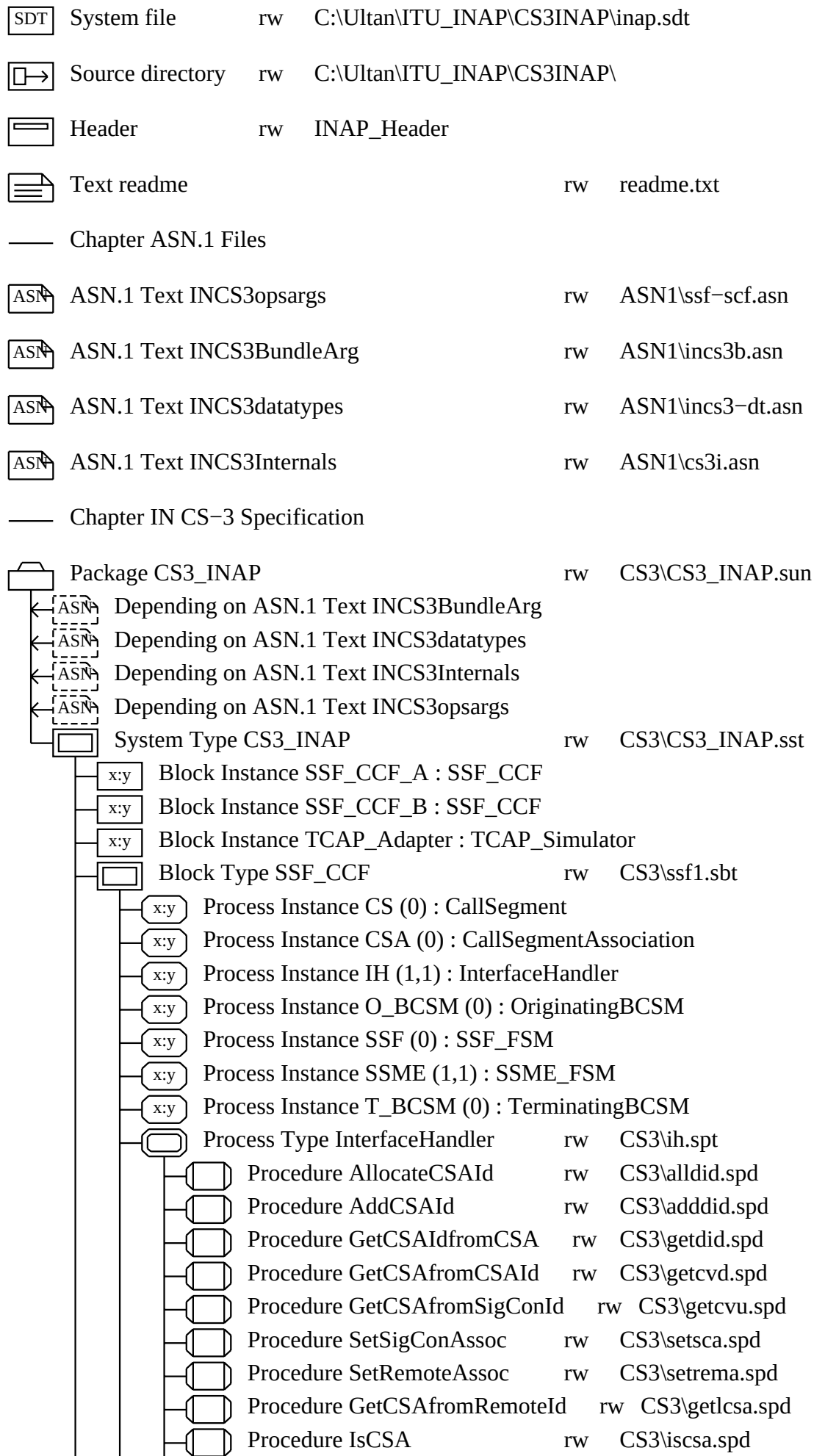
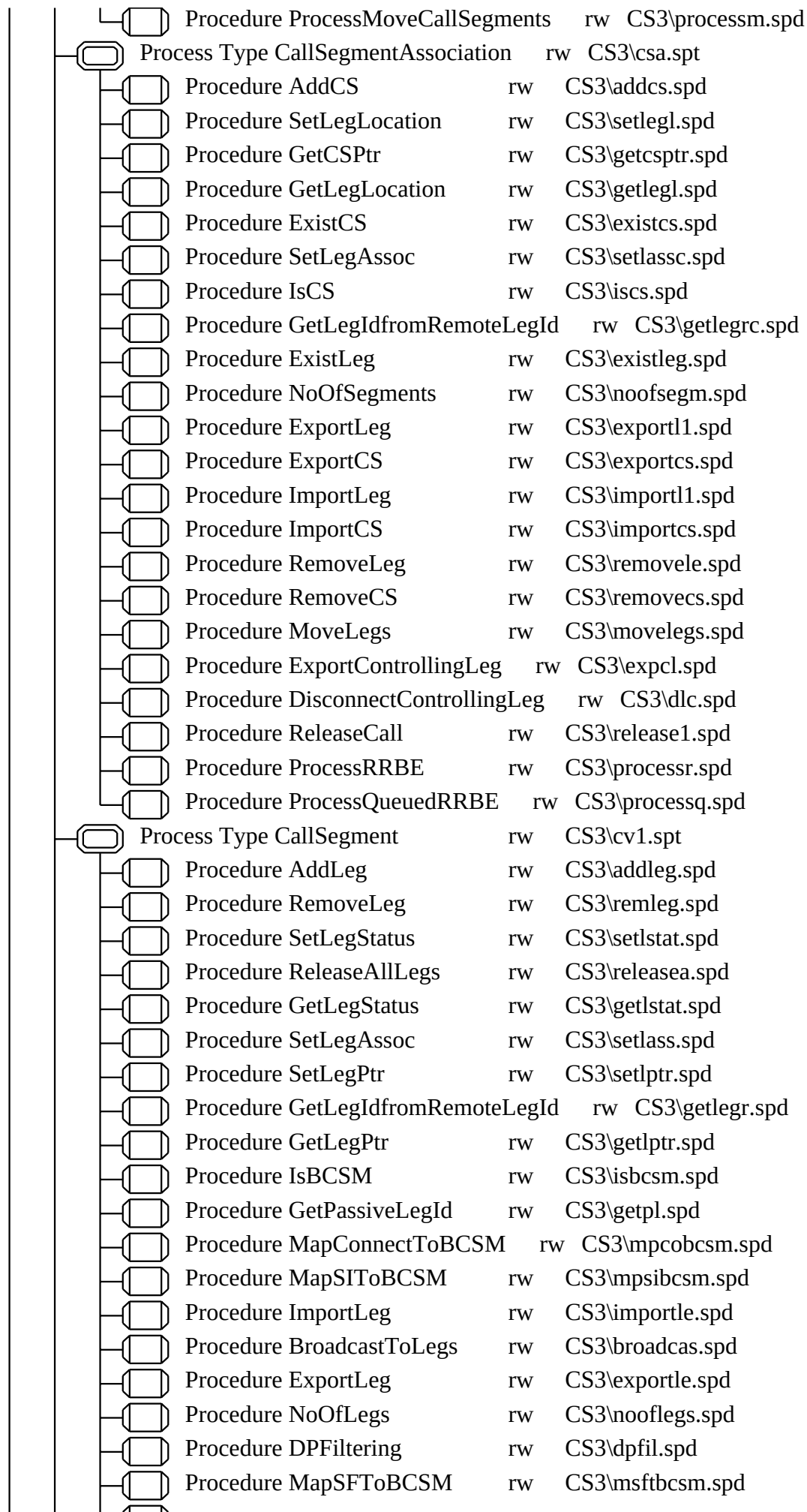


Annex A: View



Annex A: View



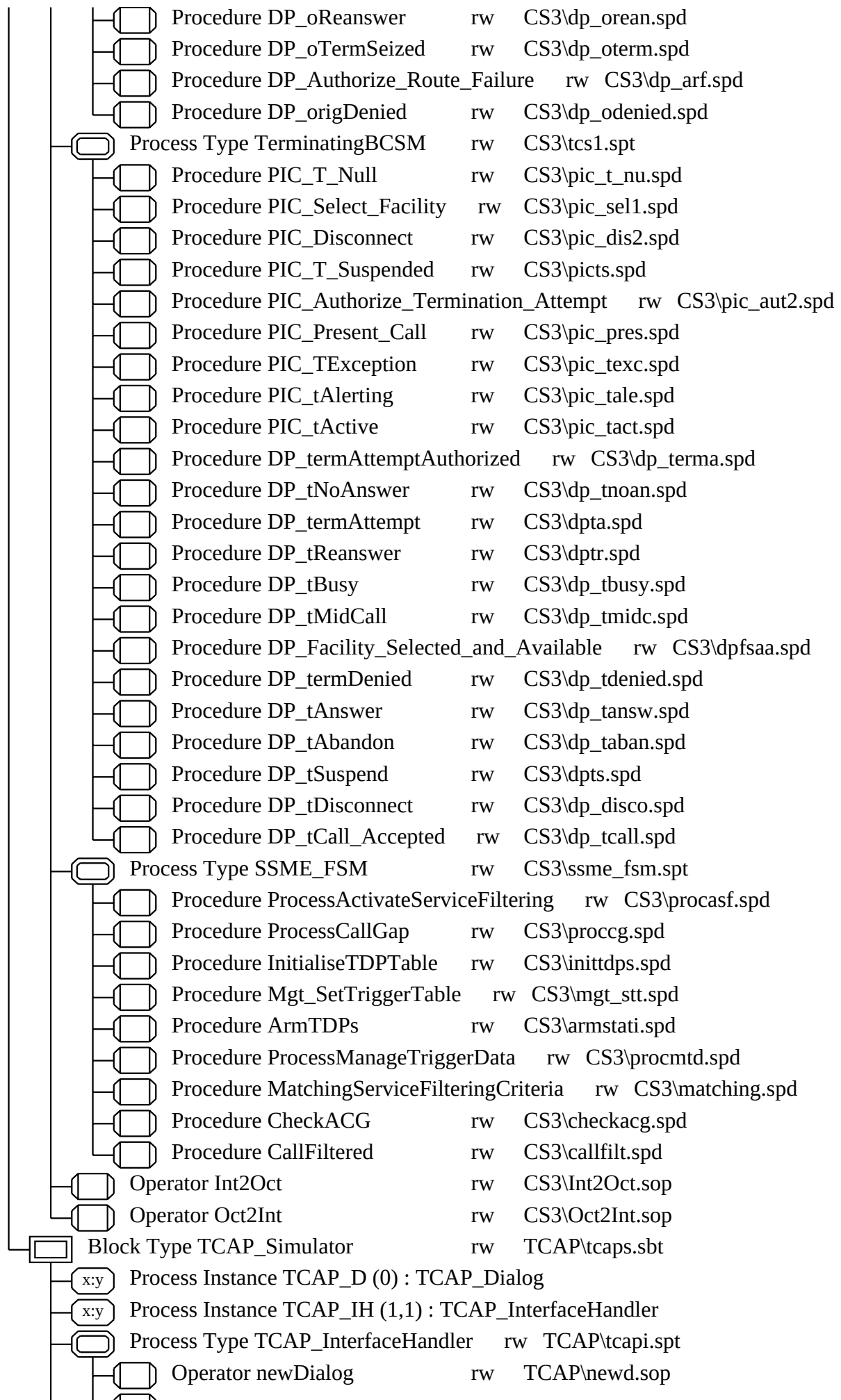
Annex A: View

	Procedure MapCIToBCSM	rw	CS3\mcitbcsn.spd
	Procedure DPFilteringUTSI	rw	CS3\dpfilter.spd
	Procedure MapAIToBCSM	rw	CS3\maitbcsn.spd
	Procedure MapSRToBCSM	rw	CS3\msrtbcsn.spd
	Process Type SSF_FSM	rw	CS3\ssf_fsm.spt
	Procedure InitialiseDPTable	rw	CS3\initlvl.spd
	Procedure ExistLeg	rw	CS3\exleg.spd
	Procedure AnyDPArmed	rw	CS3\anyevarm.spd
	Procedure IsDPArmed	rw	CS3\evarm.spd
	Procedure DisarmDPs	rw	CS3\disarm.spd
	Procedure DPArmed	rw	CS3\dparmed.spd
	Procedure CallInformationReportPending	rw	CS3\callinfo.spd
	Procedure ApplyChargingReportPending	rw	CS3\applycha.spd
	Procedure ArmTDPs	rw	CS3\armtdps.spd
	Procedure MatchingServiceFilteringCriteria	rw	CS3\msfc.spd
	Procedure IncrementOutstandingRequests	rw	CS3\ior.spd
	Procedure CheckACG	rw	CS3\cacg.spd
	Procedure OutstandingRequests	rw	CS3\or.spd
	Procedure CallFiltered	rw	CS3\cf.spd
	Procedure ProcessApplyCharging	rw	CS3\process6.spd
	Procedure ProcessContinue	rw	CS3\proces12.spd
	Procedure ProcessRequestReportBCSMEvent	rw	CS3\prreqrep.spd
	Procedure ProcessAnalyseInformation	rw	CS3\procai.spd
	Procedure ProcessCallInformationRequest	rw	CS3\process7.spd
	Procedure ProcessDisconnectForwardConnection	rw	CS3\processd.spd
	Procedure ProcessSendChargingInformation	rw	CS3\procsci.spd
	Procedure ProcessSelectRoute	rw	CS3\procsr.spd
	Procedure ProcessCancel	rw	CS3\process8.spd
	Procedure ProcessEstablishTemporaryConnection	rw	CS3\processe.spd
	Procedure ProcessEventReportBCSM	rw	CS3\prevrep.spd
	Procedure ProcessSelectFacility	rw	CS3\procsf.spd
	Procedure ProcessCollectInformation	rw	CS3\process9.spd
	Procedure ProcessFurnishChargingInformation	rw	CS3\proces13.spd
	Procedure ProcessSetServiceProfile	rw	CS3\Procssp.spd
	Procedure ProcessConnect	rw	CS3\proces10.spd
	Procedure ProcessInitiateCallAttempt	rw	CS3\processi.spd
	Procedure ProcessInitialDP	rw	CS3\pridp.spd
	Procedure ProcessReconnect	rw	CS3\ProcessReconnect.spd
	Procedure ProcessConnectToResource	rw	CS3\proces11.spd
	Procedure ProcessRequestNotificationChargingEvent	rw	CS3\proces16.spd
	Procedure ProcessDPsSpecific	rw	CS3\ProcDPS.spd
	Procedure ExportEventRecord	rw	CS3\expevl.spd
	Procedure ProcessContinueWithArgument	rw	CS3\pcwa.spd
	Procedure ProcessRequestReportUTSI	rw	CS3\process3.spd
	Procedure ImportEventRecord	rw	CS3\impevl.spd

