)
CallInformationRequest
(2, 1, { requestedInformationTypeList { releaseCause }, legID sendingSideID : '01'H }
TC_InvokeReq
(51, 4, CIRes, cIResArg : initialCallSegment : '16'H, short)
(51)
ReleaseCall
(3, 1, initialCallSegment : '16'H)
CallInformationReport
aseCause, requestedInformationValue releaseCauseValue : '16'H } }, legID sendingSideID : '01'H }, FALSE)
({ callRef 1, cause '16'H }, FALSE)
ApplicationEnd
TC_InvokeInd
(FALSE, 1)
estedInformationValue releaseCauseValue : '16'H } }, legID sendingSideID : '01'H }, TRUE)
TC_EndInd
(51, basic, FALSE)

	IN3_A_BASIC_RC_BO_01
Work item no.:	ITEM_BASIC_78
IN2 Ref(tmp)	IN2_A_BASIC_RC_BO_01
Purpose:	Verify that the SSF rejects a **ReleaseCall** invoke component received from the SCF in the **Idle** state.
Requirements refs	6.5.1.2.3/3.3, 8.2, 8.2.1, 8.2.1.2, 8.2.2, 8.2.2.2, 8.2.2.5, 8.3.2, 8.4.2, 11.32.1, 11.32.1.1.1, 11.32.3.1, 11.32.3.2, 15.1.1.2.2
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

MSC IN3_A_BASIC_RC_BO_01



6.6.17 ReportUTSI (RP) procedure

TPs for the ReportUTSI (RP) procedure are contained in EN 301 933-3 [6].

6.6.18 RequestCurrentStatusReport (RT) procedure

	IN3_A_BASIC_RT_BV_01
Work item no.:	ITEM_BASIC_128
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestCurrentStatusReport** invoke component **in the context of a call**, related to a resource being in the busy status, sends to the SCF a **RequestCurrentStatusReport** returnResult component indicating resourceStatus "busy" for the requested resource.
Requirements refs	11.10, 11.34, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestCurrentStatusReport invoke(STAT_RESOURCE_2) L1?RequestCurrentStatusReport returnResult(busy,STAT_RESOURCE_2)
Pass criteria	L1?RequestCurrentStatusReport returnResult(busy,STAT_RESOURCE_2)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RT_BV_02
Work item no.:	ITEM_BASIC_129
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestCurrentStatusReport** invoke component **in the context of a call**, related to a resource being in the idle status, sends to the SCF a **RequestCurrentStatusReport** returnResult component indicating resourceStatus "idle" for the requested resource.
Requirements refs	11.10, 11.34, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L1!RequestCurrentStatusReport invoke(STAT_RESOURCE_2) L1?RequestCurrentStatusReport returnResult(idle,STAT_RESOURCE_2)
Pass criteria	L1?RequestCurrentStatusReport returnResult(idle,STAT_RESOURCE_2)
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RT_BV_03
Work item no.:	ITEM_BASIC_130
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestCurrentStatusReport** invoke component **outside the context of a call**, related to a resource being in the busy status, sends to the SCF a TC-END message containing a **RequestCurrentStatusReport** returnResult component indicating resourceStatus "busy" for the requested resource.
Requirements refs	11.10, 11.34, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestCurrentStatusReport invoke(STAT_RESOURCE_2) L2!TC-BEGIN L2?TC-END L2?RequestCurrentStatusReport returnResult(busy,STAT_RESOURCE_2)
Pass criteria	L2?TC-END L2?RequestCurrentStatusReport returnResult(busy,STAT_RESOURCE_2)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RT_BV_04
Work item no.:	ITEM_BASIC_131
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestCurrentStatusReport** invoke component **outside the context of a call**, related to a resource being in the idle status, sends to the SCF a TC-END message containing a **RequestCurrentStatusReport** returnResult component indicating resourceStatus "idle" for the requested resource.
Requirements refs	11.10, 11.34, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L2!RequestCurrentStatusReport invoke(STAT_RESOURCE_2) L2!TC-BEGIN L2?TC-END L2?RequestCurrentStatusReport returnResult(idle,STAT_RESOURCE_2)
Pass criteria	L2?TC-END L2?RequestCurrentStatusReport returnResult(idle,STAT_RESOURCE_2)
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RT_BV_05
Work item no.:	ITEM_BASIC_132
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestCurrentStatusReport** invoke component **in the context of a call**, related to a resource being in the busy status, followed by another **RequestCurrentStatusReport** invoke component related to another resource being in the busy status, and having been received before the first operation has been answered, sends to the SCF two **RequestCurrentStatusReport** returnResult components indicating resourceStatus "busy", one for each of the requested resources.
Requirements refs	11.10, 11.34, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestCurrentStatusReport invoke(STAT_RESOURCE_1) L1!RequestCurrentStatusReport invoke(STAT_RESOURCE_2) L1?RequestCurrentStatusReport returnResult(busy,STAT_RESOURCE_1) L1?RequestCurrentStatusReport returnResult(busy,STAT_RESOURCE_2)
Pass criteria	L1?RequestCurrentStatusReport returnResult(busy,STAT_RESOURCE_1) L1?RequestCurrentStatusReport returnResult(busy,STAT_RESOURCE_2)
Postamble:	ReleaseABandIgnoreStat

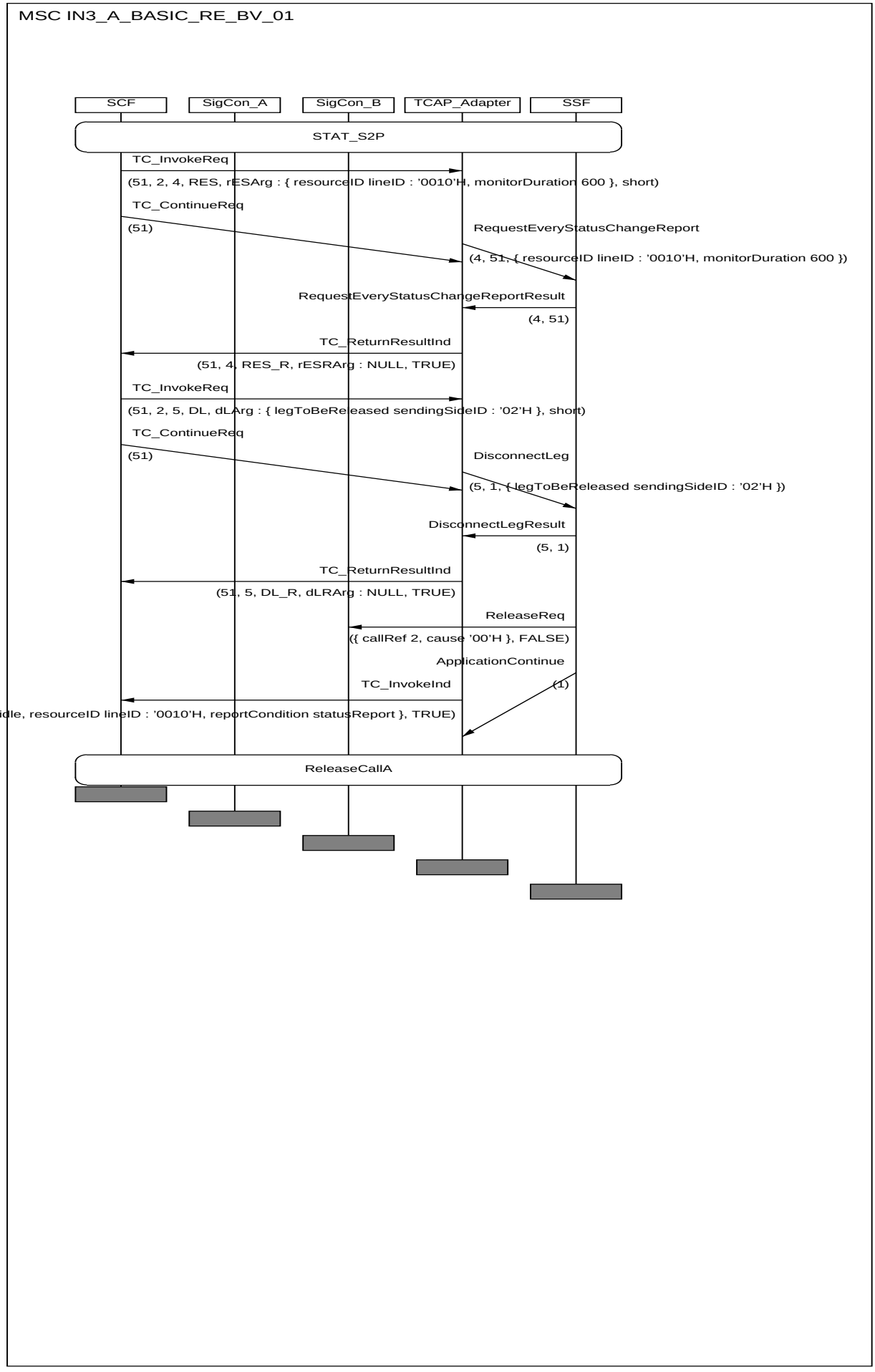
	IN3_A_BASIC_RT_BI_01
Work item no.:	ITEM_BASIC_133
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestCurrentStatusReport** invoke component **in the context of a call**, where mandatory parameter resourceID is missing, sends to the SCF a **RequestCurrentStatusReport** returnError component with error code "missingParameter".
Requirements refs	11.10, 11.34, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestCurrentStatusReport invoke(resourceID omitted) L1?RequestCurrentStatusReport returnError(missingParameter)
Pass criteria	L1?RequestCurrentStatusReport returnError(missingParameter)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RT_BI_02
Work item no.:	ITEM_BASIC_134
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestCurrentStatusReport** invoke component **outside the context of a call**, where parameter resourceID does not identify a resource being known to the SSF, sends to the SCF a TC-END message containing a **RequestCurrentStatusReport** returnError component with error code "unknownResource".
Requirements refs	11.10, 11.34, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestCurrentStatusReport invoke(STAT_RESOURCE_U) L2!TC-BEGIN L2?TC-END L2?RequestCurrentStatusReport returnError(missingParameter)
Pass criteria	L2?TC-END L2?RequestCurrentStatusReport returnError(missingParameter)
Postamble:	ReleaseABandIgnoreStat

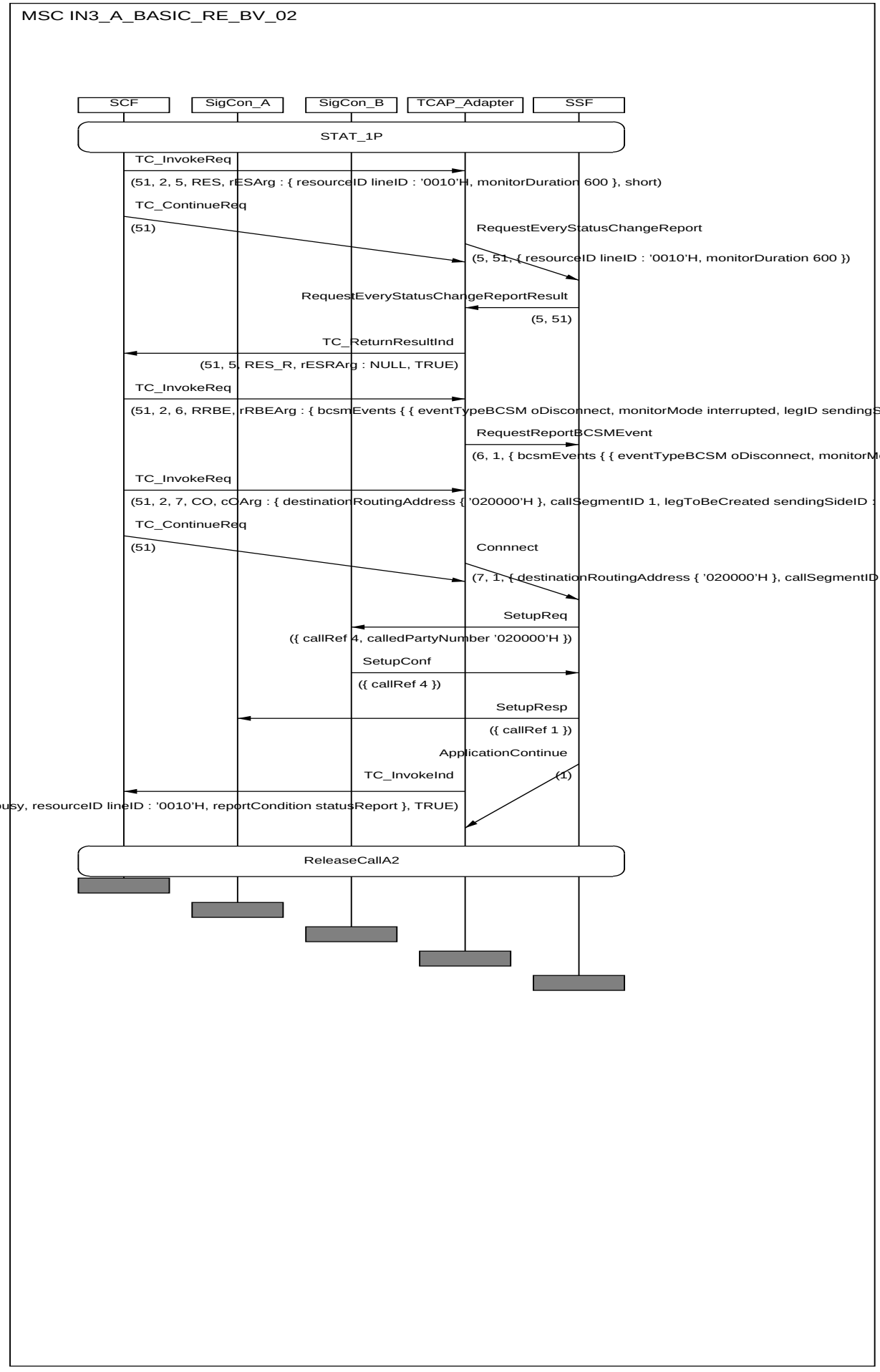
	IN3_A_BASIC_RT_BO_01
Work item no.:	ITEM_BASIC_158
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestCurrentStatusReport** invoke component **outside the context of a call**, related to a resource being in the busy status, followed (in the same context) by another **RequestCurrentStatusReport** invoke component related to another resource being in any status, sends to the SCF a **RequestCurrentStatusReport** returnResult component indicating resourceStatus "busy" for the first invoked operation, and sends a **RequestCurrentStatusReport** returnError component indicating error code "taskRefused" or "unexpectedComponentSequence" for the second invoked operation.
Requirements refs	11.10, 11.34, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	RCSR_Interruptable
Preamble:	STAT_S2P
Test description	L2!RequestCurrentStatusReport invoke(invID1,STAT_RESOURCE_1) L2!TC-BEGIN L2!RequestCurrentStatusReport invoke(invID2,STAT_RESOURCE_2) L2!TC-CONTINUE L2?TC-CONTINUE L2?RequestCurrentStatusReport returnError(invID2,"missingParameter" or "unexpectedComponentSequence") L2?TC-END L2?RequestCurrentStatusReport returnResult(invID1,busy,STAT_RESOURCE_1)
Pass criteria	L2?TC-CONTINUE L2?RequestCurrentStatusReport returnError(invID2,"missingParameter" or "unexpectedComponentSequence") L2?TC-END L2?RequestCurrentStatusReport returnResult(invID1,busy,STAT_RESOURCE_1)
Postamble:	ReleaseABandIgnoreStat

6.6.19 RequestEveryStatusChangeReport (RE) procedure

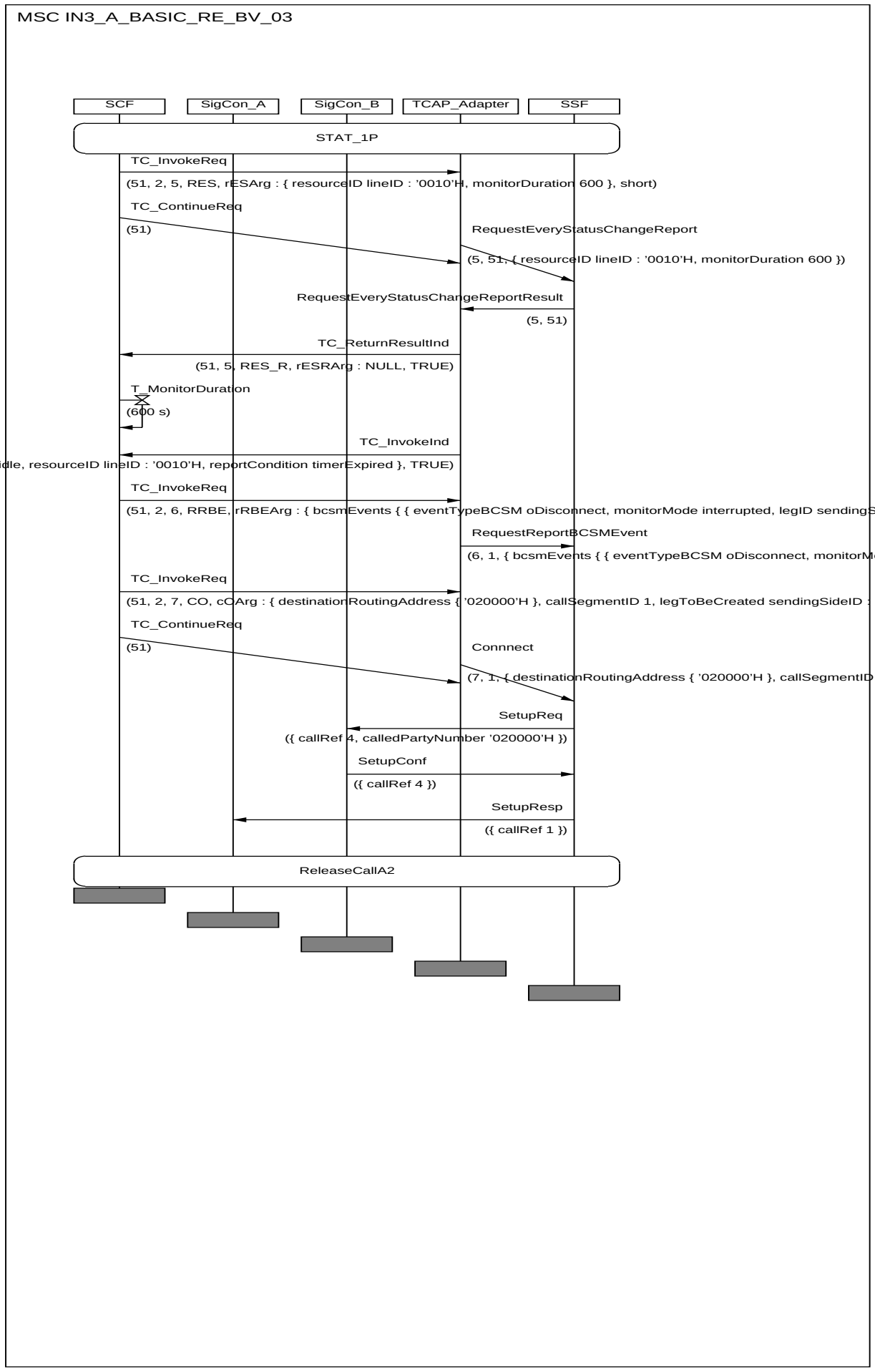
	IN3_A_BASIC_RE_BV_01
Work item no.:	ITEM_BASIC_135
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, related to a resource being in the busy status, sends to the SCF a **RequestEveryStatusChangeReport** returnResult component. Verify also that the SSF sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "statusReport", when the related resource changes to the idle status before the monitorDuration (if present in the **RequestEveryStatusChangeReport** invoke component) expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR_ANY) L1?RequestEveryStatusChangeReport returnResult L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq and StatusReport may appear in any order.
Pass criteria	L1?RequestEveryStatusChangeReport returnResult L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseAandIgnoreStat


MSC IN3_A_BASIC_RE_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
STAT_S2P
TC_InvokeReq
(51, 2, 4, RES, rESArg : { resourceID lineID : '0010'H, monitorDuration 600 }, short)
TC_ContinueReq
(51)
RequestEveryStatusChangeReport
(4, 51, { resourceID lineID : '0010'H, monitorDuration 600 })
RequestEveryStatusChangeReportResult
(4, 51)
TC_ReturnResultInd
(51, 4, RES_R, rESRArg : NULL, TRUE)
TC_InvokeReq
(51, 2, 5, DL, dLArg : { legToBeReleased sendingSideID : '02'H }, short)
TC_ContinueReq
(51)
DisconnectLeg
(5, 1, { legToBeReleased sendingSideID : '02'H })
DisconnectLegResult
(5, 1)
TC_ReturnResultInd
(51, 5, DL_R, dLRArg : NULL, TRUE)
ReleaseReq
({ callRef 2, cause '00'H }, FALSE)
ApplicationContinue
(1)
TC_InvokeInd
: { resourceStatus idle, resourceID lineID : '0010'H, reportCondition statusReport }, TRUE)
ReleaseCallA

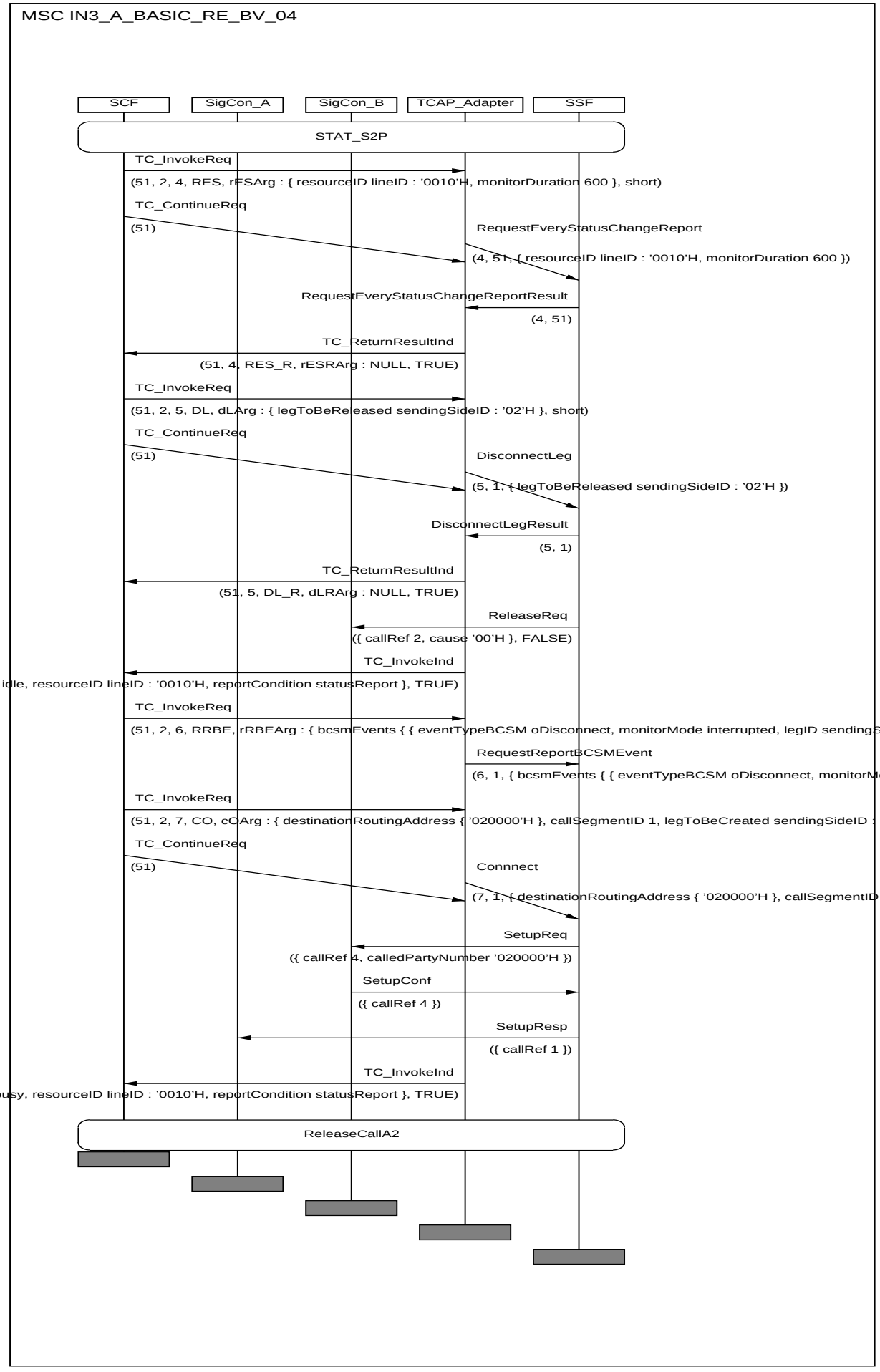
	IN3_A_BASIC_RE_BV_02
Work item no.:	ITEM_BASIC_136
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, related to a resource being in the idle status, sends to the SCF a **RequestEveryStatusChangeReport** returnResult component. Verify also that the SSF sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "statusReport", when the related resource changes to the busy status before the monitorDuration (if present in the **RequestEveryStatusChangeReport** invoke component) expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR_ANY) L1?RequestEveryStatusChangeReport returnResult L1!RequestReportBCSMEvent(3,interrupted,oDisconnect) L1!Connect(3,1,resource2) CP1_2?SetUpReq CP1_2!SetUpConf L1?StatusReport invoke(**busy**,STAT_RESOURCE_2,statusReport) NOTE: The StatusReport and the SetUpReq may appear in any order. CP1_1?SetUpResp
Pass criteria	L1?RequestEveryStatusChangeReport returnResult L1?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseABandIgnoreStat


MSC IN3_A_BASIC_RE_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
STAT_1P
TC_InvokeReq
(51, 2, 5, RES, rESArg : { resourceID lineID : '0010'H, monitorDuration 600 }, short)
TC_ContinueReq
(51)
RequestEveryStatusChangeReport
(5, 51, { resourceID lineID : '0010'H, monitorDuration 600 })
RequestEveryStatusChangeReportResult
(5, 51)
TC_ReturnResultInd
(51, 5, RES_R, rESRArg : NULL, TRUE)
TC_InvokeReq
(51, 2, 6, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '03'H } } }, sh
RequestReportBCSMEvent
(6, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(51, 2, 7, CO, cOArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '03'H }, short
TC_ContinueReq
(51)
Connnect
(7, 1, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 4, calledPartyNumber '020000'H })
SetupConf
({ callRef 4 })
SetupResp
({ callRef 1 })
ApplicationContinue
(1)
TC_InvokeInd
: { resourceStatus busy, resourceID lineID : '0010'H, reportCondition statusReport }, TRUE)
ReleaseCallA2

	IN3_A_BASIC_RE_BV_03
Work item no.:	ITEM_BASIC_137
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, related to a resource being in the idle status, and having sent to the SCF the related **RequestEveryStatusChangeReport** returnResult component, sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "timerExpired", when the monitorDuration (being present in the **RequestEveryStatusChangeReport** invoke component) expires before a status change of the related resource occurs. Verify also that the SSF does **not** send a **StatusReport** invoke component if the related resource changes to the busy status **after** timer expiry.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR) L1?RequestEveryStatusChangeReport returnResult Start **timer**(STAT_MON_DUR+) L1?StatusReport invoke(idle,STAT_RESOURCE_2,**timerExpired**) L1!RequestReportBCSMEvent(3,interrupted,oDisconnect) L1!Connect(3,1,resource2) CP1_2?SetUpReq CP1_2!SetUpConf
Pass criteria	L1?StatusReport invoke(idle,STAT_RESOURCE_2,timerExpired) No StatusReport invoke received after Connect
Postamble:	ReleaseABandIgnoreStat

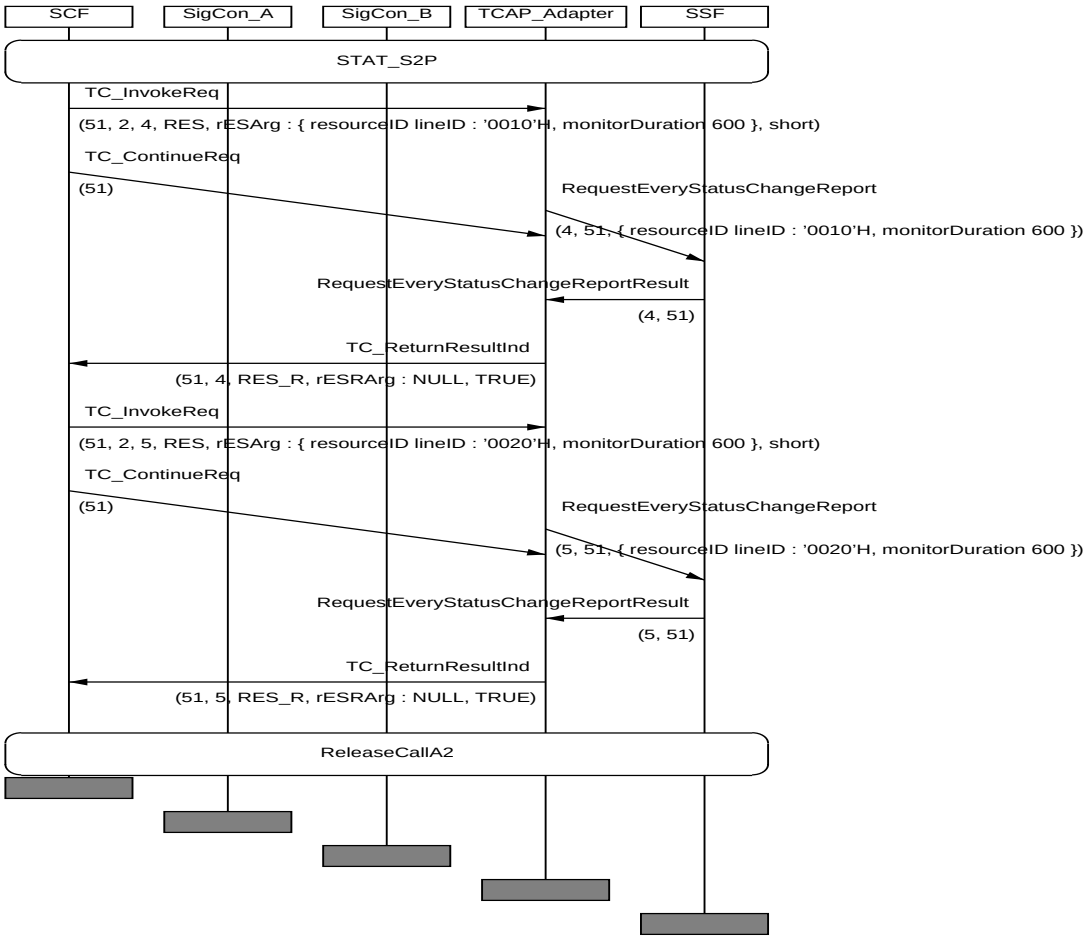

MSC IN3_A_BASIC_RE_BV_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
STAT_1P
TC_InvokeReq
(51, 2, 5, RES, rESArg : { resourceID lineID : '0010'H, monitorDuration 600 }, short)
TC_ContinueReq
(51)
RequestEveryStatusChangeReport
(5, 51, { resourceID lineID : '0010'H, monitorDuration 600 })
RequestEveryStatusChangeReportResult
(5, 51)
TC_ReturnResultInd
(51, 5, RES_R, rESRArg : NULL, TRUE)
T_MonitorDuration
(600 s)
TC_InvokeInd
: { resourceStatus idle, resourceID lineID : '0010'H, reportCondition timerExpired }, TRUE)
TC_InvokeReq
(51, 2, 6, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '03'H } } }, sh
RequestReportBCSMEvent
(6, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(51, 2, 7, CO, cCArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '03'H }, short
TC_ContinueReq
(51)
Connnect
(7, 1, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 4, calledPartyNumber '020000'H })
SetupConf
({ callRef 4 })
SetupResp
({ callRef 1 })
ReleaseCallA2

	IN3_A_BASIC_RE_BV_04
Work item no.:	ITEM_BASIC_138
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, related to a resource being in the busy status, and having sent to the SCF a **RequestEveryStatusChangeReport** returnResult component, sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "statusReport", when the related resource changes to the idle status before the monitorDuration (if present in the **RequestEveryStatusChangeReport** invoke component) expires. Verify also that the SSF sends an additional **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "statusReport", when the related resource changes again to the busy status before the monitorDuration expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR_ANY) L1?RequestEveryStatusChangeReport returnResult L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq, EventReportBCSM and StatusReport may appear in any order. L1!RequestReportBCSMEvent(3,interrupted,oDisconnect) L1!Connect(3,1,resource2) CP1_2?SetUpReq CP1_2!SetUpConf L1?StatusReport invoke(**busy**,STAT_RESOURCE_2,statusReport)
Pass criteria	L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) L1?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseABandIgnoreStat


MSC IN3_A_BASIC_RE_BV_04
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
STAT_S2P
TC_InvokeReq
(51, 2, 4, RES, rESArg : { resourceID lineID : '0010'H, monitorDuration 600 }, short)
TC_ContinueReq
(51)
RequestEveryStatusChangeReport
(4, 51, { resourceID lineID : '0010'H, monitorDuration 600 })
RequestEveryStatusChangeReportResult
(4, 51)
TC_ReturnResultInd
(51, 4, RES_R, rESRArg : NULL, TRUE)
TC_InvokeReq
(51, 2, 5, DL, dLArg : { legToBeReleased sendingSideID : '02'H }, short)
TC_ContinueReq
(51)
DisconnectLeg
(5, 1, { legToBeReleased sendingSideID : '02'H })
DisconnectLegResult
(5, 1)
TC_ReturnResultInd
(51, 5, DL_R, dLRArg : NULL, TRUE)
ReleaseReq
({ callRef 2, cause '00'H }, FALSE)
TC_InvokeInd
: { resourceStatus idle, resourceID lineID : '0010'H, reportCondition statusReport }, TRUE)
TC_InvokeReq
(51, 2, 6, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '03'H } } }, sh
RequestReportBCSMEvent
(6, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID
TC_InvokeReq
(51, 2, 7, CO, cOArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '03'H }, short
TC_ContinueReq
(51)
Connnect
(7, 1, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 4, calledPartyNumber '020000'H })
SetupConf
({ callRef 4 })
SetupResp
({ callRef 1 })
TC_InvokeInd
: { resourceStatus busy, resourceID lineID : '0010'H, reportCondition statusReport }, TRUE)
ReleaseCallA2

	IN3_A_BASIC_RE_BV_05
Work item no.:	ITEM_BASIC_139
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, related to a resource being in any status, and having sent to the SCF the related **RequestEveryStatusChangeReport** returnResult component, sends to the SCF an additional **RequestEveryStatusChangeReport** returnResult component, after having received a **RequestEveryStatusChangeReport** invoke component related to another resource (being in any status).
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestEveryStatusChangeReportinvoke(STAT_RESOURCE_1,STAT_MON_DUR_ANY) L1?RequestEveryStatusChangeReport returnResult L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR_ANY) L1?RequestEveryStatusChangeReport returnResult
Pass criteria	L1?RequestEveryStatusChangeReport returnResult L1?RequestEveryStatusChangeReport returnResult
Postamble:	ReleaseABandIgnoreStat

MSC IN3_A_BASIC_RE_BV_05



	IN3_A_BASIC_RE_BV_06
Work item no.:	ITEM_BASIC_140
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, related to a resource being in the busy status, and having sent to the SCF a **RequestEveryStatusChangeReport** returnResult component, sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "cancelled", when receiving a **CancelStatusReportRequest** invoke component for the related resource before a status change of the resource occurs and before the monitorDuration (if present in the **RequestEveryStatusChangeReport** invoke component) expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR_ANY) L1?RequestEveryStatusChangeReport returnResult L1!**Cancel**StatusReportRequest invoke(STAT_RESOURCE_2) L1?StatusReport invoke(busy,STAT_RESOURCE_2,cancelled)
Pass criteria	L1?StatusReport invoke(busy,STAT_RESOURCE_2,cancelled)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RE_BV_07
Work item no.:	ITEM_BASIC_141
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, related to a resource being in the busy status, sends to the SCF a **RequestEveryStatusChangeReport** returnResult component. Verify also that the SSF sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "statusReport", when the related resource changes to the idle status before the monitorDuration expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR) L2!TC-BEGIN L2?RequestEveryStatusChangeReport returnResult Start timer(STAT_MON_DUR) L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq and StatusReport may appear in any order L2!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L2!TC-END
Pass criteria	L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseAandIgnoreStat

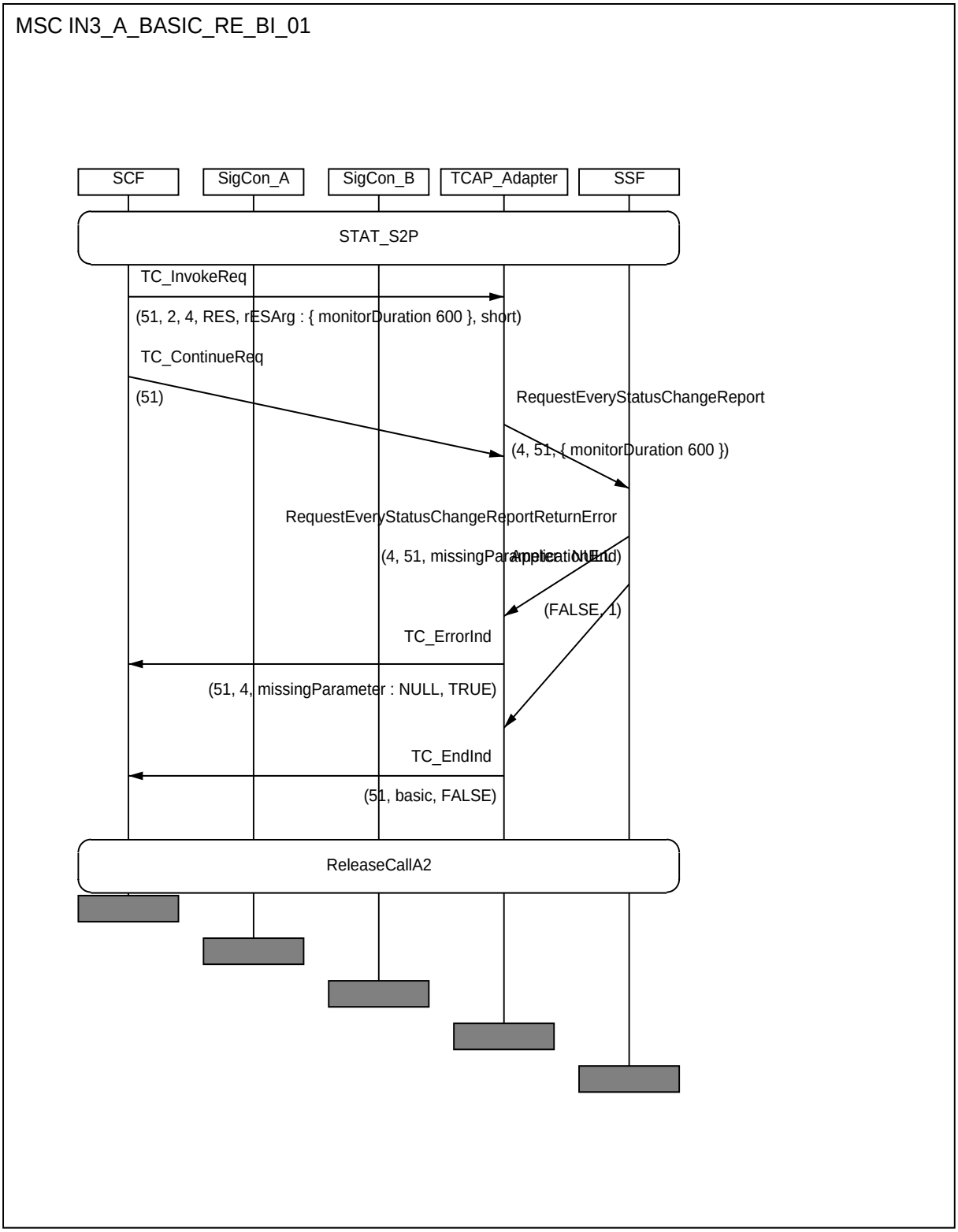
	IN3_A_BASIC_RE_BV_08
Work item no.:	ITEM_BASIC_142
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, related to a resource being in the idle status, sends to the SCF a **RequestEveryStatusChangeReport** returnResult component. Verify also that the SSF sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "statusReport", when the related resource changes to the busy status before the monitorDuration expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L2!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR) L2!TC-BEGIN L2?RequestEveryStatusChangeReport returnResult Start timer(STAT_MON_DUR) L1!RequestReportBCSMEvent(2,interrupted,oDisconnect) L1!Connect(2,1,resource2) CP1_2?SetUpReq CP1_2!SetUpConf L2?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport) NOTE: the StatusReport and the SetUpReq may appear in any order L2!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L2!TC-END
Pass criteria	L2?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RE_BV_09
Work item no.:	ITEM_BASIC_143
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, related to a resource being in the idle status, and having sent to the SCF the related **RequestEveryStatusChangeReport** returnResult component, sends to the SCF a TC-END message containing a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "timerExpired", when the monitorDuration expires before a status change of the related resource occurs.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L2!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR) L2!TC-BEGIN L2?RequestEveryStatusChangeReport returnResult Start timer(STAT_MON_DUR+) L2?TC-END L2?StatusReport invoke(idle,STAT_RESOURCE_2,timerExpired)
Pass criteria	L2?TC-END L2?StatusReport invoke(idle,STAT_RESOURCE_2,timerExpired)
Postamble:	ReleaseAandIgnoreStat

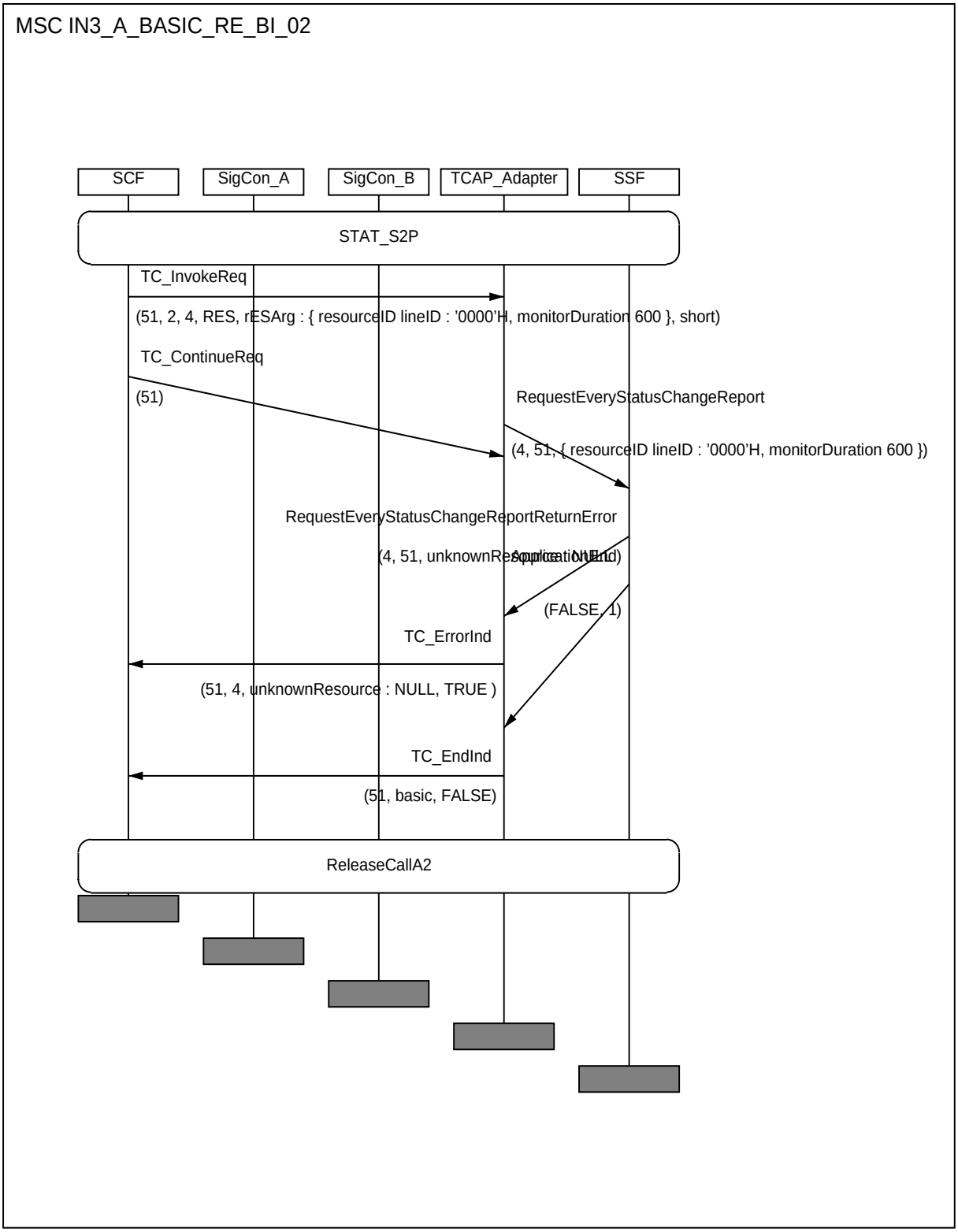
	IN3_A_BASIC_RE_BV_10
Work item no.:	ITEM_BASIC_144
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, related to a resource being in the busy status, and having sent to the SCF a **RequestEveryStatusChangeReport** returnResult component, sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "statusReport", when the related resource changes to the idle status before the monitorDuration expires. Verify also that the SSF sends an additional **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "statusReport", when the related resource changes again to the busy status before the monitorDuration expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR) L2!TC-BEGIN L2?RequestEveryStatusChangeReport returnResult L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq and StatusReport may appear in any order L1!RequestReportBCSMEvent(3,interrupted,oDisconnect) L1!Connect(3,1,resource2) CP1_2?SetUpReq CP1_2!SetUpConf L2?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport) L2!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L2!TC-END
Pass criteria	L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) L2?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RE_BV_11
Work item no.:	ITEM_BASIC_146
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, related to a resource being in the busy status, and having sent to the SCF a **RequestEveryStatusChangeReport** returnResult component, sends to the SCF a TC-END message containing a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "cancelled", when receiving a **CancelStatusReportRequest** invoke component for the related resource before a status change of the resource occurs and before the monitorDuration expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR) L2!TC-BEGIN L2?RequestEveryStatusChangeReport returnResult L2!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L2?TC-END L2?StatusReport invoke(busy,STAT_RESOURCE_2,cancelled)
Pass criteria	L2?TC-END L2?StatusReport invoke(busy,STAT_RESOURCE_2,cancelled)
Postamble:	ReleaseABandIgnoreStat