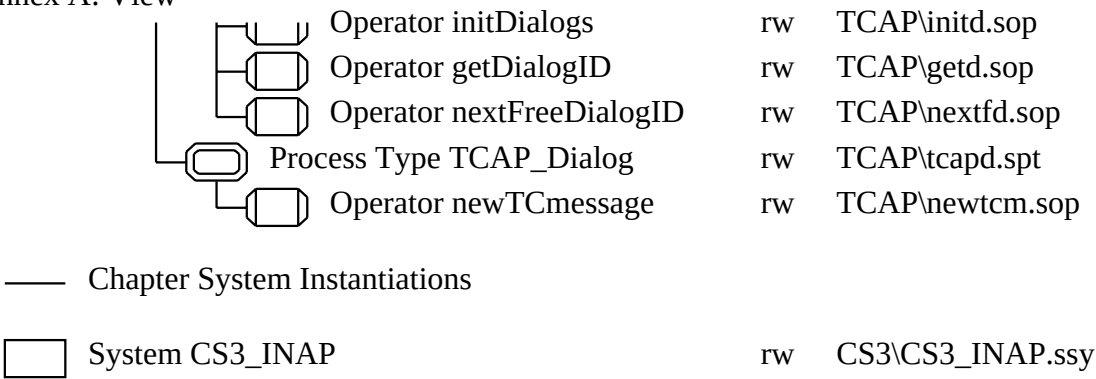
Annex A: View

	Procedure ProcessDisconnectLeg	rw	CS3\processdl.spd
	Procedure ProcessSendSTUI	rw	CS3\process4.spd
	Procedure ProcessReportUTSI	rw	CS3\process5.spd
	Procedure IsUTSIArmed	rw	CS3\isutsiar.spd
	Procedure ProcessReportFacility	rw	CS3\prf.spd
	Procedure ProcessRequestReportFacilityEvent	rw	CS3\prrfe.spd
	Procedure IsFacilityArmed	rw	CS3\isfa.spd
	Procedure ProcessSendFacility	rw	CS3\psf.spd
	Procedure ProcessAuthorizeTermination	rw	CS3\procaut.spd
	Procedure AnyFacilityArmed	rw	CS3\anyfacarm.spd
	Procedure IncrementRequests	rw	CS3\ir.spd
	Procedure DecrementRequests	rw	CS3\dr.spd
	Process Type OriginatingBCSM	rw	CS3\ocs1.spt
	Procedure PIC_O_Null	rw	CS3\pic_o_nu.spd
	Procedure PIC_Analyse_Information	rw	CS3\pic_anal.spd
	Procedure PIC_Send_Call	rw	CS3\pic_send.spd
	Procedure PIC_O_Active	rw	CS3\pic_o_ac.spd
	Procedure PIC_Authorise_Origination_Attempt	rw	CS3\pic_auth.spd
	Procedure PIC_Select_Route	rw	CS3\pic_sele.spd
	Procedure PIC_O_Alerting	rw	CS3\pic_o_al.spd
	Procedure PIC_O_Abandon	rw	CS3\pic_o_ab.spd
	Procedure PIC_Collect_Information	rw	CS3\pic_coll.spd
	Procedure PIC_Authorise_Call_Setup	rw	CS3\pic_aut1.spd
	Procedure PIC_O_Answer	rw	CS3\pic_o_an.spd
	Procedure PIC_OException	rw	CS3\pic_oexc.spd
	Procedure PIC_Collect_NDigits	rw	CS3\pic_col1.spd
	Procedure PIC_O_Suspended	rw	CS3\pic_o_su.spd
	Procedure PIC_O_Retention	rw	CS3\pic_o_re.spd
	Procedure PIC_Disconnect	rw	CS3\pic_disc.spd
	Procedure DP_origAttemptAuthorised	rw	CS3\dp_origa.spd
	Procedure DP_Route_Select_Failure	rw	CS3\dp_route.spd
	Procedure DP_oCalledPartyBusy	rw	CS3\dp_ocall.spd
	Procedure DP_oDisconnect	rw	CS3\dpdiscon.spd
	Procedure DP_Collected_Info	rw	CS3\dp_colle.spd
	Procedure DP_oAnswer	rw	CS3\dp_o_ans.spd
	Procedure DP_oMidCall	rw	CS3\dp_omidc.spd
	Procedure DP_Analysed_Information	rw	CS3\dp_analy.spd
	Procedure DP_oNoAnswer	rw	CS3\dp_onoan.spd
	Procedure DP_oAbandon	rw	CS3\dp_oaban.spd
	Procedure MapToSIRArg	rw	CS3\maptosir.spd
	Procedure MapToDP	rw	CS3\maptodp.spd
	Procedure MapFromPIC	rw	CS3\mapfpic.spd
	Procedure DP_origAttempt	rw	CS3\dp_orig1.spd
	Procedure DP_oSuspend	rw	CS3\dp_osusp.spd

Annex A: View



Annex A: View



Text readme

Q.1238.2 Annex A SDLs

Version 0.0.6, 8 December 1999

Any further information may be obtained from:

Ultan Mulligan

ETSI PEX Group

F-06921 Sophia Antipolis Cedex

France

Tel: +33 4 92 94 43 88 Fax: +33 4 93 65 38 51

ultan.mulligan@etsi.fr or pex@etsi.fr

This SDL specification is best viewed with the font settings
Arial 8 for Headings and Arial 6 for symbol texts.

ASN.1 Text INCS3opsargs

INCS3opsargs

```
--IN-CS3-SSF-SCF-ops-args {ccitt recommendation q 1238 modules(1) in-cs3-ssf-scf
--ops-args (8) version1(0)}
```

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS

```
-- common-classes,
-- common-datatypes,
-- errortypes,
-- scf-srf-classes,
-- scf-srf-datatypes,
-- ssf-scf-classes,
-- ssf-scf-datatypes,
-- operationcodes,
-- ros-InformationObjects
--FROM IN-CS3-object-identifiers
-- {ccitt recommendation q 1238 modules(1) in-cs3-object-identifiers (0) version
1(0)}
```

```
-- OPERATION
```

```
--FROM Remote-Operations-Information-Objects ros-InformationObjects
```

```
-- COMMON-BOUNDS
```

```
--FROM IN-CS3-Common-Classes common-classes
```

```
-- SCF-SSF-BOUNDS
```

```
--FROM IN-CS3-SSF-SCF-Classes ssf-scf-classes
```

```
-- SCF-SRF-BOUNDS
```

```
--FROM IN-CS3-SCF-SRF-Classes scf-srf-classes
```

```
-- opcode-activateServiceFiltering,
-- opcode-activityTest,
-- opcode-analysedInformation,
-- opcode-analyseInformation,
-- opcode-applyCharging,
-- opcode-applyChargingReport,
-- opcode-assistRequestInstructions,
-- opcode-authorizeTermination,
-- opcode-callGap,
-- opcode-callInformationReport,
-- opcode-callInformationRequest,
-- opcode-cancel,
-- opcode-cancelStatusReportRequest,
-- opcode-collectedInformation,
-- opcode-collectInformation,
-- opcode-connect,
-- opcode-connectToResource,
-- opcode-continue,
-- opcode-continueWithArgument,
-- opcode-createCallSegmentAssociation,
-- opcode-createOrRemoveTriggerData,

-- opcode-disconnectForwardConnection,
-- opcode-dFCWithArgument,
-- opcode-disconnectLeg,
-- opcode-entityReleased,
-- opcode-establishTemporaryConnection,
-- opcode-eventNotificationCharging,
-- opcode-eventReportBCSM,
-- opcode-eventReportFacility,
-- opcode-facilitySelectedAndAvailable,
-- opcode-furnishChargingInformation,
```

Annex A: ssf-scf

```

-- opcode-holdCallInNetwork,
-- opcode-initialDP,
-- opcode-initiateCallAttempt,
-- opcode-manageTriggerData,
-- opcode-mergeCallSegments,
-- opcode-moveCallSegments,
-- opcode-oAbandon,
-- opcode-oAnswer,
-- opcode-oCalledPartyBusy,
-- opcode-oDisconnect,
-- opcode-oMidCall,
-- opcode-moveLeg,
-- opcode-oNoAnswer,
-- opcode-originationAttempt,
-- opcode-originationAttemptAuthorized,
-- opcode-oSuspended,
-- opcode-reconnect,
-- opcode-releaseCall,
-- opcode-reportUTSI,
-- opcode-requestCurrentStatusReport,
-- opcode-requestEveryStatusChangeReport,
-- opcode-requestFirstStatusMatchReport,
-- opcode-requestNotificationChargingEvent,
-- opcode-requestReportBCSMEvent,
-- opcode-requestReportUTSI,
-- opcode-requestReportFacilityEvent,
-- opcode-resetTimer,
-- opcode-routeSelectFailure,
-- opcode-selectFacility,
-- opcode-selectRoute,
-- opcode-sendChargingInformation,
-- opcode-sendFacilityInformation,
-- opcode-sendSTUI,
-- opcode-serviceFilteringResponse,
-- opcode-setServiceProfile,

-- opcode-splitLeg,
-- opcode-statusReport,
-- opcode-tAnswer,
-- opcode-tBusy,
-- opcode-tDisconnect,
-- opcode-termAttemptAuthorized,
-- opcode-terminationAttempt,
-- opcode-tMidCall,
-- opcode-tNoAnswer,
-- opcode-tSuspended

--FROM IN-CS3-operationcodes operationcodes

-- Extensions,
-- Integer4,
-- InvokeId

--FROM IN-CS3-common-datatypes common-datatypes

Extensions,
Integer4,
InvokeId,
AccessCode ,

InformationToSend,
CollectedInfo ,
MailBoxID ,
InformationToRecord ,
Media,
ReceivedStatus,
RecordedMessageID,
SRFGapCriteria ,
GenericNumber,
Code,
maxRecordedMessageUnits,

ActionIndicator,

```

Annex A: ssf-scf

ActionPerformed,
 AChBillingChargingCharacteristics ,
 AdditionalCallingPartyNumber ,
 AlertingPattern,
 ApplicationTimer,
 AssistingSSPIPRoutingAddress ,
 ATMTrafficDescriptor ,

 BackwardGVNS ,
 BCSMEvent ,
 BearerCapability ,

 CalledDirectoryNumber ,
 CalledPartyBusinessGroupID,
 CalledPartyNumber ,
 CalledPartySubaddress ,
 CallingGeodeticLocation ,
 CallingPartyBusinessGroupID,
 CallingPartyNumber ,
 CallingPartysCategory,
 CallingPartySubaddress ,
 CallProcessingOperationCorrelationID,
 CallReference ,
 CallResult ,
 -- CallSegmentID ,
 Carrier ,
 Cause ,
 CCSS,
 CGEncountered,
 ChargeNumber ,
 ChargingEvent ,
 Component,
 ComponentCorrelationID,
 ComponentType,
 ControlType,
 CorrelationID ,
 CountersValue,
 CreateOrRemoveIndicator,
 CSAID ,
 CutAndPaste,
 DateAndTime,
 DefaultFaultHandling ,
 DestinationRoutingAddress ,
 Digits ,
 DisplayInformation ,
 DpSpecificCommonParameters ,
 XDuration,
 EventSpecificInformationBCSM ,
 EventSpecificInformationCharging ,
 EventTypeBCSM,
 EventTypeCharging ,
 FacilityGroup,
 FacilityGroupMember,
 FCIBillingChargingCharacteristics ,
 FeatureCode ,
 FeatureRequestIndicator,
 FilteredCallTreatment ,
 FilteringCharacteristics,
 FilteringCriteria ,
 FilteringTimeOut,
 ForwardCallIndicators,
 ForwardGVNS ,
 ForwardingCondition,
 GapCriteria ,
 GapIndicators,
 GapTreatment ,
 GenericName ,
 GenericNumbers ,
 HighLayerCompatibility,
 HoldCause,
 -- initialCallSegment,
 INprofile ,
 INServiceCompatibilityIndication ,
 INServiceCompatibilityResponse,

Annex A: ssf-scf

```

    IPAvailable ,
    IPRoutingAddress ,
    IPSSPCapabilities ,
    ISDNAccessRelatedInformation ,
--    LegID,
--    leg1,
    LocationNumber ,
--    MiscCallInfo,
    MonitorMode,
    NumberingPlan,
    OriginalCalledPartyID ,
    QoSParameter ,
    ProfileIdentifier ,

    Reason ,
    RedirectingPartyID ,
    RedirectionInformation,
    RegistratorIdentifier,
    ReportCondition,
    RoutingNumber ,

    RequestedInformationList ,
    RequestedInformationTypeList,
    RequestedUTSILList ,
    ResourceID ,
    ResourceStatus,
    ResponseCondition,
    RouteList ,
    ScfID ,
    SCIBillingChargingCharacteristics ,
    SDSSinformation ,
    ServiceInteractionIndicators ,
    ServiceInteractionIndicatorsTwo,
    ServiceKey,
    ServiceProfileIdentifier,
    TDPIIdentifier,
    TerminalType,
--    TimerID,
    TimerValue,
    TravellingClassMark ,
    TriggerData,
    TriggerDataIdentifier ,
    TriggerDPTYPE,
    Triggers ,
    TriggerResults ,
    TriggerStatus,
    TriggerType,
    USIInformation ,
    USIServiceIndicator ,
VPNIndicator ,

    numOfINProfile,
    numOfBCSMEvents,
    numOfChargingEvents,
    numOfLegs,
    numOfCSs

```

FROM INCS3datatypes --IN-CS3-SSF-SCF-datatypes ssf-scf-datatypes

-- InformationToSend

--FROM IN-CS3-scf-srf-datatypes scf-srf-datatypes

```

--    cancelFailed,
--    eTCFailed,
--    improperCallerResponse,
--    missingCustomerRecord,
--    missingParameter,
--    parameterOutOfRange,
--    requestedInfoError,
--    systemFailure,
--    taskRefused,
--    unavailableResource,
--    unexpectedComponentSequence,

```

Annex A: ssf-scf

```
-- unexpectedDataValue,
-- unexpectedParameter,
-- unknownLegID,
-- unknownResource

--FROM IN-CS3-errorTypes errorTypes
;
-- The following three definitions are local short-hand notation for convenience
.
--B1 ::= COMMON-BOUNDS      defined in Part 1 of Recommendation Q.1238
--B2 ::= SCF-SSF-BOUNDS    defined in this part (Part 2) of Recommendation Q.1238
--B3 ::= SCF-SRF-BOUNDS    defined in Part 3 of Recommendation Q.1238
```

```
-- *****
```

```
-- Definitions of MAP and ISUP types imported above.
```

```
CallReferenceNumber ::= INTEGER
```

```
ChargingMessageType ::= INTEGER
```

```
Ext-BasicServiceCode ::= CHOICE { -- Note: Ext-BasicServiceCode is not defined i
n MAP!
```

```
    bearerService [2] OCTET STRING (SIZE(1)),
    teleService [3] OCTET STRING
}
```

```
ISDNAddressString ::= OCTET STRING
```

```
OCSIApplicable ::= BOOLEAN
```

```
SubscriberState ::= INTEGER
```

```
SuppressionOfAnnouncement ::= BOOLEAN
```

```
-- *****
```

```
-- TYPE DEFINITIONS FOR IN CS-3 ERRORS (Informative).
```

```
CancelFailed ::= SEQUENCE {
    problem [0] ENUMERATED {
        unknownOperation(0),
        tooLate(1),
        operationNotCancellable(2)
    },
    operation [1] InvokeId
}
```

```
-- The parameter was not as expected (e.g. missing or out of range).
```

```
RequestedInfoError ::= ENUMERATED {
    unknownRequestedInfo(1),
    requestedInfoNotAvailable(2)
    -- other values FFS
}
```

```
-- The requested information cannot be found.
```

```
SystemFailure ::= UnavailableNetworkResource
```

```
-- The operation could not be completed due to a system failure
```

```
TaskRefused ::= ENUMERATED {
    generic(0),
    unobtainable (1),
    congestion(2)
    --other values FFS
}
```

```
-- An entity normally capable of the task requested cannot or chooses not
-- to perform the task at this
-- time. This includes error situations like congestion and unobtainable
-- address as used in e.g. the Connect operation.
```

```
UnavailableNetworkResource ::= ENUMERATED {
    unavailableResources(0),
    componentFailure(1),
    basicCallProcessingException(2),
}
```

Annex A: ssf-scf

```

    resourceStatusFailure(3),
    endUserFailure(4),
    screening(5)
}
-- Indicates the network resource that failed
-- Note that in IN CS3 the screening value can only be used by the SCF.

-- *****

-- *****
-- Due to a restriction in SDT it is not possible to denote values of types defi
ned in an imported
-- module (for example in DEFAULT statements). Therefore the types for which DEF
AULT statements are
-- used in this module, are redefined below. The corresponding type definitions
in IN-CS2-Datatypes
-- are changed to DTxxx.

```

```

CallSegmentID ::= INTEGER (1..numOfCSs)
initialCallSegment INTEGER ::= 1

```

```

LegID ::= CHOICE {
    sendingSideID    [0] LegType,
    receivingSideID  [1] LegType
}

```

```

LegType ::= OCTET STRING (SIZE(1))
leg1 LegType ::= '01'H
leg2 LegType ::= '02'H

```

```

MiscCallInfo ::= SEQUENCE {
    messageType    [0] ENUMERATED {
        request(0),
        notification(1)
    },
    dpAssignment    [1] ENUMERATED {
        individualLine(0),
        groupBased(1),
        switchBased(2)
    } OPTIONAL
}

```

```

TimerID ::= ENUMERATED {
    tssf(0)
    -- others ffs
}

```

```

-- *****

```

```

-- Operations and Arguments:

```

```

-- Direction: SCF -> SSF, Timer: Tasf
-- When receiving this operation, the SSF handles calls to destination in a spec
ified manner
-- without sending queries for every detected call. It is used for example for
providing
-- televoting or mass calling services. Simple registration functionality (count
ers) and
-- announcement control may be located at the SSF. The operation initializes th
e specified
-- counters in the SSF.

```

```

ActivateServiceFilteringArg ::= SEQUENCE {
    filteredCallTreatment    [0] FilteredCallTreatment ,
    filteringCharacteristics [1] FilteringCharacteristics,
    filteringTimeOut         [2] FilteringTimeOut ,
    filteringCriteria         [3] FilteringCriteria ,
    startTime                [4] DateAndTime OPTIONAL,
    extensions                [5] Extensions OPTIONAL
}

```

```

-- Direction: SCF -> SSF or SSF-> SCF, Timer: Tat
-- This operation is used to check for the continued existence of a relationship
between the SCF

```

Annex A: ssf-scf

-- and SSF. If the relationship is still in existence, then the SSF will respond . If no reply is received, then the SCF will assume that the SSF has failed in some way and will take the appropriate action.. As an option, this operation may be used in the reverse direction by the SSF to check for the continued existence of a relationship with the SCF

-- Direction: SSF -> SCF, Timer: Tadi

-- This operation is used to indicate availability of routing address and call type. (DP - Analysed_Info).

```
AnalysedInformationArg ::= SEQUENCE {
    dpSpecificCommonParameters    [0] DpSpecificCommonParameters ,
    dialledDigits                  [1] CalledPartyNumber    OPTIONAL,
    callingPartyBusinessGroupID    [2] CallingPartyBusinessGroupID  OPTIONAL,
    callingPartySubaddress         [3] CallingPartySubaddress  OPTIONAL,
    callingFacilityGroup           [4] FacilityGroup          OPTIONAL,
    callingFacilityGroupMember     [5] FacilityGroupMember    OPTIONAL,
    originalCalledPartyID         [6] OriginalCalledPartyID  OPTIONAL,
    prefix                         [7] Digits                OPTIONAL,
    redirectingPartyID             [8] RedirectingPartyID     OPTIONAL,
    redirectionInformation         [9] RedirectionInformation  OPTIONAL,
    routeList                      [10] RouteList             OPTIONAL,
    travellingClassMark            [11] TravellingClassMark    OPTIONAL,
    extensions                     [12] Extensions           OPTIONAL,
    featureCode                    [13] FeatureCode           OPTIONAL,
    accessCode                     [14] AccessCode            OPTIONAL,
    carrier                        [15] Carrier               OPTIONAL,
    componentType                  [16] ComponentType         OPTIONAL,
    component                      [17] Component             OPTIONAL,
    componentCorrelationID         [18] ComponentCorrelationID  OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to the trigger detection point processing rules

-- to specify when these parameters are included in the message.

-- Direction: SCF -> SSF, Timer: Tai

-- This operation is used to request the SSF to perform the originating basic call processing actions

-- to analyse destination information that is either collected from a calling party or provided by the SCF

-- (e.g. for number translation). This includes actions to validate the information according to an office

-- or customized dialing plan, and if valid, to determine call termination information, to include the called

-- party address, the type of call (e.g. intra-network or inter-network), and carrier (if inter-network).

-- If the called party is not served by the SSF, the SSF also determines a route index based on the called

-- party address and class of service, where the route index points to a list of outgoing trunk groups.

```
AnalyseInformationArg ::= SEQUENCE {
    destinationRoutingAddress      [0] DestinationRoutingAddress ,
    alertingPattern                [1] AlertingPattern          OPTIONAL,
    iSDNAccessRelatedInformation   [2] ISDNAccessRelatedInformation  OPTIONAL,
    originalCalledPartyID         [3] OriginalCalledPartyID  OPTIONAL,
    extensions                     [4] Extensions            OPTIONAL,
    callingPartyNumber             [5] CallingPartyNumber    OPTIONAL,
    callingPartysCategory          [6] CallingPartysCategory  OPTIONAL,
    calledPartyNumber              [7] CalledPartyNumber      OPTIONAL,
    chargeNumber                   [8] ChargeNumber           OPTIONAL,
    travellingClassMark            [9] TravellingClassMark    OPTIONAL,
    carrier                        [10] Carrier               OPTIONAL,
    serviceInteractionIndicators    [11] ServiceInteractionIndicators  OPTIONAL,
    iNServiceCompatibilityResponse [12] INServiceCompatibilityResponse  OPTIONAL,
    forwardGVNS                    [13] ForwardGVNS          OPTIONAL,
    backwardGVNS                   [14] BackwardGVNS          OPTIONAL,
    serviceInteractionIndicatorsTwo [15] ServiceInteractionIndicatorsTwo  OPTIONAL
,
    correlationID                  [16] CorrelationID          OPTIONAL,
    scfID                          [17] ScfID                  OPTIONAL,
}
```

Annex A: ssf-scf

```

    callSegmentID      [18] CallSegmentID    OPTIONAL,
    legToBeCreated      [19] LegID          OPTIONAL
}

```

-- Direction: SCF -> SSF, Timer: Tac
 -- This operation is used for interacting from the SCF with the SSF charging mechanisms.
 -- The ApplyChargingReport operation provides the feedback from the SSF to the SCF.
 -- This operation can also be used to instruct the SSF to release the call regarding some condition.

```

ApplyChargingArg ::= SEQUENCE {
    aChBillingChargingCharacteristics [0] AChBillingChargingCharacteristics
    ,
    partyToCharge [2] LegID OPTIONAL,
    extensions [3] Extensions OPTIONAL,
    releaseIndication [4] BOOLEAN OPTIONAL,
    releaseCause [5] Cause OPTIONAL
}

```

-- The TAG value 1 should not be used for future extensions (used in CS-1 by regions).
 -- The partyToCharge parameter indicates the party in the call to which the ApplyCharging operation should be applied. If it is not present, then it is applied to the A-party.
 -- The releaseIndicator shall be present and set to TRUE if the release condition is specified in the aChBillingCharacteristics parameter.

-- Direction: SSF -> SCF, Timer: Tacr
 -- This operation is used by the SSF to report to the SCF the occurrence of a specific charging event
 -- as requested by the SCF using the ApplyCharging operation.

ApplyChargingReportArg ::= CallResult
 -- Note: When the SSF sends the ApplyChargingReport operation as the last event from the Call Segment, the
 -- lastEventIndicator parameter such as the CallInformationReport operation is needed for indicating whether
 -- the event is last to the SCF. However, because there is no consideration for the parameter expansion in the
 -- CS-1, this parameter cannot be added. There are two alternatives for the solution. One is to be included
 -- into the CallResult parameter. And the other is to specify a new operation with this parameter. The latter is
 -- ffs.

-- Direction: SSF -> SCF or SRF -> SCF, Timer: Tari
 -- This operation is used when there is an assist or a hand-off procedure and may be sent by the SSF
 -- or SRF to the SCF. This operation is sent by the assisting SSF to SCF, when the initiating SSF has
 -- set up a connection to the SRF or to the assisting SSF as a result of receiving an EstablishTemporaryConnection
 -- or Connect/SelectRoute operation (in the case of hand-off) from the SCF.

```

AssistRequestInstructionsArg ::= SEQUENCE {
    correlationID [0] CorrelationID,
    iPAvailable [1] iPAvailable OPTIONAL,
    iPSSPCapabilities [2] iPSSPCapabilities OPTIONAL,
    extensions [3] Extensions OPTIONAL
}

```

-- OPTIONAL denotes network operator specific use. The value of the correlationID may be the
 -- Called Party Number supplied by the initiating SSF.

-- Direction: SCF -> SSF, Timer: Tatr
 -- This operation is used to request the SSF to resume terminating call processing action at the
 -- Authorize_Termination PIC of the call based on the information received from the SCF.

Annex A: ssf-scf

```

AuthorizeTerminationArg ::= SEQUENCE {
    alertingPattern          [0] AlertingPattern  OPTIONAL,
    callingPartyNumber       [1] CallingPartyNumber  OPTIONAL,
    destinationNumberRoutingAddress [2] CalledPartyNumber  OPTIONAL,
    displayInformation       [3] DisplayInformation  OPTIONAL,
    iSDNAccessRelatedInformation [4] ISDNAccessRelatedInformation  OPTIONAL,
    originalCalledPartyID    [5] OriginalCalledPartyID  OPTIONAL,
    travellingClassMark      [6] TravellingClassMark  OPTIONAL,
    extensions               [7] Extensions  OPTIONAL,
    iNServiceCompatibilityResponse [8] INServiceCompatibilityResponse  OPTIONAL,
    forwardGVNS              [9] ForwardGVNS  OPTIONAL,
    backwardGVNS             [10] BackwardGVNS  OPTIONAL,
    legID                    [11] LegID  OPTIONAL
}

```

--- OPTIONAL parameters are only provided if modifications are desired to basic call processing values.

--- Direction: SCF -> SSF, Timer: Tcg
 --- This operation is used to request the SSF to reduce the rate at which specific service requests are sent to the SCF. Use of this operation by the SCF to gap queries and updates at the SDF is outside the scope of this capability set.

```

CallGapArg ::= SEQUENCE {
    gapCriteria          [0] GapCriteria ,
    gapIndicators        [1] GapIndicators,
    controlType          [2] ControlType  OPTIONAL,
    gapTreatment         [3] GapTreatment  OPTIONAL,
    extensions           [4] Extensions  OPTIONAL
}

```

--- OPTIONAL denotes network operator optional. If gapTreatment is not present, the SSF will use a default treatment depending on network operator implementation.

--- Direction: SSF -> SCF, Timer: Tcirp
 --- This operation is used to send specific call information for a single call to the SCF as requested by the SCF in a previous CallInformationRequest.

```

CallInformationReportArg ::= SEQUENCE {
    requestedInformationList [0] RequestedInformationList ,
    correlationID            [1] CorrelationID  OPTIONAL,
    extensions               [2] Extensions  OPTIONAL,
    legID                    [3] LegID  OPTIONAL,
    lastEventIndicator       [4] BOOLEAN  DEFAULT FALSE
}

```

--- OPTIONAL denotes network operator optional.
 --- The lastEventIndicator parameter is set with 'TRUE' when the report is last in the Call Segment.
 --- In the CS-1, the lastEventIndicator should not be sent, and the meaning of DEFAULT is not applied. The SCF must decide whether the report is last without this parameter.

--- Direction: SCF -> SSF, Timer: Tcirp
 --- This operation is used to request the SSF to record specific information about a single call and report it to the SCF (with a CallInformationReport operation).

```

CallInformationRequestArg ::= SEQUENCE {
    requestedInformationTypeList [0] RequestedInformationTypeList,
    correlationID                [1] CorrelationID  OPTIONAL,
    extensions                   [2] Extensions  OPTIONAL,
    legID                        [3] LegID  OPTIONAL
}

```

--- OPTIONAL denotes network operator optional.

--- Direction: SCF -> SSF, or SCF -> SRF, Timer: Tcan
 --- This operation cancels the correlated previous operation or all previous requests
 --- This operation can also be used to cancel all outstanding requests and enable the state machine (SSF)

Q.1238.2 : December 1999

Annex A: ssf-scf

-- to go to idle. In this case the Cancel operation does not specify any specific operation to be cancelled.
 --- For the SCF-SRF operations that can be cancelled, refer to Q.1238.3

```
CancelArg ::= CHOICE {
  invokeID          [0] InvokeID,
  allRequests       [1] NULL,
  callSegmentToCancel [2] SEQUENCE {
    invokeID          [0] InvokeID,
    callSegmentID     [1] CallSegmentID
  },
  allRequestsForCallSegment [3] CallSegmentID
}
```

-- The InvokeID has the same value as that which was used for the SCF-SRF operation, i.e. is used to identify
 --- the correlated previous SCF-SRF operation to be cancelled.

-- Direction: SCF -> SSF, Timer: Tcsr
 -- This operation cancels the following processes: RequestFirstStatusMatchReport
 and
 -- RequestEveryStatusChangeReport.

```
CancelStatusReportRequestArg ::= SEQUENCE {
  resourceID [0] ResourceID OPTIONAL,
  extensions [1] Extensions OPTIONAL
}
```

-- Direction: SSF -> SCF, Timer: Tcdi
 -- This operation is used to indicate availability of complete initial information package/dialing string from
 -- originating party. (This event may have already occurred in the case of end-to-end signaling, in which case
 -- the waiting duration in this PIC is zero.) (DP - Collected_Info). For additional information on this operation
 -- and its use with open numbering plans, refer to clause 24.