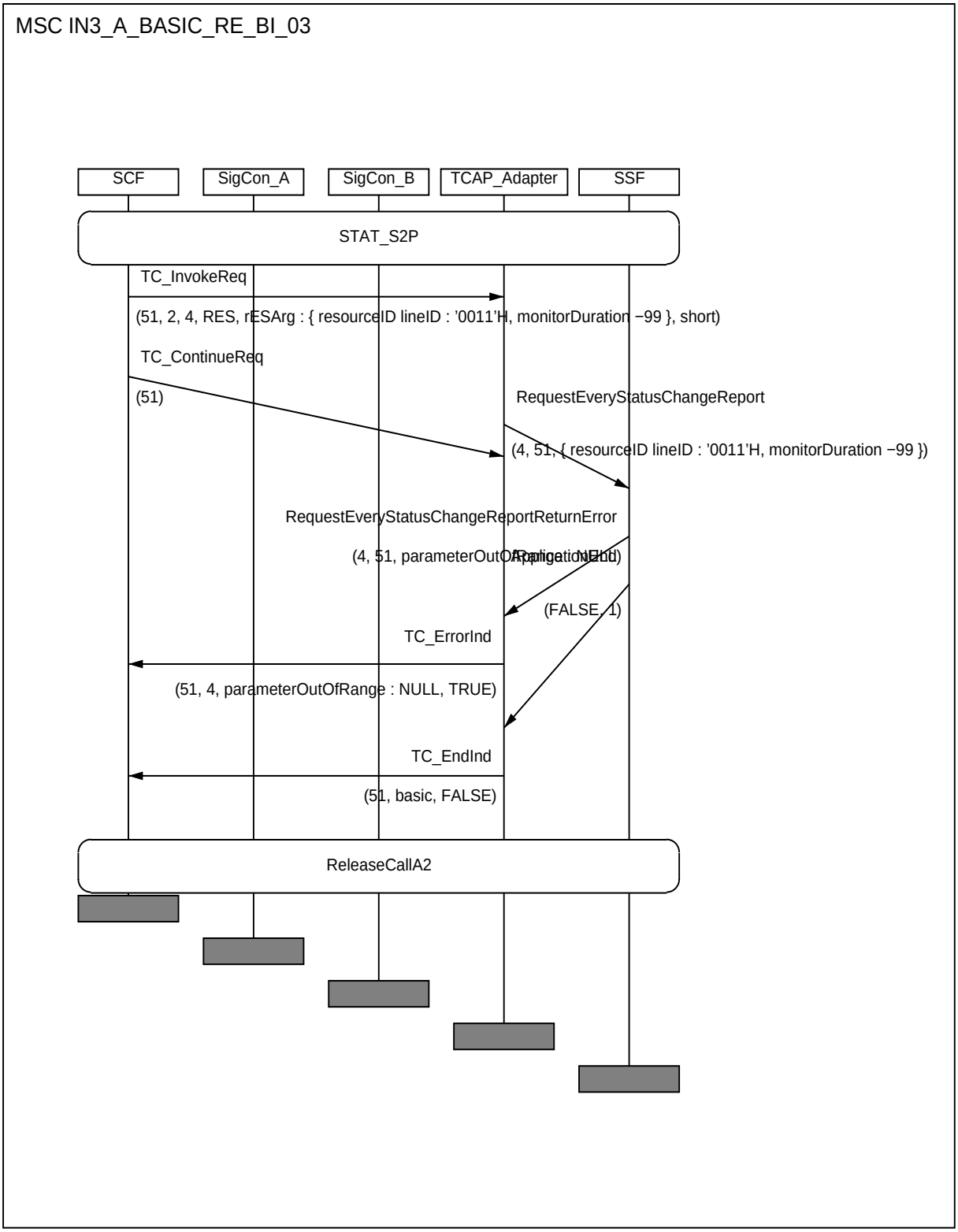
	IN3_A_BASIC_RE_BI_01
Work item no.:	ITEM_BASIC_147
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, where mandatory parameter resourceID is missing, sends to the SCF a **RequestEveryStatusChangeReport** returnError component with error code "missingParameter".
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestEveryStatusChangeReport invoke(resourceID omitted,STAT_MON_DUR_ANY) L1?RequestEveryStatusChangeReport returnError(missingParameter)
Pass criteria	L1?RequestEveryStatusChangeReport returnError(missingParameter)
Postamble:	ReleaseABandIgnoreStat


MSC IN3_A_BASIC_RE_BI_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
STAT_S2P
TC_InvokeReq
(51, 2, 4, RES, rESArg : { monitorDuration 600 }, short)
TC_ContinueReq
(51)
RequestEveryStatusChangeReport
(4, 51, { monitorDuration 600 })
RequestEveryStatusChangeReportReturnError
(4, 51, missingParameter : NULL)
(FALSE, 1)
TC_ErrorInd
(51, 4, missingParameter : NULL, TRUE)
TC_EndInd
(51, basic, FALSE)
ReleaseCallA2

	IN3_A_BASIC_RE_BI_02
Work item no.:	ITEM_BASIC_148
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, where parameter resourceID does not identify a resource known to the SSF, sends to the SCF a **RequestEveryStatusChangeReport** returnError component with error code "unknownResource".
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_U,STAT_MON_DUR_ANY) L1?RequestEveryStatusChangeReport returnError(unknownResource)
Pass criteria	L1?RequestEveryStatusChangeReport returnError(unknownResource)
Postamble:	ReleaseABandIgnoreStat


MSC IN3_A_BASIC_RE_BI_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
STAT_S2P
TC_InvokeReq
(51, 2, 4, RES, rESArg : { resourceID lineID : '0000'H, monitorDuration 600 }, short)
TC_ContinueReq
(51)
RequestEveryStatusChangeReport
(4, 51, { resourceID lineID : '0000'H, monitorDuration 600 })
RequestEveryStatusChangeReportReturnError
(4, 51, unknownResource : NULL)
(FALSE, 1)
TC_ErrorInd
(51, 4, unknownResource : NULL, TRUE)
TC_EndInd
(51, basic, FALSE)
ReleaseCallA2

	IN3_A_BASIC_RE_BI_03
Work item no.:	ITEM_BASIC_149
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, where parameter monitorDuration has a value < -2, sends to the SCF a **RequestEveryStatusChangeReport** returnError component with error code "parameterOutOfRange".
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,monitorDuration: < -2) L1?RequestEveryStatusChangeReport returnError(parameterOutOfRange)
Pass criteria	L1?RequestEveryStatusChangeReport returnError(parameterOutOfRange)
Postamble:	ReleaseABandIgnoreStat


MSC IN3_A_BASIC_RE_BI_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
STAT_S2P
TC_InvokeReq
(51, 2, 4, RES, rESArg : { resourceID lineID : '0011'H, monitorDuration −99 }, short)
TC_ContinueReq
(51)
RequestEveryStatusChangeReport
(4, 51, { resourceID lineID : '0011'H, monitorDuration −99 })
RequestEveryStatusChangeReportReturnError
(4, 51, parameterOutOfRange, NULL)
(FALSE, 1)
TC_ErrorInd
(51, 4, parameterOutOfRange : NULL, TRUE)
TC_EndInd
(51, basic, FALSE)
ReleaseCallA2

	IN3_A_BASIC_RE_BI_04
Work item no.:	ITEM_BASIC_150
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, related to a resource being in the "busy" status, and having sent to the SCF a **RequestEveryStatusChangeReport** returnResult component, sends to the SCF a **CancelStatusReportRequest** returnError component indicating errorCode "missingParameter", when receiving a **CancelStatusReportRequest** invoke component where mandatory parameter resourceID is missing. Verify also that the SSF sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "statusReport", when the related resource changes to the idle status before the monitorDuration (if present in the **RequestEveryStatusChangeReport** invoke component) expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR_ANY) L1?RequestEveryStatusChangeReport returnResult L1!CancelStatusReportRequest invoke(resourceID omitted) L1?CancelStatusReportRequest returnError(missingParameter) L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq and StatusReport may appear in any order.
Pass criteria	L1?CancelStatusReportRequest returnError(missingParameter) L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RE_BI_05
Work item no.:	ITEM_BASIC_151
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, where mandatory parameter resourceID is missing, sends to the SCF a TC-END message containing a **RequestEveryStatusChangeReport** returnError component with error code "missingParameter".
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestEveryStatusChangeReport invoke(resourceID omitted,STAT_MON_DUR) L2!TC-BEGIN L2?TC-END L2?RequestEveryStatusChangeReport returnError(missingParameter)
Pass criteria	L2?TC-END L2?RequestEveryStatusChangeReport returnError(missingParameter)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RE_BI_06
Work item no.:	ITEM_BASIC_152
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, where parameter resourceID does not identify a resource known to the SSF, sends to the SCF a TC-END message containing a **RequestEveryStatusChangeReport** returnError component with error code "unknownResource".
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_U,STAT_MON_DUR) L2!TC-BEGIN L2?TC-END L2?RequestEveryStatusChangeReport returnError(unknownResource)
Pass criteria	L2?RequestEveryStatusChangeReport returnError(unknownResource)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RE_BI_07
Work item no.:	ITEM_BASIC_153
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, where parameter monitorDuration has a value > 86400, sends to the SCF a TC-END message containing a **RequestEveryStatusChangeReport** returnError component with error code "parameterOutOfRange".
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,monitorDuration: > 86400) L2!TC-BEGIN L2?TC-END L2?RequestEveryStatusChangeReport returnError(parameterOutOfRange)
Pass criteria	L2?RequestEveryStatusChangeReport returnError(parameterOutOfRange)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RE_BI_08
Work item no.:	ITEM_BASIC_154
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, where mandatory parameter monitorDuration is missing, sends to the SCF a TC-END message containing a **RequestEveryStatusChangeReport** returnError component with error code "missingParameter".
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,monitorDuration omitted) L2!TC-BEGIN L2?TC-END L2?RequestEveryStatusChangeReport returnError(missingParameter)
Pass criteria	L2?RequestEveryStatusChangeReport returnError(missingParameter)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RE_BI_09
Work item no.:	ITEM_BASIC_155
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, related to a resource being in the "idle" status, and having sent to the SCF a **RequestEveryStatusChangeReport** returnResult component, sends to the SCF a **CancelStatusReportRequest** returnError component indicating errorCode "missingParameter", when receiving a **CancelStatusReportRequest** invoke component where mandatory parameter resourceID is missing. Verify also that the SSF sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "statusReport", when the related resource changes to the busy status before the monitorDuration expires.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR) L2?RequestEveryStatusChangeReport returnResult Start timer(STAT_MON_DUR+) L2!CancelStatusReportRequest invoke(resourceID omitted) L2?CancelStatusReportRequest returnError(missingParameter) L2!DisconnectLeg(2) L2?DisconnectLeg ReturnResult CP1_2?ReleaseReq L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq and StatusReport may appear in any order L2!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L2!TC-END
Pass criteria	L2?CancelStatusReportRequest returnError(missingParameter) L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RE_BO_01
Work item no.:	ITEM_BASIC_156
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, not having received from the SCF a **RequestEveryStatusChangeReport** (or RequestFirstStatusMatchReport) invoke component **in the context of a call**, sends to the SCF a **CancelStatusReportRequest** returnError component, indicating error code "cancelFailed" or "taskRefused", when receiving in the context of the call a **CancelStatusReportRequest** invoke component for a resource used in the call.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L1?CancelStatusReportRequest returnError("cancelFailed" or "taskRefused")
Pass criteria	L1?CancelStatusReportRequest returnError("cancelFailed" or "taskRefused")
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RE_BO_02
Work item no.:	ITEM_BASIC_157
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **in the context of a call**, related to a resource being in the busy status and containing a monitorDuration, and having sent to the SCF a **RequestEveryStatusChangeReport** returnResult component, sends to the SCF a **CancelStatusReportRequest** returnError component, indicating error code "cancelFailed" or "taskRefused", when receiving a **CancelStatusReportRequest** invoke component **after** the monitorDuration has expired (and the corresponding **StatusReport** has been sent).
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestEveryStatusChangeReport invoke(STAT_RESOURCE_2,STAT_MON_DUR) L1?RequestEveryStatusChangeReport returnResult Start timer(STAT_MON_DUR+) L1?StatusReport invoke(busy,STAT_RESOURCE_2,timerExpired) L1!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L1?CancelStatusReportRequest returnError("cancelFailed" or "taskRefused")
Pass criteria	L1?CancelStatusReportRequest returnError("cancelFailed" or "taskRefused")
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RE_BO_03
Work item no.:	ITEM_BASIC_145
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestEveryStatusChangeReport** invoke component **outside the context of a call**, related to a resource being in the busy status, followed (in the same context) by another **RequestEveryStatusChangeReport** invoke component related to another resource being in any status, sends to the SCF a **RequestEveryStatusChangeReport** returnResult component indicating resourceStatus "busy" for the first invoked operation, and sends a **RequestEveryStatusChangeReport** returnError component indicating error code "taskRefused" or "unexpectedComponentSequence" for the second invoked operation.
Requirements refs	11.10, 11.35, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestEveryStatusChangeReport invoke(invID1,STAT_RESOURCE_2,STAT_MON_DUR) L2!TC-BEGIN L2?RequestEveryStatusChangeReport returnResult(invID1) L2!RequestEveryStatusChangeReport invoke(invID2,STAT_RESOURCE_1,STAT_MON_DUR) L2!TC-CONTINUE L2?RequestEveryStatusChangeReport returnError(invID2,"taskRefused" or "unexpectedComponentSequence") L2!CancelStatusReportRequest invoke(invID1,"STAT_RESOURCE_2) L2!TC-END
Pass criteria	L2?RequestEveryStatusChangeReport returnError(invID2,"taskRefused" or "unexpectedComponentSequence")
Postamble:	ReleaseABandIgnoreStat

6.6.20 RequestFirstStatusMatchReport (RF) procedure

	IN3_A_BASIC_RF_BV_01
Work item no.:	ITEM_BASIC_159
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, related to a resource being in the busy status and where the resourceStatus received indicates "idle", sends to the SCF a **RequestFirstStatusMatchReport** returnResult component. Verify also that the SSF sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "statusReport", when the related resource changes to the idle status before the monitorDuration (if present in the **RequestFirstStatusMatchReport** invoke component) expires.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnResult L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq and StatusReport may appear in any order
Pass criteria	L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RF_BV_02
Work item no.:	ITEM_BASIC_160
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, related to a resource being in the busy status and where the resourceStatus received indicates "busy", sends to the SCF a **RequestFirstStatusMatchReport** returnResult component, followed by a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "statusReport".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,busy,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnResult L1?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Pass criteria	L1?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BV_03
Work item no.:	ITEM_BASIC_161
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, related to a resource being in the idle status and where the resourceStatus received indicates "busy", sends to the SCF a **RequestFirstStatusMatchReport** returnResult component. Verify also that the SSF sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "statusReport", when the related resource changes to the busy status before the monitorDuration (if present in the **RequestFirstStatusMatchReport** invoke component) expires.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,busy,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnResult L1!RequestReportBCSMEvent(2,interrupted,oDisconnect) L1!Connect(2,1,resource2) CP1_2?SetUpReq CP1_2!SetUpConf L1?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport) NOTE: The StatusReport and the SetUpReq may appear in any order.
Pass criteria	L1?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BV_04
Work item no.:	ITEM_BASIC_162
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, related to a resource being in the idle status and where the resourceStatus received indicates "busy", and having sent to the SCF the related **RequestFirstStatusMatchReport** returnResult component, sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "timerExpired", when the monitorDuration (being present in the **RequestFirstStatusMatchReport** invoke component) expires before a status change of the related resource occurs. Verify also that the SSF does **not** send a **StatusReport** invoke component if the related resource changes to the busy status **after** timer expiry.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,busy,STAT_MON_DUR) L1?RequestFirstStatusMatchReport returnResult Start timer(STAT_MON_DUR+) L1?StatusReport invoke(idle,STAT_RESOURCE_2,timerExpired) L1!RequestReportBCSMEvent(2,interrupted,oDisconnect) L1!Connect(2,1,resource2) CP1_2?SetUpReq CP1_2!SetUpConf
Pass criteria	L1?StatusReport invoke(idle,STAT_RESOURCE_2,timerExpired), No StatusReport received after Connect
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BV_05
Work item no.:	ITEM_BASIC_163
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, related to a resource being in the "busy" status and where the resourceStatus received indicates "idle" status, and having sent to the SCF the related **RequestFirstStatusMatchReport** returnResult component, sends to the SCF an additional **RequestFirstStatusMatchReport** returnResult component, after having received a **RequestFirstStatusMatchReport** invoke component related to another resource (being in any status).
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(invID1,STAT_RESOURCE_1,idle,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnResult(invID1) L1!RequestFirstStatusMatchReport invoke(invID2,STAT_RESOURCE_2,idle,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnResult(invID2)
Pass criteria	L1?RequestFirstStatusMatchReport returnResult(invID1) L1?RequestFirstStatusMatchReport returnResult(invID2)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BV_06
Work item no.:	ITEM_BASIC_164
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, related to a resource being in the busy status and where the resourceStatus received indicates "idle", and having sent to the SCF a **RequestFirstStatusMatchReport** returnResult component, sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "cancelled", when receiving a **CancelStatusReportRequest** invoke component for the related resource before a status change of the resource occurs and before the monitorDuration (if present in the **RequestFirstStatusMatchReport** invoke component) expires. Verify also that the SSF does **not** send a **StatusReport** invoke component if the related resource changes to the idle status **after** cancellation.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnResult L1!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L1?StatusReport invoke(busy,STAT_RESOURCE_2,cancelled) L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq
Pass criteria	L1?StatusReport invoke(busy,STAT_RESOURCE_2,cancelled), No StatusReport invoke after Disconnect
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RF_BV_07
Work item no.:	ITEM_BASIC_165
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, related to a resource being in the busy status and where the resourceStatus received indicates "idle", sends to the SCF a **RequestFirstStatusMatchReport** returnResult component. Verify also that the SSF sends to the SCF a TC-END message containing a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "statusReport", when the related resource changes to the idle status before the monitorDuration expires.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,STAT_MON_DUR) L2!TC-BEGIN L2?RequestFirstStatusMatchReport returnResult L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq L2?TC-END L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq, EventReportBCSM and StatusReport may appear in any order.
Pass criteria	L2?RequestFirstStatusMatchReport returnResult, L2?TC-END L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RF_BV_08
Work item no.:	ITEM_BASIC_166
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, related to a resource being in the busy status and where the resourceStatus received indicates "busy", sends to the SCF a **RequestFirstStatusMatchReport** returnResult component, followed by a TC-END message containing a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "statusReport".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,busy,STAT_MON_DUR) L2!TC-BEGIN L2?RequestFirstStatusMatchReport returnResult L2?TC-END L2?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Pass criteria	L2?TC-END L2?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BV_09
Work item no.:	ITEM_BASIC_167
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, related to a resource being in the "idle" status and where the resourceStatus received indicates "busy", sends to the SCF a **RequestFirstStatusMatchReport** returnResult component. Verify also that the SSF sends to the SCF a TC-END message containing a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "statusReport", when the related resource changes to the busy status before the monitorDuration expires.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L2!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,busy,STAT_MON_DUR) L2!TC-BEGIN L2?RequestFirstStatusMatchReport returnResult L1!RequestReportBCSMEvent(2,interrupted,oDisconnect) L1!Connect(2,1,resource2) CP1_2?SetUpReq CP1_2!SetUpConf L2?TC-END L2?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport) NOTE: The StatusReport and the SetUpReq may appear in any order.
Pass criteria	L2?RequestFirstStatusMatchReport returnResult, L2?TC-END L2?StatusReport invoke(busy,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BV_10
Work item no.:	ITEM_BASIC_168
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, related to a resource being in the idle status and where the resourceStatus received indicates "busy", and having sent to the SCF the related **RequestFirstStatusMatchReport** returnResult component, sends to the SCF a TC-END message containing a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "timerExpired", when the monitorDuration expires before a status change of the related resource occurs.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_1P
Test description	L2!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,busy,STAT_MON_DUR) L2!TC-BEGIN L2?RequestFirstStatusMatchReport returnResult Start timer(STAT_MON_DUR+) L2?TC-END L2?StatusReport invoke(idle,STAT_RESOURCE_2,timerExpired)
Pass criteria	L2?TC-END L2?StatusReport invoke(idle,STAT_RESOURCE_2,timerExpired)
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RF_BV_11
Work item no.:	ITEM_BASIC_169
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, related to a resource being in the busy status and where the resourceStatus received indicates "idle", and having sent to the SCF a **RequestFirstStatusMatchReport** returnResult component, sends to the SCF a TC-END message containing a **StatusReport** invoke component indicating resourceStatus = "busy" and reportCondition = "cancelled", when receiving a **CancelStatusReportRequest** invoke component for the related resource before a status change of the resource occurs and before the monitorDuration expires.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,STAT_MON_DUR) L2!TC-BEGIN L2?RequestFirstStatusMatchReport returnResult L1!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L2?TC-END L2?StatusReport invoke(busy,STAT_RESOURCE_2,cancelled)
Pass criteria	L2?TC-END L2?StatusReport invoke(busy,STAT_RESOURCE_2,cancelled)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BI_01
Work item no.:	ITEM_BASIC_170
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, where mandatory parameter resourceID is missing, sends to the SCF a **RequestFirstStatusMatchReport** returnError component with error code "missingParameter".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	RFSMMandatory
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(resourceID omitted,busy,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnError(missingParameter)
Pass criteria	L1?RequestFirstStatusMatchReport returnError(missingParameter)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BI_02
Work item no.:	ITEM_BASIC_171
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, where parameter resourceID does not identify a resource known to the SSF, sends to the SCF a **RequestFirstStatusMatchReport** returnError component with error code "unknownResource".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_U,idle,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnError(unknownResource)
Pass criteria	L1?RequestFirstStatusMatchReport returnError(unknownResource)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BI_03
Work item no.:	ITEM_BASIC_172
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, where mandatory parameter resourceStatus is missing, sends to the SCF a **RequestFirstStatusMatchReport** returnError component with error code "missingParameter".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	RFSMMandatory
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,resourceStatus omitted,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnError(missingParameter)
Pass criteria	L1?RequestFirstStatusMatchReport returnError(missingParameter)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BI_04
Work item no.:	ITEM_BASIC_173
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, where parameter monitorDuration has a value < -2, sends to the SCF a **RequestFirstStatusMatchReport** returnError component with error code "parameterOutOfRange".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,monitorDuration < -2) L1?RequestFirstStatusMatchReport returnError(parameterOutOfRange)
Pass criteria	L1?RequestFirstStatusMatchReport returnError(parameterOutOfRange)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BI_05
Work item no.:	ITEM_BASIC_174
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, related to a resource being in the "busy" status and where the resourceStatus received indicates "idle", and having sent to the SCF a **RequestFirstStatusMatchReport** returnResult component, sends to the SCF a **CancelStatusReportRequest** returnError component indicating errorCode "missingParameter", when receiving a **CancelStatusReportRequest** invoke component where mandatory parameter resourceID is missing. Verify also that the SSF sends to the SCF a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "statusReport", when the related resource changes to the idle status before the monitorDuration (if present in the **RequestFirstStatusMatchReport** invoke component) expires.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,STAT_MON_DUR_ANY) L1?RequestFirstStatusMatchReport returnResult L1!CancelStatusReportRequest invoke(resourceID omitted) L1?CancelStatusReportRequest returnError(missingParameter) L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq and StatusReport may appear in any order.
Pass criteria	L1?CancelStatusReportRequest returnError(missingParameter), L1?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RF_BI_06
Work item no.:	ITEM_BASIC_175
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, where mandatory parameter resourceID is missing, sends to the SCF a TC-END message containing a **RequestFirstStatusMatchReport** returnError component with error code "missingParameter".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	RFSMMandatory
Preamble:	STAT_S2P
Test description	L2!RequestFirstStatusMatchReport invoke(resourceID omitted,idle,STAT_MON_DUR_ANY) L2!TC-BEGIN L2?TC-END L2?RequestFirstStatusMatchReport returnError(missingParameter)
Pass criteria	L2?TC-END L2?RequestFirstStatusMatchReport returnError(missingParameter)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BI_07
Work item no.:	ITEM_BASIC_176
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, where parameter resourceID does not identify a resource known to the SSF, sends to the SCF a TC-END message containing a **RequestFirstStatusMatchReport** returnError component with error code "unknownResource".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_U,idle,STAT_MON_DUR_ANY) L2!TC-BEGIN L2?TC-END L2?RequestFirstStatusMatchReport returnError(unknownResource)
Pass criteria	L2?TC-END L2?RequestFirstStatusMatchReport returnError(unknownResource)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BI_08
Work item no.:	ITEM_BASIC_177
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, where parameter monitorDuration has a value > 86400, sends to the SCF a TC-END message containing a **RequestFirstStatusMatchReport** returnError component with error code "parameterOutOfRange".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,monitorDuration > 86400) L2!TC-BEGIN L2?TC-END L2?RequestFirstStatusMatchReport returnError(parameterOutOfRange)
Pass criteria	L2?TC-END L2?RequestFirstStatusMatchReport returnError(parameterOutOfRange)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BI_09
Work item no.:	ITEM_BASIC_178
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, where mandatory parameter monitorDuration is missing, sends to the SCF a TC-END message containing a **RequestFirstStatusMatchReport** returnError component with error code "missingParameter".
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,monitorDuration omitted) L2!TC-BEGIN L2?TC-END L2?RequestFirstStatusMatchReport returnError(missingParameter)
Pass criteria	L2?TC-END L2?RequestFirstStatusMatchReport returnError(missingParameter)
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BI_10
Work item no.:	ITEM_BASIC_179
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, related to a resource being in the "busy" status, and having sent to the SCF a **RequestFirstStatusMatchReport** returnResult component, sends to the SCF a **CancelStatusReportRequest** returnError component indicating errorCode "missingParameter", when receiving a **CancelStatusReportRequest** invoke component where mandatory parameter resourceID is missing. Verify also that the SSF sends to the SCF a TC-END message containing a **StatusReport** invoke component indicating resourceStatus = "idle" and reportCondition = "statusReport", when the related resource changes to the idle status before the monitorDuration expires.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,STAT_MON_DUR_ANY) L2!TC-BEGIN L2?RequestFirstStatusMatchReport returnResult L2!CancelStatusReportRequest invoke(resourceID omitted) L2?CancelStatusReportRequest returnError(missingParameter) L1!DisconnectLeg(2) L1?DisconnectLeg ReturnResult CP1_2?ReleaseReq L2?TC-END L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport) NOTE: The ReleaseReq and StatusReport may appear in any order.
Pass criteria	L2?CancelStatusReportRequest returnError(missingParameter), L2?TC-END L2?StatusReport invoke(idle,STAT_RESOURCE_2,statusReport)
Postamble:	ReleaseAandIgnoreStat

	IN3_A_BASIC_RF_BO_01
Work item no.:	ITEM_BASIC_180
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **in the context of a call**, related to a resource being in the busy status, and where the resourceStatus received indicates "idle" and contains the monitorDuration parameter, and having sent to the SCF a **RequestFirstStatusMatchReport** returnResult component, sends to the SCF a **CancelStatusReportRequest** returnError component, indicating error code "cancelFailed" or "taskRefused", when receiving a **CancelStatusReportRequest** invoke component **after** the monitorDuration has expired (and the corresponding **StatusReport** has been sent).
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L1!RequestFirstStatusMatchReport invoke(STAT_RESOURCE_2,idle,STAT_MON_DUR) L1?RequestFirstStatusMatchReport returnResult Start timer(STAT_MON_DUR+) L1?StatusReport invoke(busy,STAT_RESOURCE_2,timerExpired) L1!CancelStatusReportRequest invoke(STAT_RESOURCE_2) L1?CancelStatusReportRequest returnError("cancelFailed" or "taskRefused")
Pass criteria	L1?CancelStatusReportRequest returnError("cancelFailed" or "taskRefused")
Postamble:	ReleaseABandIgnoreStat

	IN3_A_BASIC_RF_BO_02
Work item no.:	ITEM_BASIC_181
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a **RequestFirstStatusMatchReport** invoke component **outside the context of a call**, related to a resource being in the "busy" status and where the resourceStatus received indicates "idle", and having sent to the SCF the related **RequestFirstStatusMatchReport** returnResult component, sends to the SCF a **RequestFirstStatusMatchReport** returnError component with error code "taskRefused" or "unexpectedComponentSequence", after having received an additional **RequestFirstStatusMatchReport** invoke component related to another resource (being in any status) in the same context.
Requirements refs	11.10, 11.36, 11.47, 15.1.1.1.1.3, 15.1.1.1.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	STAT_S2P
Test description	L2!RequestFirstStatusMatchReport invoke(invID1,STAT_RESOURCE_1,idle,STAT_MON_DUR_ANY) L2!TC-BEGIN L2?RequestFirstStatusMatchReport returnResult(invID1) L2!RequestFirstStatusMatchReport invoke(invID2,STAT_RESOURCE_2,idle,STAT_MON_DUR_ANY) L2?RequestFirstStatusMatchReport returnError(invID2,"taskRefused" or "unexpectedComponentSequence") L2!CancelStatusReportRequest invoke(STAT_RESOURCE_1) L2?TC-END L2?StatusReport invoke(busy,STAT_RESOURCE_1,cancelled)
Pass criteria	L2?RequestFirstStatusMatchReport returnError(invID2,"taskRefused" or "unexpectedComponentSequence")
Postamble:	ReleaseABandIgnoreStat

6.6.21 RequestNotificationChargingEvent (RN) procedure

	IN3_A_BASIC_RN_BV_01
Work item no.:	ITEM_BASIC_356
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Waiting for instructions" FSM for CS state (before the called B-party has received the call), containing one ChargingEvent specification indicating monitorMode=interrupted, legID=2 and eventTypeTariff = chargingTariffInformation, sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=interrupted, legID=2, eventTypeTariff= chargingTariffInformation, eventSpecificInformationTariff = crgt), after having received an ApplicationTransfer message from SigConB containing ChargingTariffInformation(advice-of-charge). Check also that **no** ApplicationTransfer message containing ChargingTariffInformation is sent to SigConA (interrupted mode).
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=interrupted, legID=2, eventTypeTariff= chargingTariffInformation)) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=2, eventTypeTariff= chargingTariffInformation, eventSpecificInformationTariff = crgt) CP1_2!SetupConf (This indicates start of charging) CP1_1?SetupResp NOTE: APM(ChargingAcknowledgementInformation(accepted)) and EventNotificationCharging may be received in any order.
Pass criteria	L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=2, eventTypeTariff= chargingTariffInformation, eventSpecificInformationTariff = crgt) **No** ApplicationTransfer message containing ChargingTariffInformation is sent to SigConA
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RN_BV_02
Work item no.:	ITEM_BASIC_357
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Waiting for instructions" FSM for CS state (before the called B-party has received the call), containing one ChargingEvent specification indicating monitorMode=interrupted, legID=2 and eventTypeTariff = startCharging, and having received an ApplicationTransfer message from SigConB containing ChargingTariffInformation where the chargingControlIndicators indicate delayUntilStart="delay-start-of-tariffing", does **not** send an EventNotificationCharging invoke component to the SCF when receiving the ANSWER message (SetupConf signal) from SigConB. Verify also that the SSF sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=interrupted, legID=2, eventTypeTariff= startCharging, eventSpecificInformationTariff = start), after having received an ApplicationTransfer message from SigConB containing the StartCharging parameter.
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=interrupted, legID=2, eventTypeTariff= startCharging)) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,delay-start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This does not indicate start of charging) CP1_1?SetupResp Wait a while, to assure that no EventNotificationCharging invoke is received CP1_2!APM(StartCharging) L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=2, eventTypeTariff= startCharging, eventSpecificInformationTariff = start)
Pass criteria	no EventNotificationCharging invoke is received after SetupResp L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=2, eventTypeTariff= startCharging, eventSpecificInformationTariff = start)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RN_BV_03
Work item no.:	ITEM_BASIC_358
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Waiting for instructions" FSM for CS state (before the called B-party has received the call), containing one ChargingEvent specification indicating monitorMode=interrupted, legID=2 and eventTypeTariff = startCharging, and having received an ApplicationTransfer message from SigConB containing ChargingTariffInformation where the chargingControlIndicators indicate delayUntilStart="start-tariffing", sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=interrupted, legID=2, eventTypeTariff= startCharging) when receiving the ANSWER message (SetupConf signal) from SigConB.
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=interrupted, legID=2, eventTypeTariff= startCharging)) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) CP1_1?SetupResp L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=2, eventTypeTariff= startCharging, eventSpecificInformationTariff = start IF_PRESENT) NOTE: SetupResp and EventNotificationCharging invoke may appear in any order Tmp. NOTE: It is currently not clear if eventSpecificInformationTariff is present in this case.
Pass criteria	L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=2, eventTypeTariff= startCharging, eventSpecificInformationTariff = start IF_PRESENT)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RN_BV_04
Work item no.:	ITEM_BASIC_359
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Monitoring" FSM for CS state (before the called B-party has released the call), containing one ChargingEvent specification indicating monitorMode=notifyAndContinue, legID=2 and eventTypeTariff = stopCharging, and having received an ApplicationTransfer message from SigConB containing the StopCharging parameter, sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging, eventSpecificInformationTariff = stop). Verify also that the SSF does not clear the call upon receiving the ApplicationTransfer message containing the StopCharging parameter.
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) CP1_1?SetupResp L1!RequestNotificationChargingEvent invoke((monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging)) Wait a while CP1_2!APM(StopCharging) L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging, eventSpecificInformationTariff = stop) Wait a while to check that no ReleaseReq is received
Pass criteria	L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging, eventSpecificInformationTariff = stop) No ReleaseReq is received after APM(StopCharging)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RN_BV_05
Work item no.:	ITEM_BASIC_360
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Monitoring" FSM for CS state (before the called B-party has released the call), containing one ChargingEvent specification indicating monitorMode=notifyAndContinue, legID=2 and eventTypeTariff = stopCharging, and having received a ReleaseInd message from SigConB, sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging).
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) CP1_1?SetupResp L1!RequestNotificationChargingEvent invoke((monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging)) Wait a while CP1_2!ReleaseInd(Normal cause) L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging, eventSpecificInformationTariff = stop IF_PRESENT) CP1_1?ReleaseReq(Normal cause) Tmp. NOTE 1: It is currently not clear if eventSpecificInformationTariff is present in this case. NOTE 2: EventNotificationCharging invoke and ReleaseReq may appear in any order.
Pass criteria	L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging, eventSpecificInformationTariff = stop IF_PRESENT)
Postamble:	None

	IN3_A_BASIC_RN_BV_06
Work item no.:	ITEM_BASIC_361
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Waiting for instructions" FSM for CS state (before the called B-party has received the call), containing one ChargingEvent specification indicating monitorMode=notifyAndContinue, legID=2 and eventTypeTariff = addOnChargingInformation, sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=notifyAndContinue, legID=2, eventTypeTariff= addOnChargingInformation, eventSpecificInformationTariff = aocrg), after having received an ApplicationTransfer message from SigConB containing AddOnChargingInformation (after start of charging).
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=notifyAndContinue, legID=2, eventTypeTariff= addOnChargingInformation)) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_2!SetupConf (This indicates start of charging) CP1_1?SetupResp Wait a while CP1_2!APM(addOnchargingInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,ADD_ON_CHARGE)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= addOnChargingInformation, eventSpecificInformationTariff = aocrg) NOTE: APM(ChargingAcknowledgementInformation(accepted)) and EventNotificationCharging invoke may be received in any order.
Pass criteria	L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= addOnChargingInformation, eventSpecificInformationTariff = aocrg)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RN_BV_07
Work item no.:	ITEM_BASIC_362
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Waiting for instructions" FSM for CS state (before the called B-party has received the call), containing one ChargingEvent specification indicating monitorMode=interrupted, legID=1 and eventTypeTariff = chargingAcknowledgementInformation, sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=interrupted, legID=1, eventTypeTariff= chargingAcknowledgementInformation, eventSpecificInformationTariff = crga), after having received an ApplicationTransfer message from SigConA containing ChargingAcknowledgementInformation.
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=interrupted, legID=1, eventTypeTariff= chargingAcknowledgementInformation)) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_1?APM(ChargingTariffInformation(advice-of-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_1!APM(ChargingAcknowledgementInformation(accepted)) L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=1, eventTypeTariff= chargingAcknowledgementInformation, eventSpecificInformationTariff = aocrg)
Pass criteria	L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=1, eventTypeTariff= chargingAcknowledgementInformation, eventSpecificInformationTariff = aocrg)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RN_BV_08
Work item no.:	ITEM_BASIC_363
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Waiting for instructions" FSM for CS state (before the called B-party has received the call), containing one ChargingEvent specification indicating monitorMode=interrupted, legID=1 and eventTypeTariff = chargingAcknowledgementTimerExpired, sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=interrupted, legID=1, eventTypeTariff= chargingAcknowledgementTimerExpired), when SigConA does not acknowledge the received ApplicationTransfer message containing ChargingTariffInformation.
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateIntoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=interrupted, legID=1, eventTypeTariff= chargingAcknowledgementInformation)) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) CP1_1?APM(ChargingTariffInformation(advice-of-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) Wait for timeout Tcrga L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=1, eventTypeTariff= chargingAcknowledgementTimerExpired, eventSpecificInformationTariff = *)
Pass criteria	L1?EventNotificationCharging invoke(monitorMode=interrupted, legID=1, eventTypeTariff= chargingAcknowledgementTimerExpired, eventSpecificInformationTariff = *)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RN_BV_09
Work item no.:	ITEM_BASIC_364
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Waiting for instructions" FSM for CS state (before the called B-party has received the call), containing two ChargingEvent specifications, indicating (monitorMode=notifyAndContinue, legID=2, eventTypeTariff = chargingTariffInformation) and (monitorMode=notifyAndContinue, legID=2, eventTypeTariff = stopCharging), sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=notifyAndContinue, legID=2, eventTypeTariff= chargingTariffInformation, eventSpecificInformationTariff = crgt), after having received an ApplicationTransfer message from SigConB containing ChargingTariffInformation(subscriber-charge). Verify also that the SSF sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging, eventSpecificInformationTariff = stop), after having received an ApplicationTransfer message (StopCharging) from SigConB.
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=notifyAndContinue, legID=2, eventTypeTariff = chargingTariffInformation), (monitorMode=notifyAndContinue, legID=2, eventTypeTariff = stopCharging)) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= chargingTariffInformation, eventSpecificInformationTariff = crgt) NOTE: APM(ChargingAcknowledgementInformation(accepted)) and EventNotificationCharging may be received in any order. CP1_2!SetupConf (This indicates start of charging) CP1_1?SetupResp Wait a while CP1_2!APM(StopCharging) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging, eventSpecificInformationTariff = stop) NOTE: APM(ChargingAcknowledgementInformation(accepted)) and EventNotificationCharging invoke may be received in any order.
Pass criteria	L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= chargingTariffInformation, eventSpecificInformationTariff = crgt) L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= stopCharging, eventSpecificInformationTariff = stop)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RN_BV_10
Work item no.:	ITEM_BASIC_365
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Waiting for instructions" FSM for CS state (before the called B-party has received the call), containing two ChargingEvent specifications, indicating (monitorMode=notifyAndContinue, legID=2, eventTypeTariff = chargingTariffInformation) and (monitorMode=interrupted, legID=2, eventTypeTariff = stopCharging), sends to the SCF an EventNotificationCharging invoke component containing (monitorMode=notifyAndContinue, legID=2, eventTypeTariff= chargingTariffInformation, eventSpecificInformationTariff = crgt), after having received an ApplicationTransfer message from SigConB containing ChargingTariffInformation(subscriber-charge). Verify also that the SSF does not send to the SCF an EventNotificationCharging invoke component when having received from the SCF a RequestNotificationChargingEvent invoke component containing one ChargingEvent specification indicating monitorMode=transparent, legID=2 and eventTypeTariff = stopCharging, **before** receiving from SigConB an ApplicationTransfer message (StopCharging).
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Preamble:	O_OSA_RNC
Selection Cond.	
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=notifyAndContinue, legID=2, eventTypeTariff = chargingTariffInformation), (monitorMode=interrupted, legID=2, eventTypeTariff = stopCharging)) L1!Connect(2,1) CP1_2?SetUpReq CP1_2!APM(ChargingTariffInformation(subscriber-charge,immediate-tariff-change-with-restart,start-of-tariffing,CHARGING_TARIFF)) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= chargingTariffInformation, eventSpecificInformationTariff = crgt) NOTE: APM(ChargingAcknowledgementInformation(accepted)) and EventNotificationCharging may be received in any order. CP1_2!SetupConf (This indicates start of charging) CP1_1?SetupResp Wait a while L1!RequestNotificationChargingEvent invoke((monitorMode=transparent, legID=2, eventTypeTariff = stopCharging)) CP1_2!APM(StopCharging) CP1_2?APM(ChargingAcknowledgementInformation(accepted)) No EventNotificationCharging invoke received
Pass criteria	L1?EventNotificationCharging invoke(monitorMode=notifyAndContinue, legID=2, eventTypeTariff= chargingTariffInformation, eventSpecificInformationTariff = crgt) No EventNotificationCharging invoke received after APM(StopCharging)
Postamble:	ReleaseCallA2