```
CollectedInformationArg ::= SEQUENCE {
  dpSpecificCommonParameters [0] DpSpecificCommonParameters,
  dialledDigits [1] CalledPartyNumber OPTIONAL,
  callingPartyBusinessGroupID [2] CallingPartyBusinessGroupID OPTIONAL,
  callingPartySubaddress [3] CallingPartySubaddress OPTIONAL,
  callingFacilityGroup [4] FacilityGroup OPTIONAL,
  callingFacilityGroupMember [5] FacilityGroupMember OPTIONAL,
  originalCalledPartyID [6] OriginalCalledPartyID OPTIONAL,
  prefix [7] Digits OPTIONAL,
  redirectingPartyID [8] RedirectingPartyID OPTIONAL,
  redirectionInformation [9] RedirectionInformation OPTIONAL,
  travellingClassMark [10] TravellingClassMark OPTIONAL,
  extensions [11] Extensions OPTIONAL,
  featureCode [12] FeatureCode OPTIONAL,
  accessCode [13] AccessCode OPTIONAL,
  carrier [14] Carrier OPTIONAL,
  componentType [15] ComponentType OPTIONAL,
  component [16] Component OPTIONAL,
  componentCorrelationID [17] ComponentCorrelationID OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to clause 2.1.2 for the trigger detection point processing rules to specify
 -- when these parameters are included in the message.

-- Direction: SCF -> SSF, Timer: Tci
 -- This operation is used to request the SSF to perform the originating basic call processing actions to prompt
 -- a calling party for destination information, then collect destination information according to a specified
 -- numbering plan (e.g. for virtual private networks).

```
CollectInformationArg ::= SEQUENCE {
  alertingPattern [0] AlertingPattern OPTIONAL,
  numberingPlan [1] NumberingPlan OPTIONAL,
  originalCalledPartyID [2] OriginalCalledPartyID OPTIONAL,
  travellingClassMark [3] TravellingClassMark OPTIONAL,
  extensions [4] Extensions OPTIONAL,
}
```

Annex A: ssf-scf

```

callingPartyNumber      [5] CallingPartyNumber  OPTIONAL,
dialledDigits           [6] CalledPartyNumber   OPTIONAL,
serviceInteractionIndicators [7] ServiceInteractionIndicators OPTIONAL,
iNServiceCompatibilityResponse [8] INServiceCompatibilityResponse OPTIONAL,
forwardGVNS             [9] ForwardGVNS        OPTIONAL,
backwardGVNS            [10] BackwardGVNS       OPTIONAL,
serviceInteractionIndicatorsTwo [11] ServiceInteractionIndicatorsTwo OPTIONAL,
,
callSegmentID           [12] CallSegmentID     OPTIONAL,
legToBeCreated          [13] LegID             OPTIONAL
}

```

-- Direction: SCF -> SSF, Timer: Tcon
 -- This operation is used to request the SSF to perform the call processing actions to route or forward a call to
 -- a specified destination. To do so, the SSF may or may not use destination information from the calling party
 -- (e.g. dialed digits) and existing call setup information (e.g. route index to a list of trunk groups), depending on
 -- the information provided by the SCF.
 -- - When address information is only included in the Connect operation, call processing resumes at PIC3 in
 -- the O-BCSM.
 -- - When address information and routing information is included, call processing resumes at PIC4.

```

ConnectArg ::= SEQUENCE {
  destinationRoutingAddress [0] DestinationRoutingAddress ,
  alertingPattern           [1] AlertingPattern      OPTIONAL,
  correlationID             [2] CorrelationID        OPTIONAL,
  cutAndPaste               [3] CutAndPaste          OPTIONAL,
  forwardingCondition       [4] ForwardingCondition  OPTIONAL,
  iSDNAccessRelatedInformation [5] ISDNAccessRelatedInformation OPTIONAL,
  originalCalledPartyID     [6] OriginalCalledPartyID OPTIONAL,
  routeList                 [7] RouteList            OPTIONAL,
  scfID                     [8] ScfID                OPTIONAL,
  travellingClassMark       [9] TravellingClassMark  OPTIONAL,
  extensions                [10] Extensions          OPTIONAL,
  carrier                   [11] Carrier              OPTIONAL,
  serviceInteractionIndicators [26] ServiceInteractionIndicators OPTIONAL,
  callingPartyNumber        [27] CallingPartyNumber  OPTIONAL,
  callingPartysCategory     [28] CallingPartysCategory OPTIONAL,
  redirectingPartyID        [29] RedirectingPartyID   OPTIONAL,
  redirectionInformation    [30] RedirectionInformation OPTIONAL,
  displayInformation        [12] DisplayInformation  OPTIONAL,
  forwardCallIndicators     [13] ForwardCallIndicators OPTIONAL,
  genericNumbers            [14] GenericNumbers      OPTIONAL,
  serviceInteractionIndicatorsTwo [15] ServiceInteractionIndicatorsTwo OPTIONAL,
  iNServiceCompatibilityResponse [16] INServiceCompatibilityResponse OPTIONAL,
  forwardGVNS               [17] ForwardGVNS        OPTIONAL,
  backwardGVNS              [18] BackwardGVNS       OPTIONAL,
  chargeNumber              [19] ChargeNumber       OPTIONAL,
  callSegmentID             [20] CallSegmentID       OPTIONAL,
  legToBeCreated            [21] LegID               OPTIONAL,
  sDSSinformation           [22] SDSSinformation     OPTIONAL,
  calledDirectoryNumber     [23] CalledDirectoryNumber OPTIONAL,
  aTMTrafficDescriptor      [24] ATMTrafficDescriptor OPTIONAL,
  bearerCapability          [25] BearerCapability     OPTIONAL,
  qoSParameter              [31] QoSParameter       OPTIONAL
}

```

-- For alerting pattern, OPTIONAL denotes that this parameter only applies if SSF is the terminating local
 -- exchange for the subscriber.

-- Direction: SCF -> SSF, Timer: Tctr
 -- This operation is used to connect a call from the SSP to the physical entity containing the SRF.

```

ConnectToResourceArg ::= SEQUENCE {
  resourceAddress CHOICE {
    ipRoutingAddress [0] IPRoutingAddress ,
    legID            [1] LegID,
    ipAddressAndLegID [2] SEQUENCE {

```


Annex A: ssf-scf

```

    ipRoutingAddress      [0] IPRoutingAddress ,
    legID                 [1] LegID
  },
  nnone                   [3] NULL,
  callSegmentID           [5] CallSegmentID ,
  ipAddressAndCallSegment [6] SEQUENCE {
    ipRoutingAddress      [0] IPRoutingAddress ,
    callSegmentID         [1] CallSegmentID
  }
},
extensions                [4] Extensions OPTIONAL,
serviceInteractionIndicators [30] ServiceInteractionIndicators OPTIONAL,
serviceInteractionIndicatorsTwo [7] ServiceInteractionIndicatorsTwo OPTIONAL,
uSIServiceIndicator        [35] USIServiceIndicator OPTIONAL,
uSIInformation             [36] USIInformation OPTIONAL
}

```

-- Direction: SCF -> SSF, Timer: Tcue
 -- This operation is used to request the SSF to proceed with call processing at the DP at which it
 -- previously suspended call processing to await SCF instructions (i.e. proceed to the next point
 -- in call in the BCSM). The SSF continues call processing without substituting new data from SCF.
 -- This operation is not valid for a single call segment CSA with more than 2 legs or a multi call segment CSA.
 -- Direction: SCF -> SSF, Timer: Tcwa
 -- This operation is used to request the SSF to proceed with call processing at the DP at which it previously
 -- suspended call processing to await SCF instructions.
 -- It is also used to provide additional service related information to a User (Called Party or Calling Party) whilst
 -- the call processing proceeds.

```

ContinueWithArgumentArg ::= SEQUENCE {
  legorCSID CHOICE {
    legID      [0] LegID,
    csID       [9] CallSegmentID }      DEFAULT legID:sendingSideID:leg1
  ,
  alertingPattern [1] AlertingPattern OPTIONAL,
  genericName     [2] GenericName OPTIONAL,
  iNServiceCompatibilityResponse [3] INServiceCompatibilityResponse OPTIONAL,
  forwardGVNS     [4] ForwardGVNS OPTIONAL,
  backwardGVNS    [5] BackwardGVNS OPTIONAL,
  extensions      [6] Extensions OPTIONAL,
  serviceInteractionIndicatorsTwo [7] ServiceInteractionIndicatorsTwo OPTIONAL
L,
  sDSSinformation [8] SDSSinformation OPTIONAL
}

```

-- Direction SCF -> SSF, Timer Tcsa
 -- This operation is used to create a new CSA. The new CSA will not contain any Call Segments after creation.
 -- The SSF is responsible for specifying a new CSA identifier for the created CSA which is unique within
 -- the SSF.

```

CreateCallSegmentAssociationArg ::= SEQUENCE {
  extensions [0] Extensions OPTIONAL
}

```

```

CreateCallSegmentAssociationResultArg ::= SEQUENCE {
  newCallSegmentAssociation [0] CSAID
}

```

-- Direction: SCF -> SSF, Class 1, Timer: Tcrt
 -- This operation is used by the SCF outside the context of a call to create a new trigger detection point in the SSF/CCF by downloading trigger data (e.g. triggering criteria, ServiceKey, SCF address,...) or to remove an existing trigger.

```

CreateOrRemoveTriggerDataArg ::= SEQUENCE {

```

Annex A: ssf-scf

```

createOrRemoveIndicator [ 0 ] CreateOrRemoveIndicator, -- DEFAULT create,
dPName [1] EventTypeBCSM OPTIONAL,
triggerDPTYPE [2] TriggerDPTYPE, -- DEFAULT tdp-r,
serviceKey [3] ServiceKey OPTIONAL,
profileIdentifier [4] ProfileIdentifier OPTIONAL,
triggerData [5] TriggerData
OPTIONAL,
defaultFaultHandling [6] DefaultFaultHandling
OPTIONAL,
tDPIIdentifier [7] TDPIIdentifier
OPTIONAL,
extensions [8] Extensions OPTIONAL
}

```

```

CreateOrRemoveTriggerDataResultArg ::= SEQUENCE {
triggerStatus [0] TriggerStatus,
tDPIIdentifier [1] TDPIIdentifier ,

registratorIdentifier [2] RegistratorIdentifier OPTIONAL,
extensions [3] Extensions OPTIONAL
}

```

-- Direction: SCF -> SSF, Timer: Tdfc
 -- This operation is used to disconnect a forward temporary connection or a connection to a resource.

-- Direction: SCF -> SSF, Timer: Tdfcwa
 -- This operation is used to disconnect a forward temporary connection or a connection to a resource.

DisconnectForwardConnectionWithArgumentArg ::= SEQUENCE {

```

partyToDisconnect CHOICE {
legID [0] LegID,
callSegmentID [1] CallSegmentID
},
extensions [2] Extensions OPTIONAL,
uSIServiceIndicator [3] USIServiceIndicator OPTIONAL,
uSIInformation [4] USIInformation OPTIONAL
}

```

-- Direction: SCF -> SSF, Timer: Tdl
 -- This operation is issued by the SCF to release a specific leg associated with the call and retain any other legs not specified in the DisconnectLeg. Any leg may be disconnected, including the controlling leg, without completely releasing all legs.

```

DisconnectLegArg ::= SEQUENCE {
legToBeReleased [0] LegID,
releaseCause [1] Cause OPTIONAL,
extensions [2] Extensions OPTIONAL
}

```

-- Direction SSF -> SCF, Timer: Ter
 -- This operation is used by SSF to inform the SCF of an error/exception

```

EntityReleasedArg ::= CHOICE {
cSFailure [0] SEQUENCE{
callSegmentID [0] CallSegmentID ,
reason [1] Reason OPTIONAL,
cause [2] Cause OPTIONAL
},
bCSMFailure [1] SEQUENCE{
legID [0] LegID,
reason [1] Reason OPTIONAL,
cause [2] Cause OPTIONAL
}
}

```

Annex A: ssf-scf

```

    }
}

```

-- Direction: SCF -> SSF, Timer: Tetc
 -- This operation is used to create a connection to a resource for a limited period of time
 -- (e.g. to play an announcement, to collect user information); it implies the use of the assist procedure.

```

EstablishTemporaryConnectionArg ::= SEQUENCE {
    assistingSSPIPRoutingAddress [0] AssistingSSPIPRoutingAddress,
    correlationID [1] CorrelationID OPTIONAL,
    partyToConnect CHOICE {
        legID [2] LegID,
        callSegmentID [7] CallSegmentID OPTIONAL,
    },
    scfID [3] ScfID OPTIONAL,
    extensions [4] Extensions OPTIONAL,
    carrier [5] Carrier OPTIONAL,
    serviceInteractionIndicators [30] ServiceInteractionIndicators OPTIONAL,
    serviceInteractionIndicatorsTwo [6] ServiceInteractionIndicatorsTwo OPTIONAL
}

```

-- Direction: SSF -> SCF, Timer: Tenc
 -- This operation is used by the SSF to report to the SCF the occurrence of a specific charging event
 -- type as previously requested by the SCF in a RequestNotificationChargingEvent operation.

```

EventNotificationChargingArg ::= SEQUENCE {
    eventTypeCharging [0] EventTypeCharging,
    eventSpecificInformationCharging [1] EventSpecificInformationCharging OPTIONAL,
    legID [2] LegID OPTIONAL,
    extensions [3] Extensions OPTIONAL,
    monitorMode [30] MonitorMode -- DEFAULT notifyAndContinue
}

```

-- OPTIONAL denotes network operator specific use.

-- Direction: SSF -> SCF, Timer: Terb
 -- This operation is used to notify the SCF of a call-related event (e.g. BCSM events such as busy or no answer) previously requested by the SCF in a RequestReportBCSMEvent operation.

```

EventReportBCSMArg ::= SEQUENCE {
    eventTypeBCSM [0] EventTypeBCSM,
    bcsmeventCorrelationID [1] CorrelationID OPTIONAL,
    eventSpecificInformationBCSM [2] EventSpecificInformationBCSM OPTIONAL,
    legID [3] LegID OPTIONAL,
    miscCallInfo [4] MiscCallInfo DEFAULT {messageType request},
    extensions [5] Extensions OPTIONAL,
    componentType [6] ComponentType OPTIONAL,
    component [7] Component OPTIONAL,
    componentCorrelationID [8] ComponentCorrelationID OPTIONAL
}

```

OP

-- SSF->SCF, Timer: Terf
 -- This operation is issued by the SSF to report the SCF the event, that was previously requested by the
 -- SCF, the CCF/SSF receives a DSS1 message which contains a FACILITY IE. Criteria for the report, like
 -- reception of the ReturnResult which is specified with ComponentType, is optionally checked
 -- before issuing this operation.

```

EventReportFacilityArg ::= SEQUENCE {
    componentType [0] ComponentType OPTIONAL,
    component [1] Component OPTIONAL,
    legID [2] LegID OPTIONAL,
    componentCorrelationID [3] ComponentCorrelationID OPTIONAL,
}

```

Annex A: ssf-scf

```

    extensions          [4] Extensions    OPTIONAL
  }

```

-- Direction: SSF -> SCF, Timer: Tfs
 -- This operation is used for indication of a call termination attempt from the terminating half BCSM. (DP - Facility_Selected_And_Available).

```

FacilitySelectedAndAvailableArg ::= SEQUENCE {
  dpSpecificCommonParameters    [0] DpSpecificCommonParameters ,
  calledPartyBusinessGroupID     [1] CalledPartyBusinessGroupID  OPTIONAL,
  calledPartySubaddress          [2] CalledPartySubaddress       OPTIONAL,
  callingPartyBusinessGroupID    [3] CallingPartyBusinessGroupID OPTIONAL,
  callingPartyNumber             [4] CallingPartyNumber          OPTIONAL,
  originalCalledPartyID          [5] OriginalCalledPartyID       OPTIONAL,
  redirectingPartyID             [6] RedirectingPartyID          OPTIONAL,
  redirectionInformation          [7] RedirectionInformation      OPTIONAL,
  routeList                     [8] RouteList                    OPTIONAL,
  travellingClassMark            [9] TravellingClassMark         OPTIONAL,
  extensions                     [10] Extensions                OPTIONAL,
  componentType                  [11] ComponentType              OPTIONAL,
  component                      [12] Component                  OPTIONAL,
  componentCorrelationID         [13] ComponentCorrelationID     OPTIONAL
}

```

-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point processing rules
 -- to specify when these parameters are included in the message.

-- Direction: SCF -> SSF, Timer: Tfci
 -- This operation is used to request the SSF to generate, register a call record or to include some information
 -- in the default call record. The registered call record is intended for offline charging of the call.

```

FurnishChargingInformationArg ::= FCIBillingChargingCharacteristics

```

-- Direction: SCF -> SSF, Timer: Then
 -- This operation is used to provide the capability of queueing a call during the setup phase (e.g. to provide
 -- a call completion to busy, the call would be queued until the destination becomes free).

```

HoldCallInNetworkArg ::= CHOICE {
  holdcause [0] HoldCause,
  empty     [1] NULL
}

```

-- holdcause is optional and denotes network operator specific use.

-- Direction: SSF -> SCF, Timer: Tidp
 -- This operation is used after a TDP to indicate request for service.

```

InitialDPArg ::= SEQUENCE {
  serviceKey          [0] ServiceKey          OPTIONAL,
  dialledDigits       [1] CalledPartyNumber    OPTIONAL,
  calledPartyNumber   [2] CalledPartyNumber    OPTIONAL,
  callingPartyNumber  [3] CallingPartyNumber   OPTIONAL,
  callingPartyBusinessGroupID [4] CallingPartyBusinessGroupID OPTIONAL,
  callingPartysCategory [5] CallingPartysCategory OPTIONAL,
  callingPartySubaddress [6] CallingPartySubaddress OPTIONAL,
  cGEncountered       [7] CGEncountered        OPTIONAL,
  iPSSPCapabilities   [8] IPSSPCapabilities    OPTIONAL,
  iPAvailable          [9] IPAvailable          OPTIONAL,
  locationNumber       [10] LocationNumber      OPTIONAL,
  miscCallInfo         [11] MiscCallInfo        OPTIONAL,
  originalCalledPartyID [12] OriginalCalledPartyID OPTIONAL,
  serviceProfileIdentifier [13] ServiceProfileIdentifier OPTIONAL,
  terminalType         [14] TerminalType        OPTIONAL,
  extensions           [15] Extensions          OPTIONAL,
  triggerType          [16] TriggerType         OPTIONAL,
  highLayerCompatibility [23] HighLayerCompatibility OPTIONAL,
  serviceInteractionIndicators [24] ServiceInteractionIndicators OPTIONAL,
  additionalCallingPartyNumber [25] AdditionalCallingPartyNumber OPTIONAL,
  forwardCallIndicators [26] ForwardCallIndicators OPTIONAL,
  bearerCapability     [27] BearerCapability     OPTIONAL,
}

```

Annex A: ssf-scf

```

eventTypeBCSM          [28] EventTypeBCSM          OPTIONAL,
redirectingPartyID      [29] RedirectingPartyID      OPTIONAL,
redirectionInformation  [30] RedirectionInformation  OPTIONAL,
cause                  [17] Cause                    OPTIONAL,
componentType          [18] ComponentType           OPTIONAL,
component              [19] Component               OPTIONAL,
componentCorrelationID [20] ComponentCorrelationID       OPTIONAL,
iSDNAccessRelatedInformation [21] ISDNAccessRelatedInformation OPTIONAL,
iNServiceCompatibilityIndication [22] INServiceCompatibilityIndication
                        OPTIONAL,
genericNumbers         [31] GenericNumbers          OPTIONAL,
serviceInteractionIndicatorsTwo [32] ServiceInteractionIndicatorsTwo OPTIONAL,
forwardGVNS            [33] ForwardGVNS             OPTIONAL,
createdCallSegmentAssociation [34] CSAID             OPTIONAL,
uSIServiceIndicator    [35] USIServiceIndicator     OPTIONAL,
uSIInformation         [36] USIInformation          OPTIONAL,
carrier                [37] Carrier                 OPTIONAL,
cCSS                  [38] CCSS                     OPTIONAL,
vPNIndicator           [39] VPNIndicator            OPTIONAL,
cNInfo                [40] OCTET STRING(SIZE(1..13)) OPTIONAL,
callReference          [41] CallReference           OPTIONAL,
routingNumber          [42] RoutingNumber           OPTIONAL,
aTMTrafficDescriptor   [43] ATMTrafficDescriptor    OPTIONAL,
qoSParameter           [45] QoSParameter            OPTIONAL,
callingGeodeticLocation [46] CallingGeodeticLocation  OPTIONAL
}

```

-- OPTIONAL for iPSSPCapabilities, iPAvailable, cGEncountered, and miscCallInfo denotes network

-- operator specific use.

-- OPTIONAL for dialledDigits, callingPartyNumber, and callingPartysCategory refer to clause 2 for the trigger

-- detection point processing rules to specify when these parameters are included in the message.

-- OPTIONAL for terminalType indicates that this parameter applies only at originating or terminating

-- local exchanges if the SSF has this information.

-- encoding of cNInfo is similar to the NNI specific information in the APP parameter in Q.765.1

--first octet contains Q.765.1 CNID indicator, only bit 5 and 6 are significant,

This is followed by zero to 12 octets

--that contains the Q.765.1 CNID parameter, i.e. the Corporate Telecommunication s Network Identifier , if present..

-- Direction: SCF -> SSF, Timer: Tica

-- This operation is used to request the SSF to create a new call to one calling party using address

-- information provided by the SCF.

```

InitiateCallAttemptArg ::= SEQUENCE {
    destinationRoutingAddress [0] DestinationRoutingAddress,
    alertingPattern           [1] AlertingPattern           OPTIONAL,
    iSDNAccessRelatedInformation [2] ISDNAccessRelatedInformation OPTIONAL,
    travellingClassMark        [3] TravellingClassMark      OPTIONAL,
    extensions                 [4] Extensions               OPTIONAL,
    serviceInteractionIndicators [29] ServiceInteractionIndicators OPTIONAL,
    callingPartyNumber         [30] CallingPartyNumber      OPTIONAL,
    legToBeCreated             [5] LegID                     DEFAULT sendingSideID:leg1,
    newCallSegment             [6] CallSegmentID            DEFAULT initialCallSegment,
    iNServiceCompatibilityResponse [7] INServiceCompatibilityResponse OPTIONAL,
    serviceInteractionIndicatorsTwo [8] ServiceInteractionIndicatorsTwo OPTIONAL,
    correlationID              [9] CorrelationID            OPTIONAL,
    scfID                      [10] ScfID                    OPTIONAL,
    carrier                    [11] Carrier                 OPTIONAL,
    callReference              [12] CallReference           OPTIONAL,
    calledDirectoryNumber      [13] CalledDirectoryNumber    OPTIONAL
}

```

-- Direction: SCF -> SSF, Class 1, Timer: Tmtd

-- This operation is used to activate, deactivate or retrieve

-- the status of one or several trigger detection point linked to a subscriber record known at the switch,

-- e.g. related to an access line.

Q.1238.2 : December 1999

Annex A: ssf-scf

```

ManageTriggerDataArg ::= SEQUENCE {
    actionIndicator    [0] ActionIndicator,
    triggerDataIdentifier CHOICE {
        profileAndDP    [1] TriggerDataIdentifier ,
        -- one trigger
        profile          [5] ProfileIdentifier
    },
    registratorIdentifier [2] RegistratorIdentifier OPTIONAL,
    extensions            [3] Extensions OPTIONAL,
    tDPIIdentifier        [4] TDPIIdentifier OPTIONAL
}

```

```

ManageTriggerDataResultArg ::= CHOICE
{
    oneTriggerResult SEQUENCE {
        actionPerformed [0] ActionPerformed,
        extensions      [1] Extensions OPTIONAL},
    severalTriggerResult [1] SEQUENCE {
        results [0] TriggerResults ,
        extensions [1] Extensions OPTIONAL }
}

```

-- Direction: SCF -> SSF. Timer: T mc
 -- This operation is issued by the SCF to merge two associated CSs , into one CS .

```

MergeCallSegmentsArg ::= SEQUENCE {
    sourceCallSegment [0] CallSegmentID ,
    targetCallSegment [1] CallSegmentID DEFAULT initialCallSegment,
    extensions        [2] Extensions OPTIONAL
}

```

-- Direction: SCF -> SSF, Timer Tmcs
 -- This operation moves a CS from the source CSA to the the target CSA

```

MoveCallSegmentsArg ::= SEQUENCE {
    targetCallSegmentAssociation [0] CSAID ,

    callSegments [1] SEQUENCE SIZE (1..numOfCSs)
    OF SEQUENCE {
        sourceCallSegment [0] CallSegmentID DEFAULT initialCallSegment,
        newCallSegment    [1] CallSegmentID
    },
    legs [2] ExportLegsType,
    extensions [3] Extensions OPTIONAL
}

```

-- Direction : SCF ->SSF, Timer: T ml
 -- This operation is issued by the SCF to move a leg from one CS to another with which it is associated.

```

MoveLegArg ::=SEQUENCE {
    legIDToMove [0] LegID,
    targetCallSegment [1] CallSegmentID DEFAULT 1,
    extensions [2] Extensions OPTIONAL
}

```

-- For the OPTIONAL parameters, refer to clause 2 for the trigger
 -- detection point processing rules to specify when these parameters are
 -- included in the message.

-- Direction: SSF -> SCF. Timer: T ob
 -- This operation is issued by the SSF after detecting a valid trigger condition at the O_Abandon DP or to
 -- report an oAbandon event requested by the RequestReportBCSMEvent.

```

OAbandonArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters ,
    callSegmentID [1] CallSegmentID ,
    releaseCause [2] Cause OPTIONAL,
    extensions [3] Extensions OPTIONAL
}

```

-- For the OPTIONAL parameters, refer to the trigger detection point rules to specify

Annex A: ssf-scf

-- when these parameter are included in the message.
 -- Type definition for PointInCall is outside the scope of this capability sets.
 Use of T/EDP-R is outside
 -- the scope of this capability set.

-- Direction: SSF -> SCF, Timer: Toa
 -- This operation is used for indication from the terminating half BCSM that the call is accepted and answered
 -- by terminating party (e.g. terminating party goes offhook, Q.931 Connect message received, ISDN-UP Answer
 -- message received) (DP - O_Answer).

```
OAnswerArg ::= SEQUENCE {
    dpSpecificCommonParameters      [0] DpSpecificCommonParameters ,
    callingPartyBusinessGroupID      [1] CallingPartyBusinessGroupID  OPTIONAL,
    callingPartySubaddress            [2] CallingPartySubaddress      OPTIONAL,
    callingFacilityGroup              [3] FacilityGroup              OPTIONAL,
    callingFacilityGroupMember        [4] FacilityGroupMember        OPTIONAL,
    originalCalledPartyID             [5] OriginalCalledPartyID      OPTIONAL,
    redirectingPartyID                [6] RedirectingPartyID          OPTIONAL,
    redirectionInformation             [7] RedirectionInformation      OPTIONAL,
    routeList                         [8] RouteList                  OPTIONAL,
    travellingClassMark               [9] TravellingClassMark         OPTIONAL,
    extensions                        [10] Extensions                 OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point processing rules
 -- to specify when these parameters are included in the message.
 -- Direction: SSF -> SCF, Timer: Tob
 -- This operation is used for Indication from the terminating half BCSM that the terminating party is busy
 -- (DP - O_Called_Party_Busy). .

```
OCalledPartyBusyArg ::= SEQUENCE {
    dpSpecificCommonParameters      [0] DpSpecificCommonParameters ,
    busyCause                        [1] Cause                      OPTIONAL,
    callingPartyBusinessGroupID      [2] CallingPartyBusinessGroupID  OPTIONAL,
    callingPartySubaddress            [3] CallingPartySubaddress      OPTIONAL,
    callingFacilityGroup              [4] FacilityGroup              OPTIONAL,
    callingFacilityGroupMember        [5] FacilityGroupMember        OPTIONAL,
    originalCalledPartyID             [6] OriginalCalledPartyID      OPTIONAL,
    prefix                           [7] Digits                    OPTIONAL,
    redirectingPartyID                [8] RedirectingPartyID          OPTIONAL,
    redirectionInformation             [9] RedirectionInformation      OPTIONAL,
    routeList                         [10] RouteList                  OPTIONAL,
    travellingClassMark               [11] TravellingClassMark         OPTIONAL,
    extensions                        [12] Extensions                 OPTIONAL,
    carrier                           [13] Carrier                   OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point processing rules
 -- to specify when these parameters are included in the message.
 -- Direction: SSF -> SCF, Timer: Tod
 -- This operation is used for a disconnect indication (e.g. onhook, Q.931 Disconnect message, SS7 Release message)
 -- is received from the originating party, or received from the terminating party via the terminating half BCSM.
 -- (DP - O_Disconnect). .

```
ODisconnectArg ::= SEQUENCE {
    dpSpecificCommonParameters      [0] DpSpecificCommonParameters ,
    callingPartyBusinessGroupID      [1] CallingPartyBusinessGroupID  OPTIONAL,
    callingPartySubaddress            [2] CallingPartySubaddress      OPTIONAL,
    callingFacilityGroup              [3] FacilityGroup              OPTIONAL,
    callingFacilityGroupMember        [4] FacilityGroupMember        OPTIONAL,
    releaseCause                      [5] Cause                      OPTIONAL,
    routeList                         [6] RouteList                  OPTIONAL,
    extensions                        [7] Extensions                 OPTIONAL,
    carrier                           [8] Carrier                   OPTIONAL,
    connectTime                       [9] Integer4                  OPTIONAL,
    componentType                     [10] ComponentType             OPTIONAL,
    component                         [11] Component                 OPTIONAL,
    componentCorrelationID            [12] ComponentCorrelationID     OPTIONAL
}
```

Annex A: ssf-scf

-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point processing rules
 -- to specify when these parameters are included in the message.