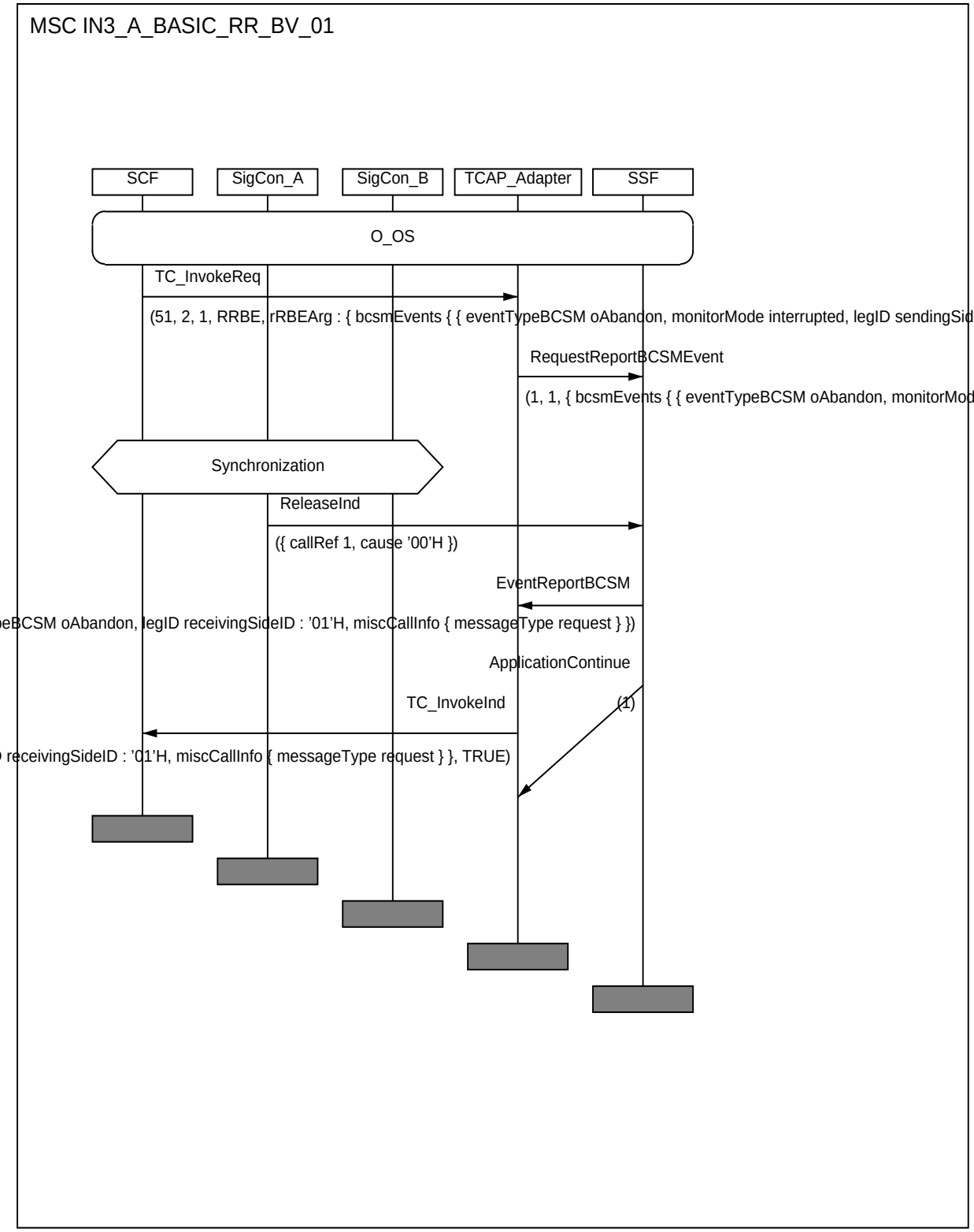
	IN3_A_BASIC_RN_BI_01
Work item no.:	ITEM_BASIC_366
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component without mandatory parameter "monitorMode", sends to the SCF aRequestNotificationChargingEvent returnError component with errorCode "missingParameter".
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Selection Cond.	
Preamble:	O_OSA_RNC
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=OMIT, legID=2, eventTypeTariff = stopCharging)) L1?RequestNotificationChargingEvent returnError(missingParameter)
Pass criteria	L1?RequestNotificationChargingEvent returnError(missingParameter)
Postamble:	ReleaseCallA

	IN3_A_BASIC_RN_BI_02
Work item no.:	ITEM_BASIC_367
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component with parameter "legID" containing the number of a non-existing leg, sends to the SCF aRequestNotificationChargingEvent returnError component with errorCode "unknownLegID".
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Selection Cond.	
Preamble:	O_OSA_RNC
Test description	L1!FurnishChargingInformation invoke(tariffFromSuccExchange = takeIntoAccountTranslateNoAOC, iINRecordIndicators = FCI_IN_CRI1, partyToCharge = FCI_PARTY_TO_CHARGE) L1!RequestNotificationChargingEvent invoke((monitorMode=interrupted, legID=3, eventTypeTariff = stopCharging)) L1?RequestNotificationChargingEvent returnError(unknownLegID)
Pass criteria	L1?RequestNotificationChargingEvent returnError(unknownLegID)
Postamble:	ReleaseCallA

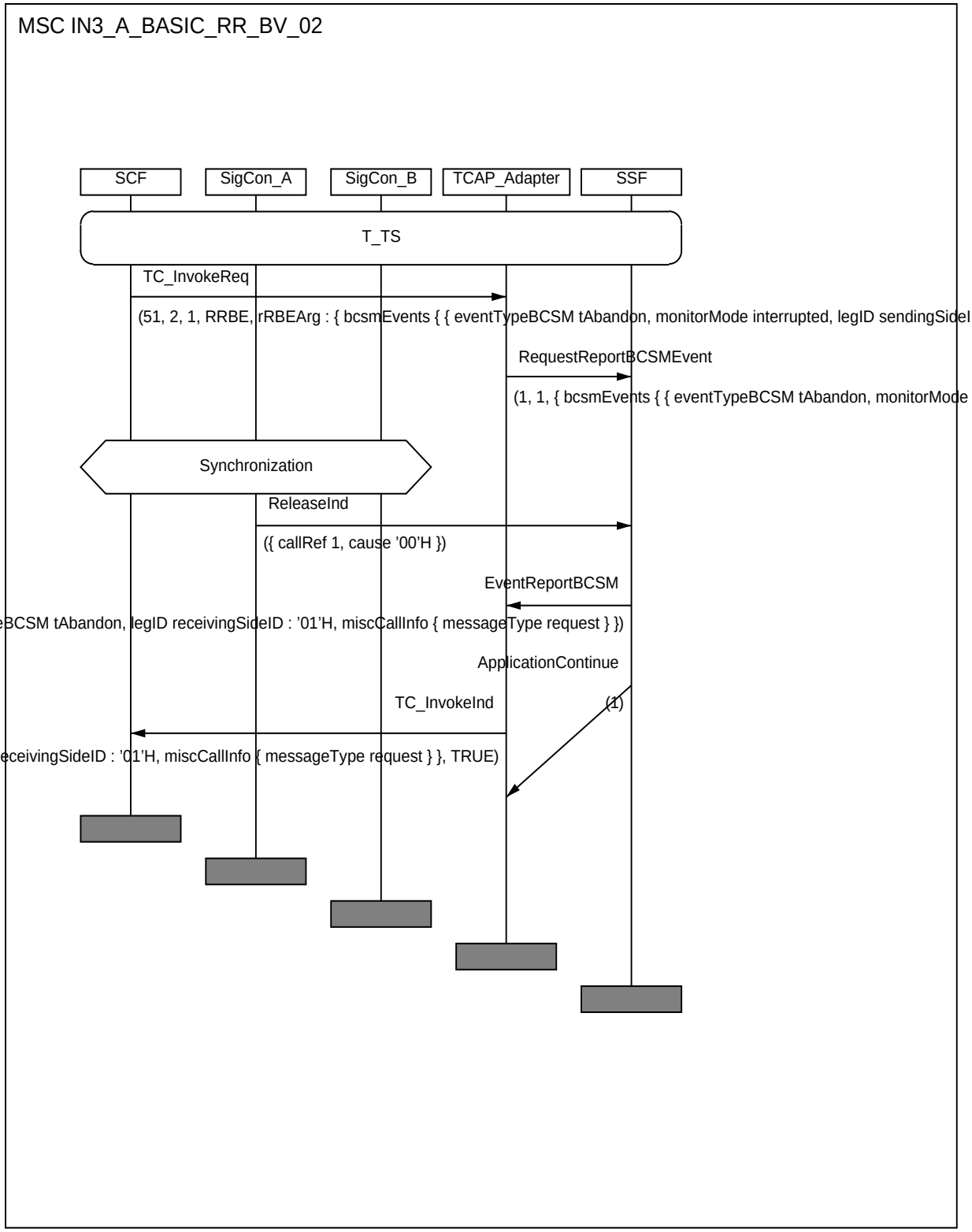
	IN3_A_BASIC_RN_BO_01
Work item no.:	ITEM_BASIC_368
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having received from the SCF a RequestNotificationChargingEvent invoke component in the "Idle" state, sends to the SCF aRequestNotificationChargingEvent returnError component with errorCode "unexpectedComponentSequence".
Requirements refs	6.6.3.3.2.1, 8.2.1.2, 8.2.2, 8.2.2, 11.23, 11.37.1, 11.37, 12.59, 12.61, 12.63, 12.64, 16, 7
Selection Cond.	
Preamble:	None
Test description	L1!RequestNotificationChargingEvent invoke((monitorMode=interrupted, legID=3, eventTypeTariff = stopCharging)) L1!TCAP-BEGIN L1?RequestNotificationChargingEvent returnError(unexpectedComponentSequence)
Pass criteria	L1?RequestNotificationChargingEvent returnError(unexpectedComponentSequence)
Postamble:	None

6.6.22 RequestReportBCSMEvent (RR) procedure

	IN3_A_BASIC_RR_BV_01
Work item no.:	ITEM_BASIC_84
IN2 Ref(tmp)	IN2_A_BASIC_RR_CA_01
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oAbandon**, if the O_Abandon DP has been armed (monitorMode=interrupted) and the calling party abandons the call before it is answered.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oAbandon - monitorMode=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


MSC IN3_A_BASIC_RR_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAbandon, monitorMode interrupted, legID sendingSideID : '01'H } } }, short
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oAbandon, monitorMode interrupted, legID s
Synchronization
ReleaseInd
({ callRef 1, cause '00'H })
EventReportBCSM
1, { eventTypeBCSM oAbandon, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
oAbandon, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)

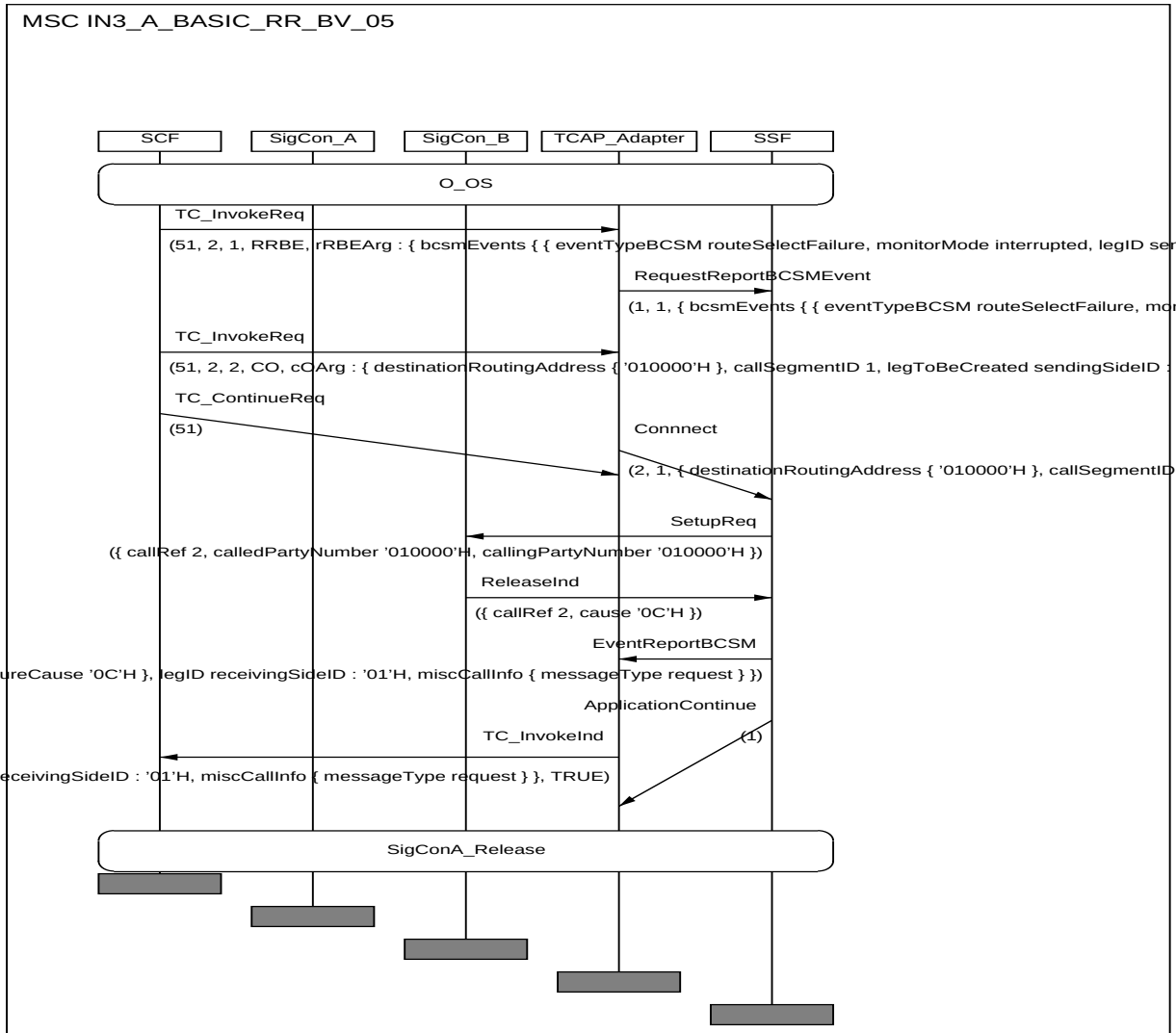
	IN3_A_BASIC_RR_BV_02
Work item no.:	ITEM_BASIC_85
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_01
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tAbandon**, if the T_Abandon DP has been armed (monitorMode=interrupted) and the calling party abandons the call before it is answered.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tAbandon - monitorMode=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 IN3_A_BASIC_RR_BV_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM tAbandon, monitorMode interrupted, legID sendingSideID : '01'H } } }, short
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM tAbandon, monitorMode interrupted, legID s
Synchronization
ReleaseInd
({ callRef 1, cause '00'H })
EventReportBCSM
1, { eventTypeBCSM tAbandon, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
M tAbandon, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)

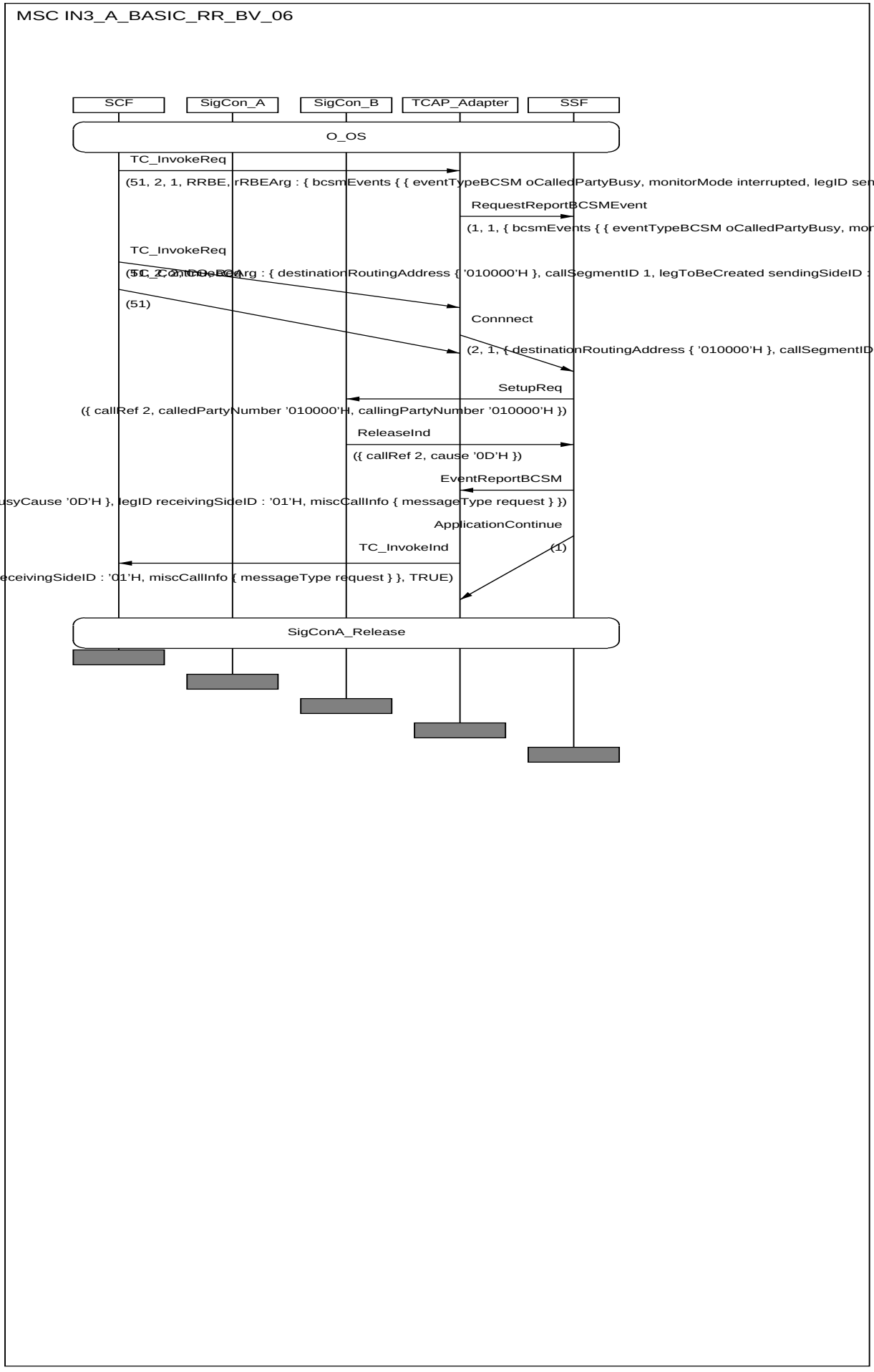
	IN3_A_BASIC_RR_BV_03
Work item no.:	ITEM_BASIC_86
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_02 (see also IN3_A_BASIC_CI_CA_01)
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **collectedInfo**, when the Collected_Information DP has been armed (monitorMode=interrupted), the SCF has sent a **CollectInformation** operation and the calling party sends the remaining digits to complete the call information.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.11.1, 11.11.1.1.1, 11.11.3.1, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS_ColIn Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=collectedInfo - monitorMode=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

	IN3_A_BASIC_RR_BV_04
Work item no.:	ITEM_BASIC_87
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_03
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **analysedInfo**, when the Analysed_Information DP has been armed (monitorMode=interrupted) and the calling party sends the remaining digits to complete the call information.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=analysedInfo - monitorMode=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

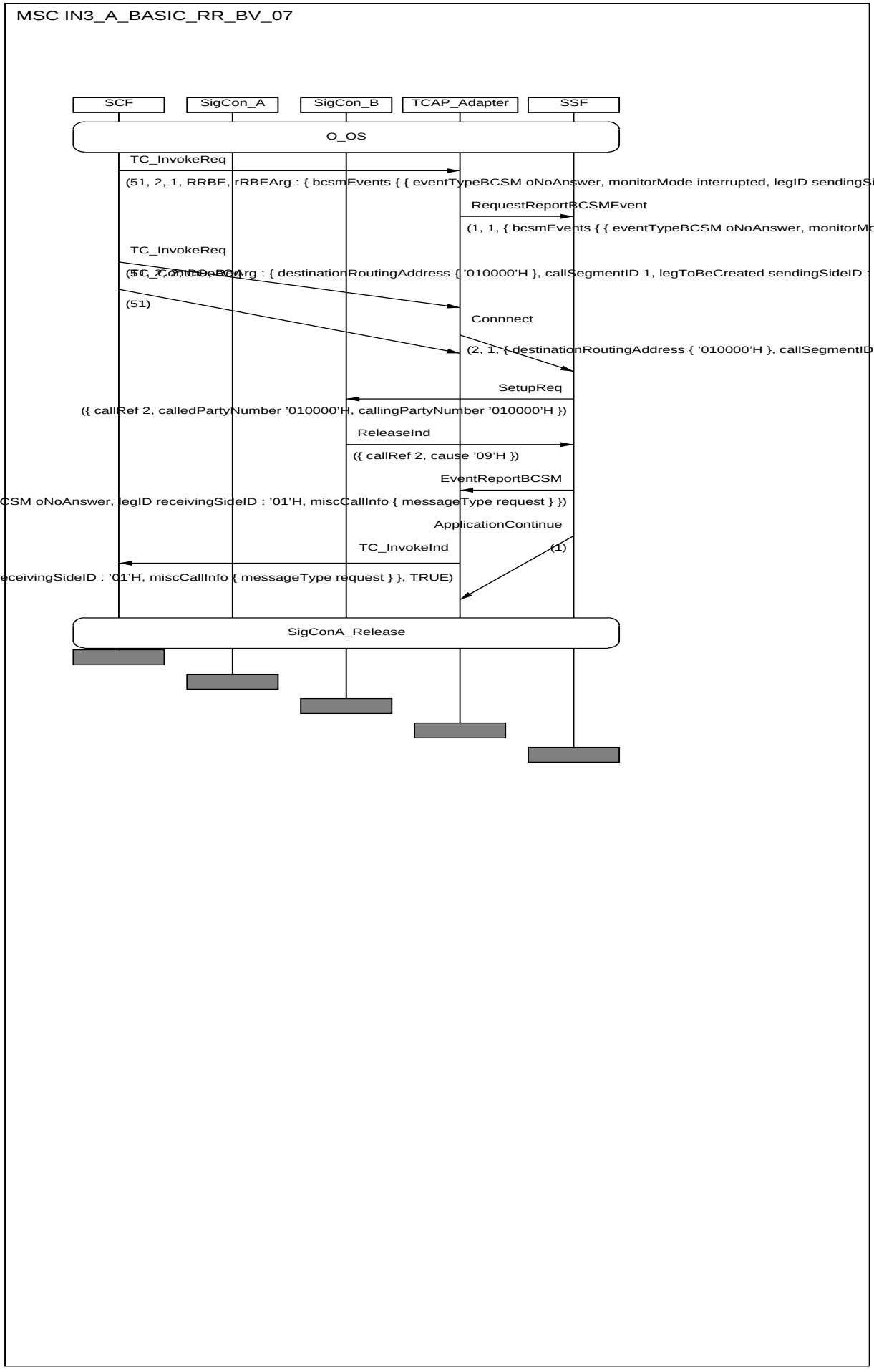
	IN3_A_BASIC_RR_BV_05
Work item no.:	ITEM_BASIC_88
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_04
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **routeSelectFailure**, when the Route_Select_Failure DP has been armed (monitorMode=interrupted), the SCF has sent a **Connect** invoke component to request the connection of the call to SigConB, but SigConB releases the call **indicating** routeFailure2.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=routeSelectFailure - monitorMode=interrupted followed by a **Connect** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call **(ReleaseInd)** because of error: routeFailure2 ("0C"H)
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** routeSelectFailure
Postamble:	SigConA_Release


MSC IN3_A_BASIC_RR_BV_05
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode interrupted, legID sendingSideID : '01'H }
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode interrupted
TC_InvokeReq
(51, 2, 2, CO, cOArg : { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
TC_ContinueReq
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '0C'H })
EventReportBCSM
reSpecificInfo : { failureCause '0C'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
(1)
TC_InvokeInd
ause '0C'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)
SigConA_Release

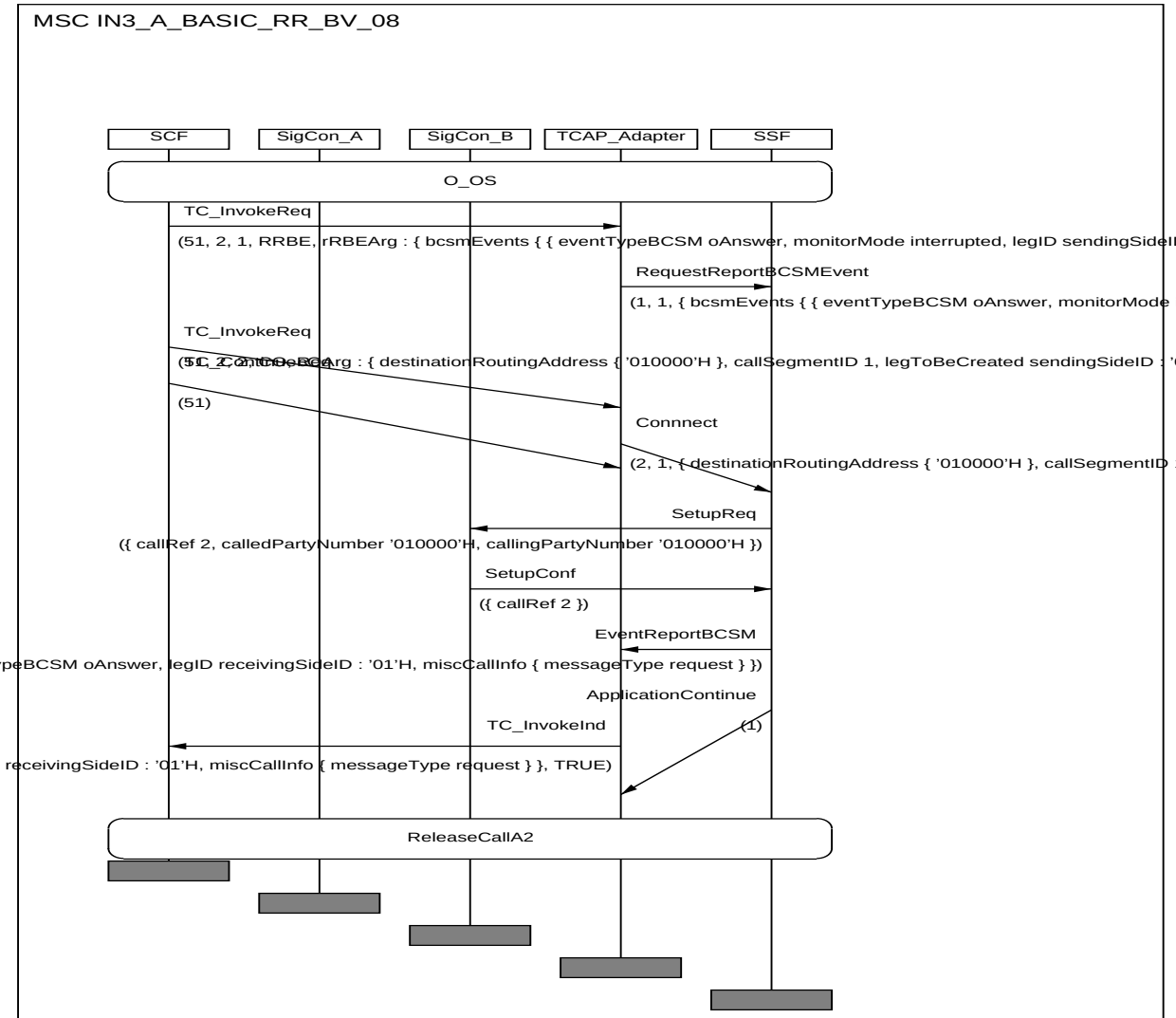
	IN3_A_BASIC_RR_BV_06
Work item no.:	ITEM_BASIC_89
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_05
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oCalledPartyBusy**, when the O_Called_Party_Busy DP has been armed (monitorMode=interrupted), the SCF has sent a **Connect** invoke component to request the connection of the call to SigConB, but SigConB releases the call **indicating** bPtyBusy_UDUB.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oCalledPartyBusy - monitorMode=interrupted followed by a **Connect** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call **(ReleaseInd)** with bPtyBusy_UDUB (cause "0D"H)
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** oCalledPartyBusy
Postamble:	SigConA_Release


MSC IN3_A_BASIC_RR_BV_06
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted, legID sendingSideID : '01'H } }
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode interrupted
TC_InvokeReq
(51, 2, 1, CRBE, cRBEArg : { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '0D'H })
EventReportBCSM
usySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
ause '0D'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)
SigConA_Release

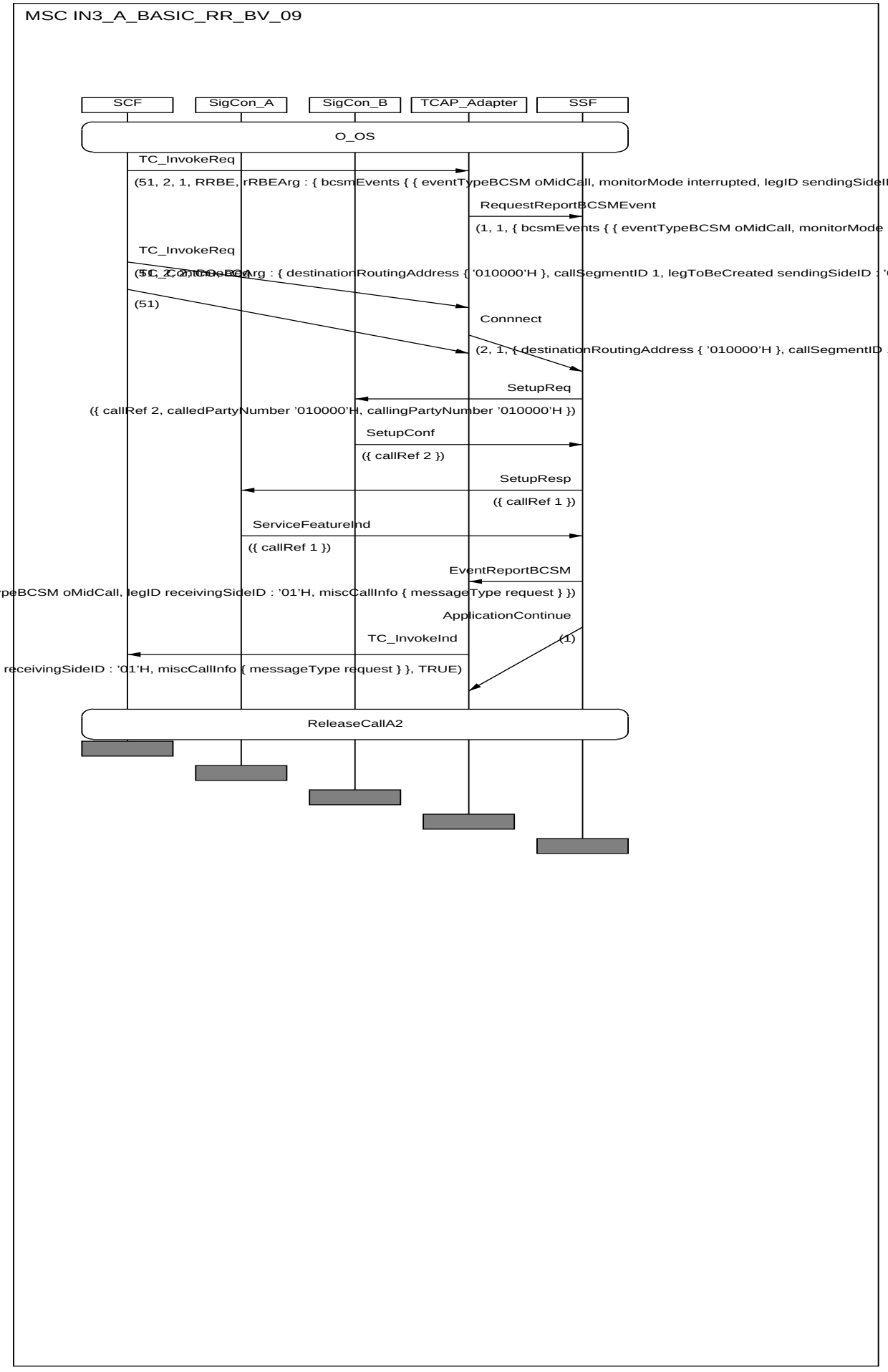
	IN3_A_BASIC_RR_BV_07
Work item no.:	ITEM_BASIC_90
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_06
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oNoAnswer**, when the O_No_Answer DP has been armed (monitorMode=interrupted), the SCF has sent a **Connect** invoke component to request the connection of the call to SigConB, but the SSF releases the call because SigConB does not answer.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oNoAnswer - monitorMode=interrupted followed by a **Connect** invoke Then SSF sends a SetupReq to SigCon B and releases the call **(ReleaseInd)** because SigConB does not answer
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** oNoAnswer
Postamble:	SigConA_Release


MSC IN3_A_BASIC_RR_BV_07
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID sendingSideID : '01'H } } }, sho
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode interrupted, legID
TC_InvokeReq
(51, 2, 1, Connect, connectArg : { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '09'H })
EventReportBCSM
1, { eventTypeBCSM oNoAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
oNoAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)
SigConA_Release

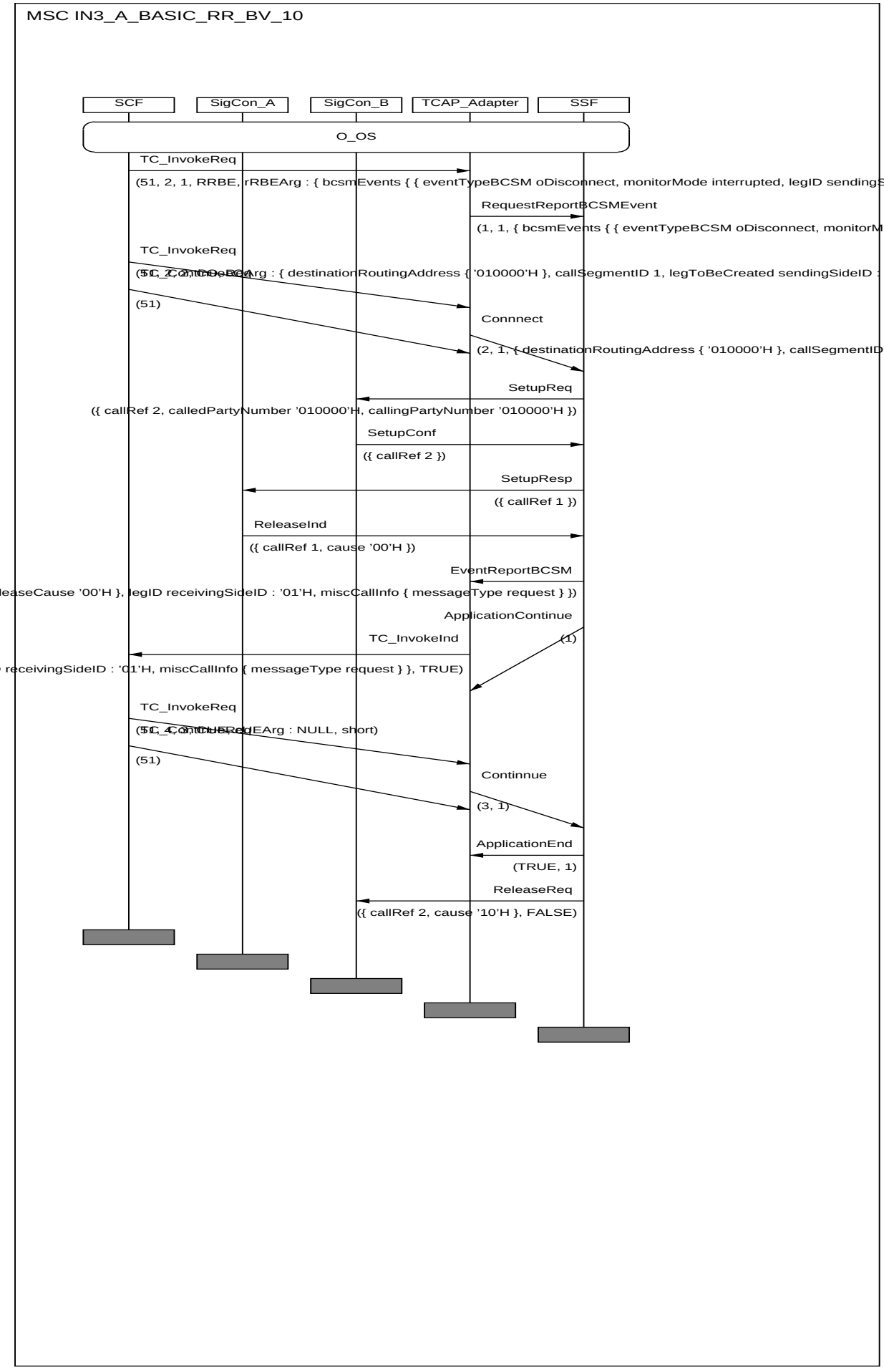
	IN3_A_BASIC_RR_BV_08
Work item no.:	ITEM_BASIC_91
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_07
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oAnswer**, when the O_Answer DP has been armed (monitorMode=interrupted), the SCF has sent a **Connect** invoke component to request the connection of the call to SigConB, and SigConB answers the call.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oAnswer - monitorMode=interrupted followed by a **Connect** invoke Then SSF sends a SetupReq to SigCon B SigCon B answers the call (SetupConf from SigConB to SSF)
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** oAnswer
Postamble:	ReleaseCallAB_cause_00


MSC IN3_A_BASIC_RR_BV_08
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID sendingSideID : '01'H } } }, short
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID se
TC_InvokeReq
destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
1, { eventTypeBCSM oAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
(1)
TC_InvokeInd
M oAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)
ReleaseCallA2

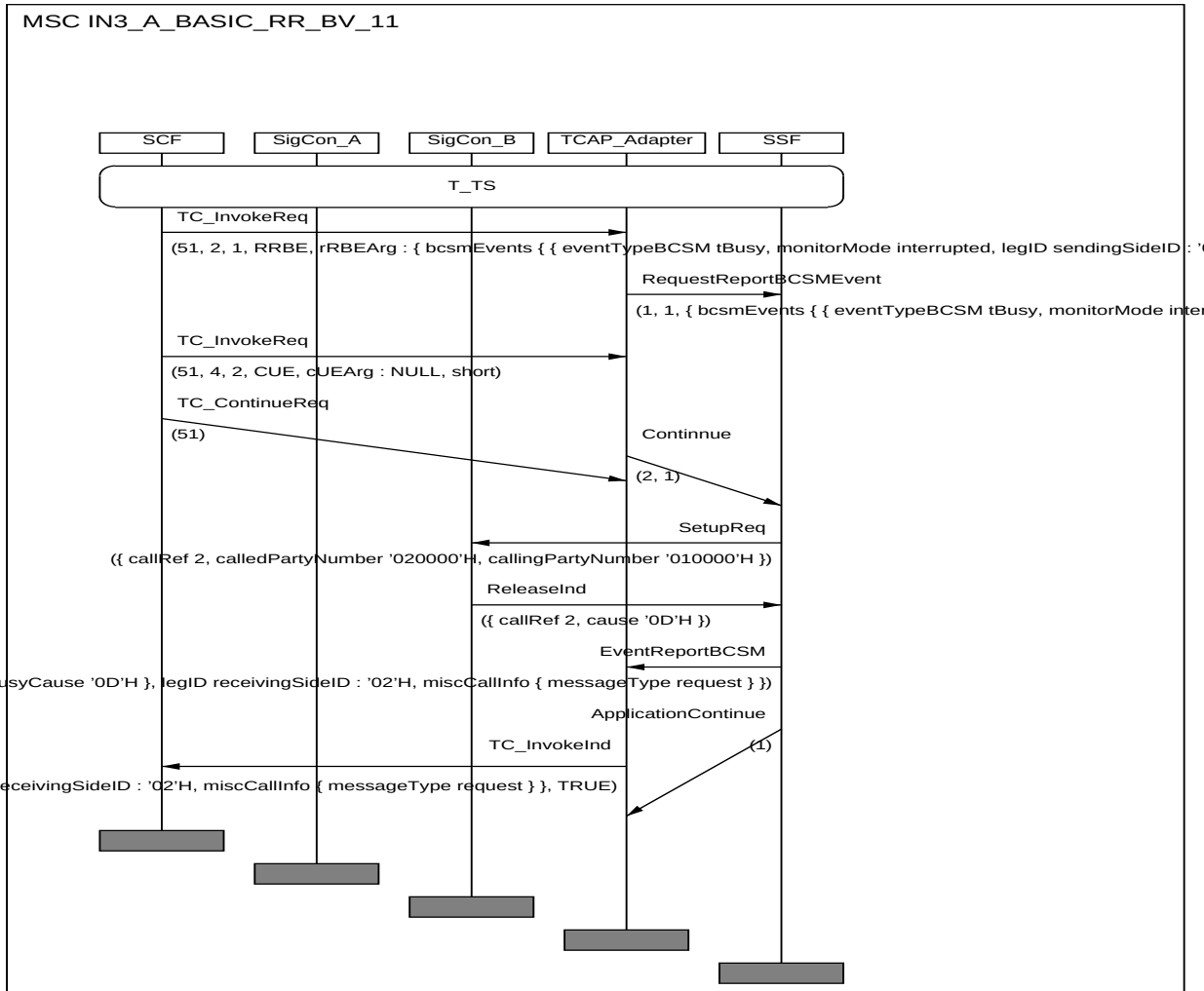
	IN3_A_BASIC_RR_BV_09
Work item no.:	ITEM_BASIC_92
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_08
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oMidCall**, when the O_Mid_Call DP has been armed (monitorMode=interrupted), SigConA and SigConB have been connected, and SigConA initiates a service (ServiceFeatureInd sent to the SSF).
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oMidCall - monitorMode=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 IN3_A_BASIC_RR_BV_09
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oMidCall, monitorMode interrupted, legID sendingSideID : '01'H } } }, short
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oMidCall, monitorMode interrupted, legID se
TC_InvokeReq
(51, 2, Connect, connectArg : { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ServiceFeatureInd
({ callRef 1 })
EventReportBCSM
1, { eventTypeBCSM oMidCall, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
M oMidCall, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)
ReleaseCallA2

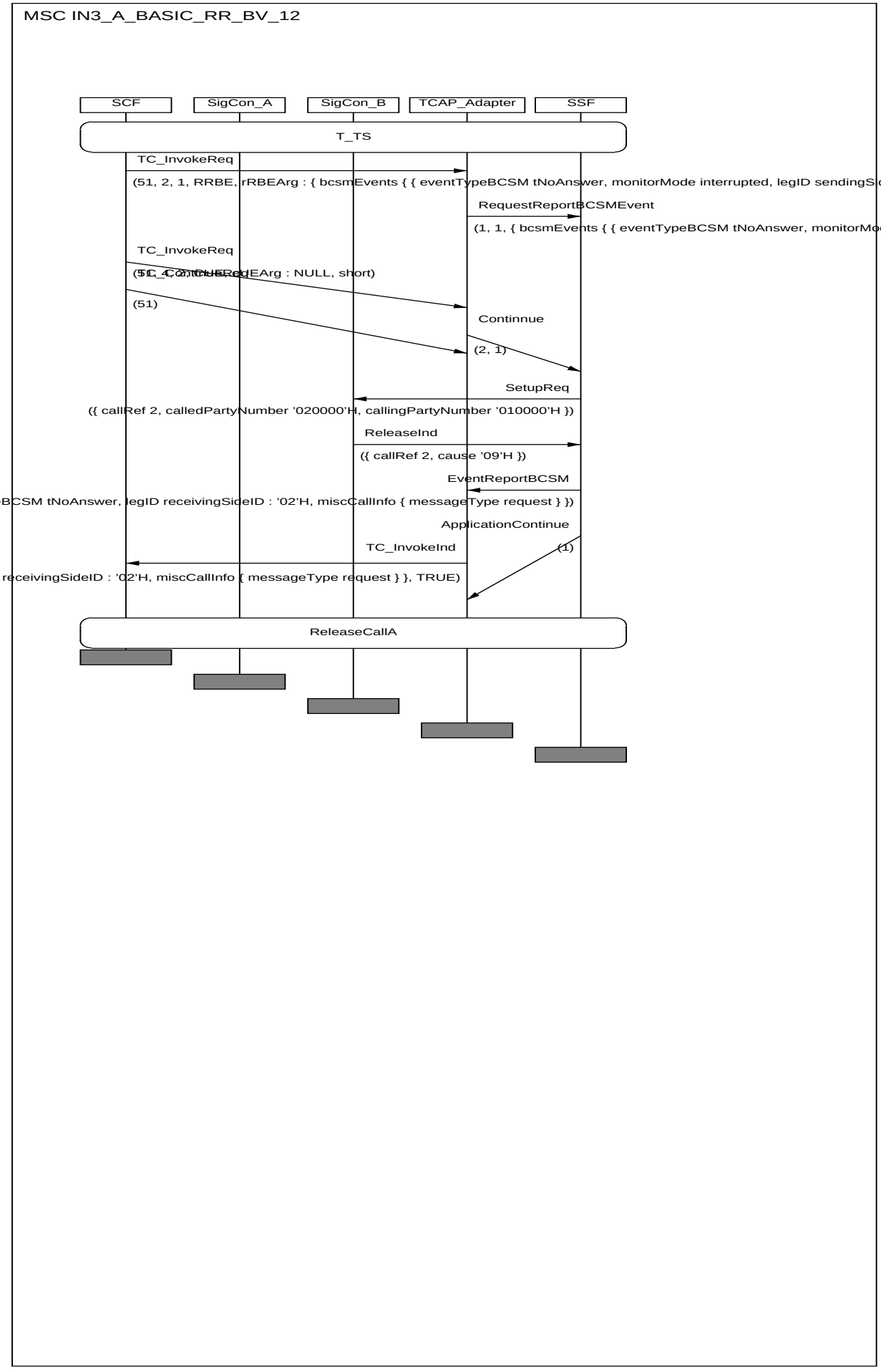
	IN3_A_BASIC_RR_BV_10
Work item no.:	ITEM_BASIC_93
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_09
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oDisconnect**, when the O_Disconnect DP has been armed (monitorMode=interrupted), SigConA and SigConB have been connected, and SigConA then clears the call. Check also that SigConB receives a ReleaseReq.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oDisconnect - monitorMode=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 IN3_A_BASIC_RR_BV_10
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID
TC_InvokeReq
(51, 2, TC_InvokeReq, CC, destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ReleaseInd
({ callRef 1, cause '00'H })
EventReportBCSM
tSpecificInfo : { releaseCause '00'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
(1)
TC_InvokeInd
ause '00'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)
TC_InvokeReq
(51, 4, TC_Continue, CC, rRBEArg : NULL, short)
(51)
Continnue
(3, 1)
ApplicationEnd
(TRUE, 1)
ReleaseReq
({ callRef 2, cause '10'H }, FALSE)

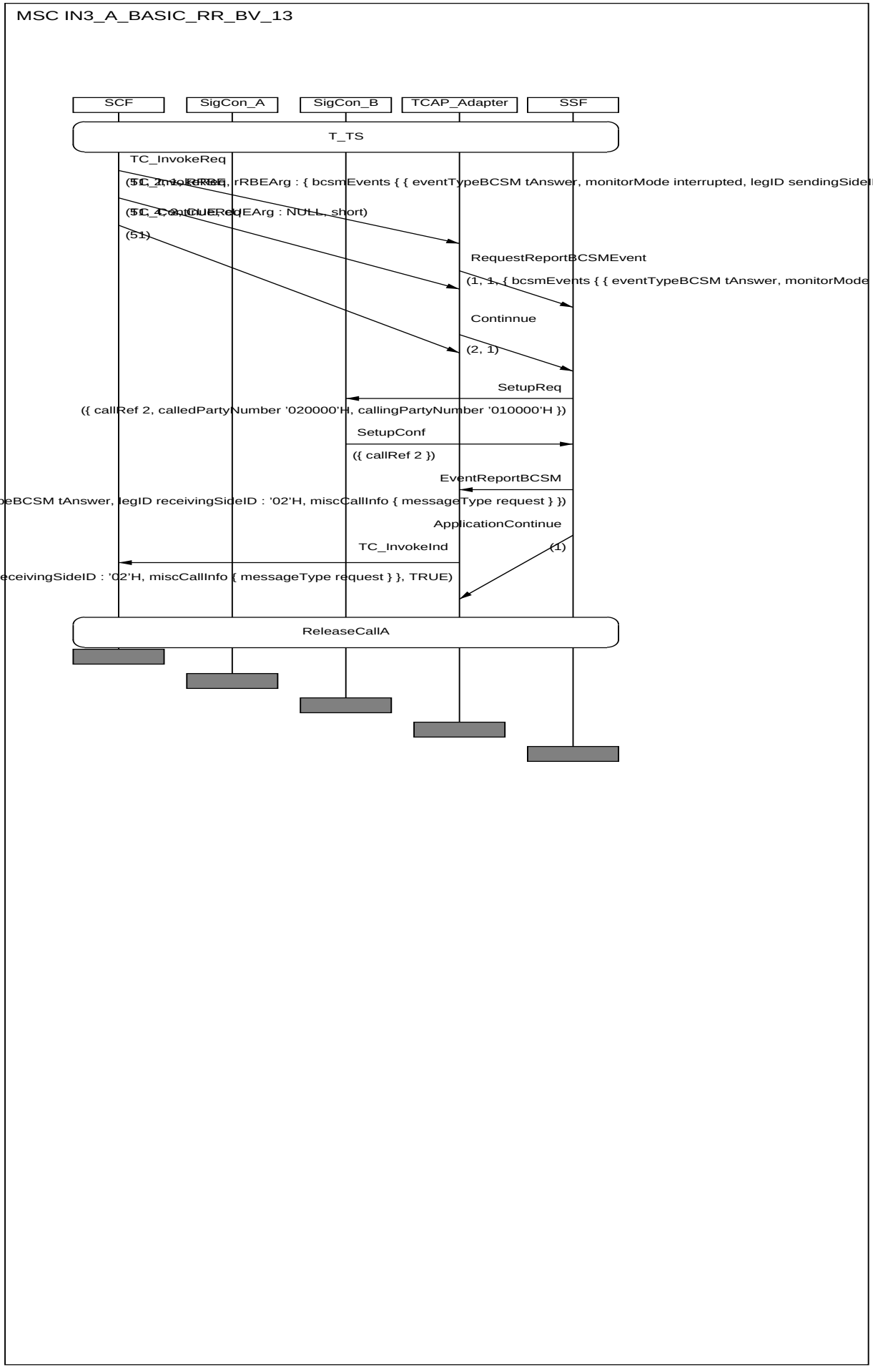
	IN3_A_BASIC_RR_BV_11
Work item no.:	ITEM_BASIC_94
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_10
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tBusy**, when the T_Busy DP has been armed (monitorMode=interrupted), and SigConB releases theSetupReq sent from the SSF with the indication of bPtyBusy_UDUB.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg2) - eventTypeBCSM=tBusy - monitorMode=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 IN3_A_BASIC_RR_BV_11
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM tBusy, monitorMode interrupted, legID sendingSideID : '02'H } } }, short
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM tBusy, monitorMode interrupted, legID sendi
TC_InvokeReq
(51, 4, 2, CUE, cUEArg : NULL, short)
TC_ContinueReq
(51)
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '0D'H })
EventReportBCSM
usySpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
(1)
TC_InvokeInd
ause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)

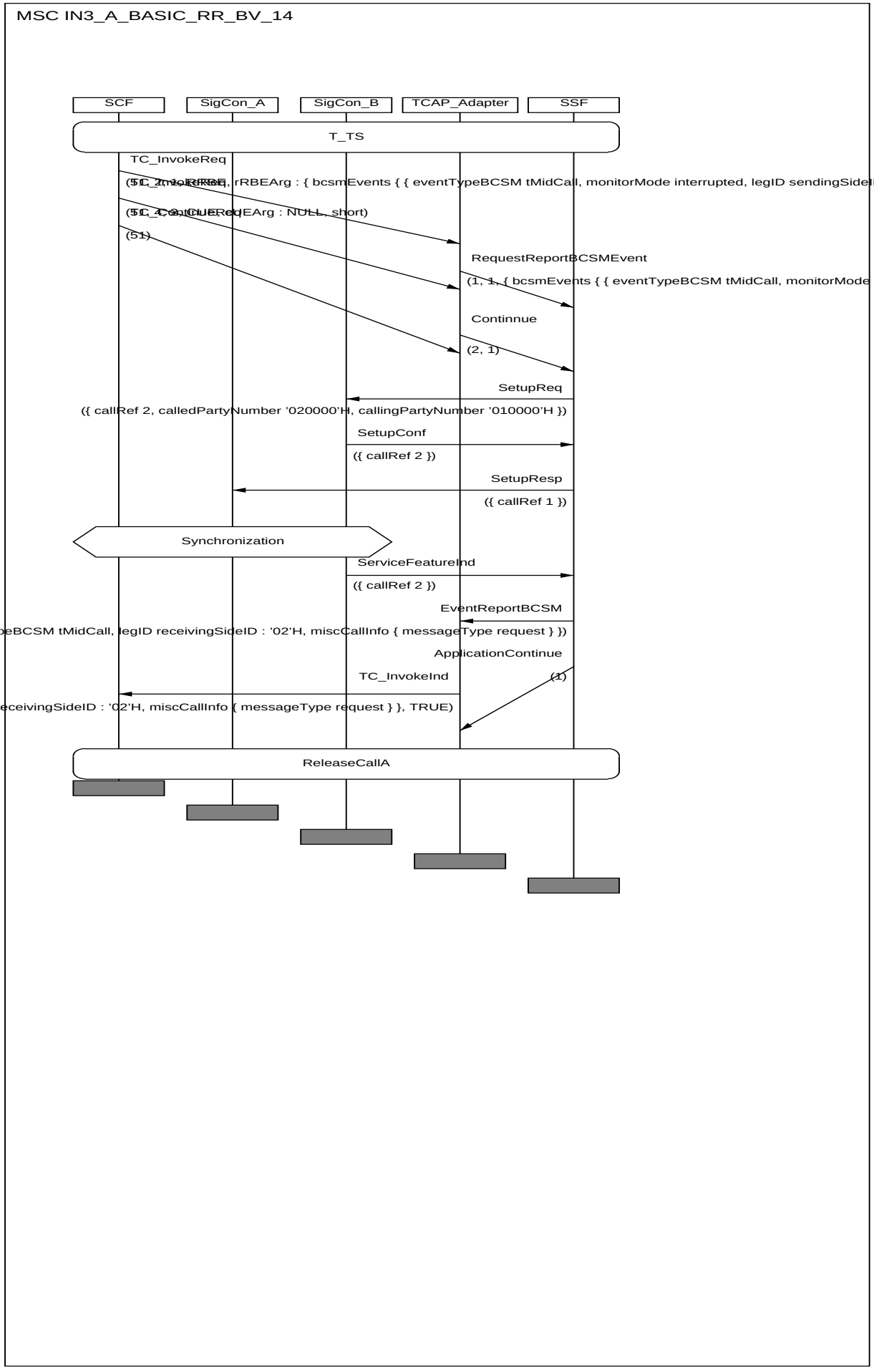
	IN3_A_BASIC_RR_BV_12
Work item no.:	ITEM_BASIC_95
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_11
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tNoAnswer**, when the T_No_Answer DP has been armed (monitorMode=interrupted), and SigConB releases theSetupReq sent from the SSF because SigConB does not answer.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg2) - eventTypeBCSM=tNoAnswer - monitorMode=interrupted followed by a **Continue** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call (**ReleaseInd** sent) because SigConB does not answer
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** tNoAnswer
Postamble:	ReleaseCallA


MSC IN3_A_BASIC_RR_BV_12
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM tNoAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, shor
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM tNoAnswer, monitorMode interrupted, legID
TC_InvokeReq
(51, 4, Continue, cEArg : NULL, short)
(51)
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '09'H })
EventReportBCSM
1, { eventTypeBCSM tNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
tNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
ReleaseCallA

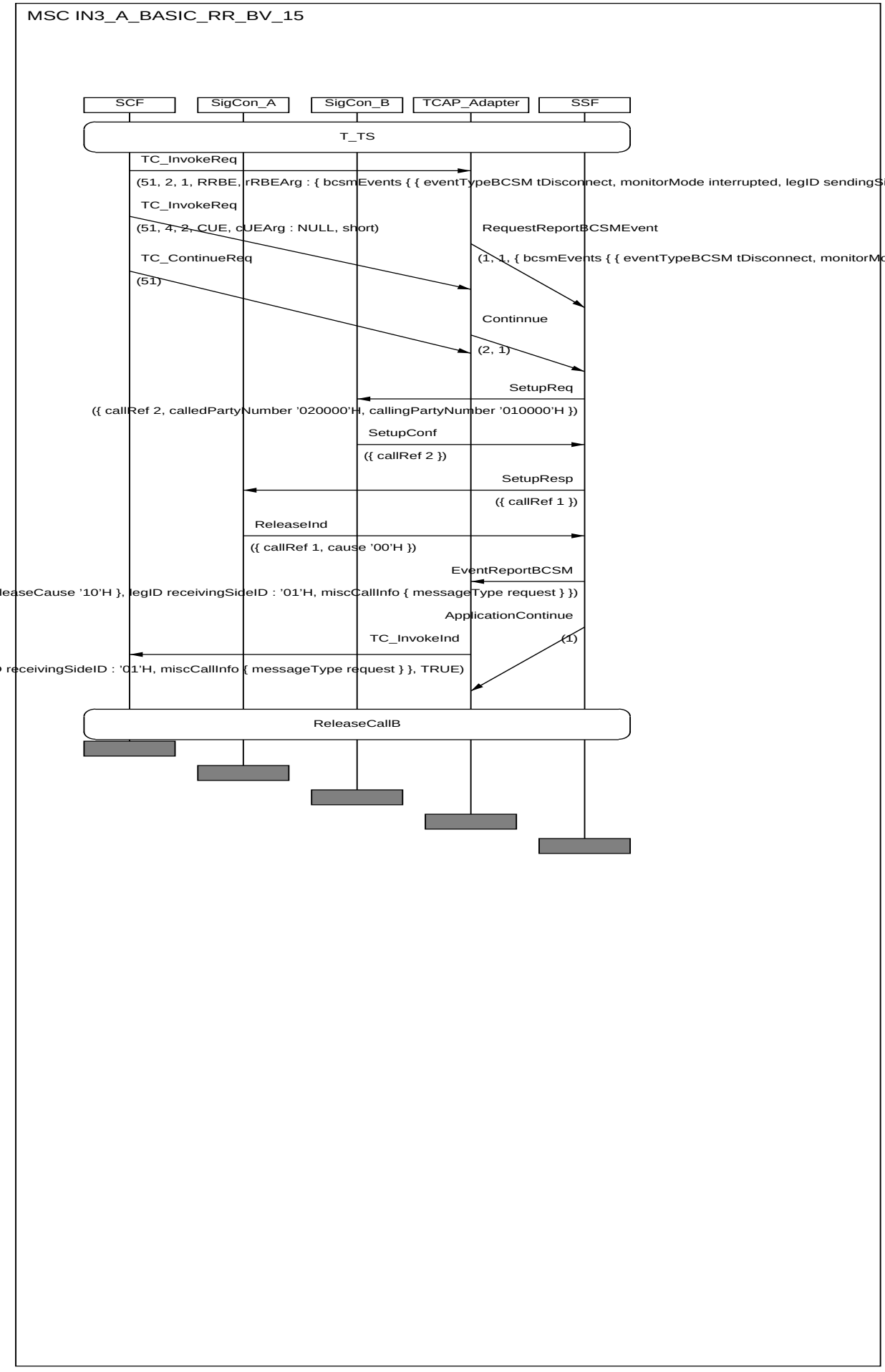
	IN3_A_BASIC_RR_BV_13
Work item no.:	ITEM_BASIC_96
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_12
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tAnswer**, when the T_Answer DP has been armed (monitorMode=interrupted), and SigConB confirms theSetupReq sent from the SSF.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg2) - eventTypeBCSM=tAnswer - monitorMode=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 IN3_A_BASIC_RR_BV_13
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, rRBEArg : { bcsmEvents { { eventTypeBCSM tAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, short
(51, 4, cUEArg : NULL, short)
(51)
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM tAnswer, monitorMode interrupted, legID se
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
1, { eventTypeBCSM tAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
SM tAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
ReleaseCallA

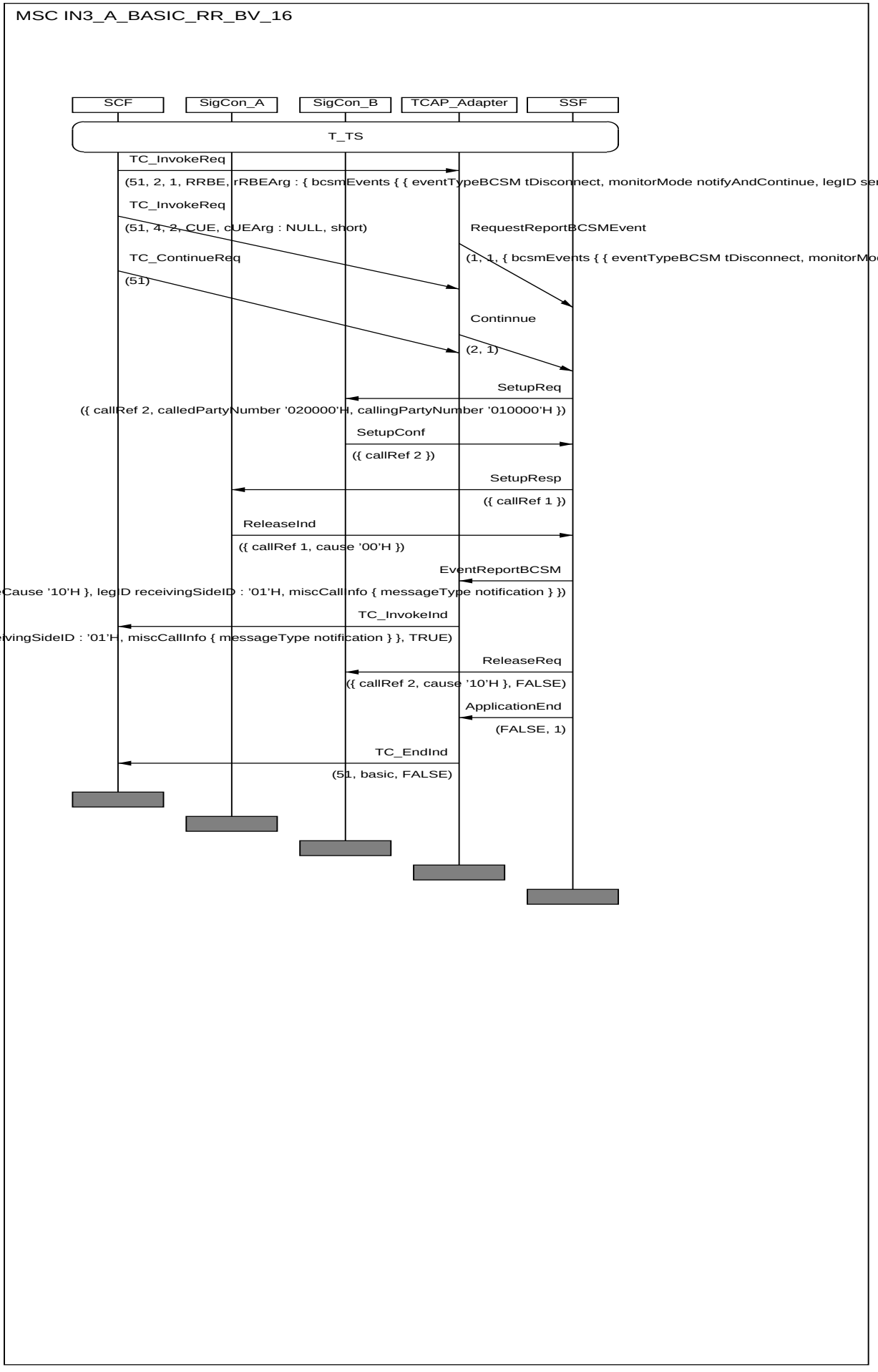
	IN3_A_BASIC_RR_BV_14
Work item no.:	ITEM_BASIC_97
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_13
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tMidCall**, when the T_Mid_Call DP has been armed (monitorMode=interrupted), SigConA and SigConB have been connected, and SigConB initiates a service (ServiceFeatureInd sent to the SSF).
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg2) - eventTypeBCSM= tMidCall - monitorMode=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 IN3_A_BASIC_RR_BV_14
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, rRBEArg : { bcsmEvents { { eventTypeBCSM tMidCall, monitorMode interrupted, legID sendingSideID : '02'H } } }, short
(51, 4, cUEArg : NULL, short)
(51)
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM tMidCall, monitorMode interrupted, legID se
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
Synchronization
ServiceFeatureInd
({ callRef 2 })
EventReportBCSM
1, { eventTypeBCSM tMidCall, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
SM tMidCall, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
ReleaseCallA

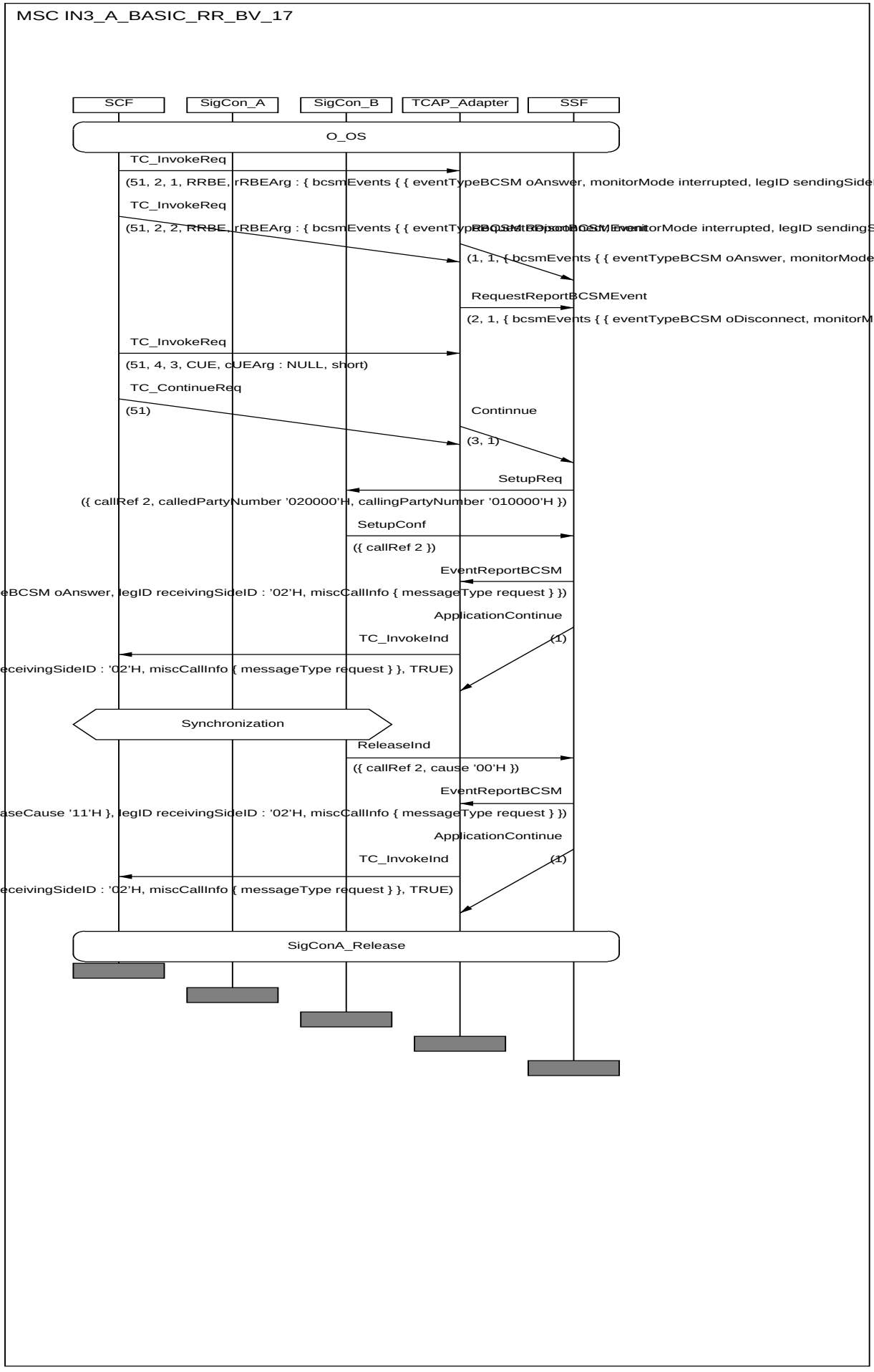
	IN3_A_BASIC_RR_BV_15
Work item no.:	ITEM_BASIC_98
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_14
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tDisconnect**, when the T_Disconnect DP has been armed (monitorMode=interrupted), SigConA and SigConB have been connected, and SigConA then clears the call.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg1) - eventTypeBCSM= tDisconnect - monitorMode=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 IN3_A_BASIC_RR_BV_15
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM tDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sho
TC_InvokeReq
(51, 4, 2, CUE, cUEArg : NULL, short)
RequestReportBCSMEvent
TC_ContinueReq
(1, 1, { bcsmEvents { { eventTypeBCSM tDisconnect, monitorMode interrupted, legID
(51)
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ReleaseInd
({ callRef 1, cause '00'H })
EventReportBCSM
tSpecificInfo : { releaseCause '10'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
ause '10'H }, legID receivingSideID : '01'H, miscCallInfo { messageType request } }, TRUE)
ReleaseCallB

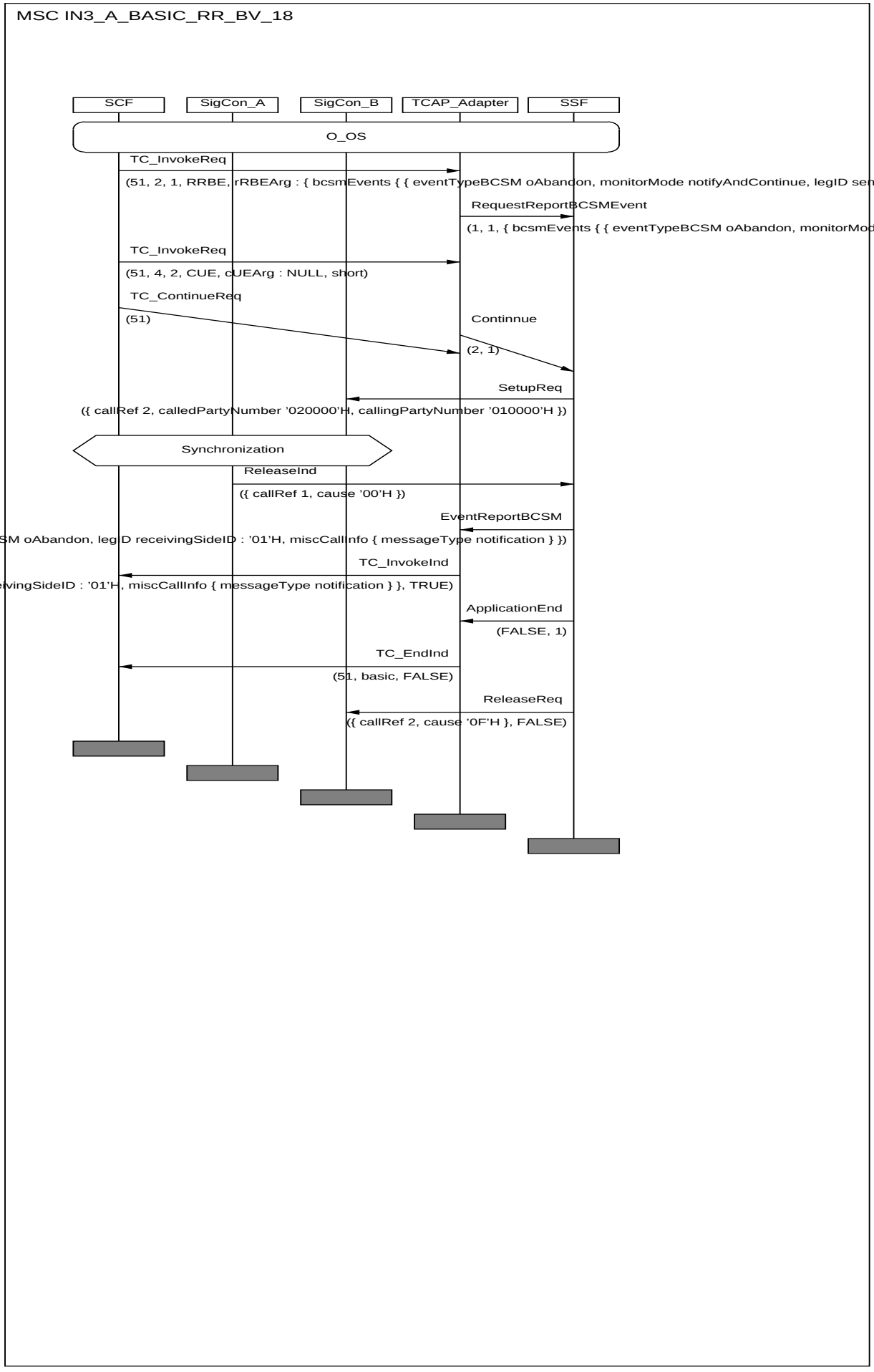
	IN3_A_BASIC_RR_BV_16
Work item no.:	ITEM_BASIC_99
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_15
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tDisconnect**, when the T_Disconnect DP has been armed (monitorMode=notifyAndContinue), SigConA and SigConB have been connected, and SigConA then clears the call.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg1) - eventTypeBCSM= tDisconnect - monitorMode=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 IN3_A_BASIC_RR_BV_16
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM tDisconnect, monitorMode notifyAndContinue, legID sendingSideID : '01'H }
TC_InvokeReq
(51, 4, 2, CUE, cUEArg : NULL, short)
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM tDisconnect, monitorMode notifyAndContinu
TC_ContinueReq
(51)
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ReleaseInd
({ callRef 1, cause '00'H })
EventReportBCSM
ecificInfo : { releaseCause '10'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } })
TC_InvokeInd
e '10'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } }, TRUE)
ReleaseReq
({ callRef 2, cause '10'H }, FALSE)
ApplicationEnd
(FALSE, 1)
TC_EndInd
(51, basic, FALSE)

	IN3_A_BASIC_RR_BV_17
Work item no.:	ITEM_BASIC_100
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_17
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oAnswer**, followed by an **EventReportBCSM** with the indication of eventTypeBCSM = **oDisconnect**, when the O_Answer and O_Disconnect DPs have been armed (monitorMode=interrupted), SigConA and SigConB are connected, and SigConB then clears the call.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF - SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg2) - eventTypeBCSM= oAnswer - monitorMode=interrupted - eventTypeBCSM= oDisconnect - monitorMode=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 IN3_A_BASIC_RR_BV_17
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID sendingSideID : '02'H } } }, short
TC_InvokeReq
(51, 2, 2, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '02'H } } }, sh
(1, 1, { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode interrupted, legID se
RequestReportBCSMEvent
(2, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(51, 4, 3, CUE, cUEArg : NULL, short)
TC_ContinueReq
(51)
Continnue
(3, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
1, { eventTypeBCSM oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
M oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
Synchronization
ReleaseInd
({ callRef 2, cause '00'H })
EventReportBCSM
tSpecificInfo : { releaseCause '11'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } })
ApplicationContinue
TC_InvokeInd
(1)
ause '11'H }, legID receivingSideID : '02'H, miscCallInfo { messageType request } }, TRUE)
SigConA_Release

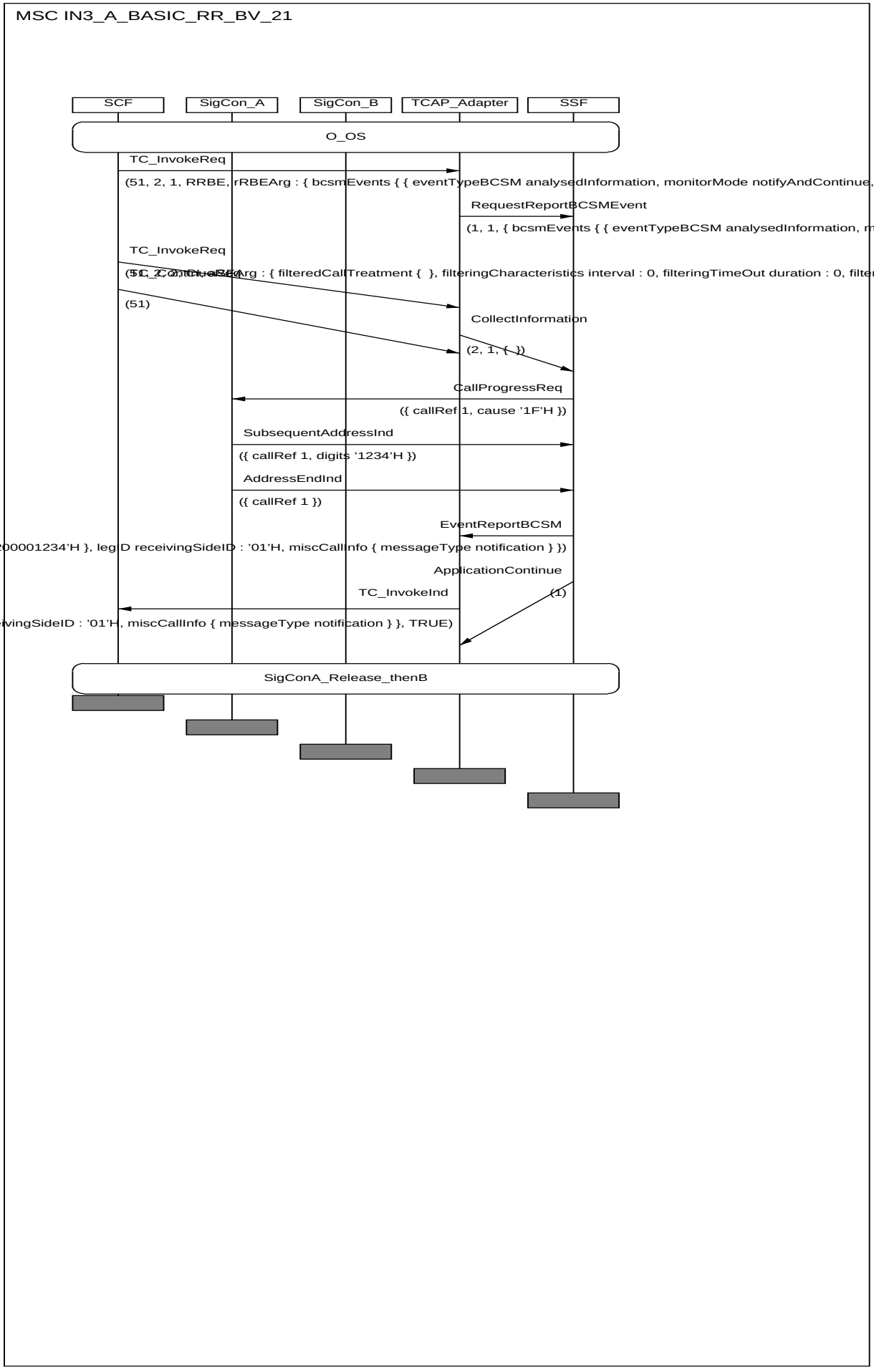
	IN3_A_BASIC_RR_BV_18
Work item no.:	ITEM_BASIC_101
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_18
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oAbandon**, if the O_Abandon DP has been armed (monitorMode=notifyAndContinue) and the calling party abandons the call before it is answered.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg1) - eventTypeBCSM=oAbandon - monitorMode=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


MSC IN3_A_BASIC_RR_BV_18
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAbandon, monitorMode notifyAndContinue, legID sendingSideID : '01'H } }
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oAbandon, monitorMode notifyAndContinue
TC_InvokeReq
(51, 4, 2, CUE, cUEArg : NULL, short)
TC_ContinueReq
(51)
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
Synchronization
ReleaseInd
({ callRef 1, cause '00'H })
EventReportBCSM
1, { eventTypeBCSM oAbandon, legID receivingSideID : '01'H, miscCallInfo { messageType notification } })
TC_InvokeInd
Abandon, legID receivingSideID : '01'H, miscCallInfo { messageType notification } }, TRUE)
ApplicationEnd
(FALSE, 1)
TC_EndInd
(51, basic, FALSE)
ReleaseReq
({ callRef 2, cause '0F'H }, FALSE)

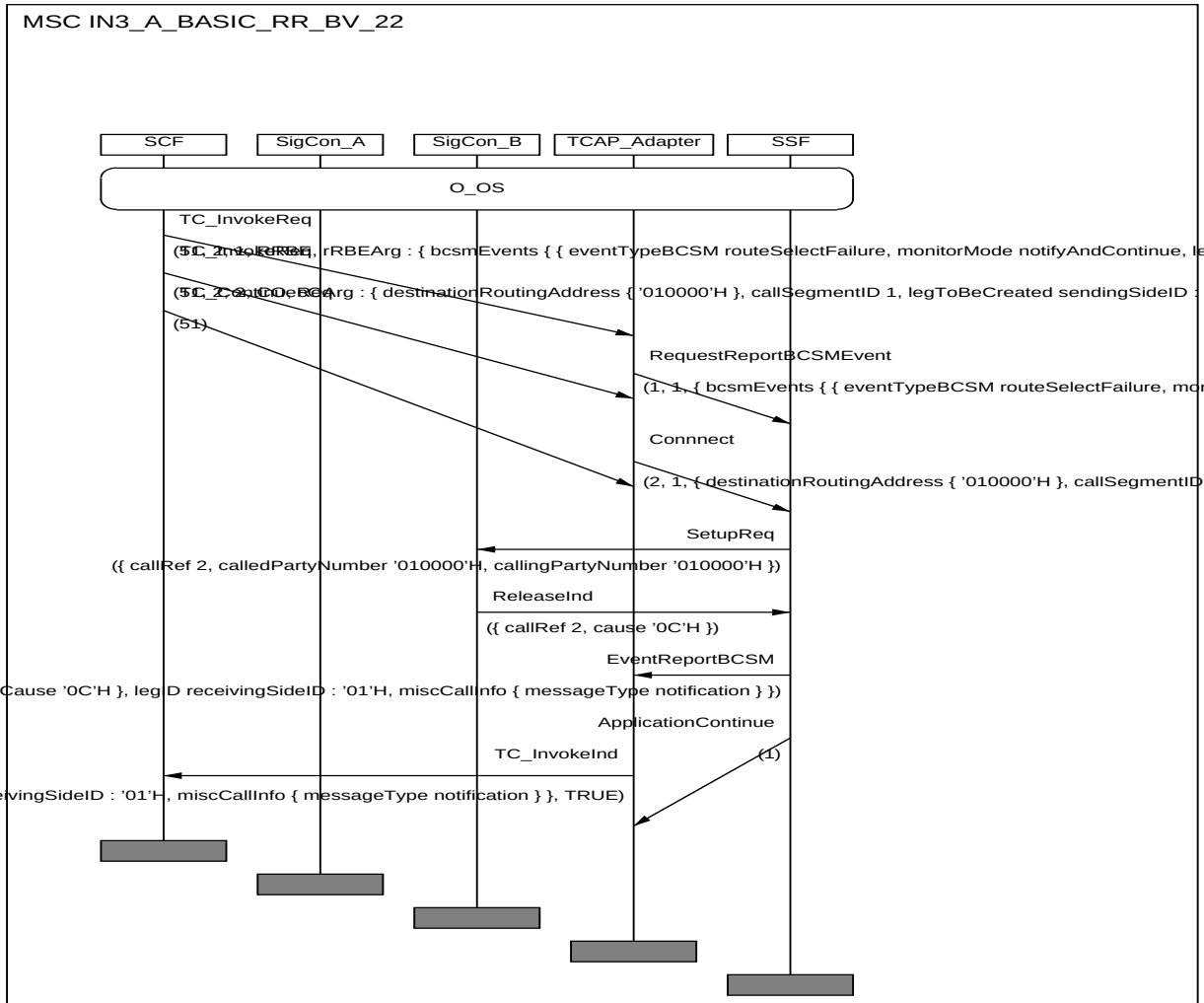
	IN3_A_BASIC_RR_BV_19
Work item no.:	ITEM_BASIC_102
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_19
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tAbandon**, if the T_Abandon DP has been armed (monitorMode=notifyAndContinue) and the calling party abandons the call before it is answered.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tAbandon - monitorMode=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

	IN3_A_BASIC_RR_BV_20
Work item no.:	ITEM_BASIC_103
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_20
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **collectedInfo**, when the Collected_Information DP has been armed (monitorMode=notifyAndContinue) and the calling party sends the remaining digits to complete the call information.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=collectedInfo - monitorMode= 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

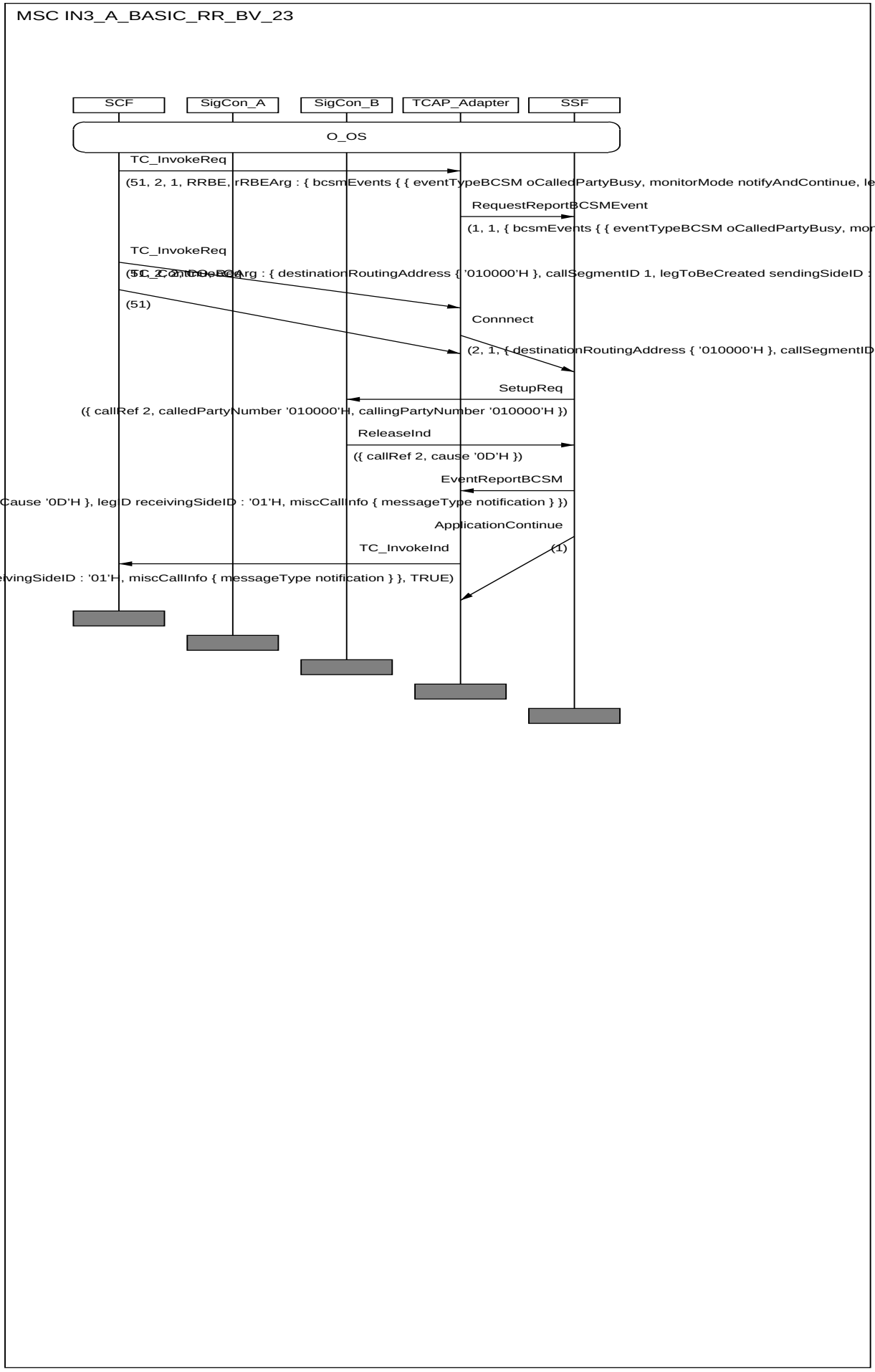
	IN3_A_BASIC_RR_BV_21
Work item no.:	ITEM_BASIC_104
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_21
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **analysedInfo**, when the Analysed_Information DP has been armed (monitorMode=notifyAndContinue), the SCF has sent a **CollectInformation** operation and the calling party sends the remaining digits to complete the call information.
Requirements refs	6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.11, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS_ColIn Preamble contains an InitialDP without complete digits for CalledPartyNumber
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=analysedInfo monitorMode= 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 IN3_A_BASIC_RR_BV_21
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM analysedInformation, monitorMode notifyAndContinue, legID sendingSideID
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM analysedInformation, monitorMode notifyAn
TC_InvokeReq
(51, 2, 1, CueInRBE, cueInRBEArg : { filteredCallTreatment { }, filteringCharacteristics interval : 0, filteringTimeOut duration : 0, filteringCriteria serviceKe
(51)
CollectInformation
(2, 1, { })
CallProgressReq
({ callRef 1, cause '1F'H })
SubsequentAddressInd
({ callRef 1, digits '1234'H })
AddressEndInd
({ callRef 1 })
EventReportBCSM
lledPartynumber '0200001234'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } })
ApplicationContinue
(1)
TC_InvokeInd
1234'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } }, TRUE)
SigConA_Release_thenB