-- Direction: SSF -> SCF, Timer: Tomc
 -- This operation is used to indicate a feature request is received from the originating party
 -- (e.g. hook flash, ISDN feature activation, Q.931 HOLD or RETrieve message). (DP - O_Mid_Call).

```
MidCallArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters,
    calledPartyBusinessGroupID [1] CalledPartyBusinessGroupID OPTIONAL,
    calledPartySubaddress [2] CalledPartySubaddress OPTIONAL,
    callingPartyBusinessGroupID [3] CallingPartyBusinessGroupID OPTIONAL,
    callingPartySubaddress [4] CallingPartySubaddress OPTIONAL,
    featureRequestIndicator [5] FeatureRequestIndicator OPTIONAL,
    extensions [6] Extensions OPTIONAL,
    carrier [7] Carrier OPTIONAL,
    componentType [8] ComponentType OPTIONAL,
    component [9] Component OPTIONAL,
    componentCorrelationID [10] ComponentCorrelationID OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to the trigger detection point processing rules
 -- to specify when these parameters are included in the message.

-- Direction: SSF -> SCF, Timer: Tona
 -- This operation is used for indication from the terminating half BCSM that the terminating party does not
 -- answer within a specified time period (DP - O_No_Answer).

```
ONoAnswerArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters,
    callingPartyBusinessGroupID [1] CallingPartyBusinessGroupID OPTIONAL,
    callingPartySubaddress [2] CallingPartySubaddress OPTIONAL,
    callingFacilityGroup [3] FacilityGroup OPTIONAL,
    callingFacilityGroupMember [4] FacilityGroupMember OPTIONAL,
    originalCalledPartyID [5] OriginalCalledPartyID OPTIONAL,
    prefix [6] Digits OPTIONAL,
    redirectingPartyID [7] RedirectingPartyID OPTIONAL,
    redirectionInformation [8] RedirectionInformation OPTIONAL,
    routeList [9] RouteList OPTIONAL,
    travellingClassMark [10] TravellingClassMark OPTIONAL,
    extensions [11] Extensions OPTIONAL,
    carrier [12] Carrier OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to the trigger detection point processing rules
 -- to specify when these parameters are included in the message.

-- Direction: SSF -> SCF, Timer: Tora
 -- This operation is used for indication of a call origination attempt from the originating half BCSM.
 -- (DP - Origination_Attempt).

```
OriginationAttemptArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters,
    callingPartyBusinessGroupID [1] CallingPartyBusinessGroupID OPTIONAL,
    callingPartySubaddress [2] CallingPartySubaddress OPTIONAL,
    callingFacilityGroup [3] FacilityGroup OPTIONAL,
    callingFacilityGroupMember [4] FacilityGroupMember OPTIONAL,
    carrier [5] Carrier OPTIONAL,
    travellingClassMark [6] TravellingClassMark OPTIONAL,
    extensions [7] Extensions OPTIONAL,
    componentType [8] ComponentType OPTIONAL,
    component [9] Component OPTIONAL,
    componentCorrelationID [10] ComponentCorrelationID OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point processing rules
 -- to specify when these parameters are included in the message.

Annex A: ssf-scf

-- Direction: SSF -> SCF, Timer: Toaa
 -- This operation is used to Indicate the desire to place outgoing call (e.g. of
 hook, Q.931 Setup message,
 -- ISDN-UP IAM message) and authority/ability to place outgoing call verified (D
 P -
 -- Origination_Attempt_Authorized).

```
OriginationAttemptAuthorizedArg ::= SEQUENCE {
    dpSpecificCommonParameters    [0] DpSpecificCommonParameters ,
    dialledDigits                  [1] CalledPartyNumber    OPTIONAL,
    callingPartyBusinessGroupID    [2] CallingPartyBusinessGroupID  OPTIONAL,
    callingPartySubaddress          [3] CallingPartySubaddress  OPTIONAL,
    callingFacilityGroup            [4] FacilityGroup          OPTIONAL,
    callingFacilityGroupMember      [5] FacilityGroupMember    OPTIONAL,
    travellingClassMark             [6] TravellingClassMark    OPTIONAL,
    extensions                      [7] Extensions            OPTIONAL,
    carrier                        [8] Carrier                OPTIONAL,
    componentType                  [9] ComponentType          OPTIONAL,
    component                      [10] Component             OPTIONAL,
    componentCorrelationID         [11] ComponentCorrelationID  OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point processing rules
 -- to specify when these parameters are included in the message.

-- Direction: SSF -> SCF, Timer: Tos
 -- This operation is issued by the SSF after detecting a valid trigger condition at the O_Suspend DP or to
 -- report an oSuspended event requested by the RequestReportBCSMEvent.

```
OSuspendedArg ::= SEQUENCE {
    dpSpecificCommonParameters    [0] DpSpecificCommonParameters ,
    legID                          [1] LegID                OPTIONAL,
    extensions                      [2] Extensions            OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point processing rules
 -- to specify when these parameters are included in the message.
 -- Modification to LegID is ffs. Use for T/EDP-R is ffs.

-- Direction: SCF -> SSF, Timer: Tre
 -- This operation is issued by the SCF to reestablish communication between the controlling leg and the
 -- (held) passive leg(s).

```
ReconnectArg ::= SEQUENCE {
    notificationDuration           [0] ApplicationTimer    OPTIONAL,
    alertingPattern                [1] AlertingPattern     OPTIONAL,
    displayInformation             [2] DisplayInformation  OPTIONAL,
    extensions                     [3] Extensions          OPTIONAL,
    callSegmentID                 [4] CallSegmentID        OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to clause 2.1.2 for the trigger detection point processing rules
 -- to specify when these parameters are included in the message.

-- Direction: SCF -> SSF, Timer: Trc
 -- This operation is used to tear down an existing call at any phase of the call for all parties
 -- involved in the call.

```
ReleaseCallArg ::= CHOICE {
    initialCallSegment            Cause ,
    callSegmentToRelease          [1] SEQUENCE {
        callSegment               [0] INTEGER (1..numOfCSs),
        releaseCause              [1] Cause                OPTIONAL,
        forcedRelease             [2] BOOLEAN DEFAULT FALSE
    },
    allCallSegments              [2] SEQUENCE {
        releaseCause              [0] Cause                OPTIONAL,
        timeToRelease             [1] TimerValue            OPTIONAL,
        forcedRelease             [2] BOOLEAN DEFAULT FALSE
    }
}
```

Annex A: ssf-scf

```

    }
}
-- A default cause value of decimal 31 (normal unspecified) should be coded appropriately.

```

```

-- Direction: SSF -> SCF, Timer: Tru
-- This operation is issued by the SSF in the context of the USI feature. It is used to report the receipt
-- of a User to Service Information (USI) to the SCF.

```

```

ReportUTSIArg ::= SEQUENCE {
    uSIServiceIndicator [0] USIServiceIndicator,
    legID [1] LegID DEFAULT receivingSideID:leg1,
    uSIInformation [2] USIInformation,
    extensions [3] Extensions OPTIONAL
}

```

```

-- Direction: SCF -> SSF, Timer: Trcs
-- This operation is used to request the SSF to report immediately the busy/idle status of a physical
-- termination resource.

```

```

RequestCurrentStatusReportArg ::= ResourceID

```

```

RequestCurrentStatusReportResultArg ::= SEQUENCE {
    resourceStatus [0] ResourceStatus,
    resourceID [1] ResourceID OPTIONAL,
    extensions [2] Extensions OPTIONAL
}

```

```

-- Direction: SCF -> SSF, Timer: Tres
-- This operation is used to request the SSF to report every change of busy/idle status of a physical
-- termination resource.

```

```

RequestEveryStatusChangeReportArg ::= SEQUENCE {
    resourceID [0] ResourceID,
    correlationID [1] CorrelationID OPTIONAL,
    monitorDuration [2] XDuration OPTIONAL,
    extensions [3] Extensions OPTIONAL
}

```

```

-- For correlationID OPTIONAL denotes network operator optional.
-- monitorDuration is required if outside the context of a call. It is not expected if we are in the context
-- of a call, because in that case the end of the call implicitly means the end of the monitoring.

```

```

-- Direction: SCF -> SSF, Timer: Trfs
-- This operation is used to request the SSF to report the first change busy/idle to the specified status of
-- a physical termination resource.

```

```

RequestFirstStatusMatchReportArg ::= SEQUENCE {
    resourceID [0] ResourceID OPTIONAL,
    resourceStatus [1] ResourceStatus OPTIONAL,
    correlationID [2] CorrelationID OPTIONAL,
    monitorDuration [3] XDuration OPTIONAL,
    extensions [4] Extensions OPTIONAL,
    bearerCapability [5] BearerCapability OPTIONAL
}

```

```

-- For correlationID OPTIONAL denotes network operator optional.
-- monitorDuration is required if outside the context of a call. It is not expected if we are in the context
-- of a call, because in that case the end of the call implicitly means the end of the monitoring.

```

```

-- Direction: SCF -> SSF, Timer: Trnc
-- This operation is used by the SCF to instruct the SSF on how to manage the charging events
-- which are received from other FE's and not under control of the service logging instance.

```

```

RequestNotificationChargingEventArg ::= SEQUENCE SIZE(1..numOfChargingEvents) OF

```

Annex A: ssf-scf

ChargingEvent

-- Direction: SCF -> SSF, Timer: Trrb
 -- This operation is used to request the SSF to monitor for a call-related event (e.g. BCSM events such as busy or no answer), then send a notification back to the SCF when the event is detected.
 -- It is proposed that Event Detection Point (EDP) processing is always initiated by RequestReportBCSMEvent and the
 -- EDP may be acknowledged with either an EventReportBCSM or by a DP-specific operations:
 -- NOTE - the application context should identify whether BCSM Event Handling Package is being used, or whether DP Specific Event Handling Package is being used.
 -- for a particular IN, only one of the two alternatives identified by the respective Packages should be selected (i.e., only one approach should be selected for a given application context).
 -- Every EDP must be explicitly armed by the SCF via a RequestReportBCSMEvent operation. No implicit arming of EDPs at the SSF after reception of any operation (different from RequestReportBCSMEvent) from the SCF is allowed.

```
RequestReportBCSMEventArg ::= SEQUENCE {
    bcsmEvents          [0] SEQUENCE SIZE(1..numOfBCSMEvents) OF
                          BCSMEvent,
    bcsmEventCorrelationID [1] CorrelationID OPTIONAL,
    extensions          [2] Extensions OPTIONAL
}
```

-- Indicates the BCSM related events for notification.
 -- For correlationID OPTIONAL denotes network operator optional.

-- SCF->SSF, Timer: Trrfe
 -- This operation is issued by the SCF to request the SSF to report the SCF the event that the CCF/SSF receives a DSS1 message which contains a FACILITY IE during a BCSM being suspended at a DP.

```
RequestReportFacilityEventArg ::= SEQUENCE{
    componentTypes      [0] SEQUENCE SIZE(1..3)
OF ComponentType,      -- DEFAULT {any},
    legID              [1] LegID OPTIONAL,
    componentCorrelationID [2] ComponentCorrelationID OPTIONAL,
    monitorDuration     [3] XDuration,
    extensions          [4] Extensions OPTIONAL
}
```

-- componentTypes specifies the component types which should be reported to the SCF.

-- monitorDuration specifies the monitor duration.

-- Direction: SCF -> SSF, Timer: Trru
 -- This operation is issued by the SCF in the context of the USI feature to request the SSF to monitor for a User to Service Information (UTSI) information element, which are received from a user.

```
RequestReportUTSIArg ::= SEQUENCE {
    requestedUTSIList [0] RequestedUTSIList,
    extensions        [1] Extensions OPTIONAL,
    legID             [2] LegID DEFAULT sendingSideID:leg1
}
```

-- Direction: SCF -> SSF, Timer: Trt
 -- This operation is used to request the SSF to refresh an application timer in the SSF.

```
ResetTimerArg ::= SEQUENCE {
    timerID      [0] TimerID DEFAULT tssf,
    timervalue   [1] TimerValue,
    extensions   [2] Extensions OPTIONAL,
```

Annex A: ssf-scf

```

    callSegmentID    [3] CallSegmentID    OPTIONAL
  }

```

-- Direction: SSF -> SCF, Timer: Trsf
 -- This operation is used to indicate that the SSP is unable to select a route (e.g. unable to determine a correct route, no more routes on route list) or indication from the terminating half BCSM that a call cannot be presented to the terminating party (e.g. network congestion) (DP - Route_Select_Failure).

```

RouteSelectFailureArg ::= SEQUENCE {
    dpSpecificCommonParameters    [0] DpSpecificCommonParameters ,
    dialledDigits                  [1] CalledPartyNumber    OPTIONAL,
    callingPartyBusinessGroupID    [2] CallingPartyBusinessGroupID  OPTIONAL,
    callingPartySubaddress         [3] CallingPartySubaddress  OPTIONAL,
    callingFacilityGroup           [4] FacilityGroup    OPTIONAL,
    callingFacilityGroupMember     [5] FacilityGroupMember  OPTIONAL,
    failureCause                   [6] Cause    OPTIONAL,
    originalCalledPartyID          [7] OriginalCalledPartyID  OPTIONAL,
    prefix                         [8] Digits    OPTIONAL,
    redirectingPartyID             [9] RedirectingPartyID  OPTIONAL,
    redirectionInformation         [10] RedirectionInformation  OPTIONAL,
    routeList                     [11] RouteList   OPTIONAL,
    travellingClassMark            [12] TravellingClassMark  OPTIONAL,
    extensions                     [13] Extensions  OPTIONAL,
    carrier                       [14] Carrier    OPTIONAL
}

```

-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point processing
 -- rules to specify when these parameters are included in the message.

-- Direction: SCF -> SSF, Timer: Tsf
 -- This operation is used to request the SSF to perform the terminating basic call processing
 -- actions to select the terminating line if it is idle, or selects an idle line from a multi-line hunt

-- group, or selects an idle trunk from a trunk group, as appropriate. If no idle line or trunk is available, the SSF determines that the terminating facility is busy.

```

SelectFacilityArg ::= SEQUENCE {
    alertingPattern                [0] AlertingPattern  OPTIONAL,
    destinationNumberRoutingAddress [1] CalledPartyNumber  OPTIONAL,
    iSDNAccessRelatedInformation   [2] ISDNAccessRelatedInformation  OPTIONAL,
    calledFacilityGroup            [3] FacilityGroup    OPTIONAL,
    calledFacilityGroupMember      [4] FacilityGroupMember  OPTIONAL,
    originalCalledPartyID          [5] OriginalCalledPartyID  OPTIONAL,
    extensions                     [6] Extensions    OPTIONAL,
    displayInformation             [7] DisplayInformation  OPTIONAL,
    serviceInteractionIndicators    [8] ServiceInteractionIndicators  OPTIONAL,
    iNServiceCompatibilityResponse [9] INServiceCompatibilityResponse  OPTIONAL,
    forwardGVNS                   [10] ForwardGVNS    OPTIONAL,
    backwardGVNS                  [11] BackwardGVNS    OPTIONAL,
    serviceInteractionIndicatorsTwo [12] ServiceInteractionIndicatorsTwo  OPTIONAL,
    correlationID                  [13] CorrelationID   OPTIONAL,
    scfID                          [14] ScfID          OPTIONAL,
    callSegmentID                  [15] CallSegmentID   OPTIONAL,
    legToBeCreated                 [16] LegID          OPTIONAL
}

```

OP

OP

-- OPTIONAL parameters are only provided if modifications desired to basic call processing values.