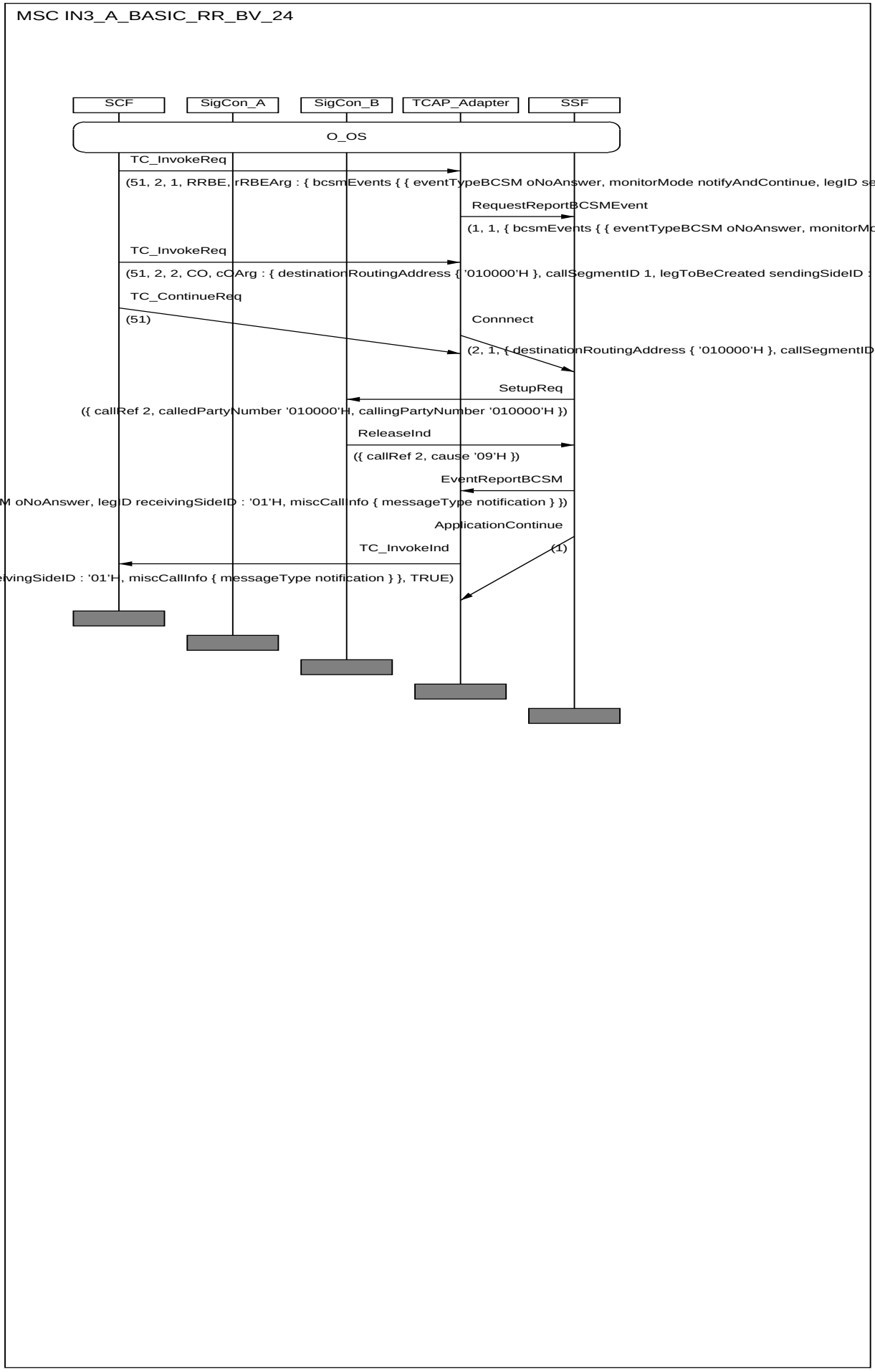
	IN3_A_BASIC_RR_BV_22
Work item no.:	ITEM_BASIC_105
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_22
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **routeSelectFailure**, when the Route_Select_Failure DP has been armed (monitorMode=notifyAndContinue), the SCF has sent a **Connect** invoke component to request the connection of the call to SigConB, but SigConB releases the call **indicating** routeFailure2.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg1) - eventTypeBCSM=routeSelectFailure - monitorMode= notifyAndContinue followed by a **Connect** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call **(ReleaseInd)** with cause routeFailure2
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** routeSelectFailure
Postamble:	None


MSC IN3_A_BASIC_RR_BV_22
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, rRBEArg : { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode notifyAndContinue, legID sendingSideID :
(51, 2, CallDeBCA
(51)
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM routeSelectFailure, monitorMode notifyAndC
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '0C'H })
EventReportBCSM
pecificInfo : { failureCause '0C'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } })
ApplicationContinue
TC_InvokeInd
(1)
e '0C'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } }, TRUE)

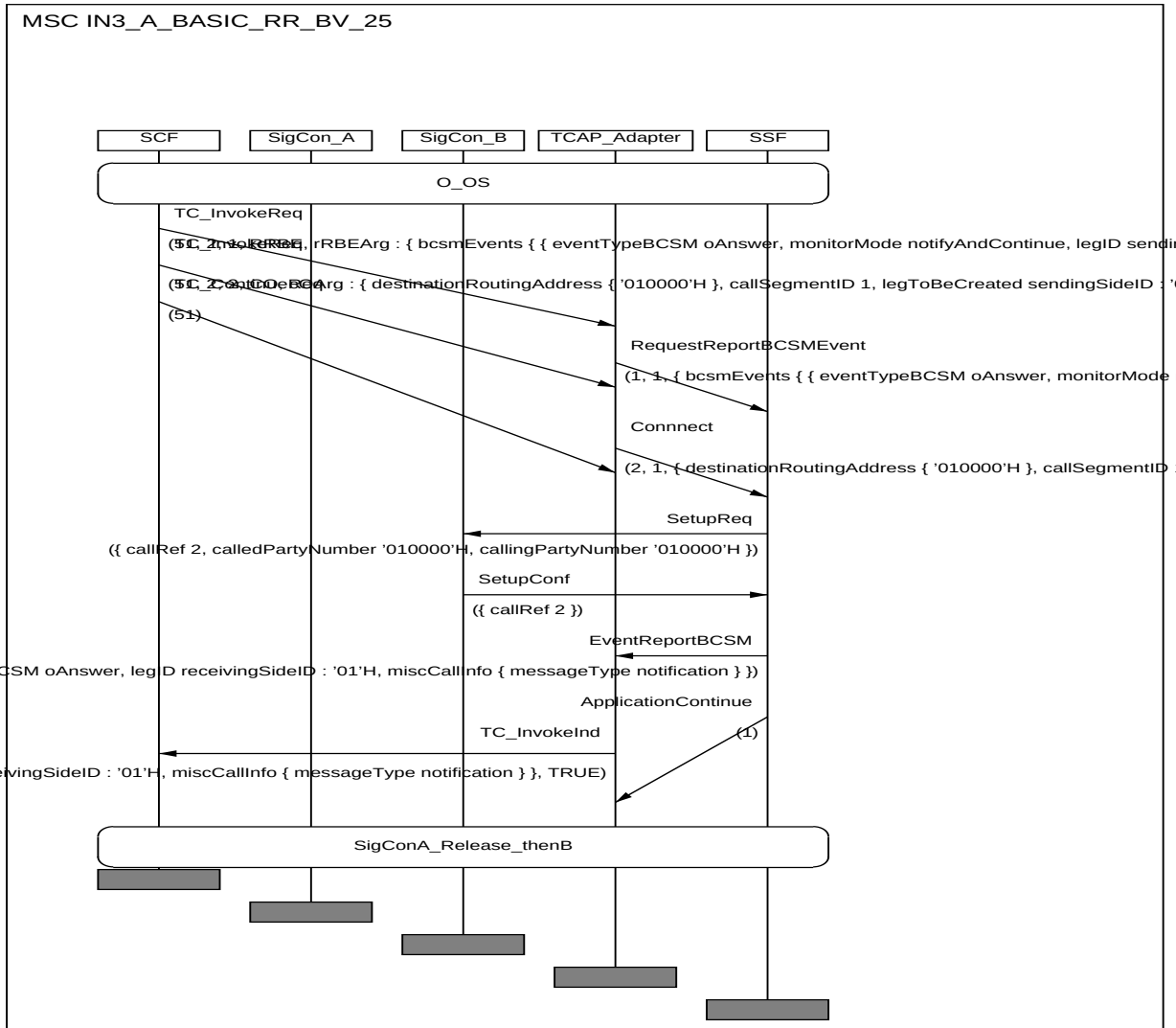
	IN3_A_BASIC_RR_BV_23
Work item no.:	ITEM_BASIC_106
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_23
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oCalledPartyBusy**, when the O_Called_Party_Busy DP has been armed (monitorMode=notifyAndContinue), the SCF has sent a **Connect** invoke component to request the connection of the call to SigConB, but SigConB releases the call **indicating** bPtyBusy_UDUB.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg1) - eventTypeBCSM=oCalledPartyBusy - monitorMode= 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 IN3_A_BASIC_RR_BV_23
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode notifyAndContinue, legID sendingSideID :
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oCalledPartyBusy, monitorMode notifyAndC
TC_InvokeReq
(51, 2, 1, CRG, cRGArg : { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '0D'H })
EventReportBCSM
SpecificInfo : { busyCause '0D'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } })
ApplicationContinue
(1)
TC_InvokeInd
e '0D'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } }, TRUE)

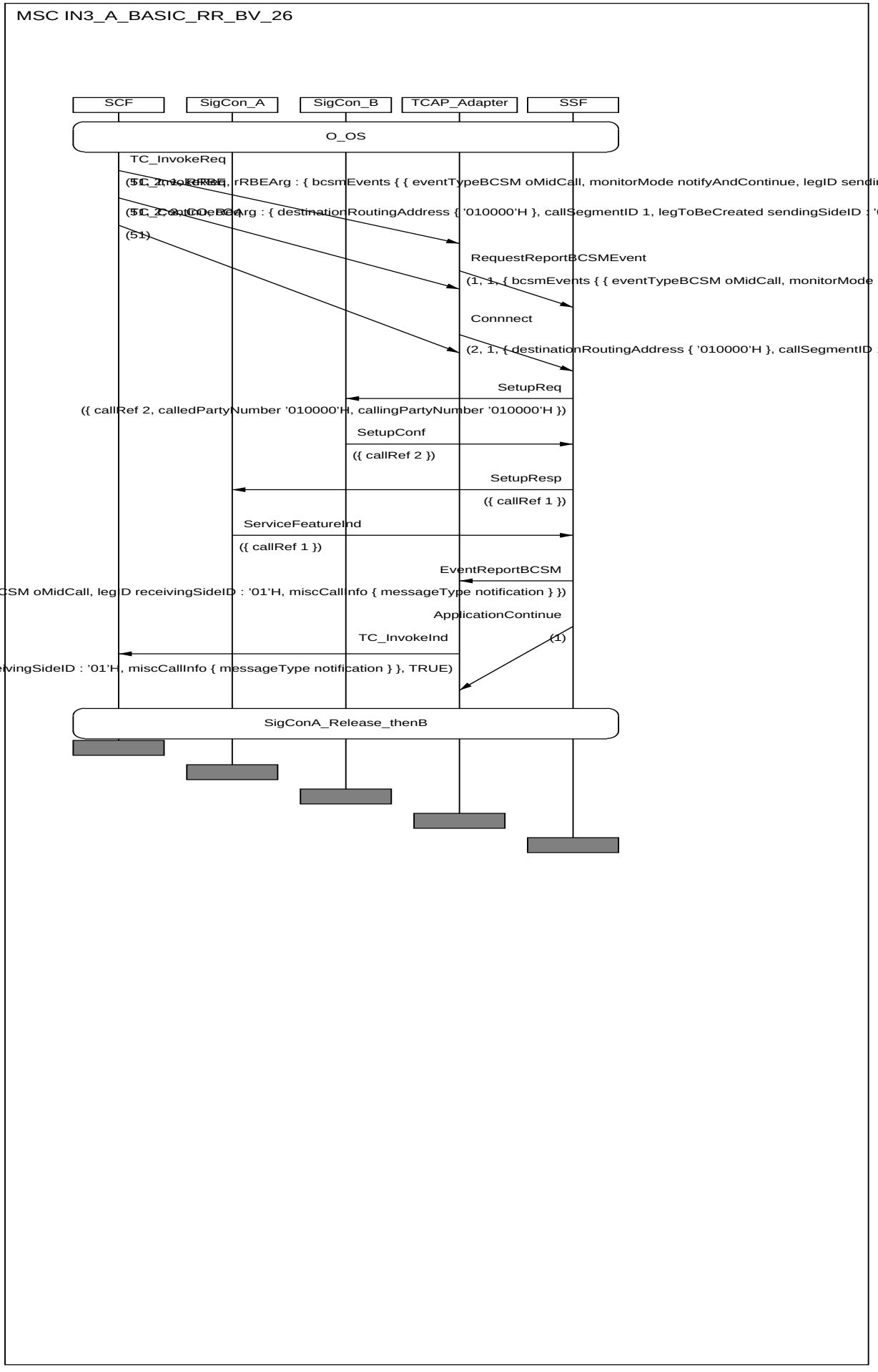
	IN3_A_BASIC_RR_BV_24
Work item no.:	ITEM_BASIC_107
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_24
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oNoAnswer**, when the O_No_Answer DP has been armed (monitorMode=notifyAndContinue), the SCF has sent a **Connect** invoke component to request the connection of the call to SigConB, but SigConB releases the call because SigConB does not answer.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oNoAnswer - monitorMode= notifyAndContinue followed by a **Connect** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call **(ReleaseInd)** because SigConB does not answer
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** oNoAnswer
Postamble:	none


MSC IN3_A_BASIC_RR_BV_24
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode notifyAndContinue, legID sendingSideID : '01'H }
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oNoAnswer, monitorMode notifyAndContinu
TC_InvokeReq
(51, 2, 2, CO, cOArg : { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
TC_ContinueReq
(51)
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '09'H })
EventReportBCSM
1, { eventTypeBCSM oNoAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType notification } })
ApplicationContinue
(1)
TC_InvokeInd
oAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType notification } }, TRUE)

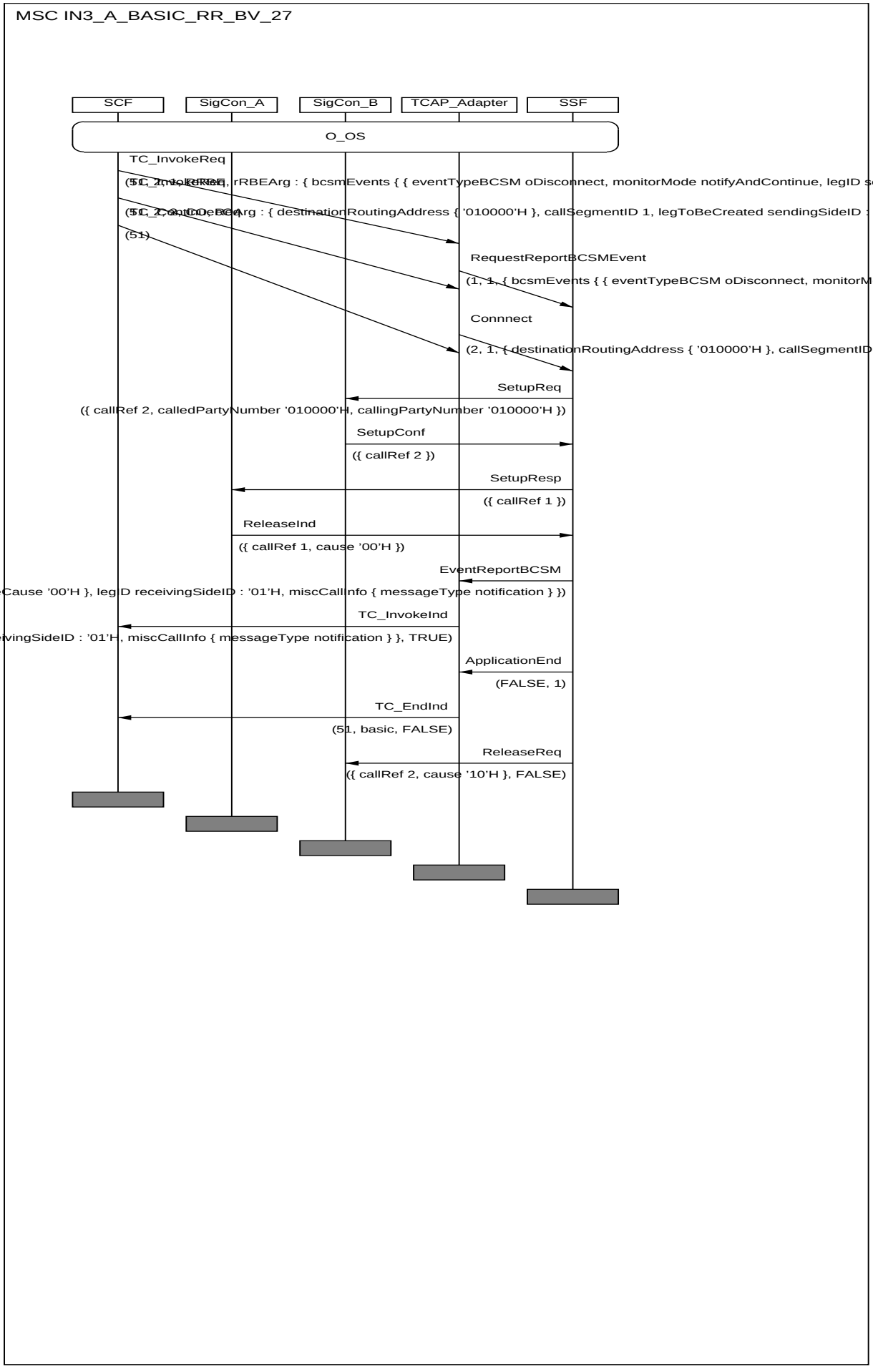
	IN3_A_BASIC_RR_BV_25
Work item no.:	ITEM_BASIC_108
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_25
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oAnswer**, when the O_Answer DP has been armed (monitorMode=notifyAndContinue), the SCF has sent a **Connect** invoke component to request the connection of the call to SigConB, and SigConB answers the call.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oAnswer - monitorMode= 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 IN3_A_BASIC_RR_BV_25
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, rRBEArg : { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode notifyAndContinue, legID sendingSideID : '01'H } } }
(51, 2, CallDetectArg : { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
(51)
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oAnswer, monitorMode notifyAndContinue,
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
1, { eventTypeBCSM oAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType notification } })
ApplicationContinue
TC_InvokeInd
(1)
oAnswer, legID receivingSideID : '01'H, miscCallInfo { messageType notification } }, TRUE)
SigConA_Release_thenB

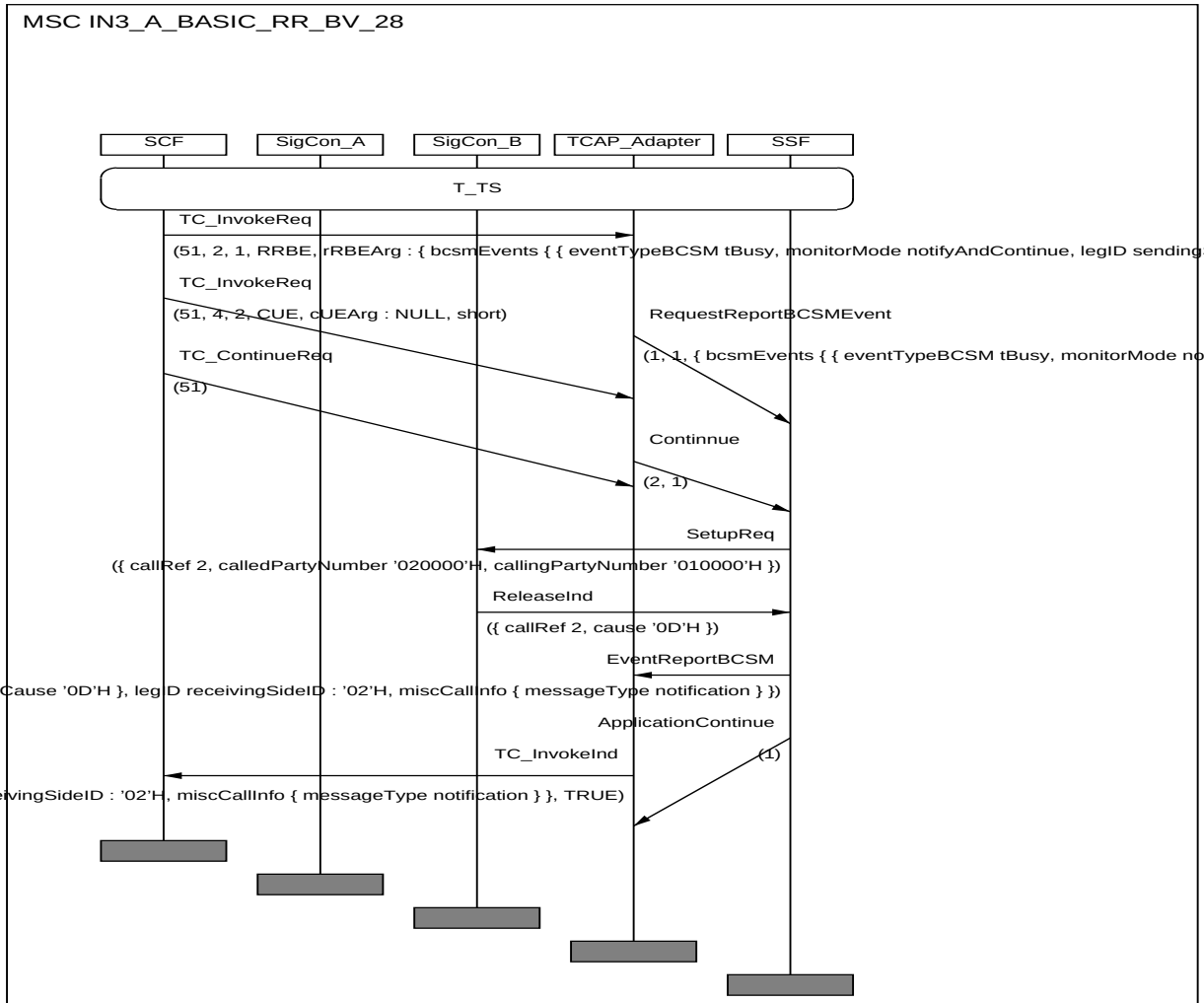
	IN3_A_BASIC_RR_BV_26
Work item no.:	ITEM_BASIC_109
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_26
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oMidCall**, when the O_Mid_Call DP has been armed (monitorMode=notifyAndContinue), SigConA and SigConB have been connected, and SigConA initiates a service (ServiceFeatureInd sent to the SSF).
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oMidCall - monitorMode= 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 IN3_A_BASIC_RR_BV_26
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
RequestReportBCSMEvent
Connnect
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ServiceFeatureInd
({ callRef 1 })
EventReportBCSM
ApplicationContinue
TC_InvokeInd
(1)
SigConA_Release_thenB

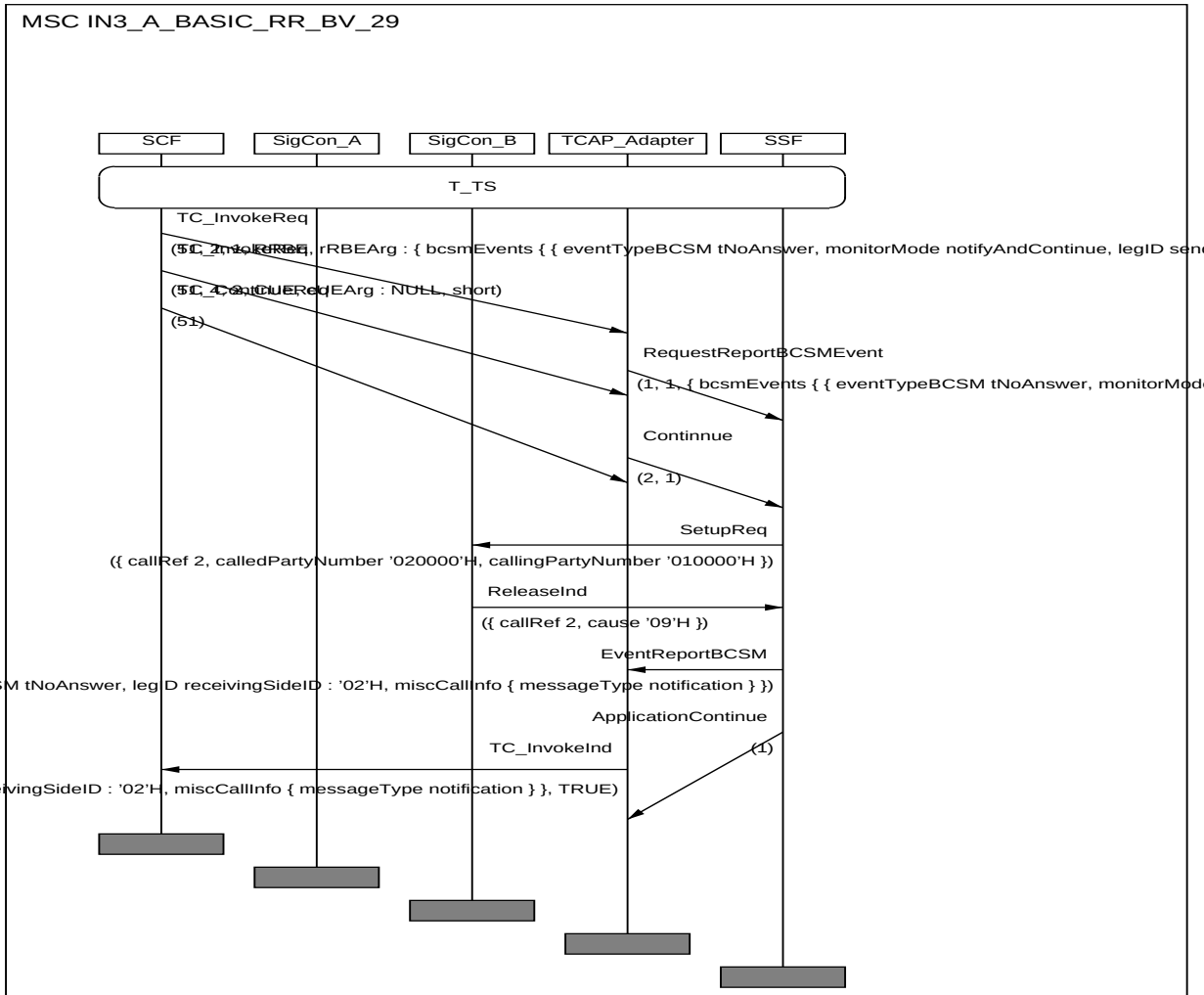
	IN3_A_BASIC_RR_BV_27
Work item no.:	ITEM_BASIC_110
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_27
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oDisconnect**, when the O_Disconnect DP has been armed (monitorMode=notifyAndContinue, legID=sendingSideID: "01"H), SigConA and SigConB have been connected, and SigConA then clears the call. Check also that SigConB receives a ReleaseReq.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oDisconnect - monitorMode= 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 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
Postamble:	none


MSC IN3_A_BASIC_RR_BV_27
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode notifyAndContinue, legID sendingSideID : '01'H
(51, 2, rRBEArg : { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated sendingSideID : '01'H }, short
(51)
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode notifyAndContinu
Connnect
(2, 1, { destinationRoutingAddress { '010000'H }, callSegmentID 1, legToBeCreated s
SetupReq
({ callRef 2, calledPartyNumber '010000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ReleaseInd
({ callRef 1, cause '00'H })
EventReportBCSM
ecificInfo : { releaseCause '00'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } })
TC_InvokeInd
e '00'H }, legID receivingSideID : '01'H, miscCallInfo { messageType notification } }, TRUE)
ApplicationEnd
(FALSE, 1)
TC_EndInd
(51, basic, FALSE)
ReleaseReq
({ callRef 2, cause '10'H }, FALSE)

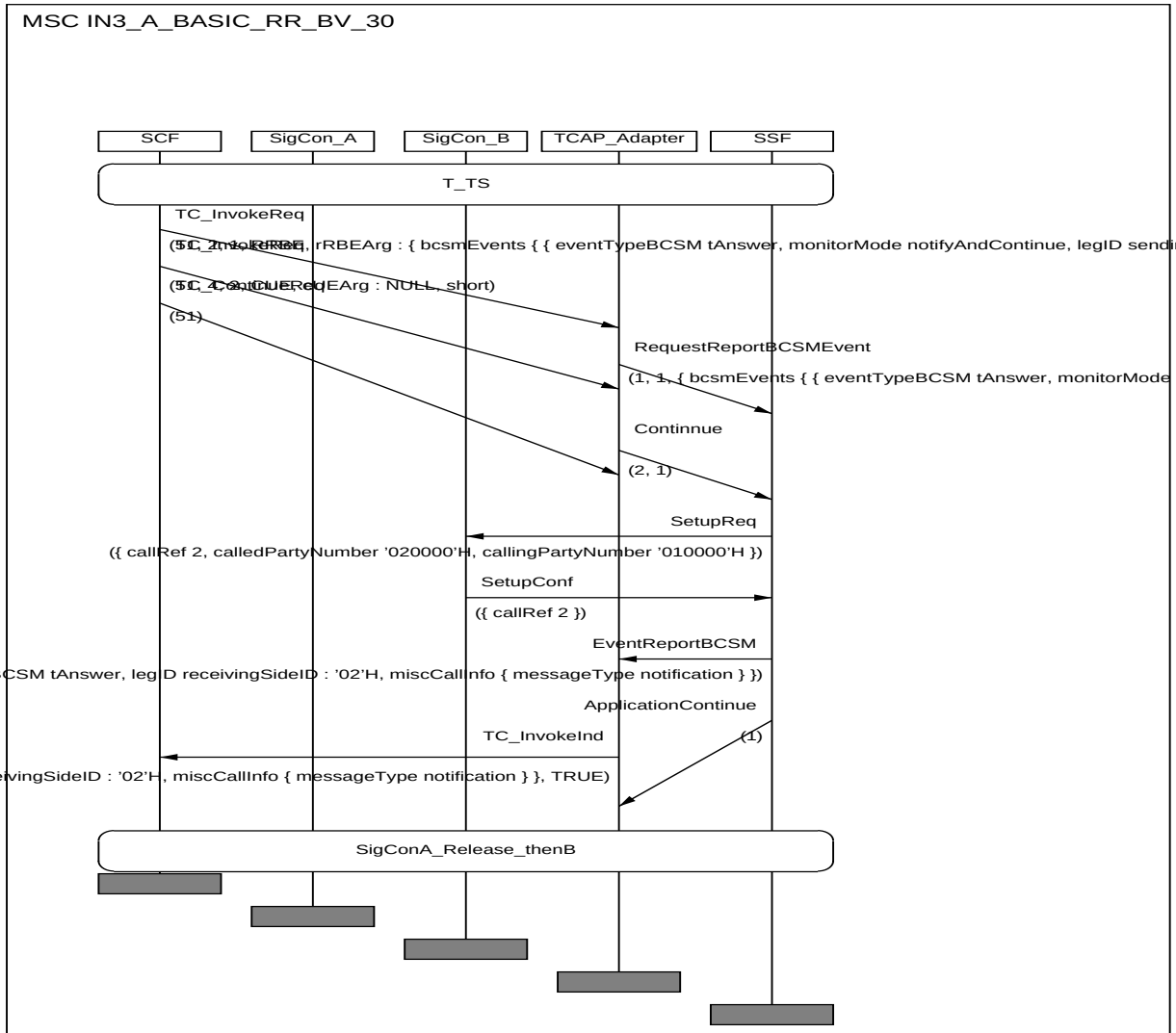
	IN3_A_BASIC_RR_BV_28
Work item no.:	ITEM_BASIC_111
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_28
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tBusy**, when the T_Busy DP has been armed (monitorMode=notifyAndContinue), and SigConB releases theSetupReq sent from the SSF with the indication of bPtyBusy_UDUB.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg2) - eventTypeBCSM=tBusy - monitorMode= notifyAndContinue followed by a **Continue** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call (**ReleaseInd** sent) because SigConB is busy (UDUB="0D"H)
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** tBusy
Postamble:	None


MSC IN3_A_BASIC_RR_BV_28
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM tBusy, monitorMode notifyAndContinue, legID sendingSideID : '02'H } } }, sh
TC_InvokeReq
(51, 4, 2, CUE, cUEArg : NULL, short)
RequestReportBCSMEvent
TC_ContinueReq
(1, 1, { bcsmEvents { { eventTypeBCSM tBusy, monitorMode notifyAndContinue, legI
(51)
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '0D'H })
EventReportBCSM
SpecificInfo : { busyCause '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType notification } })
ApplicationContinue
TC_InvokeInd
(1)
e '0D'H }, legID receivingSideID : '02'H, miscCallInfo { messageType notification } }, TRUE)

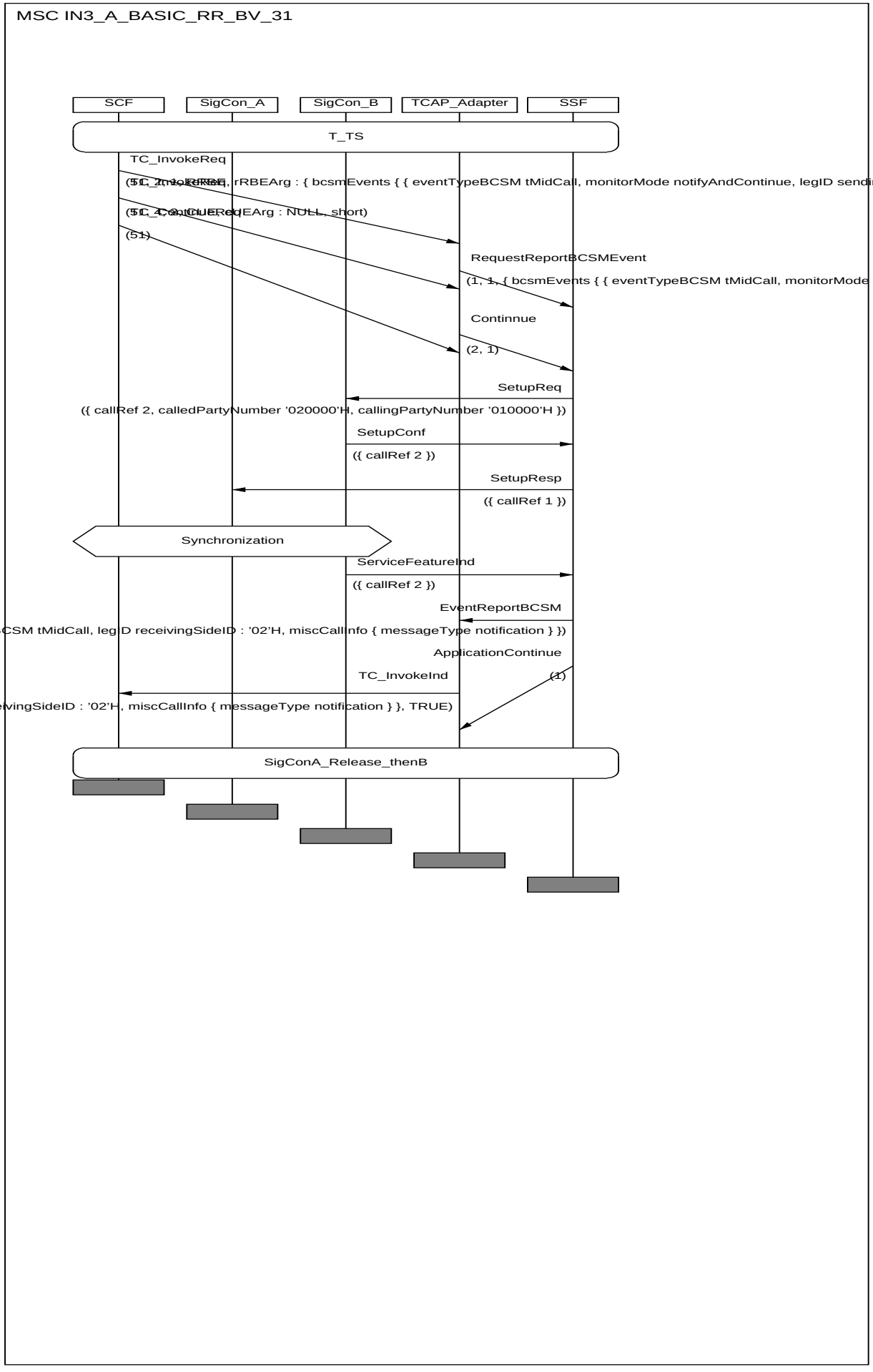
	IN3_A_BASIC_RR_BV_29
Work item no.:	ITEM_BASIC_112
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_29
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tNoAnswer**, when the T_No_Answer DP has been armed (monitorMode=notifyAndContinue), and SigConB releases theSetupReq sent from the SSF because SigConB does not answer.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg2) - eventTypeBCSM=tNoAnswer - monitorMode= notifyAndContinue followed by a **Continue** invoke Then SSF sends a SetupReq to SigCon B SigCon B releases the call (**ReleaseInd** sent) because SigConB does not answer
Pass criteria	Check that SSF sends to SCF an **EventReportBCSM** with the indication of **eventTypeBCSM=** tNoAnswer
Postamble:	None


MSC IN3_A_BASIC_RR_BV_29
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, rRBEArg : { bcsmEvents { { eventTypeBCSM tNoAnswer, monitorMode notifyAndContinue, legID sendingSideID : '02'H }
(51, 4, cUEArg : NULL, short)
(51)
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM tNoAnswer, monitorMode notifyAndContinue
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ReleaseInd
({ callRef 2, cause '09'H })
EventReportBCSM
1, { eventTypeBCSM tNoAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType notification } })
ApplicationContinue
TC_InvokeInd
(1)
oAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType notification } }, TRUE)

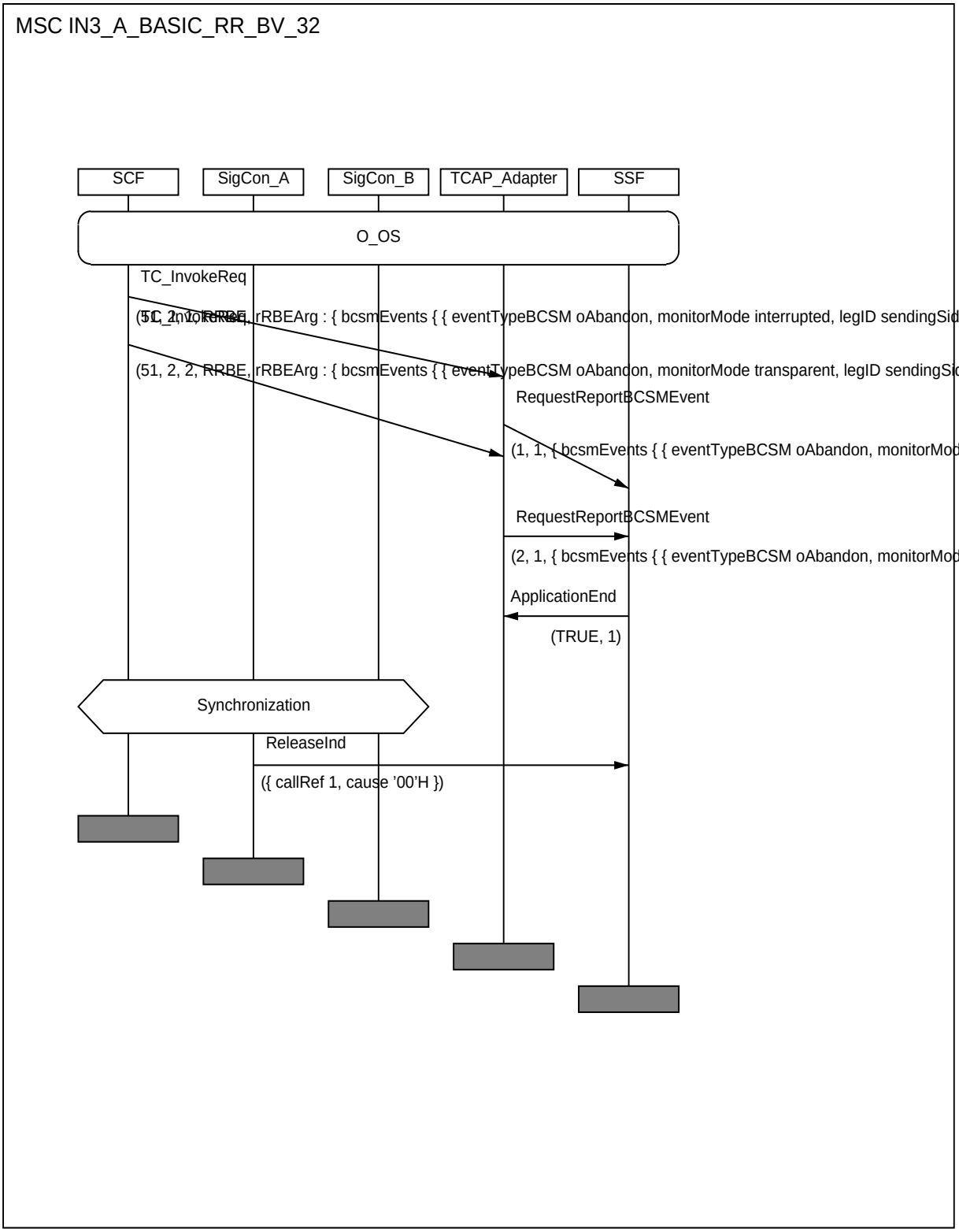
	IN3_A_BASIC_RR_BV_30
Work item no.:	ITEM_BASIC_113
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_30
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tAnswer**, when the T_Answer DP has been armed (monitorMode=notifyAndContinue), and SigConB confirms theSetupReq sent from the SSF.
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=tAnswer - monitorMode= 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 IN3_A_BASIC_RR_BV_30
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, rRBEArg : { bcsmEvents { { eventTypeBCSM tAnswer, monitorMode notifyAndContinue, legID sendingSideID : '02'H } } },
(51, 4, cUEArg : NULL, short)
(51)
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM tAnswer, monitorMode notifyAndContinue,
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
EventReportBCSM
1, { eventTypeBCSM tAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType notification } })
ApplicationContinue
(1)
TC_InvokeInd
tAnswer, legID receivingSideID : '02'H, miscCallInfo { messageType notification } }, TRUE)
SigConA_Release_thenB

	IN3_A_BASIC_RR_BV_31
Work item no.:	ITEM_BASIC_114
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_31
Purpose:	Verify that the SSF sends to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **tMidCall**, when the T_Mid_Call DP has been armed (monitorMode=notifyAndContinue), SigConA and SigConB have been connected, and SigConB initiates a service (ServiceFeatureInd sent to the SSF).
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.14, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	T_TS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg2) - eventTypeBCSM= tMidCall - monitorMode= 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 IN3_A_BASIC_RR_BV_31
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
T_TS
TC_InvokeReq
(51, 2, rRBEArg : { bcsmEvents { { eventTypeBCSM tMidCall, monitorMode notifyAndContinue, legID sendingSideID : '02'H } } },
(51, 4, cUEArg : NULL, short)
(51)
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM tMidCall, monitorMode notifyAndContinue,
Continnue
(2, 1)
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
Synchronization
ServiceFeatureInd
({ callRef 2 })
EventReportBCSM
1, { eventTypeBCSM tMidCall, legID receivingSideID : '02'H, miscCallInfo { messageType notification } })
ApplicationContinue
TC_InvokeInd
(1)
tMidCall, legID receivingSideID : '02'H, miscCallInfo { messageType notification } }, TRUE)
SigConA_Release_thenB