-- Direction: SCF -> SSF, Timer: Tsr
 -- This operation is used to request the SSF to perform the originating basic call processing actions to determine routing information and select a route for a call, based either on call information available to the SSF, or on call information provided by the SCF (e.g. for alternate ro

Annex A: ssf-scf

uting), to include the

-- called party address, type of call, carrier, route index, and one or more alternate route indices.

-- Based on the routing information, the SSF attempts to select a primary route for the call, and if the

-- route is busy, attempts to select an alternate route. The SSF may fail to select a route for the call

-- if all routes are busy.

```

SelectRouteArg ::= SEQUENCE {
    destinationRoutingAddress [0] DestinationRoutingAddress ,
    alertingPattern           [1] AlertingPattern      OPTIONAL,
    correlationID              [2] CorrelationID       OPTIONAL,
    iSDNAccessRelatedInformation [3] ISDNAccessRelatedInformation  OPTIONAL,
    originalCalledPartyID     [4] OriginalCalledPartyID OPTIONAL,
    routeList                  [5] RouteList           OPTIONAL,
    scfID                       [6] ScfID               OPTIONAL,
    travellingClassMark        [7] TravellingClassMark  OPTIONAL,
    extensions                  [8] Extensions          OPTIONAL,
    carrier                     [9] Carrier             OPTIONAL,
    serviceInteractionIndicators [10] ServiceInteractionIndicators OPTIONAL,
    iNServiceCompatibilityResponse [11] INServiceCompatibilityResponse OPTIONAL,
    forwardGVNS                 [12] ForwardGVNS        OPTIONAL,
    backwardGVNS                 [13] BackwardGVNS       OPTIONAL,
    serviceInteractionIndicatorsTwo [14] ServiceInteractionIndicatorsTwo OPTIONAL,
    callSegmentID               [15] CallSegmentID      OPTIONAL,
    legToBeCreated              [16] LegID              OPTIONAL
}

```

-- OPTIONAL parameters are only provided if modifications desired to basic call processing values.

-- Direction: SCF -> SSF, Timer: Tsci

-- This operation is used to instruct the SSF on the charging information to send by the SSF.

-- The charging information can either be sent back by means of signalling or internal

-- if the SSF is located in the local exchange. In the local exchange

-- this information may be used to update the charge meter or to create a standard call record.

```

SendChargingInformationArg ::= SEQUENCE {
    sCIBillingChargingCharacteristics [0] SCIBillingChargingCharacteristics ,
    partyToCharge                     [1] LegID,
    extensions                         [2] Extensions      OPTIONAL,
    nocharge                          [3] BOOLEAN          OPTIONAL
}

```

-- SCF->SSF, Timer: Tsfi

-- This operation is issued by the SCF during a BCSM being suspended at a DP to request the CCF/SSF

-- sending a FACILITY IE to a user with a specified DSS1 message.

```

SendFacilityInformationArg ::= SEQUENCE{
    componentType [0] ComponentType,
    legID         [1] LegID      OPTIONAL,
    componentCorrelationID [2] ComponentCorrelationID  OPTIONAL,
    component      [3] Component,
    callProcessingOperationCorrelationID [4] CallProcessingOperationCorrelationID,
    -- DEFAULT facility,
    extensions     [5] Extensions  OPTIONAL
}

```

-- FACILITY information will be delivered with the specified DSS1 message. The message is specified with the

-- callProcessingOperationCorrelationID

-- Direction: SCF -> SSF. Timer: Tss

-- This operation is issued by the SCF in the context of the USI feature. It is used to request the SSF

-- to send a Service to User Information (USI information) data element to the indicated user.

```

SendSTUIArg ::= SEQUENCE {
    uSIServiceIndicator [0] USIServiceIndicator ,

```

Annex A: ssf-scf

```

legID      [1] LegID   DEFAULT sendingSideID:leg1,
uSIInformation [2] USIInformation ,
extensions  [3] Extensions   OPTIONAL
}

```

-- Direction: SSF -> SCF, Timer: Tsfr

-- This operation is used to send back to the SCF the values of counters specified in a previous

-- ActivateServiceFiltering operation

```

ServiceFilteringResponseArg ::= SEQUENCE {
  countersValue      [0] CountersValue,
  filteringCriteria   [1] FilteringCriteria ,
  extensions          [2] Extensions   OPTIONAL,
  responseCondition   [3] ResponseCondition   OPTIONAL
}

```

-- Direction SCF-SSF, Timer Tsep

-- This operation is used to request the SSF to activate/de-activate a list of trigger for one of the parties of the call.

```

SetServiceProfileArg ::= SEQUENCE {
  iNprofiles  [0] SEQUENCE SIZE (1..numOfINProfile)
               OF INprofile ,
  extensions  [1] Extensions   OPTIONAL
}

```

-- Direction: SCF -> SSF. Timer: Tsl

-- This operation is issued by the SCF to separate one joined leg from a multi-way connection

-- or a single 2 party Call segment.

```

SplitLegArg ::= SEQUENCE {
  legToBeSplit  [0] LegID,
  newCallSegment [1] INTEGER (2..numOfCSs),
  extensions     [2] Extensions   OPTIONAL
}

```

-- Direction: SSF -> SCF, Timer: Tsrp

-- This operation is used as a response to RequestFirstStatusMatchReport or

-- RequestEveryStatusChangeReport operations.

```

StatusReportArg ::= SEQUENCE {
  resourceStatus [0] ResourceStatus   OPTIONAL,
  correlationID  [1] CorrelationID   OPTIONAL,
  resourceID     [2] ResourceID   OPTIONAL,
  extensions     [3] Extensions   OPTIONAL,
  reportCondition [4] ReportCondition   OPTIONAL
}

```

-- For correlationID, OPTIONAL denotes network operator optional.

-- resourceID is required when the SSF sends a report as an answer to a previous request when the

-- correlationID was present.

-- Direction: SSF -> SCF, Timer: Tta

-- This operation is used to indicate that the call is accepted and answered by terminating party

-- (e.g. terminating party goes offhook, Q.931 Connect message received, ISDN-UP Answer message

-- received) (DP - T_Answer). .

```

TAnswerArg ::= SEQUENCE {
  dpSpecificCommonParameters [0] DpSpecificCommonParameters ,
  calledPartyBusinessGroupID  [1] CalledPartyBusinessGroupID   OPTIONAL,
  calledPartySubaddress       [2] CalledPartySubaddress   OPTIONAL,
  calledFacilityGroup         [3] FacilityGroup   OPTIONAL,
  calledFacilityGroupMember   [4] FacilityGroupMember   OPTIONAL,
  extensions                  [5] Extensions   OPTIONAL,
  componentType               [6] ComponentType   OPTIONAL,
  component                   [7] Component   OPTIONAL,
  componentCorrelationID      [8] ComponentCorrelationID   OPTIONAL
}

```

-- Direction: SSF -> SCF, Timer: Ttb

-- This operation is used to indicate all resources in group busy (DP- TBusy).

Annex A: ssf-scf

```

TBusyArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters ,
    busyCause [1] Cause OPTIONAL,
    calledPartyBusinessGroupID [2] CalledPartyBusinessGroupID OPTIONAL,
    calledPartySubaddress [3] CalledPartySubaddress OPTIONAL,
    originalCalledPartyID [4] OriginalCalledPartyID OPTIONAL,
    redirectingPartyID [5] RedirectingPartyID OPTIONAL,
    redirectionInformation [6] RedirectionInformation OPTIONAL,
    routeList [7] RouteList OPTIONAL,
    travellingClassMark [8] TravellingClassMark OPTIONAL,
    extensions [9] Extensions OPTIONAL
}
-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point processing rules
-- to specify when these parameters are included in the message.

-- Direction: SSF -> SCF, Timer: Ttd
-- This operation is used for a disconnect indication (e.g. onhook, Q.931 Disconnect message,
-- SS7 Release message) is received from the terminating party, or received from the originating party
-- via the originating half BCSM. (DP - T_Disconnect).

TDisconnectArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters ,
    calledPartyBusinessGroupID [1] CalledPartyBusinessGroupID OPTIONAL,
    calledPartySubaddress [2] CalledPartySubaddress OPTIONAL,
    calledFacilityGroup [3] FacilityGroup OPTIONAL,
    calledFacilityGroupMember [4] FacilityGroupMember OPTIONAL,
    releaseCause [5] Cause OPTIONAL,
    extensions [6] Extensions OPTIONAL,
    connectTime [7] Integer4 OPTIONAL,
    componentType [8] ComponentType OPTIONAL,
    component [9] Component OPTIONAL,
    componentCorrelationID [10] ComponentCorrelationID OPTIONAL
}

-- Direction: SSF -> SCF, Timer: Ttaa
-- This operation is used for indication of incoming call received from originating half BCSM and authority
-- to route call to a specified terminating resource (or group) verified. (DP - Termination_Authorized).

TermAttemptAuthorizedArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters ,
    calledPartyBusinessGroupID [1] CalledPartyBusinessGroupID OPTIONAL,
    calledPartySubaddress [2] CalledPartySubaddress OPTIONAL,
    callingPartyBusinessGroupID [3] CallingPartyBusinessGroupID OPTIONAL,
    originalCalledPartyID [4] OriginalCalledPartyID OPTIONAL,
    redirectingPartyID [5] RedirectingPartyID OPTIONAL,
    redirectionInformation [6] RedirectionInformation OPTIONAL,
    routeList [7] RouteList OPTIONAL,
    travellingClassMark [8] TravellingClassMark OPTIONAL,
    extensions [9] Extensions OPTIONAL,
    callingPartySubaddress [10] CallingPartySubaddress OPTIONAL
}

-- Direction: SSF -> SCF, Timer: Ttra
-- This operation is used for indication of a call termination attempt from the terminating half BCSM. (DP - Termination_Attempt).

TerminationAttemptArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters ,
    calledPartyBusinessGroupID [1] CalledPartyBusinessGroupID OPTIONAL,
    calledPartySubaddress [2] CalledPartySubaddress OPTIONAL,
    callingPartyBusinessGroupID [3] CallingPartyBusinessGroupID OPTIONAL,
    callingPartySubaddress [4] CallingPartySubaddress OPTIONAL,
    originalCalledPartyID [5] OriginalCalledPartyID OPTIONAL,
    redirectingPartyID [6] RedirectingPartyID OPTIONAL,
    redirectionInformation [7] RedirectionInformation OPTIONAL,
    routeList [8] RouteList OPTIONAL,
    travellingClassMark [9] TravellingClassMark OPTIONAL,
    extensions [10] Extensions OPTIONAL
}
-- For the OPTIONAL parameters, refer to clause 2 for the trigger detection point

```

Annex A: ssf-scf

t processing rules

-- to specify when these parameters are included in the message.

-- Direction: SSF -> SCF, Timer: Ttmc

-- This operation is used to indicate that a feature request is received from the terminating party (e.g. hook

-- flash, ISDN feature activation Q.931 HOLD or RETrieve message). (DP - T_Mid_Call).

-- Direction: SSF -> SCF, Timer: Ttna

-- This operation is used to indicate that the terminating party does not answer within a specified duration.

-- (DP - T_No_Answer). .

```
TNoAnswerArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters ,
    calledPartyBusinessGroupID [1] CalledPartyBusinessGroupID OPTIONAL,
    calledPartySubaddress [2] CalledPartySubaddress OPTIONAL,
    calledFacilityGroup [3] FacilityGroup OPTIONAL,
    calledFacilityGroupMember [4] FacilityGroupMember OPTIONAL,
    originalCalledPartyID [5] OriginalCalledPartyID OPTIONAL,
    redirectingPartyID [6] RedirectingPartyID OPTIONAL,
    redirectionInformation [7] RedirectionInformation OPTIONAL,
    travellingClassMark [8] TravellingClassMark OPTIONAL,
    extensions [9] Extensions OPTIONAL,
    componentType [10] ComponentType OPTIONAL,
    component [11] Component OPTIONAL,
    componentCorrelationID [12] ComponentCorrelationID OPTIONAL
}
```

-- Direction: SSF -> SCF, Timer: T ts

-- This operation is issued by the SSF after detecting a valid trigger condition at the T_Suspend DP or to

-- report a tSuspended event requested by the RequestReportBCSMEvent.

```
TSuspendedArg ::= SEQUENCE {
    dpSpecificCommonParameters [0] DpSpecificCommonParameters ,
    legID [1] LegID OPTIONAL,
    extensions [2] Extensions OPTIONAL
}
```

-- For the OPTIONAL parameters, refer to clause 2.1..27 for the trigger detection point processing rules

-- to specify when these parameters are included in the message.

-- Use for T/EDP-R is ffs.

-- Direction: SCF -> SRF, Timer: Tpa

-- This operation is to be used after Establish Temporary Connection (assist procedure with a second SSP)

-- or a Connect to Resource (no assist) operation. It may be used for inband interaction with an analog user,

-- or for interaction with an ISDN user. In the former case, the SRF is usually collocated with the SSF for

-- standard tones (congestion tone...) or standard announcements. In the latter case, the SRF is always

-- collocated with the SSF in the switch. Any error is returned to the SCF. The timer associated with this

-- operation must be of a sufficient duration to allow its linked operation to be correctly correlated.

```
PlayAnnouncementArg ::= SEQUENCE {
    informationToSend [0] InformationToSend ,
    disconnectFromIPForbidden [1] BOOLEAN DEFAULT TRUE,
    requestAnnouncementComplete [2] BOOLEAN DEFAULT TRUE,
    extensions [3] Extensions OPTIONAL,
    connectedParty CHOICE {
        legID [4] LegID,
        callSegmentID [5] CallSegmentID
    } OPTIONAL
}
```

-- Direction: SCF -> SRF, Timer: Tpc

Annex A: ssf-scf

-- This operation is used to interact with a user to collect information.

```
PromptAndCollectUserInformationArg ::= SEQUENCE {
    collectedInfo    [0] CollectedInfo,
    disconnectFromIPForbidden [1] BOOLEAN    DEFAULT TRUE,
    informationToSend [2] InformationToSend  OPTIONAL,
    extensions       [3] Extensions         OPTIONAL,
    callSegmentID    [4] CallSegmentID      OPTIONAL
}
```

```
ReceivedInformationArg ::= CHOICE {
    digitsResponse [0] Digits,
    iA5Response    [1] IA5String,
    modemdetected  [2] BOOLEAN
}
```

-- Direction: SCF -> SRF, Timer: Tprm
 -- Used to prompt a user to store a message

```
PromptAndReceiveMessageArg ::= SEQUENCE {
    disconnectFromIPForbidden [0] BOOLEAN    DEFAULT TRUE,
    informationToSend         [1] InformationToSend  OPTIONAL,
    extensions                [3] Extensions         OPTIONAL,
    subscriberID              [4] GenericNumber      OPTIONAL,
    mailBoxID                  [5] MailBoxID          OPTIONAL,
    informationToRecord        [6] InformationToRecord,
    media                      [7] Media,             -- DEFAULT voiceMail,
    callSegmentID              [8] CallSegmentID      OPTIONAL
}
```

```
MessageReceivedArg ::= SEQUENCE {
    receivedStatus [0] ReceivedStatus,
    recordedMessageID [1] RecordedMessageID  OPTIONAL,
    recordedMessageUnits [2] INTEGER(1..maxRecordedMessageUnits)  OPTIONAL,
    extensions        [3] Extensions         OPTIONAL
}
```

-- reportUTSI OPERATION
 -- This operation is described in part 2 of the recommendation Q.1238.