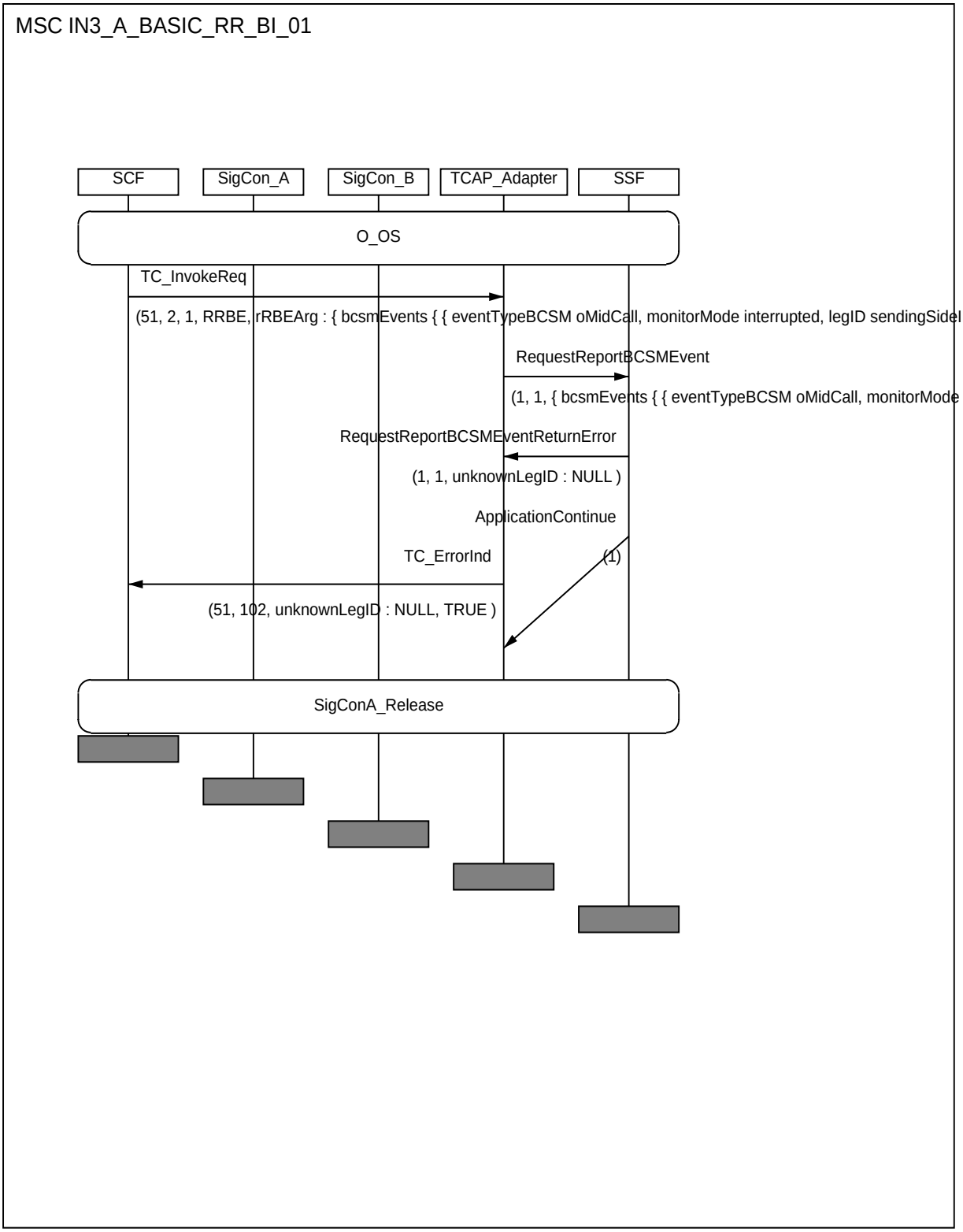
	IN3_A_BASIC_RR_BV_32
Work item no.:	ITEM_BASIC_115
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_32
Purpose:	Verify that the SSF does not send to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oAbandon** when the calling party abandons the call, if the O_Abandon DP had first been armed (monitorMode=interrupted), and then disarmed (monitorMode=transparent).
Requirements refs	6.6.3.3.2.1, 8.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters (leg1) - - eventTypeBCSM=oAbandon - monitorMode=interrupted then SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oAbandon - monitorMode=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


MSC IN3_A_BASIC_RR_BV_32
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAbandon, monitorMode interrupted, legID sendingSideID : '01'H } } }, short
(51, 2, 2, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oAbandon, monitorMode transparent, legID sendingSideID : '01'H } } }, shor
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oAbandon, monitorMode interrupted, legID s
RequestReportBCSMEvent
(2, 1, { bcsmEvents { { eventTypeBCSM oAbandon, monitorMode transparent, legID
ApplicationEnd
(TRUE, 1)
Synchronization
ReleaseInd
({ callRef 1, cause '00'H })

	IN3_A_BASIC_RR_BV_33
Work item no.:	ITEM_BASIC_116
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_33
Purpose:	Verify that the SSF does not send to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oAnswer** when the called party answers the call, if the O_Answer DP had first been armed (monitoringMode=interrupted), and then disarmed (monitoringMode=transparent).
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - - eventTypeBCSM=oAnswer - monitoringMode=interrupted SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM=oAnswer - monitoringMode=transparent followed by a **Connect** invoke then SSF sends a SetupReq to SigCon B SigCon B answers the call (SetupConf from SigCon B to SSF)
Pass criteria	Check that SSF does not send to SCF an **EventReportBCSM**
Postamble:	SigConA_Release_thenB

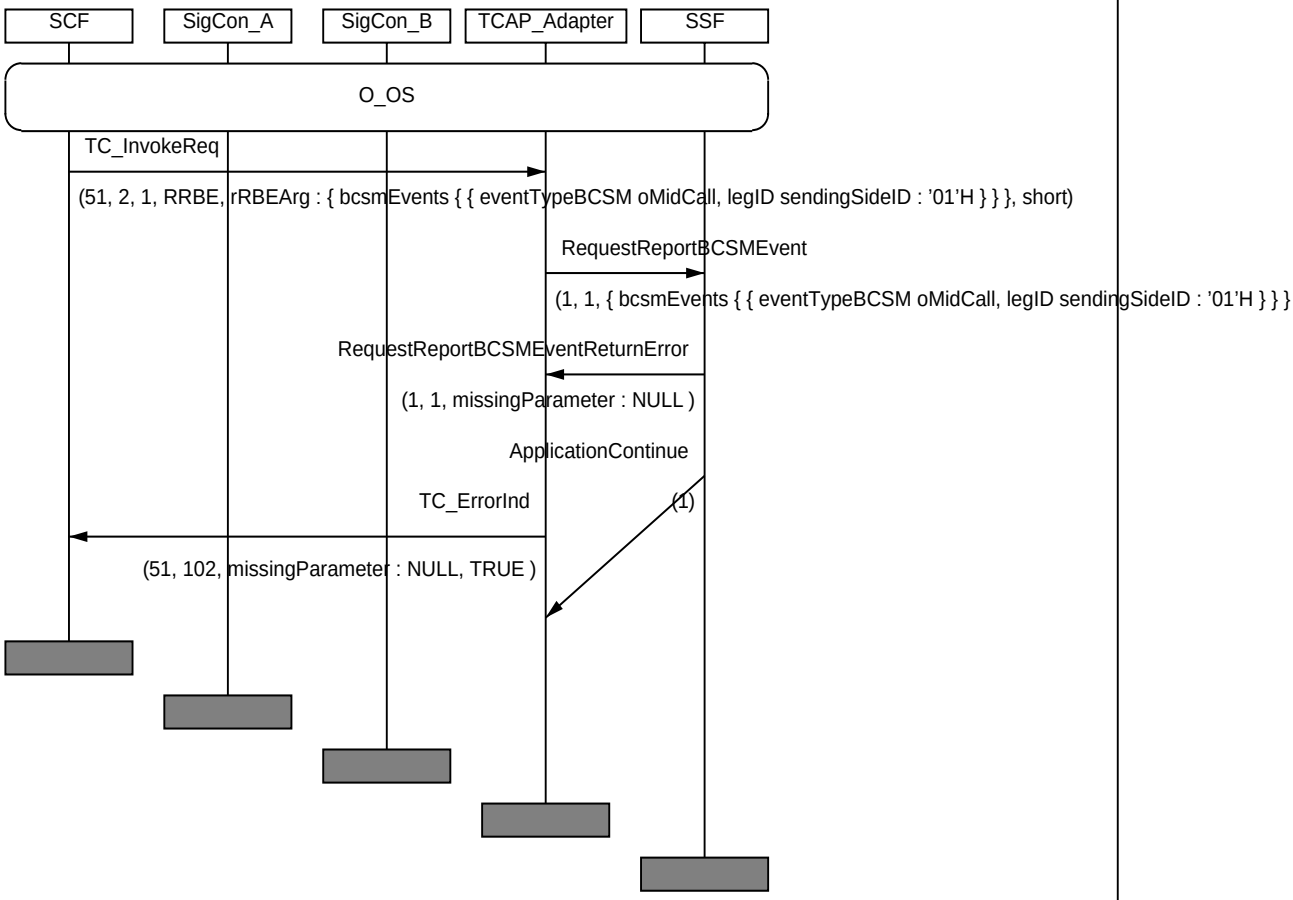
	IN3_A_BASIC_RR_BV_34
Work item no.:	ITEM_BASIC_117
IN2 Ref(tmp)	IN2_A_BASIC_RR_BV_34
Purpose:	Verify that the SSF does not send to the SCF an **EventReportBCSM** with the indication of eventTypeBCSM = **oDisconnect** when the calling party clears the call after having received an answer, if the O_Disconnect DP had first been armed (monitoringMode=interrupted), and then disarmed (monitoringMode=transparent).
Requirements refs	6.5.1.2.3, 6.6.3.3.2.1, 6.6.3.4.2.7, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 8.3.2, 8.4.2, 11.12, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.1
Selection Cond.	
Preamble:	O_OS
Test description	SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oDisconnect - monitoringMode= interrupted then SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oDisconnect - monitoringMode= transparent followed by a **Connect** invoke Then SSF establishes the call (a SetupReq to SigCon B. SetupConf from SigConB to SSF, then SetupResp to SigConB) SigCon A (calling party) clears the call after it is answered (ReleaseInd sent)
Pass criteria	Check that SSF does not send to SCF an **EventReportBCSM**
Postamble:	None

	IN3_A_BASIC_RR_BI_01
Work item no.:	ITEM_BASIC_118
IN2 Ref(tmp)	IN2_A_BASIC_RR_BI_01
Purpose:	Verify that the SSF sends to the SCF an **RequestReportBCSMEvent** returnError component with the error indication of **unknownLegId** after receiving from the SCF an **RequestReportBCSMEvent** invoke component containing parameters eventTypeBCSM= oMidCall, monitorMode=interrupted and the legID has an invalid value. Verify also that after call establishment the SSF does not send any **EventReportBCSM**.
Requirements refs	6.6.3.3.2.1, 8.2, 8.2, 8.2.1.2, 8.2.2, 8.2.2.5, 8.2.2.6, 8.2.2.7, 11.24.1, 11.24.1.1.1, 11.24.2.1, 11.38.1, 11.38.1.1.1, 11.38.3.2, 13.1.5.1, 15.1.1.2.2
Selection Cond.	
Preamble:	O_OS
Test description	SCF - SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameters - eventTypeBCSM= oMidCall - monitorMode=interrupted - legID=invalid value
Pass criteria	- Check that SSF sends to SCF a **RequestReportBCSMEvent** error with the indication of **unknownLegId** - When call Set-up is established, check that SSF is not sending to SCF any **EventReportBCSM**
Postamble:	SigConA_Release


MSC IN3_A_BASIC_RR_BI_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oMidCall, monitorMode interrupted, legID sendingSideID : '03'H } } }, short
RequestReportBCSMEvent
(1, 1, { bcsmEvents { { eventTypeBCSM oMidCall, monitorMode interrupted, legID se
RequestReportBCSMEventReturnError
(1, 1, unknownLegID : NULL)
ApplicationContinue
TC_ErrorInd
(1)
(51, 102, unknownLegID : NULL, TRUE)
SigConA_Release

	IN3_A_BASIC_RR_BI_02
Work item no.:	ITEM_BASIC_119
IN2 Ref(tmp)	IN2_A_BASIC_RR_BI_02
Purpose:	Verify that the SSF rejects an **RequestReportBCSMEvent** invoke component with mandatory parameter **monitorMode** missing.
Requirements refs	6.6.3.3.2.1, 8.2, 8.2.1.2, 8.2.2, 11.38.1, 11.38.1.1.1, 11.38.3.1, 11.38.3.2, 15.1.1.2.2
Selection Cond.	
Preamble:	O_OS
Test description	SCF - SCF sends to SSF **RequestReportBCSMEvent** invoke containing parameter - eventTypeBCSM=oMidCall and mandatory parameter **monitorMode** is omitted.
Pass criteria	- Check that SSF sends to the SCF a **RequestReportBCSMEvent** error component ("missing parameter")
Postamble:	SigConA_Release

MSC IN3_A_BASIC_RR_BI_02



6.6.23 ResetTimer (RS) procedure

	IN3_A_BASIC_RS_BV_01
Work item no.:	ITEM_BASIC_263
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having sent an InitialDP invoke component to the SCF and being in the "Waiting for instructions" state, does not send a ResetTimer returnError component, when receiving a valid ResetTimer invoke component containing a timerValue "pointing" behind the default InitialDP Tssf timeout value. Verify also that the SSF does not abort the TCAP dialog when the default InitialDP Tssf timeout value expires (the restarted Tssf timer still being running), before a response is received from the SCF.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_OS
Test description	Start timer(TSSF_TIMEOUT_IDP_DEFAULT - 5 %) SWait for TSSF_TIMEOUT_IDP_DEFAULT - 5 % L1!ResetTimer invoke(TSSF_TIMEOUT_RESET) Start timer(TSSF_TIMEOUT_RESET - 5 %)
Pass criteria	The SSF does not abort the TCAP dialog before TSSF_TIMEOUT_RESET - 5 % occurs and does not send a ResetTimer returnError component
Postamble:	ReleaseCallA

	IN3_A_BASIC_RS_BV_02
Work item no.:	ITEM_BASIC_264
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having sent an InitialDP invoke component to the SCF, being in the "Waiting for instructions" state, having invoked the ResetTimer operation successfully for one time, and having received a RequestReportBCSMEvent invoke component afterwards, does not send a ResetTimer returnError component, when receiving a valid ResetTimer invoke component for the second time. Verify also that the SSF does not abort the TCAP dialog when the Tssf timeout value associated with the first ResetTimer operation expires (the restarted Tssf timer still being running because of the second ResetTimer operation), before a further response is received from the SCF.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_OS
Test description	Start timer(TSSF_TIMEOUT_IDP_DEFAULT - 5 %) Wait for TSSF_TIMEOUT_IDP_DEFAULT - 5 % L1!ResetTimer invoke(TSSF_TIMEOUT_RESET) Start timer(TSSF_TIMEOUT_RESET - 5 %) Wait for TSSF_TIMEOUT_RESET - 5 % L1!RequestReportBCSMEvent(1,interrupted,oDisconnect) L1!ResetTimer invoke(TSSF_TIMEOUT_RESET) Start timer(TSSF_TIMEOUT_RESET - 5 %) Wait for TSSF_TIMEOUT_RESET - 5 % (second time)
Pass criteria	The SSF does not abort the TCAP dialog and does not send a ResetTimer returnError component before TSSF_TIMEOUT_RESET - 5 % (second time) occurs
Postamble:	ReleaseCallA

	IN3_A_BASIC_RS_BV_03
Work item no.:	ITEM_BASIC_265
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having sent an InitialDP invoke component to the SCF, being in the "Waiting for instructions" state and having invoked the ResetTimer operation successfully for one time, aborts the TCAP dialog when the Tssf timeout value associated with the ResetTimer operation expires.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_OS
Test description	Start timer(TSSF_TIMEOUT_IDP_DEFAULT - 5 %) Wait for TSSF_TIMEOUT_IDP_DEFAULT - 5 % L1!ResetTimer invoke(TSSF_TIMEOUT_RESET) Start timer(TSSF_TIMEOUT_RESET + 5 %)
Pass criteria	The SSF aborts the TCAP dialog before TSSF_TIMEOUT_RESET + 5 % occurs
Postamble:	ReleaseCallA

	IN3_A_BASIC_RS_BV_04
Work item no.:	ITEM_BASIC_266
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the SplitLeg operation and being in the "Waiting for instructions" state, does not send a ResetTimer returnError component, when receiving a valid ResetTimer invoke component containing a timerValue "pointing" behind the default Tssf timeout value. Verify also that the SSF does not send an EntityReleased invoke component when the default Tssf timeout value expires (the restarted Tssf timer still being running), before a response is received from the SCF.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_S2P
Test description	L1!SplitLeg(1,2) L1?SplitLegReturnResult Start timer(TSSF_TIMEOUT_DEFAULT - 5 %) Wait for TSSF_TIMEOUT_DEFAULT - 5 % L1!ResetTimer invoke(TSSF_TIMEOUT_RESET) Start timer(TSSF_TIMEOUT_RESET - 5 %)
Pass criteria	The SSF does not abort the TCAP dialog before TSSF_TIMEOUT_RESET - 5 % occurs and does not send a ResetTimer returnError component
Postamble:	ReleaseCallA

	IN3_A_BASIC_RS_BV_05
Work item no.:	ITEM_BASIC_267
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the SplitLeg operation, being in the "Waiting for instructions" state and having invoked the ResetTimer operation successfully for one time, does not send a ResetTimer returnError component, when receiving a valid ResetTimer invoke component for the second time. Verify also that the SSF does not send an EntityReleased invoke component when the Tssf timeout value associated with the first ResetTimer operation expires (the restarted Tssf timer still being running because of the second ResetTimer operation), before a further response is received from the SCF.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_S2P
Test description	L1!SplitLeg(1,2) L1?SplitLegReturnResult Start timer(TSSF_TIMEOUT_DEFAULT - 5 %) Wait for TSSF_TIMEOUT_DEFAULT - 5 % L1!ResetTimer invoke(TSSF_TIMEOUT_RESET) Start timer(TSSF_TIMEOUT_RESET - 5 %) Wait for TSSF_TIMEOUT_DEFAULT - 5 % L1!ResetTimer invoke(TSSF_TIMEOUT_RESET) Start timer(TSSF_TIMEOUT_RESET - 5 %) (second time)
Pass criteria	The SSF does not abort the TCAP dialog before TSSF_TIMEOUT_RESET - 5 % (second time) occurs and does not send a ResetTimer returnError component
Postamble:	ReleaseCallA

	IN3_A_BASIC_RS_BV_06
Work item no.:	ITEM_BASIC_268
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having performed the SplitLeg operation, being in the "Waiting for instructions" state and having invoked the ResetTimer operation successfully for one time, sends an EntityReleased invoke component when the Tssf timeout value associated with the ResetTimer operation expires.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_S2P
Test description	L1!SplitLeg(1,2) L1?SplitLegReturnResult Start timer(TSSF_TIMEOUT_DEFAULT - 5 %) Wait for TSSF_TIMEOUT_DEFAULT - 5 % L1!ResetTimer invoke(TSSF_TIMEOUT_RESET) Start timer(TSSF_TIMEOUT_RESET + 5 %)
Pass criteria	L1?EntityReleased invoke before timeout TSSF_TIMEOUT_RESET + 5 %
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RS_BI_01
Work item no.:	ITEM_BASIC_269
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having sent an InitialDP invoke component to the SCF and being in the "Waiting for instructions" state, sends a ResetTimer returnError component with errorCode "missingParameter", when receiving a ResetTimer invoke component where mandatory parameter "timervalue" is missing.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_S2P
Test description	L1!SplitLeg(1,2) L1?SplitLegReturnResult L1!ResetTimer invoke(parameter "timervalue" omitted)
Pass criteria	L1?ResetTimer returnError(missingParameter)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RS_BI_02
Work item no.:	ITEM_BASIC_270
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having sent an InitialDP invoke component to the SCF and being in the "Waiting for instructions" state, sends a ResetTimer returnError component with errorCode "unexpectedDataValue", when receiving a ResetTimer invoke component where parameter "csID" has a value not identifying an existing CS.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_S2P
Test description	L1!SplitLeg(1,2) L1?SplitLegReturnResult L1!ResetTimer invoke(TSSF_TIMEOUT_RESET, csID=3)
Pass criteria	L1?ResetTimer returnError(unexpectedDataValue)
Postamble:	ReleaseCallA2

	IN3_A_BASIC_RS_BO_01
Work item no.:	ITEM_BASIC_271
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having sent an InitialDP invoke component to the SCF, being in the "Waiting for instructions" state and having invoked the ResetTimer once successfully, sends a ResetTimer returnError component with errorCode "taskRefused" or "unexpectedComponentSequence", when receiving a valid ResetTimer invoke component for the second time.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_OS
Test description	Start timer(TSSF_TIMEOUT_IDP_DEFAULT - 5 %) Wait for TSSF_TIMEOUT_IDP_DEFAULT - 5 % L1!ResetTimer invoke(TSSF_TIMEOUT_RESET) Start timer(TSSF_TIMEOUT_RESET - 5 %) Wait for TSSF_TIMEOUT_RESET - 5 % L1!ResetTimer invoke(TSSF_TIMEOUT_RESET)
Pass criteria	L1?ResetTimer returnError("taskRefused" or "unexpectedComponentSequence")
Postamble:	ReleaseCallA

	IN3_A_BASIC_RS_BO_02
Work item no.:	ITEM_BASIC_272
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, being in the "Monitoring" state, sends a ResetTimer returnError component with errorCode "taskRefused" or "unexpectedComponentSequence", when receiving a ResetTimer invoke component.
Requirements refs	8.2.2, 11.40
Selection Cond.	
Preamble:	O_S2P
Test description	L1!ResetTimer invoke(TSSF_TIMEOUT_RESET)
Pass criteria	L1?ResetTimer returnError("taskRefused" or "unexpectedComponentSequence")
Postamble:	ReleaseCallA2

6.6.24 SendChargingInformation (SCI) procedure

6.6.24.1 General information on testing SCI

Since the SCI procedure is more complex than other procedures, and testing it means also testing large parts of ES 201 296 [7], some general information and assumptions are presented before formulating the Test Purposes themselves:

1) The following abbreviations are used:

   ANS Answer message

   CCP Connection Control Point

   CDP Charge Determination Point

CGP Charge Generation Point

CRI Charging Reference ID (NI plus RI)

CRP/CGP CRI CRI provided by the CRP/CGP

CRI*n*B CRI provided by Network No. *n* (n=1..7) on the path to the B-party

CRI*n*C CRI provided by Network No. *n* (n=1..2) on the path to the C-party

SSF CRI CRI provided by the SSF

SCF CRI CRI provided by the SCF

CRP Charge Registration Point

DNID Destination Identification field in a charging message (contains NI and RI)

NI Network ID

NI*n*B NI of Network No. *n* (n=1...7) on the path to the B-party

NI*n*C NI of Network No. *n* (n=1...2) on the path to the C-party

NW Network

OLE Originating Local Exchange

ONID Originating Identification field in a charging message (contains NI and RI)

RI Reference ID

The values for the "**delayUntilStart**" bit:

- "start tariffing, if it is not already started, without waiting for the "start" signal"
- "delay start of tariffing up to the receipt of the "start" signal"

are **replaced** by the short forms:

- "immediate start of tariffing"
- "delay start of tariffing"

2) In the tests the SSF operates as CDP or CCP, subject to SCF control. No tests for the SSF operating as CGP/CRP are provided here. This is done with the tests of other operations (e.g. AC, FCI). Also tariff variations are not tested here.

3) The SSF model "SSP on transit level with incoming and outgoing ISUP signalling and charging of the calling line is done at the originating local exchange (OLE)" is assumed (see clause 16.3 of EN 301 931-2 [2]). The partyToCharge is always reached via leg 1 (= controlling leg (O_BCSM)). The OLE is a combined CGP/CRP and the SSF provides the destination NI of the OLE where provision of "backward NI" is required from the SSF.

4) "Start of charging" for a connection is assumed to be defined by the transmission/reception of the ANSWER message. However it is indicated more generally as "SetUpConf" in the test descriptions.

5) It is assumed that ISUP signalling according to ES 201 296 [7] is implemented in the SSF. No network operator-specific signalling is taken into account.

6) Except for the case where a special IN number is called, the addressed B/C-parties belong to networks different from the network of the SSF.

7) The following assumptions regarding the **interpretation of ES 201 296** [7] are made for an SSF acting as a **connection control point** and being permitted to relay charge messages from destination CDPs/CCPs:

   - When the SSF receives an APM(crgt) message from a succeeding exchange indicating "delay until start", the SSF relays the message unchanged to the leg of the party to charge. The SSF does **not generate** an APM(start) message for this charging tariff in any phase of the connection, but relays APM(start) messages (and other charging messages) received from the destination CDP/CCP to the leg of the party to charge.

NOTE: It is assumed that the entity having set the "delay" condition for the APM(crgt) message is also responsible for generating the APM(start) message.

   - When the SSF receives an APM(crgt) message indicating "immediate tariff change" **before** having sent the "start of charging signal" to OLE, the SSF relays the APM(crgt) message to the leg of the party to charge, but changes the "immediate tariff change" into "delay until start", leaving unchanged all other fields of the relayed APM(crgt) message.

     The SSF is now in charge to **generate** an APM(start) message for this charging tariff after having sent the "start of charging signal" on the leg of the party to charge, but still relays charging messages received from the destination CDP/CCP to the leg of the party to charge.

   - When the SSF receives an APM(crgt) message indicating "immediate tariff change" **after** having sent the "start of charging signal" (ANS message) on the leg of the party to charge, the SSF relays the APM(crgt) message to the leg of the party to charge without a change. The SSF does **not generate** an APM(start) message for this charging tariff.

8) In some cases where several components are to be contained together in one TCAP message (usually including the initial SCI invoke component), the TC-CONTINUE request primitive is explicitly shown. This done because an SCI invoke component containing "charging control" information must be sent by the SCF in the context of a "connection" operation (CUE, CWA, CON).

9) It is assumed that the charging information received by an SSF acting as CDP is not modified by the SSF for reasons of bilateral network provider agreements (see clause 6.1b of ES 201 296 [7]).

6.6.24.2 Send Charging InformationTest Purposes

	IN3_A_BASIC_SCI_BV_01
Work item no.:	ITEM_BASIC_371
Purpose:	**Keywords**: SCI(No charging message, relayCharges=noIndication, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt(advice-of-charge only) unchanged. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CGP/CRP an APM(crgt, "advice-of-charge only") message received from the charging network via the succeeding exchange. Verify also that the SSF relays the APM(crga) message received from the CGP/CRP to the succeeding exchange.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296**: 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_02
Work item no.:	ITEM_BASIC_372
Purpose:	**Keywords**: SCI(No charging message, relayCharges=noIndication, relayAOC=TRUE, sSFdetermination=noDetermination), do not relay crgt, exception handling. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "noDetermination", does not relay to the CGP/CRP an APM(crgt, "subscriber charge") message received from the charging network via the succeeding exchange, and performs exception handling for the call.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, table 44 **ES 201 296**: 6.2, 6.2.2, 6.2.3, 6.2.5, 6.2.7
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) [TSPX_EXCEPT_CRGT_RELEASE] CP1_2?ReleaseReq L1?EntityReleased(leg2) [NOT TSPX_EXCEPT_CRGT_RELEASE] CP1_2?APM(crga("not accepted", ONID = SSF CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_EXCEPT_RELEASE_B(TSPX_EXCEPT_CRGT_RELEASE)

	IN3_A_BASIC_SCI_BV_03
Work item no.:	ITEM_BASIC_373
Purpose:	**Keywords**: SCI(No charging message, relayCharges=noIndication, relayAOC=TRUE, sSFdetermination=destinationRoutingAddr), relay crgt(advice-of-charge only) unchanged. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddr", does not generate an APM(crgt) message, but relays to the CGP/CRP an APM(crgt, "advice-of-charge only") message received from the charging network via the succeeding exchange, if the destination routing address passed from the SCF does not enable the SSF to determine the charging tariff.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296**: 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddress") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_04
Work item no.:	ITEM_BASIC_374
Purpose:	**Keywords**: SCI(No charging message, relayCharges=noIndication, relayAOC=TRUE, sSFdetermination=destinationRoutingAddr), generate crgt, act as CDP. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddr", sends an APM(crgt, "subscriber charge") message to the CRP/CGP (acts as CDP) if the destination routing address passed from the SCF enables the SSF to determine the charging tariff. Check also that the SSF accepts the APM(crga) message received from the CGP/CRP without a response.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296**: 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5
Selection Expr.:	SelCDPRoutingB
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR_CH)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddress") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", delayUntilStart = ?, valid charging tariff, ONID = SSF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI)) CP1_2!SetUpConf CP1_1?SetupResp NOTE: An APM(start) message may be received at CP1_1 after SetupResp, if the "delayUntilStart" bit in the APM(crgt) message is set to "1".
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_05
Work item no.:	ITEM_BASIC_375
Purpose:	**Keywords**: SCI(No charging message, relayCharges=noIndication, relayAOC=TRUE, sSFdetermination=destinationRoutingAddr), SSF does not generate crgt message, relays crgt(subscriber charge, immediate), sends start. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddr", decides that the charges have to be determined in a succeeding exchange and does not send an APM(crgt) message to the CRP/CGP, if the destination routing address passed from the SCF does not enable the SSF to determine the charging tariff. Check also that the SSF relays to the CGP/CRP an APM(crgt, "subscriber charge", "immediate start of tariffing") message received from the charging network via the succeeding exchange, unmodified except for setting delayUntilStart = "delay start of tariffing".
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296**: 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddress") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp CP1_1?APM(start({SSF NI}, ONID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_06
Work item no.:	ITEM_BASIC_376
Purpose:	**Keywords**: SCI(No charging message, relayCharges=noIndication, relayAOC=TRUE, sSFdetermination=destinationRoutingAddr), exception handling on timeout. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddr", sends an APM(crgt, "subscriber charge") message to the CRP/CGP (acts as CDP) if the destination routing address passed from the SCF enables the SSF to determine the charging tariff. Check also that the SSF performs exception handling after timeout of Tcrga (not having received an APM(crga) message upon the transmitted APM(crgt) message).
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296**: 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5, 6.1.7
Selection Expr.:	SelCDPRoutingB
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR_CH)

Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddress")
	L1!Continue
	L1!TC_CONTINUE
	CP1_2?SetUpReq
	CP1_1?APM(crgt("subscriber charge", delayUntilStart = ?, valid charging tariff, ONID = SSF CRI, DNID = OMIT))
	Do not send APM(crga), letting Tcrga expire in the SSF
	[TSPX_EXCEPT_CRGT_TOUT_RELEASE]
	  CP1_1?ReleaseReq
	  CP1_2?ReleaseReq
	[NOT TSPX_EXCEPT_CRGT_TOUT_RELEASE]
	  CP1_2!SetUpConf
	  CP1_1?SetupResp
	  POST_SCI_RELEASE_B

Pass criteria:	
Postamble:	None

	IN3_A_BASIC_SCI_BV_07
Work item no.:	ITEM_BASIC_377
Purpose:	**Keywords:** SCI(No charging message, relayCharges=noIndication, relayAOC=TRUE, sSFdetermination=destinationRoutingAddr), exception handling on crga(not accepted). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddr", sends an APM(crgt, "subscriber charge") message to the CRP/CGP (acts as CDP) if the destination routing address passed from the SCF enables the SSF to determine the charging tariff. Check also that the SSF performs exception handling after having received an APM(crga) message, indicating "not accepted".
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5, 6.1.7
Selection Expr.:	SelCDPRoutingB
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR_CH)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "destinationRoutingAddress") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", delayUntilStart = ?, valid charging tariff, ONID = SSF CRI, DNID = OMIT)) CP1_1!APM(crga("not accepted", ONID = CGP/CRP CRI, DNID = SSF CRI)) [TSPX_EXCEPT_CRGA_RELEASE] CP1_1?ReleaseReq CP1_2?ReleaseReq [NOT TSPX_EXCEPT_CRGA_RELEASE] CP1_2!SetUpConf CP1_1?SetupResp POST_SCI_RELEASE_B
Pass criteria:	
Postamble:	None

	IN3_A_BASIC_SCI_BV_08
Work item no.:	ITEM_BASIC_378
Purpose:	**Keywords**: SCI(No charging message, relayCharges=noIndication, relayAOC=TRUE, sSFdetermination=calledInNumber), generate crgt, act as CDP. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "calledInNumber", sends an APM(crgt, "subscriber charge") message on the CGP/CRP (acts as CDP) if the called IN number enables the SSF to determine the charging tariff. Check also that the SSF accepts the APM(crga) message received from the CGP/CRP without a response.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296**: 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5
Selection Expr.:	SelCDPINAddress
Preamble:	PRE_SCI_1(TSPX_SCI_IN_ADDR_CH)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "calledInNumber") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", delayUntilStart = ?, valid charging tariff, ONID = SSF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI)) CP1_2!SetUpConf CP1_1?SetupResp NOTE: An APM(start) message may be received at CP1_1 after SetupResp, if the "delayUntilStart" bit in the APM(crgt) message is set to "1".
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_09
Work item no.:	ITEM_BASIC_379
Purpose:	**Keywords:** SCI(No charging message, relayCharges=noIndication, relayAOC=FALSE, sSFdetermination=destinationRoutingAddr), do not relay crgt(advice-of-charge only). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC = "FALSE", sSFdetermination = "destinationRoutingAddr", does not relay to the CRP/CGP an APM(crgt, "advice-of-charge only") message received from the charging network via the succeeding exchange (the message is discarded without a response).
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.2, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "FALSE", sSFdetermination = "destinationRoutingAddress") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) Wait a while to ensure no response CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_10
Work item no.:	ITEM_BASIC_380
Purpose:	**Keywords**: SCI(No charging message, relayCharges=noIndication, relayAOC=OMIT, sSFdetermination=destinationRoutingAddr), do not relay crgt(advice-of-charge only). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noIndication", relayAOC omitted, sSFdetermination = "destinationRoutingAddr", does not relay to the CRP/CGP an APM(crgt, "advice-of-charge only") message received from the charging network via the succeeding exchange (the message is discarded without a response).
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296**: 6.2, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = OMIT, sSFdetermination = "destinationRoutingAddress") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) Wait a while to ensure no response CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_11
Work item no.:	ITEM_BASIC_381
Purpose:	**Keywords:** SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt unchanged, do not send start. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP an APM(crgt, "subscriber charge", "delay start of tariffing") message received from the charging network via the succeeding exchange, unmodified. Check also that the SSF relays the APM(crga) message related to the APM(crgt) to the charging network via the succeeding exchange, and does not send an APM(start) message to the CRP/CGP after having transmitted the ANS message.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296**: 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp Wait a while to check that the SSF does not send APM(start).
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_12
Work item no.:	ITEM_BASIC_382
Purpose:	**Keywords:** SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt unchanged up to "delayUntilStart", send start. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP an APM(crgt, "subscriber charge", "immediate start of tariffing") message received from the charging network via the succeeding exchange, unmodified except for setting delayUntilStart = "delay start of tariffing". Check also that the SSF relays the APM(crga) message related to the APM(crgt) to the charging network via the succeeding exchange, sends an APM(start) message to the CRP/CGP after having transmitted the ANS message, and accepts the APM(crga) message related to the APM(start) without a response.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp CP1_1?APM(start({NI1B}, ONID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_13
Work item no.:	ITEM_BASIC_383
Purpose:	**Keywords:** SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay APM(crgt) unchanged, relay APM(start). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP an APM(crgt, "subscriber charge", "delay start of tariffing") message received from the charging network via the succeeding exchange unmodified. Check also that the SSF relays the APM(crga) message related to the APM(crgt) to the charging network via the succeeding exchange and does not send an APM(start) message to the CRP/CGP after having transmitted the ANS message. Check also that the SSF relays an APM(start) message received from the charging network via the succeeding exchange to the CRP/CGP, and also relays the APM(crga) message related to the APM(start) to the succeeding exchange, addressing the starting CCP.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp CP1_2!APM(start({NI1B}, ONID = CRI1B)) CP1_1?APM(start({NI1B}, ONID = CRI1B)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_14
Work item no.:	ITEM_BASIC_384
Purpose:	**Keywords:** SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt unchanged up to "delayUntilStart" (2 times crgt(start tariffing) messages from different NWs received). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP two APM(crgt, "subscriber charge", "immediate start of tariffing") messages received from different charging networks via the succeeding exchange, unmodified except for setting delayUntilStart = "delay start of tariffing". Check also that the SSF relays the APM(crga) messages related to the APM(crgt) to the charging networks via the succeeding exchange, sends an APM(start) message to the CRP/CGP indicating both charging network operators after having transmitted the ANS message, and accepts the APM(crga) message related to the APM(start) without a response.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT1)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT1)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI2B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI2B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI2B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI2B)) CP1_2!SetUpConf CP1_1?SetupResp CP1_1?APM(start({NI1B, NI2B}, ONID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_15
Work item no.:	ITEM_BASIC_385
Purpose:	**Keywords:** SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay 6 crgt messages from different NWs, refuse 7th crgt message. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP six APM(crgt, "subscriber charge", "immediate start of tariffing") messages received from different charging networks via the succeeding exchange, unmodified except for setting delayUntilStart = "delay start of tariffing". Check also that the SSF relays the APM(crga) messages related to the APM(crgt) to the charging networks via the succeeding exchange. Verify also that the SSF sends an APM(crga, "not accepted") message to the charging network after receiving an APM(crgt, "subscriber charge") message from the seventh charging network.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)

	IN3_A_BASIC_SCI_BV_15
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI2B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI2B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI2B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI2B)) CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI3B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI3B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI3B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI3B)) CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI4B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI4B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI4B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI4B)) CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI5B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI5B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI5B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI5B)) CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI6B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI6B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI6B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI6B)) CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI7B, DNID = OMIT)) CP1_2?APM(crga("not accepted", ONID = SSF CRI, DNID = CRI7B))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_16
Work item no.:	ITEM_BASIC_386
Purpose:	**Keywords:** SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt, additional crgt after ANS with delayUntilStart bit deleted. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", having relayed and acknowledged an APM(crgt, "subscriber charge") message received from the charging network via the succeeding exchange before the ANS message, having also received and transmitted the ANS message and started the tariff at the CRP/CGP, relays an additional APM(crgt, "subscriber charge") message, where the "delayUntilStart" bit is not present in the charging control indicators, and relays the APM(crga) message related to the APM(crgt) message to the charging network via the succeeding exchange.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.3.2, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp CP1_2!APM(crgt("subscriber charge", delayUntilStart=OMIT, valid charging tariff, ONID = CRI1B, DNID = CGP/CRP CRI)) CP1_1?APM(crgt("subscriber charge", delayUntilStart=OMIT, valid charging tariff, ONID = CRI1B, DNID = CGP/CRP CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_17
Work item no.:	ITEM_BASIC_387
Purpose:	**Keywords:** SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt, relay aocrg(subscriber charge) after ANS with delayUntilStart bit deleted. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", having relayed and acknowledged an APM(crgt, "subscriber charge") message received from the charging network via the succeeding exchange before the ANS message, having also received and transmitted the ANS message and started the tariff at the CRP/CGP, relays an APM(aocrg, "subscriber charge") message, where the "delayUntilStart" bit is not present in the charging control indicators, and relays the APM(crga) message related to the APM(aocrg) message to the charging network via the succeeding exchange.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.2, 6.2.1.1, 6.2.1.3, 6.2.2, 6.2.3, 6.2.3.1, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp CP1_2!APM(aocrg("subscriber charge", delayUntilStart=OMIT, valid Add-on charge value, ONID = CRI1B, DNID = CGP/CRP CRI)) CP1_1?APM(aocrg("subscriber charge", delayUntilStart=OMIT, valid Add-on charge value, ONID = CRI1B, DNID = CGP/CRP CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_18
Work item no.:	ITEM_BASIC_388
Purpose:	**Keywords**: SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt, relay aocrg(advice-of-charge only) after ANS with delayUntilStart bit deleted. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", having relayed and acknowledged an APM(crgt, "subscriber charge") message received from the charging network via the succeeding exchange before the ANS message, having also received and transmitted the ANS message and started the tariff at the CRP/CGP, relays an APM(aocrg, "advice-of-charge only") message, where the "delayUntilStart" bit is not present in the charging control indicators, and relays the APM(crga) message related to the APM(aocrg) message to the charging network via the succeeding exchange.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296**: 6.2, 6.2.1.1, 6.2.1.3, 6.2.2, 6.2.3, 6.2.3.1, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp CP1_2!APM(aocrg("advice-of-charge only", delayUntilStart=OMIT, valid Add-on charge value, ONID = CRI1B, DNID = CGP/CRP CRI)) CP1_1?APM(aocrg("advice-of-charge only", delayUntilStart=OMIT, valid Add-on charge value, ONID = CRI1B, DNID = CGP/CRP CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_19
Work item no.:	ITEM_BASIC_389
Purpose:	**Keywords:** SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt, refuse aocrg before ANS. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", having relayed and acknowledged an APM(crgt, "subscriber charge") message received from the charging network via the succeeding exchange before the ANS message, sends an APM(crga, "not accepted") message to the charging network via the succeeding exchange, when an APM(aocrg) message has been received before the ANS message.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!APM(aocrg("advice-of-charge only", delayUntilStart=OMIT, valid Add-on charge value, ONID = CRI1B, DNID = CGP/CRP CRI)) CP1_2?APM(crga("not accepted", ONID = SSF CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_20
Work item no.:	ITEM_BASIC_390
Purpose:	**Keywords:** SCI(No charging message, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt unchanged up to "delayUntilStart" (2 crgt messages from different NWs, different "delayUntilStart" bits). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP an APM(crgt, "subscriber charge", "immediate start of tariffing") message and an APM(crgt, "subscriber charge", "delay start of tariffing") message, received from different charging networks via the succeeding exchange, unmodified except for setting delayUntilStart = "delay start of tariffing" in the first APM(crgt) message. Check also that the SSF relays the APM(crga) messages related to the APM(crgt) to the charging networks via the succeeding exchange. Check also that the SSF sends an APM(start) message to the CRP/CGP indicating the charging network operator of the first APM(crgt) message after having transmitted the ANS message, and accepts and relays the APM(crga) message related to the APM(start) without a response. Check also that the SSF does not send an APM(start) message for the second APM(crgt) message (because the originator of this message has requested "delay until start" and is responsible for initiating the APM(start) message). Verify that the SSF relays this APM(start) message, when received from the succeeding exchange, and also relays the related APM(crga) message.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI2B, DNID = CGP/CRP CRI)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI2B, DNID = CGP/CRP CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI2B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI2B)) CP1_2!SetUpConf CP1_1?SetupResp CP1_1?APM(start({NI1B}, ONID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI)) Wait a while to ensure that no further APM(start) message is received. CP1_2!APM(start({NI2B}, ONID = CRI2B)) CP1_1?APM(start({NI2B}, ONID = CRI2B)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI2B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI2B))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_21
Work item no.:	ITEM_BASIC_391
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination) related to C-party, relay crgt unchanged up to "delayUntilStart", send start (C-party). Verify that the SSF, having received an SCI invoke component related to a C-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP an APM(crgt, "subscriber charge", "immediate start of tariffing") message received from the charging network via the succeeding exchange, unmodified except for setting delayUntilStart = "delay start of tariffing". Verify also that the SSF sends an APM(start) message to the CRP/CGP, after having received the ANS message from the succeeding exchange (C-party).
Requirem. ref.:	11.42, 12.40, 16.6.2 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_2(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Connect(3,TSPX_SCI_DESTC1_ADDR) L1!TC_CONTINUE CP1_3?SetUpReq CP1_3!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI1C, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1C, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3!SetUpConf CP1_1?APM(start({NI1C}, ONID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_BC

	IN3_A_BASIC_SCI_BV_22
Work item no.:	ITEM_BASIC_392
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination) related to C-party, relay crgt (delay) unchanged, no start sent. Verify that the SSF, having received an SCI invoke component related to a C-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP an APM(crgt, "subscriber charge", "delay start of tariffing") message received from the charging network via the succeeding exchange (C-party) unmodified. Verify also that the SSF does not send an APM(start) message to the CRP/CGP, after having received the ANS message from the succeeding exchange (C-party). Check also that the SSF relays the APM(start) message received from the succeeding exchange (C-party) to the CRP/CGP, and relays the related APM(crga) message from the CRP/CGP to the succeeding exchange (C-party).
Requirem. ref.:	11.42, 12.40, 16.6.2 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_2(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Connect(3,TSPX_SCI_DESTC1_ADDR) L1!TC_CONTINUE CP1_3?SetUpReq CP1_3!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1C, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1C, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3!SetUpConf Wait a while to verify that no APM(start) message is sent by the SSF. CP1_3!APM(start({NI1C}, ONID = CRI1C)) CP1_1?APM(start({NI1C}, ONID = CRI1C)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_BC

	IN3_A_BASIC_SCI_BV_23
Work item no.:	ITEM_BASIC_393
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination) related to C-party, relay crgt (delay) unchanged, relay start, relay stop. Verify that the SSF, having received an SCI invoke component related to a C-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP an APM(crgt, "subscriber charge", "delay start of tariffing") message received from the charging network via the succeeding exchange (C-party) unmodified. Verify also that the SSF, having relayed APM(start) and related APM(crga) messages, relays the APM(stop) message received from the succeeding exchange (C-party) to the CRP/CGP, and relays the related APM(crga) message from the CRP/CGP to the succeeding exchange (C-party).
Requirem. ref.:	11.42, 12.40, 16.6.2 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.4, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_2(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Connect(3,TSPX_SCI_DESTC1_ADDR) L1!TC_CONTINUE CP1_3?SetUpReq CP1_3!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1C, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1C, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3!SetUpConf Wait a while to verify that no APM(start) message is sent by the SSF. CP1_3!APM(start({NI1C}, ONID = CRI1C)) CP1_1?APM(start({NI1C}, ONID = CRI1C)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) Wait a while CP1_3!APM(stop("call attempt charges not applicable", {NI1C}, ONID = CRI1C)) CP1_1?APM(stop("call attempt charges not applicable", {NI1C}, ONID = CRI1C)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_BC

	IN3_A_BASIC_SCI_BV_24
Work item no.:	ITEM_BASIC_394
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination) related to C-party, relay crgt (delay) unchanged, stop message initiated. Verify that the SSF, having received an SCI invoke component related to a C-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP an APM(crgt, "subscriber charge", "delay start of tariffing") message received from the charging network via the succeeding exchange (C-party) unmodified. Verify also that the SSF, having relayed APM(start) and related APM(crga) messages, sends an APM(stop) message to the CRP/CGP, and accepts the related APM(crga) message received from the CRP/CGP without a response, when the connection to the C-party is released.
Requirem. ref.:	11.42, 12.40, 16.6.2 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.4, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_2(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Connect(3,TSPX_SCI_DESTC1_ADDR) L1!TC_CONTINUE CP1_3?SetUpReq CP1_3!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1C, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1C, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3!SetUpConf Wait a while to verify that no APM(start) message is sent by the SSF. CP1_3!APM(start({NI1C}, ONID = CRI1C)) CP1_1?APM(start({NI1C}, ONID = CRI1C)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) CP1_3?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1C)) Wait a while CP1_3!ReleaseInd CP1_1?APM(stop("call attempt charges not applicable", {NI1C}, ONID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_BC

	IN3_A_BASIC_SCI_BV_25
Work item no.:	ITEM_BASIC_395
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=relay, relayAOC=FALSE, sSFdetermination=noDetermination) related to C-party, exception handling on timeout. Verify that the SSF, having received an SCI invoke component related to a C-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "FALSE", sSFdetermination = "noDetermination", sends an APM(crgt, "subscriber charge") message on the CRP/CGP (acts as CDP) if the destination routing address passed from the SCF enables the SSF to determine the charging tariff. Check also that the SSF performs exception handling after timeout of Tcrga (not having received an APM(crga) message upon the transmitted APM(crgt) message).
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5, 6.1.7
Selection Expr.:	SelCDPRoutingC
Preamble:	PRE_SCI_2(TSPX_SCI_DESTB1_ADDR)

Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "FALSE", sSFdetermination = "noDetermination")
	L1!Connect(3,TSPX_SCI_DESTC1_ADDR_CH)
	L1!TC_CONTINUE
	CP1_1?APM(crgt("subscriber charge", delayUntilStart = ?, valid charging tariff, ONID = SSF CRI, DNID = OMIT))
	Do not send APM(crga), wait a while
	[TSPX_EXCEPT_CRGT_TOUT_RELEASE]
	  CP1_3?ReleaseReq
	  L1?EntityReleased(leg3)
	[NOT TSPX_EXCEPT_CRGT_TOUT_RELEASE]
	  CP1_2!SetUpConf
	  CP1_1?SetupResp

Pass criteria:	
Postamble:	POST_SCI_EXCEPT_RELEASE_BC

	IN3_A_BASIC_SCI_BV_26
Work item no.:	ITEM_BASIC_396
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=relay, relayAOC=TRUE, sSFdetermination=noDetermination) related to C-party, exception handling on crga(not accepted). Verify that the SSF, having received an SCI invoke component related to a C-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", sends an APM(crgt, "subscriber charge") message to the CRP/CGP (acts as CDP) if the destination routing address passed from the SCF enables the SSF to determine the charging tariff. Check also that the SSF performs exception handling after having received an APM(crga) message, indicating "not accepted".
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5, 6.1.7
Selection Expr.:	SelCDPRoutingC
Preamble:	PRE_SCI_2(TSPX_SCI_DESTB1_ADDR)

Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "FALSE", sSFdetermination = "noDetermination")
	L1!Connect(3,TSPX_SCI_DESTC1_ADDR_CH)
	L1!TC_CONTINUE
	CP1_1?APM(crgt("subscriber charge", delayUntilStart = ?, valid charging tariff, ONID = SSF CRI, DNID = OMIT))
	CP1_1!APM(crga("not accepted", ONID = CGP/CRP CRI, DNID = SSF CRI))
	[TSPX_EXCEPT_CRGA_RELEASE]
	  CP1_3?ReleaseReq
	  L1?EntityReleased(leg3)
	[NOT TSPX_EXCEPT_CRGA_RELEASE]
	  CP1_2!SetUpConf
	  CP1_1?SetupResp

Pass criteria:	
Postamble:	POST_SCI_EXCEPT_RELEASE_BC

	IN3_A_BASIC_SCI_BV_27
Work item no.:	ITEM_BASIC_397
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=relay, relayAOC="FALSE", sSFdetermination=calledInNumber), generate crgt("subscriber charge") (act as CDP). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "relay", relayAOC = "FALSE", sSFdetermination = "calledInNumber", sends an APM(crgt, "subscriber charge") message on the CGP/CRP (acts as CDP) if the called IN number enables the SSF to determine the charging tariff. Check also that the SSF accepts the APM(crga) message received from the CGP/CRP without a response.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5
Selection Expr.:	SelCDPINAddress
Preamble:	PRE_SCI_1(TSPX_SCI_IN_ADDR_CH)
Test description:	L1!SCI(No charging message, relayCharges = "noIndication", relayAOC = "FALSE", sSFdetermination = "calledInNumber") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", delayUntilStart = ?, valid charging tariff, ONID = SSF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI)) CP1_2!SetUpConf CP1_1?SetupResp NOTE: An APM(start) message may be received at CP1_1 after SetupResp, if the "delayUntilStart" bit in the APM(crgt) message is set to "1".
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_28
Work item no.:	ITEM_BASIC_398
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=noRelay, relayAOC=TRUE, sSFdetermination=noDetermination), do not relay crgt, exception handling. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noRelay", relayAOC = "TRUE", sSFdetermination = "noDetermination", does not relay to the CRP/CGP an APM(crgt, "subscriber charge") message received from the charging network via the succeeding exchange, and performs exception handling for the call.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, table 44 **ES 201 296:** 6.2, 6.2.2, 6.2.3, 6.2.5, 6.2.7
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "noRelay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) [TSPX_EXCEPT_CRGT_RELEASE] CP1_2?ReleaseReq L1?EntityReleased(leg2) [NOT TSPX_EXCEPT_CRGT_RELEASE] CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_EXCEPT_RELEASE_B(TSPX_EXCEPT_CRGT_RELEASE)

	IN3_A_BASIC_SCI_BV_29
Work item no.:	ITEM_BASIC_399
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=noRelay, relayAOC=TRUE, sSFdetermination=noDetermination), relay crgt(advice-of-charge only) unchanged. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noRelay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays to the CRP/CGP an APM(crgt, "advice-of-charge only") message received from the charging network via the succeeding exchange.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "noRelay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_30
Work item no.:	ITEM_BASIC_400
Purpose:	**Keywords:** SCI(No charging msg, relayCharges=noRelay, relayAOC=FALSE, sSFdetermination=destinationRoutingAddr), do not relay crgt(advice-of-charge only). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating relayCharges = "noRelay", relayAOC = "FALSE", sSFdetermination = "destinationRoutingAddr", does not relay to the CRP/CGP an APM(crgt, "advice-of-charge only") message received from the charging network via the succeeding exchange (the message is discarded without a response).
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.2, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "noRelay", relayAOC = "FALSE", sSFdetermination = "destinationRoutingAddress") L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_2?APM(crga("not accepted", ONID = SSF CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_31
Work item no.:	ITEM_BASIC_401
Purpose:	**Keywords:** SCI(No charging msg, no charging control), do not relay crgt, exception handling. Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and not containing chargingControl, does not relay to the CRP/CGP an APM(crgt, "subscriber charge") message received from the charging network via the succeeding exchange, and performs exception handling for the call (the message is discarded without a response).
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69, table 44 **ES 201 296:** 6.2, 6.2.2, 6.2.3, 6.2.5, 6.2.7
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)

Test description:	L1!SCI(No charging message, no charging control)
	L1!Continue
	L1!TC_CONTINUE
	CP1_2?SetUpReq
	CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT))
	[TSPX_EXCEPT_CRGT_RELEASE]
	  CP1_2?ReleaseReq
	  L1?EntityReleased(leg2)
	[NOT TSPX_EXCEPT_CRGT_RELEASE]
	  CP1_2!SetUpConf
	  CP1_1?SetupResp

Pass criteria:	
Postamble:	POST_SCI_EXCEPT_RELEASE_B

	IN3_A_BASIC_SCI_BV_32
Work item no.:	ITEM_BASIC_402
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), no charging control), APM(crgt), ENC(crga). Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message, containing no charging control, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF does not send an APM(start) message after receiving the ANS message from the succeeding exchange.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!SetUpConf CP1_1?SetupResp Wait a while to ensure that the SSF does not send an APM(start) message.
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_33
Work item no.:	ITEM_BASIC_403
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "immediate start of tariffing"), charging control omitted), APM(crgt), ENC(crga), no APM(start). Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "immediate start of tariffing") message, with charging control omitted, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "immediate start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF does not send an APM(start) message after receiving the ANS message from the succeeding exchange.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!SetUpConf CP1_1?SetupResp Wait a while to ensure that the SSF does not send an APM(start) message.
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_34
Work item no.:	ITEM_BASIC_404
Purpose:	**Keywords:** SCI(crgt("advice-of-charge only"), charging control omitted), APM(crgt), ENC(crga). Verify that the SSF, having received an SCI invoke component, containing a crgt("advice-of-charge only") message, with charging control omitted, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "advice-of-charge only") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF does not send an APM(start) message after receiving the ANS message from the succeeding exchange.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("advice-of-charge only", "immediate start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("advice-of-charge only", "immediate start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!SetUpConf CP1_1?SetupResp Wait a while to ensure that the SSF does not send an APM(start) message.
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_35
Work item no.:	ITEM_BASIC_405
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted), APM(crgt), no RNC(crga), no APM(start). Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message, with charging control omitted, and having not been requested for notification of charging event(crga), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and does not send an APM(start) message after receiving the ANS message from the succeeding exchange.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1!Continue CP1_2!SetUpConf CP1_1?SetupResp Wait a while to ensure that no APM(start) message is sent by the SSF.
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_36
Work item no.:	ITEM_BASIC_406
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "immediate start of tariffing"), charging control omitted), APM(crgt), no RNC(crga), APM(start). Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "immediate start of tariffing") message, with charging control omitted, and having not been requested for notification of charging event(crga), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an APM(start) message after receiving the ANS message from the succeeding exchange.
Requirem. ref.:	11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1!Continue CP1_2!SetUpConf CP1_1?SetupResp CP1_1?APM(start({SCF NI}, ONID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_37
Work item no.:	ITEM_BASIC_407
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted), APM(crgt), ENC(crga), no relay of crgt(subscriber charge) before ANS. Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message, with charging control omitted, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF does not relay an APM(crgt, "subscriber charge") message received from the succeeding exchange before the ANS message and performs exception handling.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5, 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5, 6.2.7
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) [TSPX_EXCEPT_CRGT_RELEASE] CP1_2?ReleaseReq L1?EntityReleased(leg2) [NOT TSPX_EXCEPT_CRGT_RELEASE] CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_EXCEPT_RELEASE_B

	IN3_A_BASIC_SCI_BV_38
Work item no.:	ITEM_BASIC_408
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted), APM(crgt), ENC(crga), no relay of crgt(subscriber charge) after ANS. Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message, with charging control omitted, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF does not relay an APM(crgt, "subscriber charge") message received from the succeeding exchange after the ANS message and performs exception handling.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5, 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5, 6.2.7
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)

Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted)
	L1!RNC(leg1, crga, interrupted)
	L1!Continue
	L1!TC_CONTINUE
	CP1_2?SetUpReq
	CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT))
	CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI))
	L1?ENC(leg1, crga)
	L1!Continue
	CP1_2!SetUpConf
	CP1_1?SetupResp
	CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT))
	[TSPX_EXCEPT_CRGT_RELEASE]
	  CP1_2?ReleaseReq
	  L1?EntityReleased(leg2)
	[NOT TSPX_EXCEPT_CRGT_RELEASE]
	  No response

Pass criteria:	
Postamble:	POST_SCI_EXCEPT_RELEASE_B

	IN3_A_BASIC_SCI_BV_39
Work item no.:	ITEM_BASIC_409
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted), APM(crgt), ENC(crga), no relay of APM(aocrg), exception handling. Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message and indicating relayCharges = "OMIT", relayAOC = "OMIT", sSFdetermination = "OMIT", and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF does not relay an APM(aocrg) message received from the succeeding exchange after the ANS message and performs exception handling.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5, 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5, 6.2.7
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)

Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted)
	L1!RNC(leg1, crga, interrupted)
	L1!Continue
	L1!TC_CONTINUE
	CP1_2?SetUpReq
	CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT))
	CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI))
	L1?ENC(leg1, crga)
	L1!Continue
	CP1_2!SetUpConf
	CP1_1?SetupResp
	CP1_2!APM(aocrg("subscriber charge", "immediate start of tariffing", valid Add-on charge value, ONID = CRI1B, DNID = OMIT))
	[TSPX_EXCEPT_AOCRG_RELEASE]
	  CP1_2?ReleaseReq
	  L1?EntityReleased(leg2)
	[NOT TSPX_EXCEPT_AOCRG_RELEASE]
	  No response

Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_40
Work item no.:	ITEM_BASIC_410
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted), APM(crgt), ENC(crga), no relay of crgt(advice-of-charge only). Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message and indicating relayCharges = "OMIT", relayAOC = "OMIT", sSFdetermination = "OMIT", and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF does not relay an APM(crgt, "advice-of-charge only") message received from the succeeding exchange before the ANS message and continues call handling.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5, 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!APM(crgt("advice-of-charge only", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crga("not accepted", ONID = SSF CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_41
Work item no.:	ITEM_BASIC_411
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted) repeated before ANS, two times APM(crgt), ENC(crga). Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message, with charging control omitted, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF, having received another SCI invoke component with the same contents except for a new charging tariff before the ANS message, and having again been requested for notification of charging event(crga, interrupted), sends the corresponding APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the exchange of the party to be charged. Verify also that the SSF does not send an APM(start) message after receiving the ANS message from the succeeding exchange.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid new charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid new charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!SetUpConf CP1_1?SetupResp Wait a while to ensure that the SSF does not send an APM(start) message.
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_42
Work item no.:	ITEM_BASIC_412
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted), APM(crgt), ENC(crga), repeat SCI/APM(crgt)/ENC(crga) after ANS. Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message, with charging control omitted, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF does not send an APM(start) message after receiving the ANS message from the succeeding exchange. Verify also that the SSF, having received another SCI invoke component related to the B-party indicating a new charging tariff and relayCharges = "OMIT", relayAOC = "OMIT", sSFdetermination = "OMIT" after the ANS message, and having again been requested for notification of charging event(crga, interrupted), sends the corresponding APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the exchange of the party to be charged.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.3, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!SetUpConf CP1_1?SetupResp Wait a while to ensure that the SSF does not send an APM(start) message. L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid new charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid new charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_43
Work item no.:	ITEM_BASIC_413
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted), APM(crgt), no APM(crga) received, ENC(timeout tcrga). Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message, with charging control omitted, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged and sends an EventNotificationCharging invoke component indicating timeout, when having received no APM(crga) message from the CRP/CGP.
Requirem. ref.:	11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) Wait to let Tcrga expire L1?ENC(timeout Tcrga) L1!Continue CP1_2!SetUpConf CP1_1?SetupResp
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_44
Work item no.:	ITEM_BASIC_414
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted), APM(crgt), APM(crga, "not accepted"), ENC(crga). Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message, with charging control omitted, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged and sends an EventNotificationCharging invoke component containing the received crga message, when having received an APM(crga) message from the CRP/CGP indicating "not accepted".
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("not accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!SetUpConf CP1_1?SetupResp Wait a while to ensure that the SSF does not send an APM(start) message.
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_45
Work item no.:	ITEM_BASIC_415
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), charging control omitted), APM(crgt), ENC(crga), APM(start), ENC(crga), APM(stop), ENC(crga). Verify that the SSF, having received an SCI invoke component, containing a crgt("subscriber charge", "delay start of tariffing") message, with charging control omitted, and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Check also that the SSF sends an APM(start) message to the CRP/CGP, related to the own network, when having received an SCI(start) invoke component from the SCF, accepts the related APM(crga) message received from the CRP/CGP, and, having been also requested for notification, sends an EventNotificationCharging invoke component containing the crga message to the SCF. Check furthermore that the SSF sends an APM(stop) message to the CRP/CGP, related to the own network, when having received an SCI(stop) invoke component from the SCF, accepts the related APM(crga) message received from the CRP/CGP, and, having been also requested for notification, sends an EventNotificationCharging invoke component containing the crga message to the SCF.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!RequestReportBCSMEvent(2,interrupted,oAnswer) L1!Continue CP1_2!SetUpConf L1?EventReportBCSM(leg2, oAnswer) L1!SCI(start({SCF NI}, ONID = SCF CRI), chargingControl = OMIT) L1!RNC(leg1, crga, interrupted) L1!Continue CP1_1?SetupResp CP1_1?APM(start({SCF NI}, ONID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue Wait a while L1!SCI(stop("call attempt charges not applicable", {SCF NI}, ONID = SCF CRI), chargingControl = OMIT) L1!RNC(leg1, crga, monitorMode) CP1_1?APM(stop("call attempt charges not applicable", {SCF NI}, ONID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga)
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_46
Work item no.:	ITEM_BASIC_416
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "immediate start of tariffing"), relayCharges =relay, relayAOC=TRUE, sSFdetermination=noDetermination), APM(crgt), APM(crga), ENC(crga), SSF relays APM(crgt)/APM(crga). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, containing a crgt("subscriber charge", "immediate start of tariffing") message and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF relays an APM(crgt,"delay start of tariffing") message received from the succeeding exchange to the CRP/CGP, unchanged, and relays the APM(crga) message received from the CRP/CGP to the succeeding exchange unchanged.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5, 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), relayCharges =relay, relayAOC=TRUE, sSFdetermination=noDetermination) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf CP1_1?SetupResp CP1_2!APM(start({NI1B}, ONID = CRI1B)) CP1_1?APM(start({NI1B}, ONID = CRI1B)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_47
Work item no.:	ITEM_BASIC_417
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), relayCharges =relay, relayAOC=TRUE, sSFdetermination=noDetermination), APM(crgt), APM(crga), ENC(crga), SSF relays APM(crgt)/APM(crga), SSF sends APM(start). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, containing a crgt("subscriber charge", "delay start of tariffing") message and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", and having been requested for notification of charging event(crga, interrupted), sends an APM(crgt, "subscriber charge", "delay start of tariffing") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(crga) message from the CRP/CGP. Verify also that the SSF relays an APM(crgt, "immediate start of tariffing") message received from the succeeding exchange to the CRP/CGP, unchanged except for replacing the delayUntilStart bit value of "immediate start of tariffing" by "delay start of tariffing", and relays the APM(crga) message received from the CRP/CGP to the succeeding exchange unchanged. Check furthermore that the SSF sends an APM(start) message to the CRP/CGP, related to the network of the relayed APM(crgt) message, when having received the ANS message from the succeeding exchange, and relays the related APM(crga) message to the charging network. Check also that the SSF sends an APM(start) message to the CRP/CGP, related to the own network, when having received an SCI(start) invoke component from the SCF, and accepts the related APM(crga) message received from the CRP/CGP.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5, 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.4, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), relayCharges =relay, relayAOC=TRUE, sSFdetermination=noDetermination) L1!RNC(leg1, crga, interrupted) LL1!RequestReportBCSMEvent(2,interrupted,oAnswer) 1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf L1?EventReportBCSM(leg2, oAnswer) L1!SCI(start({SCF NI}, ONID = SCF CRI), chargingControl = OMIT) L1!RNC(leg1, crga, interrupted) L1!Continue CP1_1?SetupResp CP1_1?APM(start({NI1B}, ONID = SSF CRI)) CP1_1?APM(start({SCF NI}, ONID = SCF CRI)) NOTE 1: Depending on the implementation, these 2 messages may be received in any order, or a single APM(start) message may be received, indicating both NI tariffs to be started. CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SSF CRI)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) NOTE 2: If only one APM(start) message was received, a single APM(crga) message is sent.
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_48
Work item no.:	ITEM_BASIC_418
Purpose:	**Keywords:** SCI(aocrg("subscriber charge"), no charging control), APM(aocrg), ENC(crga). Verify that the SSF, having received an SCI invoke component in the call connection phase, containing an aocrg("subscriber charge") message without charging control, and having been requested for notification of charging event(crga, interrupted), sends an APM(aocrg, "subscriber charge") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(aocrg) message from the CRP/CGP.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.2, 6.1.2.4, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!RequestReportBCSMEvent(2,interrupted,oAnswer) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!SetUpConf L1?EventReportBCSM(leg2, oAnswer) L1!SCI(aocrg("subscriber charge", "immediate start of tariffing", valid Add-on charge value, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue CP1_1?SetupResp CP1_1?APM(aocrg("subscriber charge", "immediate start of tariffing", valid Add-on charge value, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_49
Work item no.:	ITEM_BASIC_419
Purpose:	**Keywords:** SCI(aocrg("advice-of-charge only"), no charging control), APM(aocrg), ENC(crga). Verify that the SSF, having received an SCI invoke component in the call connection phase, containing an aocrg("advice-of-charge only") message without charging control, and having been requested for notification of charging event(crga, interrupted), sends an APM(aocrg, "advice-of-charge only") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(aocrg) message from the CRP/CGP.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.2, 6.1.2.4, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!RequestReportBCSMEvent(2,interrupted,oAnswer) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!SetUpConf L1?EventReportBCSM(leg2, oAnswer) L1!SCI(aocrg("advice-of-charge only", "immediate start of tariffing", valid Add-on charge value, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue CP1_1?SetupResp CP1_1?APM(aocrg("advice-of-charge only", "immediate start of tariffing", valid Add-on charge value, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_50
Work item no.:	ITEM_BASIC_420
Purpose:	**Keywords**: SCI(No charging message, relayCharges =relay, relayAOC=TRUE, sSFdetermination=noDetermination), SSF relays APM(crgt)/APM(crga), SCI(aocrg("subscriber charge")), APM(aocrg), APM(crga), ENC(crga), SSF relays APM(aocrg)/APM(crga). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, containing no charging message and indicating relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination", relays APM(crgt)/APM(crga) in the call setup phase. Verify also that the SSF, having received an SCI invoke component in the call connection phase, containing an aocrg("subscriber charge") message without charging control, and having been requested for notification of charging event(crga, interrupted), sends an APM(aocrg, "subscriber charge") message to the party to be charged, and sends an EventNotificationCharging invoke component containing the received crga message, after having received an APM(aocrg) message from the CRP/CGP. Verify also that the SSF relays to the CRP/CGP an APM(aocrg, "subscriber charge") message received from the succeeding exchange after ANS, unchanged, and relays the APM(crga) message received from the CRP/CGP to the succeeding exchange unchanged.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69, figure 70, table 44 **ES 201 296**: 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.2.4, 6.1.4, 6.1.5, 6.2, 6.2.1.1, 6.2.1.3, 6.2.2, 6.2.3, 6.2.3.1, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(No charging message, relayCharges = "relay", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!RequestReportBCSMEvent(2,interrupted,oAnswer) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2!SetUpConf L1?EventReportBCSM(leg2, oAnswer) L1!SCI(aocrg("advice-of-charge only", "immediate start of tariffing", valid Add-on charge value, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue CP1_1?SetupResp CP1_1?APM(aocrg("advice-of-charge only", "immediate start of tariffing", valid Add-on charge value, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) L1!Continue CP1_2!APM(aocrg("subscriber charge", delayUntilStart=OMIT, valid Add-on charge value, ONID = CRI1B, DNID = OMIT)) CP1_1?APM(aocrg("subscriber charge", delayUntilStart=OMIT, valid Add-on charge value, ONID = CRI1B, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B)) CP1_2?APM(crga("accepted", ONID = CGP/CRP CRI, DNID = CRI1B))
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BV_51
Work item no.:	ITEM_BASIC_421
Purpose:	**Keywords**: RNC(crgt("subscriber charge")), APM(crgt("subscriber charge", "immediate start of tariffing", destination Network ID = CRP/CGP)), ENC(crgt), SCI(crga), SCI(crgt). Verify that the SSF, having been requested to send notification on a crgt("subscriber charge") message received from the succeeding exchange, sends a corresponding ENC invoke component to the SCF, when receiving an APM(crgt("subscriber charge", "immediate start of tariffing", destination Network ID = CRP/CGP)) message from the succeeding exchange. Check also that the SSF sends an APM(crga) message to the succeeding exchange, when receiving an SCI(crga) invoke component from the SCF. Check furthermore that the SSF sends an APM(crgt, "subscriber charge") message to the CRP/CGP, after having received an SCI(crgt("subscriber charge", "immediate start of tariffing", destination Network ID = CRP/CGP)) invoke component from the SCF. Check also that the SSF sends an ENC(crga) invoke component to the SCF, when having been requested to send corresponding notification, and having actually received the APM(crga) message from the CRP/CGP. Verify also that the SSF does not send an APM(start) message after receiving the ANS message from the succeeding exchange.
Requirem. ref.:	11.23, 11.27, 11.42, 12.40, 16.6.2, figure 69 **ES 201 296**: 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5, 6.2, 6.2.1.1, 6.2.2, 6.2.3, 6.2.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!RNC(leg2, crgt, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = CRI1B, DNID = OMIT)) L1?ENC(leg2, crgt) L1!SCI(crga("accepted", ONID = SCF CRI, DNID = CRI1B), chargingControl = OMIT) L1!SCI(crgt("subscriber charge", "immediate start of tariffing", same charging tariff as received, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?APM(crga("accepted", ONID = SCF CRI, DNID = CRI1B)) CP1_1?APM(crgt("subscriber charge", "immediate start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) NOTE: The APM(crgt) and APM(crga) messages may be received in any order. CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1?ENC(leg1, crga) CP1_2!SetUpConf CP1_1?SetUpResp Wait a while to ensure that the SSF sends no APM(start) message.
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BI_01
Work item no.:	ITEM_BASIC_422
Purpose:	**Keywords:** SCI(Invalid legID, no charging message, relayCharges=noIndication, relayAOC=TRUE, sSFdetermination=noDetermination), SCI return error(UnknownLegID). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, not containing a ChargingTariffMessageType and indicating an unknown leg ID for the partyToBeCharged, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "noDetermination", sends an SCI return error component to the SCF indicating error "unknownLegID".
Requirem. ref.:	11.42, 14.3
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(leg=3, No charging message, relayCharges = "noIndication", relayAOC = "TRUE", sSFdetermination = "noDetermination") L1!Continue L1!TC_CONTINUE L1?SCI return error(UnknownLegID)
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BI_02
Work item no.:	ITEM_BASIC_423
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), invalid destination network identification, relayCharges =OMIT, relayAOC=OMIT, sSFdetermination=OMIT), SCI return error(unexpectedDataValue). Verify that the SSF, having received an SCI invoke component related to a B-party to be called, containing a crgt("subscriber charge", "delay start of tariffing") message and indicating invalid origination network identification, relayCharges = "OMIT", relayAOC = "OMIT", sSFdetermination = "OMIT", sends to the SCF an SCI return error component indicating error "unexpectedDataValue".
Requirem. ref.:	11.42, 14.3
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = invalid origination network identification, DNID = OMIT), charging control omitted) L1!Continue L1!TC_CONTINUE L1?SCI return error(unexpectedDataValue)
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BI_03
Work item no.:	ITEM_BASIC_424
Purpose:	**Keywords:** SCI(aocrg("subscriber charge"), invalid origination network identification, no charging control), SCI return error(unexpectedDataValue). Verify that the SSF, having received in the call connection phase an SCI invoke component, containing an aocrg("subscriber charge") message with an invalid origination network identification and without charging control, sends to the SCF an SCI return error component indicating error "unexpectedDataValue".
Requirem. ref.:	11.42, 12.40, 14.3 **ES 201 296:** 6.1, 6.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetupResp L1!SCI(aocrg("subscriber charge", valid Add-on charge value, ONID = invalid origination network identification, DNID = OMIT), charging control omitted) L1?SCI return error(unexpectedDataValue)
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BO_01
Work item no.:	ITEM_BASIC_425
Purpose:	**Keywords:** SCI(crgt), APM(crgt), do not accept repeated SCI(crgt) before receiving APM(crga) for the first crgt message. Verify that the SSF, having processed an SCI(crgt) invoke component, sends an SCI return error component to the SCF indicating error "unexpectedComponentSequence" or "taskRefused", when receiving a second SCI(crgt) invoke component before receiving the APM(crga) message from the CRP/CGP.
Requirem. ref.:	11.27, 11.42, 12.40, 14.3, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) L1!SCI(crgt("subscriber charge", "delay start of tariffing", other charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1?SCI return error("unexpectedComponentSequence" or "taskRefused")
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BO_02
Work item no.:	ITEM_BASIC_426
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), no charging control), APM(crgt), do not accept SCI(start) before receiving APM(crga). Verify that the SSF, having processed an SCI(crgt) invoke component, sends an SCI return error component to the SCF indicating error "unexpectedComponentSequence" or "taskRefused", when receiving an SCI(start) invoke component before receiving the APM(crga) message from the CRP/CGP.
Requirem. ref.:	11.27, 11.42, 12.40, 14.3, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) L1!SCI(start({SCF NI}, ONID = SCF CRI), chargingControl = OMIT) L1?SCI return error("unexpectedComponentSequence" or "taskRefused")
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BO_03
Work item no.:	ITEM_BASIC_427
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), no charging control), do not accept SCI(start) before ANS. Verify that the SSF, having processed an SCI(crgt) invoke component, sends an SCI return error component to the SCF indicating error "unexpectedComponentSequence" or "taskRefused", when receiving an SCI(start) invoke component after receiving the APM(crga) message from the CRP/CGP, but before receiving and transmitting the ANS message.
Requirem. ref.:	11.27, 11.42, 12.40, 14.3, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) L1!SCI(start({SCF NI}, ONID = SCF CRI), chargingControl = OMIT) L1?SCI return error("unexpectedComponentSequence" or "taskRefused")
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BO_04
Work item no.:	ITEM_BASIC_428
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), no charging control), do not accept SCI(stop) after ANS but before SCI(start). Verify that the SSF, having processed an SCI(crgt) invoke component, sends an SCI return error component to the SCF indicating error "unexpectedComponentSequence" or "taskRefused", when receiving an SCI(stop) invoke component before having initiated the start procedure.
Requirem. ref.:	11.27, 11.42, 12.40, 14.3, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) CP1_2!SetUpConf CP1_1?SetUpResp L1!SCI(stop("call attempt charges not applicable", {SCF NI}, ONID = SCF CRI), chargingControl = OMIT) L1?SCI return error("unexpectedComponentSequence" or "taskRefused")
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

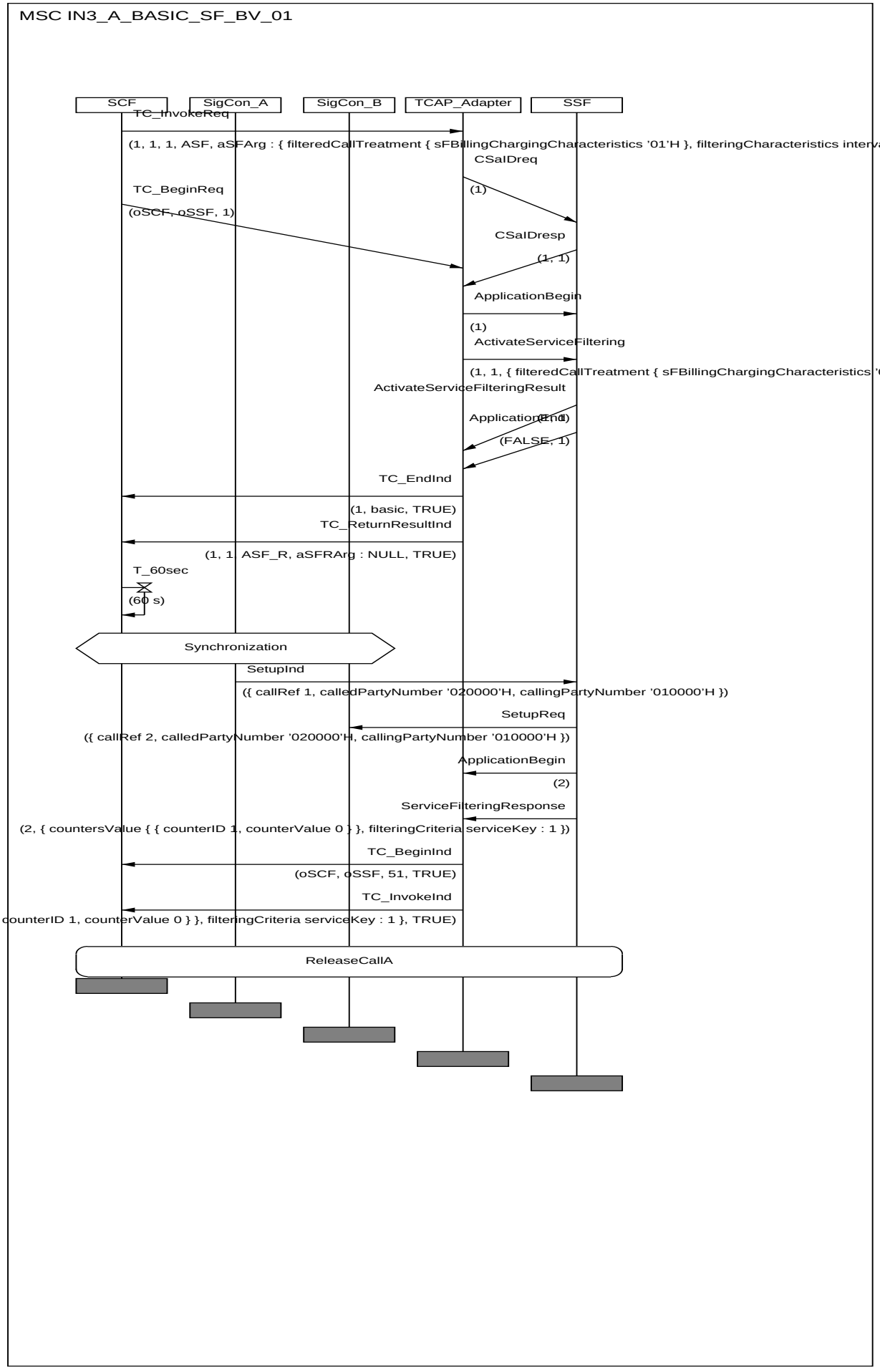
	IN3_A_BASIC_SCI_BO_05
Work item no.:	ITEM_BASIC_429
Purpose:	**Keywords:** SCI(crgt("subscriber charge", "delay start of tariffing"), no charging control), do not accept SCI(stop) after ANS, after SCI(start) but before APM(crga(start)). Verify that the SSF, having processed an SCI(crgt) invoke component, sends an SCI return error component to the SCF indicating error "unexpectedComponentSequence" or "taskRefused", when receiving an SCI(stop) invoke component before receiving acknowledgement for the procedure initiated by SCI(start).
Requirem. ref.:	11.42, 12.40, 14.3, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!SCI(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_1?APM(crgt("subscriber charge", "delay start of tariffing", valid charging tariff, ONID = SCF CRI, DNID = OMIT)) CP1_1!APM(crga("accepted", ONID = CGP/CRP CRI, DNID = SCF CRI)) CP1_2!SetUpConf CP1_1?SetUpResp L1!SCI(start({SCF NI}, ONID = SCF CRI), chargingControl = OMIT) CP1_1?APM(start({SCF NI}, ONID = SSF CRI)) L1!SCI(stop("call attempt charges not applicable", {SCF NI}, ONID = SCF CRI), chargingControl = OMIT) L1?SCI return error("unexpectedComponentSequence" or "taskRefused")
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

	IN3_A_BASIC_SCI_BO_06
Work item no.:	ITEM_BASIC_430
Purpose:	**Keywords:** SCI(aocrg("subscriber charge"), no charging control) in the call setup phase, SCI return error("unexpectedComponentSequence" or "taskRefused"). Verify that the SSF, having received in the call setup phase an SCI invoke component, containing an aocrg("subscriber charge") message without charging control, sends an SCI return error component to the SCF, indicating error "unexpectedComponentSequence" or "taskRefused".
Requirem. ref.:	11.27, 11.42, 12.40, 14.3, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!RNC(leg1, crga, interrupted) L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq L1!SCI(aocrg("subscriber charge", "immediate start of tariffing", valid Add-on charge value, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1?SCI return error("unexpectedComponentSequence" or "taskRefused")
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

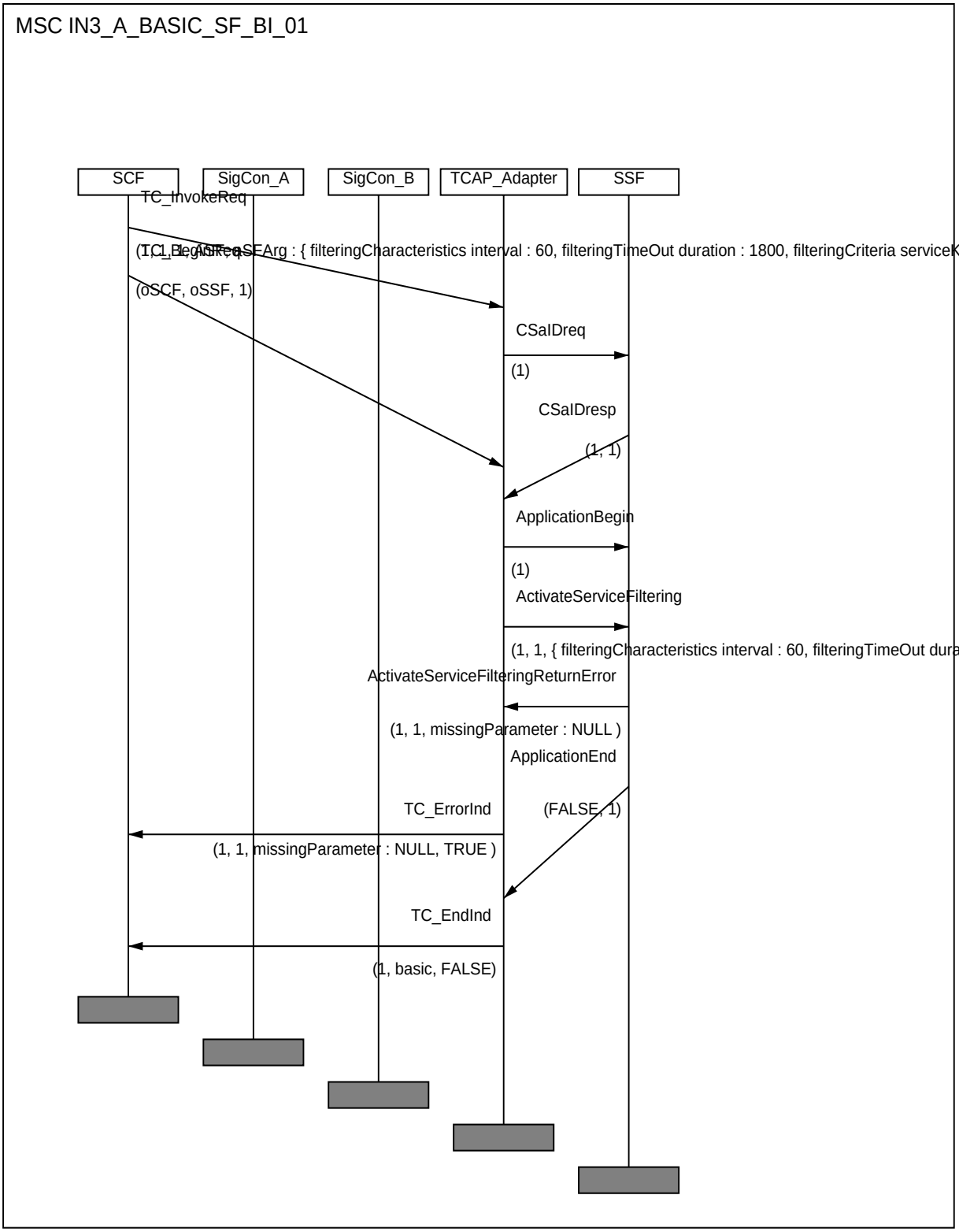
	IN3_A_BASIC_SCI_BO_07
Work item no.:	ITEM_BASIC_431
Purpose:	**Keywords:** SCI(aocrg("subscriber charge")) in the call connection phase, repeated before APM(crga). Verify that the SSF, having received an SCI invoke component in the call connection phase, containing an aocrg("subscriber charge") message and indicating relayCharges = "OMIT", relayAOC = "OMIT", sSFdetermination = "OMIT", sends an APM(aocrg, "subscriber charge") message to the party to be charged. Check also that the SSF sends to the SCF an SCI return error component, indicating error "unexpectedComponentSequence" or "taskRefused", when having received another SCI(aocrg) invoke component before having received an APM(crga) message from the CRP/CGP.
Requirem. ref.:	11.27, 11.42, 12.40, 14.3, 16.6.2, figure 69 **ES 201 296:** 6.1, 6.1.1, 6.1.1.1, 6.1.2, 6.1.4, 6.1.5
Selection Expr.:	
Preamble:	PRE_SCI_1(TSPX_SCI_DESTB1_ADDR)
Test description:	L1!Continue L1!TC_CONTINUE CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp L1!SCI(aocrg("subscriber charge", "immediate start of tariffing", valid Add-on charge value, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1!SCI(aocrg("subscriber charge", "immediate start of tariffing", valid Add-on charge value, ONID = SCF CRI, DNID = OMIT), charging control omitted) L1?SCI return error("unexpectedComponentSequence" or "taskRefused")
Pass criteria:	
Postamble:	POST_SCI_RELEASE_B

6.6.25 Service Filtering (SF) procedure

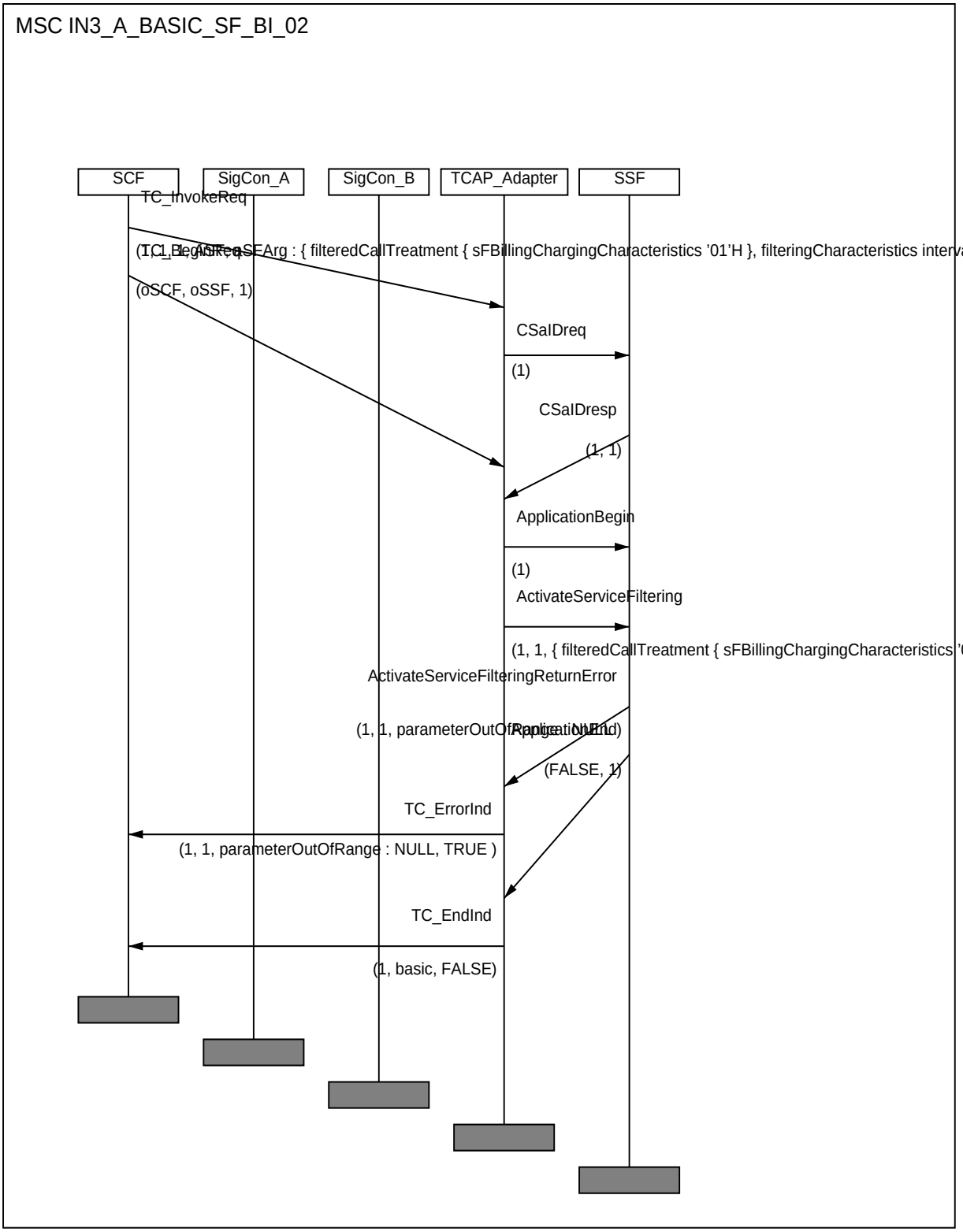
	IN3_A_BASIC_SF_BV_01
Work item no.:	ITEM_BASIC_121
IN2 Ref(tmp)	IN2_A_BASIC_SF_CA_01
Purpose:	Verify that the SSF having received an **ActivateServiceFiltering** invoke component containing mandatory parameters only (**filteredCallTreatment** including sFBillingChargingCharacteristics only; **filteringCharacteristics** (being interval); **filteringTimeOut** (being duration); and **filteringCriteria** (being serviceKey)) accepts the call being initiated after the filteringCharacteristics interval expires, and issues a **ServiceFilteringResponse** invoke component with parameters: countersValue (including 1 counterAndValue) and filteringCriteria (being serviceKey).
Requirements refs	8.1, 9.3.2, 11.1.1, 11.1.1.1.1, 11.1.3.1, 11.44.1, 11.44.1.1.1, 11.44.2.1
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 IN3_A_BASIC_SF_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
TC_InvokeReq
(1, 1, 1, ASF, aSFArg : { filteredCallTreatment { sFBillingChargingCharacteristics '01'H }, filteringCharacteristics interval : 60, filteringTimeO
CSalDreq
TC_BeginReq
(1)
(oSCF, oSSF, 1)
CSalDresp
(1, 1)
ApplicationBegin
(1)
ActivateServiceFiltering
(1, 1, { filteredCallTreatment { sFBillingChargingCharacteristics '01'H }, filteringChara
ActivateServiceFilteringResult
ApplicationEnd
(1)
(FALSE, 1)
TC_EndInd
(1, basic, TRUE)
TC_ReturnResultInd
(1, 1 ASF_R, aSFRArg : NULL, TRUE)
T_60sec
(60 s)
Synchronization
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(2)
ServiceFilteringResponse
(2, { countersValue { { counterID 1, counterValue 0 } }, filteringCriteria serviceKey : 1 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
{ countersValue { { counterID 1, counterValue 0 } }, filteringCriteria serviceKey : 1 }, TRUE)
ReleaseCallA

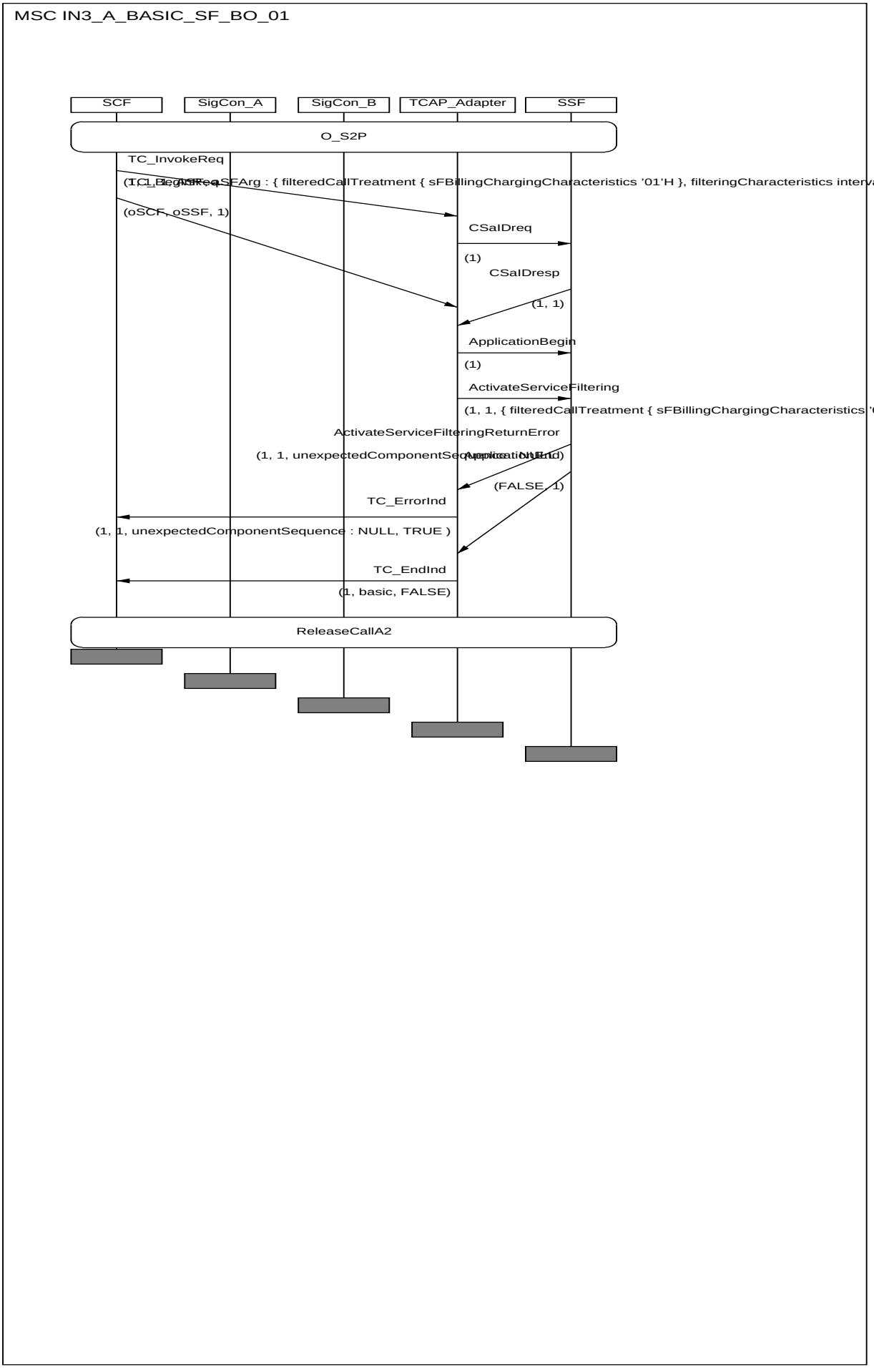
	IN3_A_BASIC_SF_BI_01
Work item no.:	ITEM_BASIC_124
IN2 Ref(tmp)	IN2_A_BASIC_SF_BI_01
Purpose:	Verify that the SSF sends to the SCF an **ActivateServiceFiltering** error component with the indication of **missing parameter**, when the SCF issues an **ActivateServiceFiltering** invoke component with missing parameter **filteredCallTreatment**.
Requirements ref.	8.1, 9.3.2, 11.1.1, 11.1.1.1.1, 11.1.3.2, 13.1.4.1.2, 15.1.1.2.2
Selection Cond.	
Preamble:	none
Test description	SCF issues **ActivateServiceFiltering** invoke with missing parameter - filteredCallTreatment
Pass criteria	- Check that SSF sends to SCF a **ActivateServiceFiltering** error with the indication of **missing parameter**
Postamble:	None.


MSC IN3_A_BASIC_SF_BI_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
TC_InvokeReq
(1, 1, ASFreqSFArg : { filteringCharacteristics interval : 60, filteringTimeOut duration : 1800, filteringCriteria serviceKey : 1 }, short
(oSCF, oSSF, 1)
CSaIDreq
(1)
CSaIDresp
(1, 1)
ApplicationBegin
(1)
ActivateServiceFiltering
(1, 1, { filteringCharacteristics interval : 60, filteringTimeOut duration : 1800, filteringC
ActivateServiceFilteringReturnError
(1, 1, missingParameter : NULL)
ApplicationEnd
TC_ErrorInd
(FALSE, 1)
(1, 1, missingParameter : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)

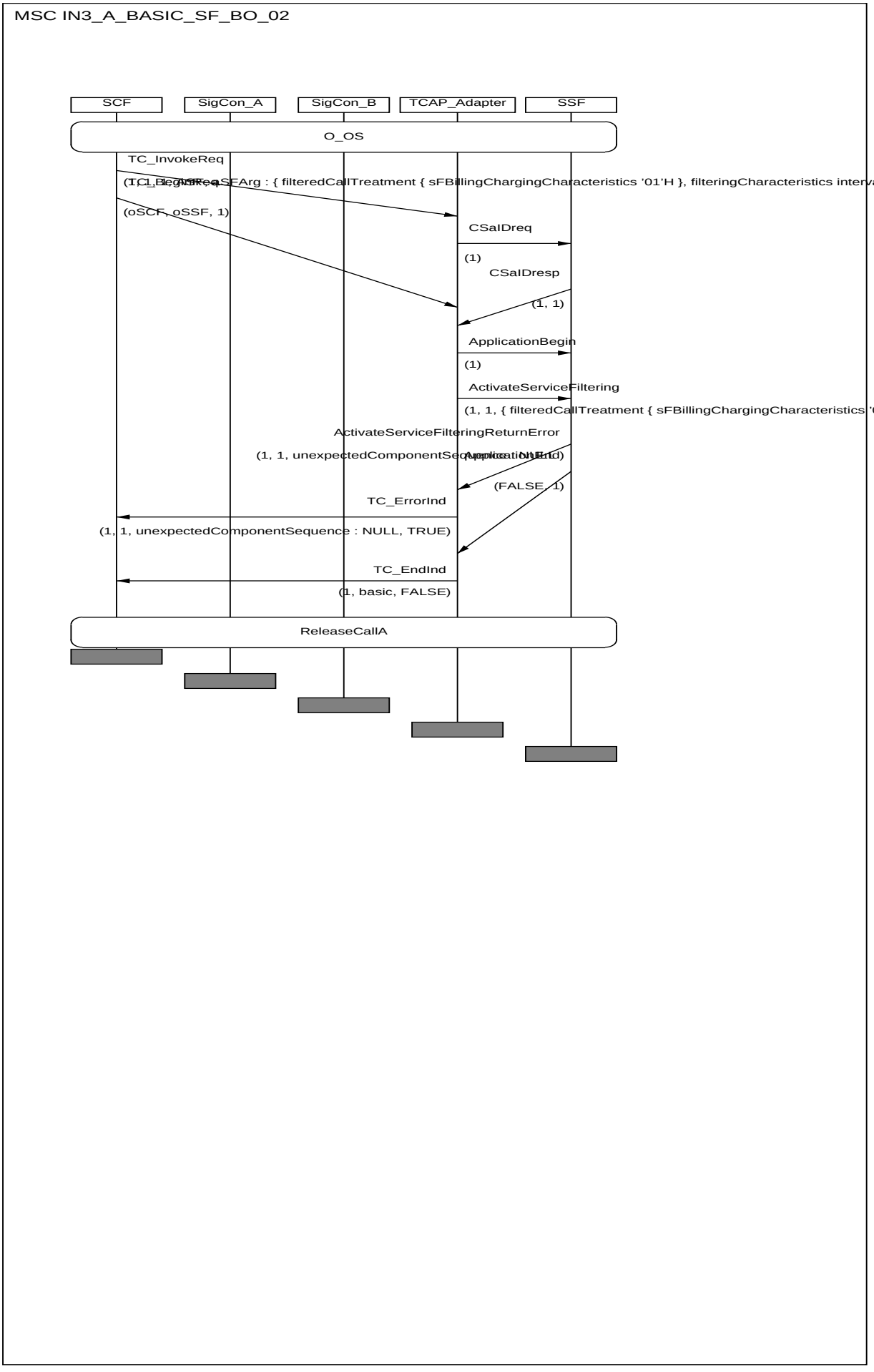
	IN3_A_BASIC_SF_BI_02
Work item no.:	ITEM_BASIC_125
IN2 Ref(tmp)	IN2_A_BASIC_SF_BI_02
Purpose:	Verify that the SSF sends to the SCF an **ActivateServiceFiltering** error component with the indication of **parameterOutOfRange**, when the SCF issues an **ActivateServiceFiltering** invoke component with parameter **filteringTimeOut** with duration > 86400 (out of range).
Requirements refs	8.1, 9.3.2, 11.1.1, 11.1.1.1.1, 11.1.3.2, 13.1.5.1, 15.1.1.2.2
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 **parameterOutOfRange**
Postamble:	None.


MSC IN3_A_BASIC_SF_BI_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
TC_InvokeReq
(1, 1, ASEqASFArg : { filteredCallTreatment { sFBillingChargingCharacteristics '01'H }, filteringCharacteristics interval : 60, filteringTimeO
(oSCF, oSSF, 1)
CSaIDreq
(1)
CSaIDresp
(1, 1)
ApplicationBegin
(1)
ActivateServiceFiltering
(1, 1, { filteredCallTreatment { sFBillingChargingCharacteristics '01'H }, filteringChara
ActivateServiceFilteringReturnError
(1, 1, parameterOutOfRange : NULL)
(FALSE, 1)
TC_ErrorInd
(1, 1, parameterOutOfRange : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)

	IN3_A_BASIC_SF_BO_01
Work item no.:	ITEM_BASIC_126
IN2 Ref(tmp)	IN2_A_BASIC_SF_BO_01
Purpose:	Test **ServiceFiltering** procedure in Monitoring state
Requirements refs	8.1, 9.3.2, 11.1.1, 11.1.1.1.1, 11.1.3.2, 13.1.9.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	O_S2P
Test description	SCF issues **ActivateServiceFiltering** invoke containing mandatory and optional parameters, with: - filteredCallTreatment including: sFBillingChargingCharacteristics, - filteringCharacteristics being interval - filteringTimeOut being duration, - filteringCriteria being addressAndService including: - calledAddressValue, - serviceKey, - callingAddressValue, - locationNumber
Pass criteria	SSF issues **ServiceFiltering error** with unexpectedComponentSequence parameter
Postamble:	ReleaseCallAB_cause_00.

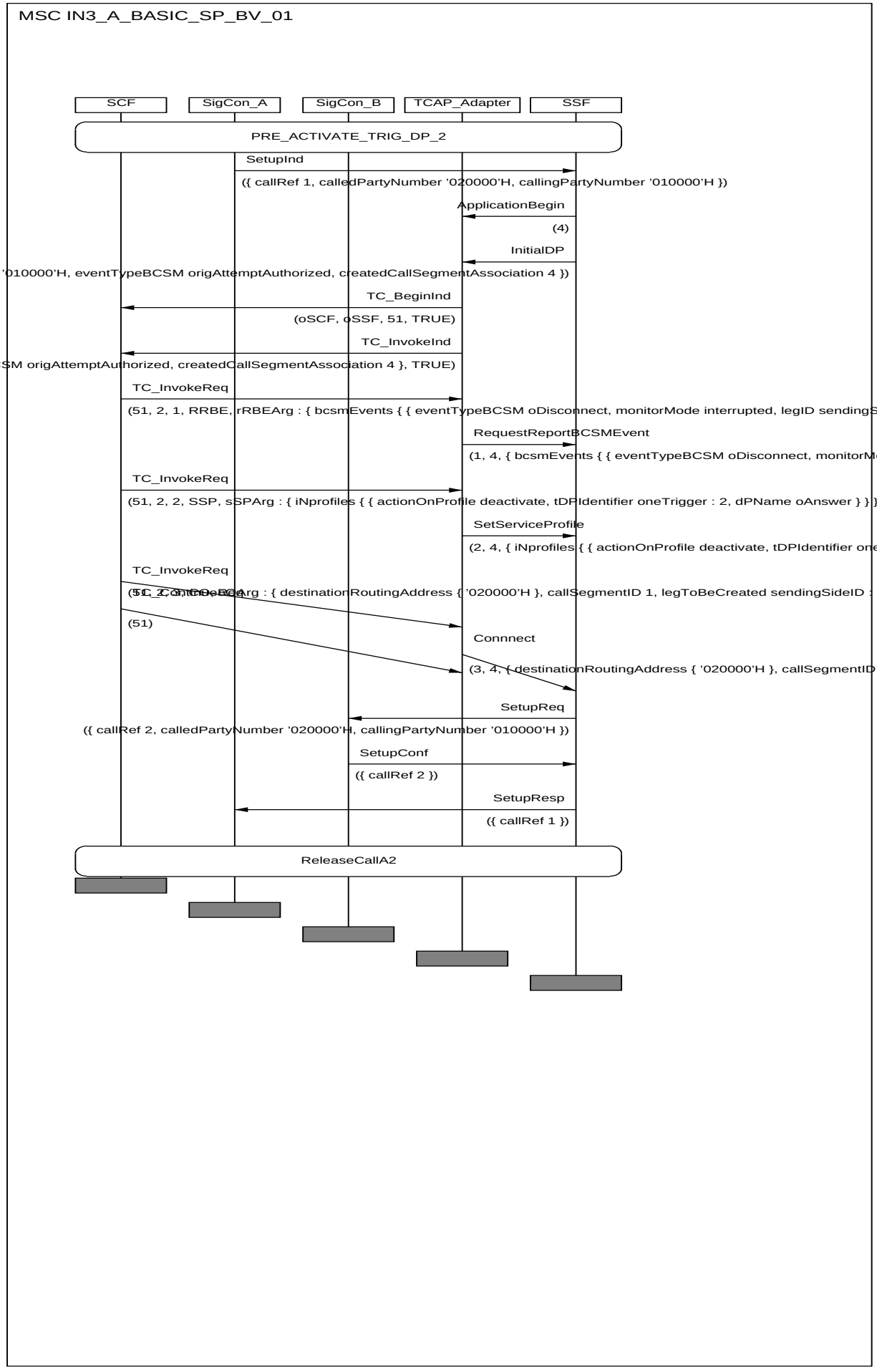

MSC IN3_A_BASIC_SF_BO_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_S2P
TC_InvokeReq
(1, 1, oSFArg : { filteredCallTreatment { sFBillingChargingCharacteristics '01'H }, filteringCharacteristics interval : 60, filteringTimeO
(oSCF, oSSF, 1)
CSaIDreq
(1)
CSaIDresp
(1, 1)
ApplicationBegin
(1)
ActivateServiceFiltering
(1, 1, { filteredCallTreatment { sFBillingChargingCharacteristics '01'H }, filteringChara
ActivateServiceFilteringReturnError
(1, 1, unexpectedComponentSequence : NULL)
(FALSE, 1)
TC_ErrorInd
(1, 1, unexpectedComponentSequence : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)
ReleaseCallA2

	IN3_A_BASIC_SF_BO_02
Work item no.:	ITEM_BASIC_127
IN2 Ref(tmp)	IN2_A_BASIC_SF_BO_02
Purpose:	test **ServiceFiltering** procedure in WaitForInstruction state
Requirements refs	8.1, 9.3.2, 11.1.1, 11.1.1.1.1, 11.1.3.2, 13.1.9.2.2, 15.1.1.2.2
Selection Cond.	
Preamble:	O_OS
Test description	SCF issues **ActivateServiceFiltering** invoke containing mandatory and optional parameters, with: - filteredCallTreatment including: sFBillingChargingCharacteristics, - filteringCharacteristics being interval - filteringTimeOut being duration, - filteringCriteria being addressAndService including: - calledAddressValue, - serviceKey, - callingAddressValue, - locationNumber
Pass criteria	SSF issues **ActivateServiceFiltering error** with unexpectedComponentSequence parameter
Postamble:	ReleaseCallA.

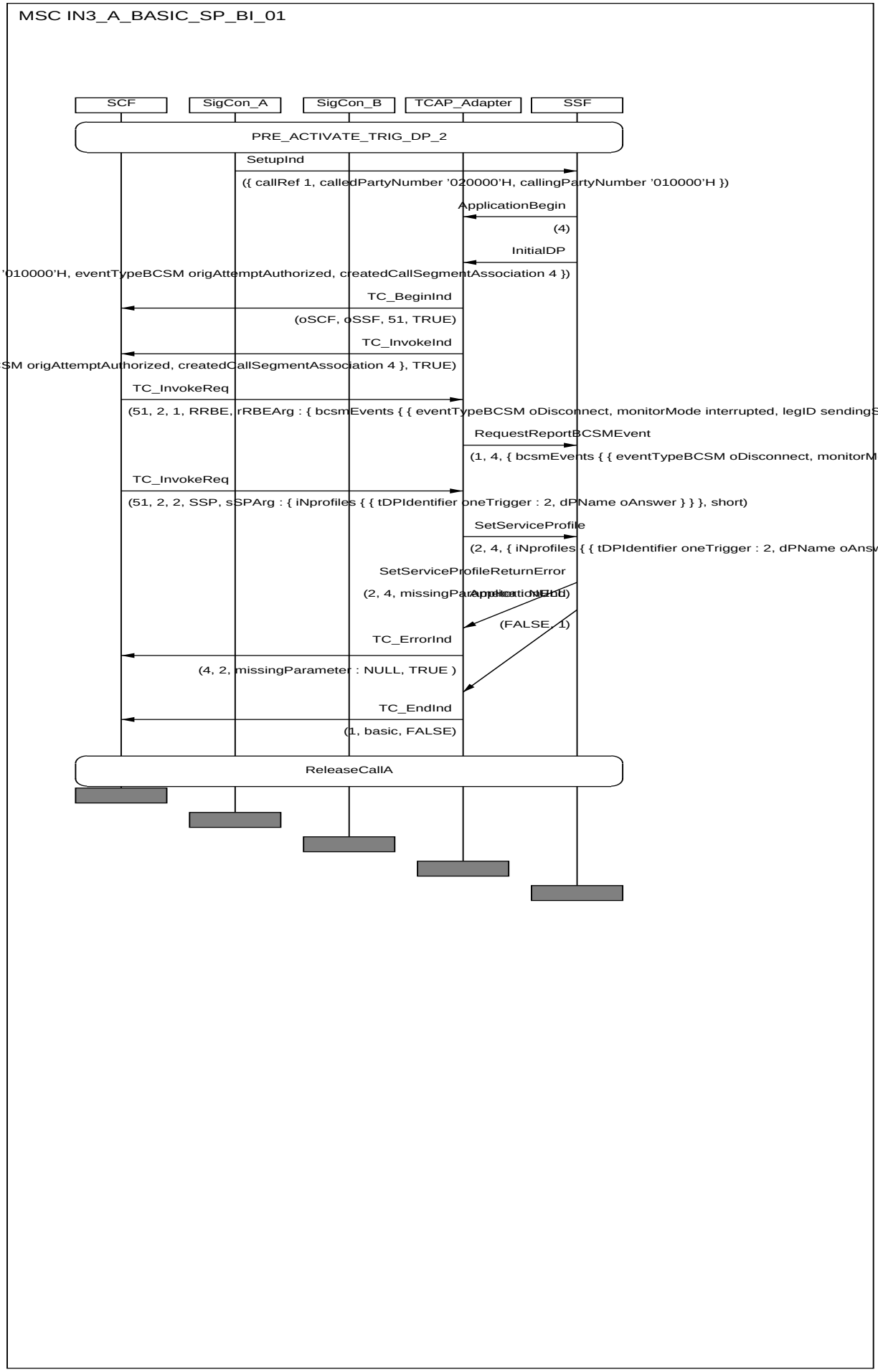

MSC IN3_A_BASIC_SF_BO_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
O_OS
TC_InvokeReq
(1, 1, { filteredCallTreatment { sFBillingChargingCharacteristics '01'H }, filteringCharacteristics interval : 60, filteringTimeO
(oSCF, oSSF, 1)
CSaIDreq
(1)
CSaIDresp
(1, 1)
ApplicationBegin
(1)
ActivateServiceFiltering
(1, 1, { filteredCallTreatment { sFBillingChargingCharacteristics '01'H }, filteringChara
ActivateServiceFilteringReturnError
(FALSE, 1)
TC_ErrorInd
(1, 1, unexpectedComponentSequence : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)
ReleaseCallA

6.6.26 SetServiceProfile (SP) procedure

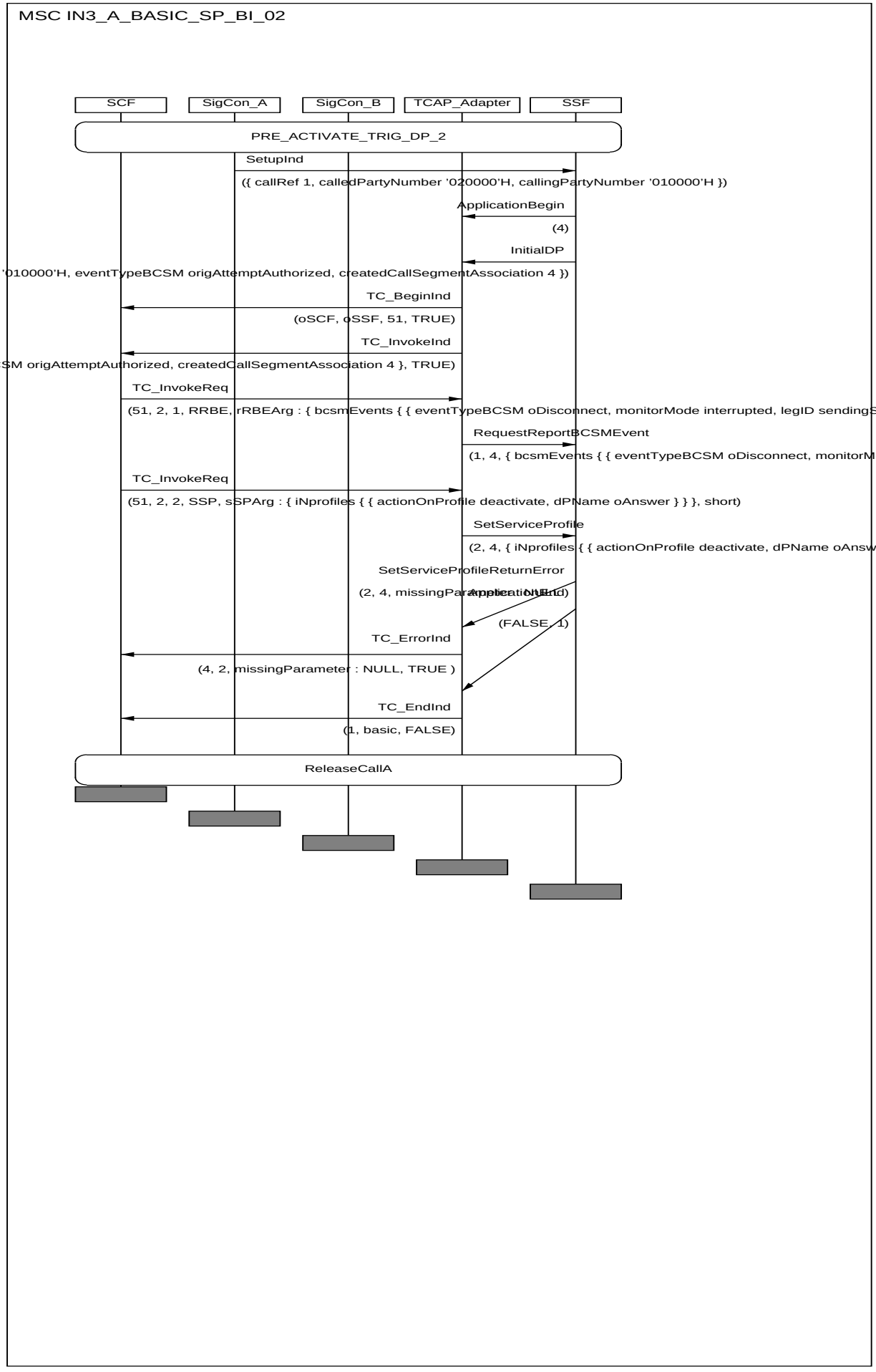
	IN3_A_BASIC_SP_BV_01
Work item no.:	ITEM_BASIC_252
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having created and activated two different Triggers, one for DP "**origAttemptAuthorized**" and one for DP "**oAnswer**", both for the same profile identified by PROFILE_ID_1, and with resulting tDPIdentifiers referred to as **tDPIdentifier1** and **tDPIdentifier2** respectively, does not send to the SCF a SetServiceProfile returnError component, when having received a SetServiceProfile invoke component containing one INprofiles element, indicating actionOnProfile = "**deactivate**" and tdPIdentifier = **tDPIdentifier2** after receiving a SetupInd message from an initiating user, containing call-related data compatible with trigger profile and trigger data of the trigger identified by tDPIdentifier1 and having sent to the SCF an InitialDP invoke component, containing parameters serviceKey (identifying the serviceKey of tDPIdentifier1) and eventTypeBCSM = "**origAttemptAuthorized**". The relationship is maintained when the call continues, by arming DP oDisconnect as EDP-R. Verify also that the SSF does not send an InitialDP invoke component when the called party answers the call (TDP-R on oAnswer is deactivated).
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.45
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_DP_2
Test description	CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(1,interrupted,oDisconnect) L1!SetServiceProfile invoke((deactivate,tDPIdentifier2,oAnswer)) Wait a while to verify that no SetServiceProfile return error is received L1!Connect(2,1) CP1_2?SetUpReq CP1_2!SetUpConf CP1_1?SetUpResp
Pass criteria	No SetServiceProfile return error is received after SetServiceProfile invoke, No InitialDP received after CP1_2!SetUpConf
Postamble:	TrigReleaseAB


MSC IN3_A_BASIC_SP_BV_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_DP_2
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(51, 2, 2, SSP, sSPArg : { iNprofiles { { actionOnProfile deactivate, tDPIdentifier oneTrigger : 2, dPName oAnswer } } }, short
SetServiceProfile
(2, 4, { iNprofiles { { actionOnProfile deactivate, tDPIdentifier oneTrigger : 2, dPName
TC_InvokeReq
(51, 2, 3, CRBE, cRBEArg : { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated sendingSideID : '02'H }, short
(51)
Connnect
(3, 4, { destinationRoutingAddress { '020000'H }, callSegmentID 1, legToBeCreated
SetupReq
({ callRef 2, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
SetupConf
({ callRef 2 })
SetupResp
({ callRef 1 })
ReleaseCallA2

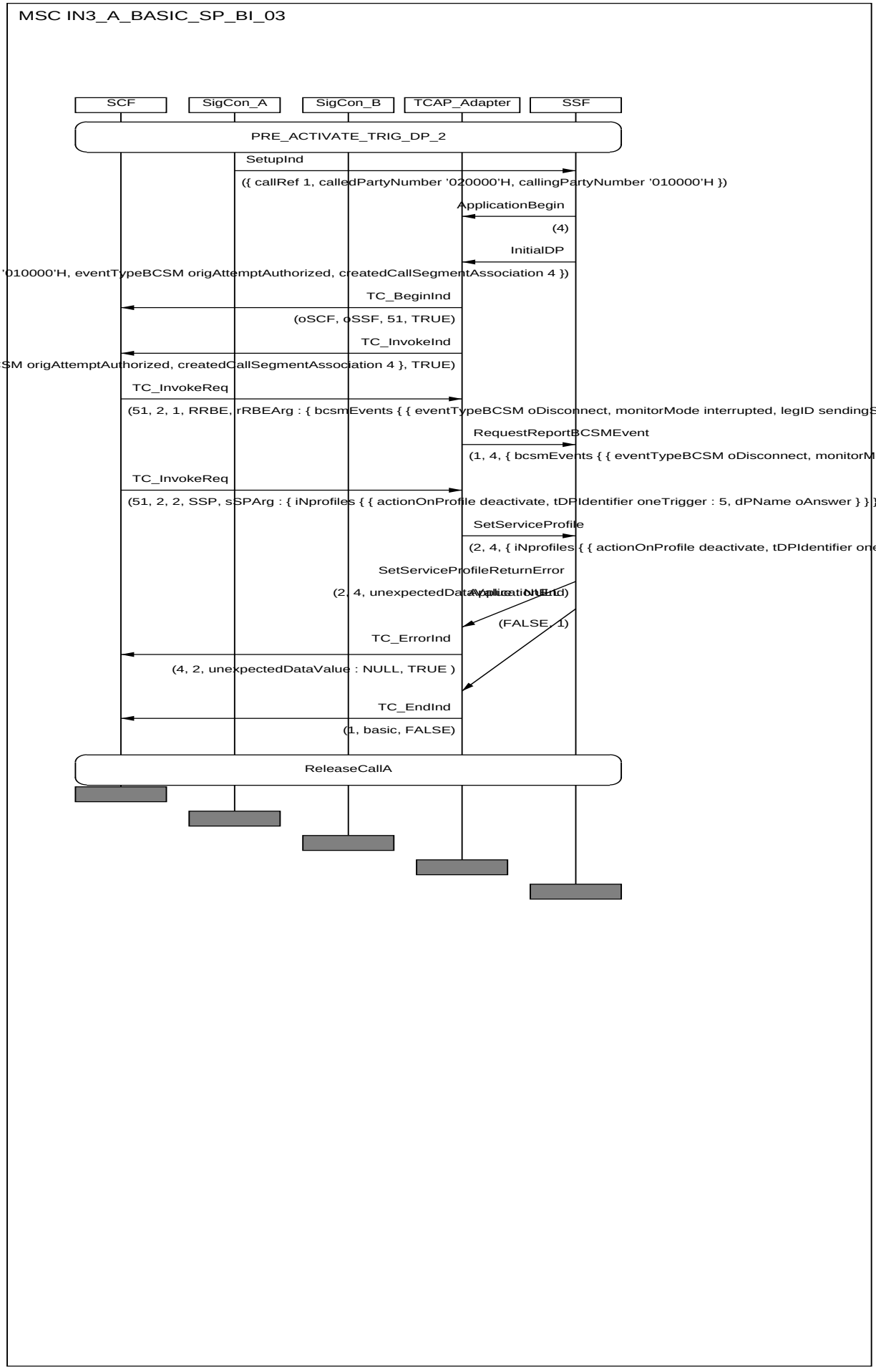
	IN3_A_BASIC_SP_BI_01
Work item no.:	ITEM_BASIC_254
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having sent to the SCF an InitialDP invoke component upon receipt of a SetupInd message, sends to the SCF a SetServiceProfile returnError component with error code "missingParameter", after having received a SetServiceProfile invoke component (with one iNProfiles element) where mandatory parameter "actionOnProfile" is missing.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.45
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_DP_2
Test description	CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(1,interrupted,oDisconnect) L1!SetServiceProfile invoke((actionOnProfile **omitted**,tDPIdentifier2,oAnswer)) L1?SetServiceProfile returnError(missingParameter)
Pass criteria	L1?SetServiceProfile returnError(missingParameter)
Postamble:	TrigReleaseA


MSC IN3_A_BASIC_SP_BI_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_DP_2
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(51, 2, 2, SSP, sSPArg : { iNprofiles { { tDPIdentifier oneTrigger : 2, dPName oAnswer } } }, short)
SetServiceProfile
(2, 4, { iNprofiles { { tDPIdentifier oneTrigger : 2, dPName oAnswer } } }
SetServiceProfileReturnError
(2, 4, missingParameter : NULL, TRUE)
(FALSE, 1)
TC_ErrorInd
(4, 2, missingParameter : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)
ReleaseCallA

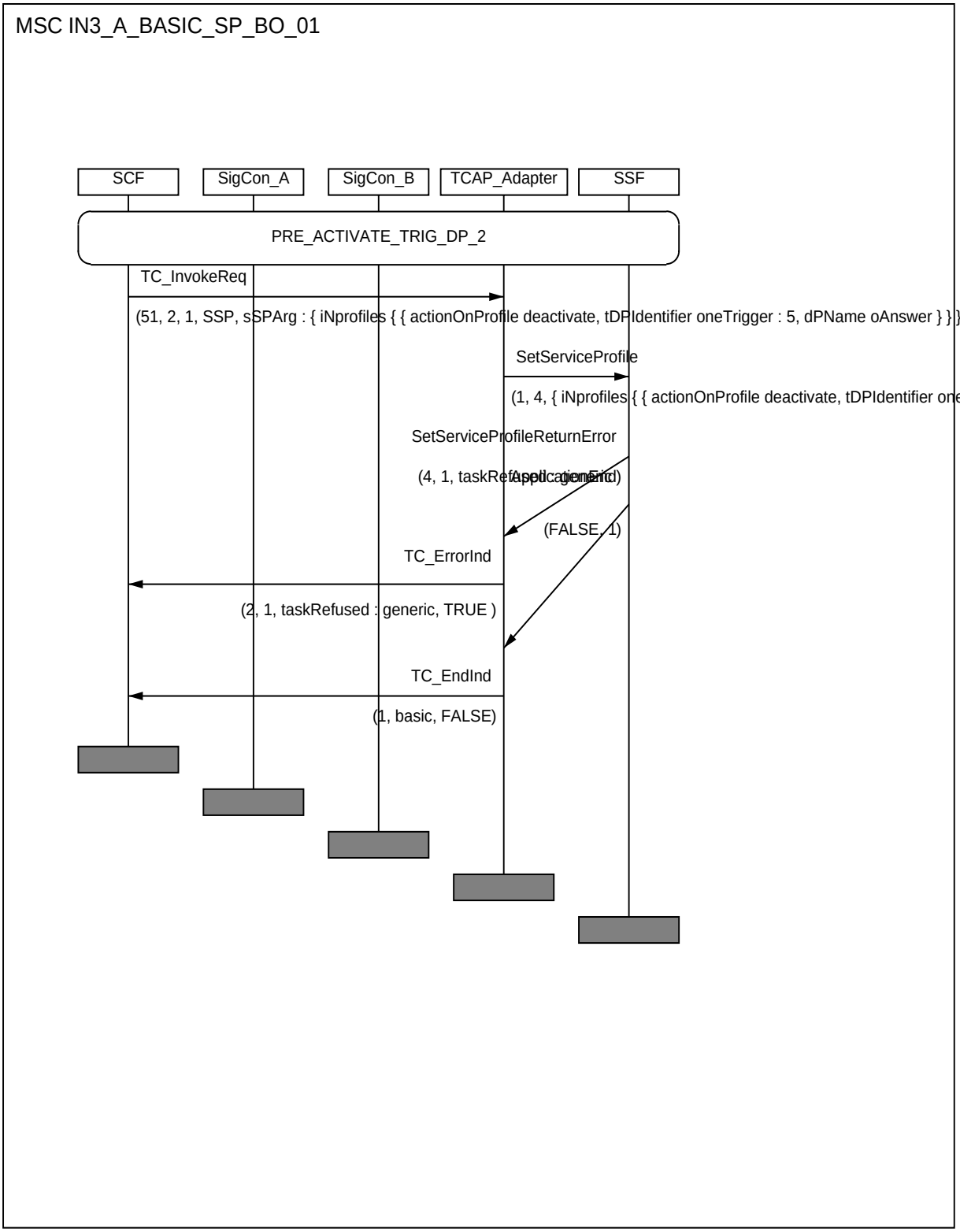
	IN3_A_BASIC_SP_BI_02
Work item no.:	ITEM_BASIC_255
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having sent to the SCF an InitialDP invoke component upon receipt of a SetupInd message, sends to the SCF a SetServiceProfile returnError component with error code "missingParameter", after having received a SetServiceProfile invoke component (with one iNProfiles element) where mandatory parameter "tDPIdentifier" is missing.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.45
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_DP_2
Test description	CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(1,interrupted,oDisconnect) L1!SetServiceProfile invoke((deactivate, tDPIdentifier **omitted**, oAnswer)) L1?SetServiceProfile returnError(missingParameter)
Pass criteria	L1?SetServiceProfile returnError(missingParameter)
Postamble:	TrigReleaseA


MSC IN3_A_BASIC_SP_BI_02
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_DP_2
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(51, 2, 2, SSP, sSPArg : { iNprofiles { { actionOnProfile deactivate, dPName oAnswer } } }, short)
SetServiceProfile
(2, 4, { iNprofiles { { actionOnProfile deactivate, dPName oAnswer } } }
SetServiceProfileReturnError
(2, 4, missingParameter : NULL)
ApplicationEnd
(FALSE, 1)
TC_ErrorInd
(4, 2, missingParameter : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)
ReleaseCallA

	IN3_A_BASIC_SP_BI_03
Work item no.:	ITEM_BASIC_256
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF, having sent to the SCF an InitialDP invoke component upon receipt of a SetupInd message, sends to the SCF a SetServiceProfile returnError component with error code "unexpectedDataValue", after having received a SetServiceProfile invoke component (with one iNProfiles element) where parameter "tDPIdentifier" does not identify a created trigger.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.45
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_DP_2
Test description	CP1_1!SetupInd(CALL-DATA-1) L1?TC-BEGIN L1?InitialDP invoke(SERVICE_KEY1, origAttemptAuthorized) L1!RequestReportBCSMEvent(1,interrupted,oDisconnect) L1!SetServiceProfile invoke((deactivate, **unassigned** tDPIdentifier, oAnswer)) L1?SetServiceProfile returnError(unexpectedDataValue)
Pass criteria	L1?SetServiceProfile returnError(unexpectedDataValue)
Postamble:	TrigReleaseA


MSC IN3_A_BASIC_SP_BI_03
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_DP_2
SetupInd
({ callRef 1, calledPartyNumber '020000'H, callingPartyNumber '010000'H })
ApplicationBegin
(4)
InitialDP
allingPartyNumber '010000'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 })
TC_BeginInd
(oSCF, oSSF, 51, TRUE)
TC_InvokeInd
00'H, eventTypeBCSM origAttemptAuthorized, createdCallSegmentAssociation 4 }, TRUE)
TC_InvokeReq
(51, 2, 1, RRBE, rRBEArg : { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legID sendingSideID : '01'H } } }, sh
RequestReportBCSMEvent
(1, 4, { bcsmEvents { { eventTypeBCSM oDisconnect, monitorMode interrupted, legI
TC_InvokeReq
(51, 2, 2, SSP, sSPArg : { iNprofiles { { actionOnProfile deactivate, tDPIdentifier oneTrigger : 5, dPName oAnswer } } }, short
SetServiceProfile
(2, 4, { iNprofiles { { actionOnProfile deactivate, tDPIdentifier oneTrigger : 5, dPName
SetServiceProfileReturnError
(2, 4, unexpectedDataValue : NULL)
(FALSE, 1)
TC_ErrorInd
(4, 2, unexpectedDataValue : NULL, TRUE)
TC_EndInd
(1, basic, FALSE)
ReleaseCallA

	IN3_A_BASIC_SP_BO_01
Work item no.:	ITEM_BASIC_257
IN2 Ref(tmp)	None
Purpose:	Verify that the SSF sends to the SCF a SetServiceProfile returnError component with error code "taskRefused" or "unexpectedComponentSequence", after having received a SetServiceProfile invoke component outside the context of a call.
Requirements refs	6.4.2, 6.4.6, 6.4.7, 6.5, 11.45
Selection Cond.	
Preamble:	PRE_ACTIVATE_TRIG_DP_2
Test description	L4!SetServiceProfile invoke((deactivate, unassigned tDPIdentifier, oAnswer)) L4?SetServiceProfile returnError("taskRefused" or "unexpectedComponentSequence")
Pass criteria	L4?SetServiceProfile returnError("taskRefused" or "unexpectedComponentSequence")
Postamble:	None


MSC IN3_A_BASIC_SP_BO_01
SCF
SigCon_A
SigCon_B
TCAP_Adapter
SSF
PRE_ACTIVATE_TRIG_DP_2
TC_InvokeReq
(51, 2, 1, SSP, sSPArg : { iNprofiles { { actionOnProfile deactivate, tDPIdentifier oneTrigger : 5, dPName oAnswer } } }, short
SetServiceProfile
(1, 4, { iNprofiles { { actionOnProfile deactivate, tDPIdentifier oneTrigger : 5, dPName
SetServiceProfileReturnError
(4, 1, taskRefused : generic)
(FALSE, 1)
TC_ErrorInd
(2, 1, taskRefused : generic, TRUE)
TC_EndInd
(1, basic, FALSE)

Annex A (informative): Description of various functional configurations

In these various configurations, the shaded area represents the Implementation Under Test (IUT).

Annex B (informative): Bibliography

ITU-T Recommendation Q.1224: "Distributed functional plane for intelligent network Capability Set 2".

History

Document history			
V1.1.1	September 2002	One-step Approval Procedure	OAP 20030110: 2002-09-11 to 2003-01-10
V1.1.1	January 2003	Publication	