-- requestReportUTSI OPERATION
 -- This operation is described in part 2 of the recommendation Q.1238.

-- sendSTUI OPERATION
 -- This operation is described in part 2 of the recommendation Q.1238.

-- Direction:SCF-> SRF. Timer :TCI
 -- This operation is issued by the SCF to deallocate the resources used to perform the
 -- instance of the "User Interaction" script : the context is released.

```
ScriptCloseArg ::= SEQUENCE {
    ulScriptId Code,
    ulScriptSpecificInfo [0] OCTET STRING  OPTIONAL,
    extensions           [1] Extensions    OPTIONAL,
    callSegmentID        [2] CallSegmentID OPTIONAL
}
```

-- Direction:SRF-> SCF. Timer :TRe
 -- This operation is issued by the SRF to return information to the SCF on the results of the
 -- execution of the instance of User Interaction script.

```
ScriptEventArg ::= SEQUENCE {
    ulScriptId Code,
    ulScriptResult [0] OCTET STRING  OPTIONAL,
    extensions      [1] Extensions    OPTIONAL,
    callSegmentID   [2] CallSegmentID OPTIONAL,
    lastEventIndicator [3] BOOLEAN    DEFAULT FALSE
}
```

-- Direction:SCF-> SRF. Timer :Tinf

Annex A: ssf-scf

```
ScriptInformationArg ::= SEQUENCE {
    ulScriptId      Code,
    ulScriptSpecificInfo  [0] OCTET STRING  OPTIONAL,
    extensions      [1] Extensions  OPTIONAL,
    callSegmentID   [2] CallSegmentID  OPTIONAL
}
```

```
-- Direction:SCF-> SRF. Timer :Tru
-- This operation is issued by the SCF to allocate the necessary resources to perform the
-- instance of the "User Interaction" script and then to activate this "User Interaction" script
-- instance. A context is partially defined for it if necessary.
```

```
ScriptRunArg ::= SEQUENCE {
    ulScriptId      Code,
    ulScriptSpecificInfo  [0] OCTET STRING OPTIONAL,
    extensions      [1] Extensions  OPTIONAL,
    disconnectFromIPForbidden  [2] BOOLEAN  DEFAULT TRUE,
    callSegmentID   [3] CallSegmentID  OPTIONAL
}
```

```
-- Direction: SRF -> SCF, Timer: Tsrr
-- This operation is used as the response to a PlayAnnouncement operation when the announcement completed
-- report indication is set.
```

```
SpecializedResourceReportArg ::= NULL
```

```
-- Direction: SRF -> SCF, Timer: Tcg
-- This operation is used to request the SCF to reduce the rate at which specific service requests are sent to the SRF.
```

```
SRFCallGapArg ::= SEQUENCE {
    sRFGapCriteria  [0] SRFGapCriteria ,
    gapIndicators   [1] GapIndicators,
    controlType     [2] ControlType  OPTIONAL,
    extensions      [3] Extensions  OPTIONAL
}
```

END

ASN.1 Text INCS3BundleArg

```

INCS3BundleArg
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
IMPORTS
    -- 29.09.1998 changed to contain SRF
    -- arguments. For that purpose INCS2SSFSCFopsargs
    -- and INCS2SCFSRFopsargs have been merged to
    -- INCS2opsargs
CancelFailed,
RequestedInfoError,
SystemFailure,
TaskRefused,
ActivateServiceFilteringArg,
AnalysedInformationArg,
AnalyseInformationArg,
ApplyChargingArg ,
ApplyChargingReportArg ,
AssistRequestInstructionsArg ,
AuthorizeTerminationArg,
CallGapArg ,
CallInformationReportArg ,
CallInformationRequestArg ,
CancelArg ,
CancelStatusReportRequestArg,
CollectedInformationArg,
CollectInformationArg ,
ConnectArg ,
ConnectToResourceArg,
ContinueWithArgumentArg ,
CreateCallSegmentAssociationArg ,
CreateCallSegmentAssociationResultArg ,
CreateOrRemoveTriggerDataArg,
CreateOrRemoveTriggerDataResultArg,
DisconnectForwardConnectionWithArgumentArg ,
DisconnectLegArg ,
EntityReleasedArg ,
EstablishTemporaryConnectionArg ,
EventNotificationChargingArg ,
EventReportBCSMArg ,
EventReportFacilityArg,
FacilitySelectedAndAvailableArg,
FurnishChargingInformationArg ,
HoldCallInNetworkArg,
InitialDPArg ,
InitiateCallAttemptArg ,
ManageTriggerDataArg ,
ManageTriggerDataResultArg ,
MergeCallSegmentsArg ,
MoveCallSegmentsArg ,
MoveLegArg ,
OAbandonArg,
OAnswerArg,
OCalledPartyBusyArg,
ODisconnectArg,
MidCallArg,
ONoAnswerArg,
OriginationAttemptArg,
OriginationAttemptAuthorizedArg,
OSuspendedArg,
ReconnectArg,
ReleaseCallArg ,
ReportUTSIArg ,
RequestCurrentStatusReportArg,
RequestEveryStatusChangeReportArg,
RequestFirstStatusMatchReportArg,
RequestNotificationChargingEventArg,
RequestReportBCSMEventArg ,
RequestReportFacilityEventArg,
RequestReportUTSIArg ,
ResetTimerArg ,
RouteSelectFailureArg,
SelectFacilityArg,

```

Annex A: incs3b

SelectRouteArg,
 SendChargingInformationArg ,
 SendFacilityInformationArg,
 SendSTUIArg ,
 ServiceFilteringResponseArg ,
 SetServiceProfileArg,
 SplitLegArg ,
 StatusReportArg,
 TAnswerArg,
 TBusyArg,
 TDisconnectArg,
 TermAttemptAuthorizedArg,
 TerminationAttemptArg,
 TNoAnswerArg,
 TSuspendedArg,

-- from now on SRF arguments

MessageReceivedArg ,
 PlayAnnouncementArg ,
 PromptAndCollectUserInformationArg ,
 PromptAndReceiveMessageArg ,
 ReceivedInformationArg ,
 ScriptCloseArg ,
 ScriptEventArg ,
 ScriptInformationArg ,
 ScriptRunArg ,
 SpecializedResourceReportArg

FROM INCS3opsargs

;

Empty ::= NULL

ArgType ::= CHOICE

{	aSFArg	[0] ActivateServiceFilteringArg,
	aSFRArg	[1] Empty,
	aTArg	[2] Empty,
	aTRArg	[3] Empty,
	aCArg	[4] ApplyChargingArg ,
	aCRArg	[5] ApplyChargingReportArg ,
	aRIArg	[6] AssistRequestInstructionsArg ,
	cGArg	[7] CallGapArg ,
	cIRArg	[8] CallInformationReportArg ,
	cIRQArg	[9] CallInformationRequestArg ,
	cANArg	[10] CancelArg ,
	clArg	[11] CollectInformationArg ,
	cONArg	[12] ConnectArg ,
	cTRArg	[13] ConnectToResourceArg,
	cWAAArg	[14] ContinueWithArgumentArg ,
	cSAArg	[15] CreateCallSegmentAssociationArg ,
	cSARArg	[16] CreateCallSegmentAssociationResultArg ,
	cUEArg	[17] Empty,
	dFCArg	[99] Empty,
	dFCWAAArg	[18] DisconnectForwardConnectionWithArgumentArg ,
	dLArg	[19] DisconnectLegArg ,
	dLRArg	[20] Empty,
	eRArg	[21] EntityReleasedArg ,
	eTCArg	[22] EstablishTemporaryConnectionArg ,
	eNCArg	[23] EventNotificationChargingArg ,
	eRBArg	[24] EventReportBCSMArg ,
	fCIArg	[25] FurnishChargingInformationArg ,
	iDPArg	[26] InitialDPArg ,
	iCAArg	[27] InitiateCallAttemptArg ,
	mTDArg	[28] ManageTriggerDataArg ,
	mTDRArg	[29] ManageTriggerDataResultArg,
	mCArg	[30] MergeCallSegmentsArg,
	mCRArg	[31] Empty,
	mCSArg	[32] MoveCallSegmentsArg,
	mCSRArg	[33] Empty,
	mLArg	[34] MoveLegArg,
	mLRArg	[35] Empty,
	rCArg	[36] ReleaseCallArg ,
	rUArg	[37] ReportUTSIArg ,

Annex A: incs3b

```

rNCArg      [38] RequestNotificationChargingEventArg ,
rRBAArg     [39] RequestReportBCSMEEventArg ,
rRUArg      [40] RequestReportUTSIArg ,
rTArg       [41] ResetTimerArg ,
sCIAArg     [42] SendChargingInformationArg ,
sSArg       [43] SendSTUIArg ,
sFRArg      [44] ServiceFilteringResponseArg ,
sLArg       [45] SplitLegArg ,
sLRArg      [46] Empty ,
-- SRF
pAArg       [47] PlayAnnouncementArg ,
mRArg       [48] MessageReceivedArg ,
pACArg      [49] PromptAndCollectUserInfoArg ,
pARArg      [50] PromptAndReceiveMessageArg ,
rIArg       [51] ReceivedInformationArg ,
sCArg       [52] ScriptCloseArg ,
sEArg       [53] ScriptEventArg ,
sIArg       [54] ScriptInformationArg ,
sCRAArg     [55] ScriptRunArg ,
sRRArg      [56] SpecializedResourceReportArg,
-- ITU CS-3
aDIAArg     [98] AnalysedInformationArg,
aIArg       [97] AnalyseInformationArg,
aUTTAArg    [96] AuthorizeTerminationArg,
cDI         [95] CollectedInformationArg,
cTDAArg     [94] CreateOrRemoveTriggerDataArg,
cTDRArg     [93] CreateOrRemoveTriggerDataResultArg,
eRFAArg     [92] EventReportFacilityArg,
fSAArg      [91] FacilitySelectedAndAvailableArg,
oABNAArg    [90] OAbandonArg,
oANSAArg    [89] OAnswerArg,
oCPBAArg    [88] OCalledPartyBusyArg,
oDISArg     [87] ODisconnectArg,
mIDCLAArg   [86] MidCallArg,
oNOANSAArg  [85] ONoAnswerArg,
oAArg       [84] OriginationAttemptArg,
oAAArg      [83] OriginationAttemptAuthorizedArg,
oSUSArg     [82] OSuspendedArg,
rECONArg    [81] ReconnectArg,
rCSRAArg    [80] RequestCurrentStatusReportArg,
rESCAArg    [79] RequestEveryStatusChangeReportArg,
rFSMAArg    [78] RequestFirstStatusMatchReportArg,
rFEAArg     [77] RequestReportFacilityEventArg,
rSFAArg     [76] RouteSelectFailureArg,
sFAArg      [75] SelectFacilityArg,
sRAArg      [74] SelectRouteArg,
sENDFArg    [73] SendFacilityInformationArg,
sSPAArg     [72] SetServiceProfileArg,
sREPAArg    [71] StatusReportArg,
tANSAArg    [70] TAnswerArg,
tBUAArg     [69] TBusyArg,
tDISArg     [68] TDisconnectArg,
tAAArg      [67] TermAttemptAuthorizedArg,
tAArg       [66] TerminationAttemptArg,
tNOANSAArg  [65] TNoAnswerArg,
tSUSArg     [64] TSuspendedArg,
hCINAArg    [63] HoldCallInNetworkArg
}

```

ErrorType ::= CHOICE

```

{
cancelPar      [0] Empty,
cancelFailedPar [1] CancelFailed,
chainingRefusedPar [2] Empty,
eTCFailedPar   [3] Empty,
improperCallerResponsePar [4] Empty,
missingCustomerRecordPar [5] Empty,
missingParameterPar [6] Empty,
parameterOutOfRangePar [7] Empty,
requestedInfoError [8] RequestedInfoError,
systemFailurePar [9] SystemFailure,
taskRefusedPar [10] TaskRefused,
unavailableResourcePar [11] Empty,
unexpectedComponentSequencePar [12] Empty,
unexpectedDataValuePar [13] Empty,
unexpectedParameterPar [14] Empty,
}

```

Annex A: incs3b

```
    unknownLegIDPar      [15] Empty,  
    unknownRecordedMessageID [16] Empty,  
    unknownResource      [17] Empty,  
    unknownSubscriber    [18] Empty  
  }  
END
```

ASN.1 Text INCS3datatypes

INCS3datatypes -- {ccitt recommendation q 1238 modules(1) in-cs3-ssf-scf-datatypes (6) version1(0)}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- IMPORTS

-- common-classes,
-- common-datatypes,
-- ssf-scf-classes,
-- scf-srf-classes,
-- scf-srf-datatypes,
-- tc-Messages

--FROM IN-CS3-object-identifiers {ccitt recommendation q 1238 modules(1) in-cs3-object-identifiers (0) version1(0)}

-- COMMON-BOUNDS

--FROM IN-CS3-Common-Classes common-classes

-- TRIGGER,

-- SCF-SSF-BOUNDS

--FROM IN-CS3-SSF-SCF-Classes ssf-scf-classes

-- SCF-SRF-BOUNDS

--FROM IN-CS3-SCF-SRF-Classes scf-srf-classes

-- Extensions {},

-- Integer4

--FROM IN-CS3-common-datatypes common-datatypes

-- InformationToSend {}

--FROM IN-CS3-scf-srf-datatypes scf-srf-datatypes

--;

-- *****

-- DEFINITIONS OF TYPES WHICH ARE IMPORTED OR THE DEFINITION IS MISSING (Informative information).

--NumberMatch ::= OCTET STRING

--ChargingTariffInformation ::= OCTET STRING

--ChargingPriceInformation ::= OCTET STRING

--AddOnChargingInformation ::= OCTET STRING

-- FacilityGroup ::= OCTET STRING

TriggerID ::= INTEGER

TriggerData ::= SEQUENCE OF TriggerID

Extensions ::= SEQUENCE SIZE (1..numOfExtensions) OF ExtensionField

ExtensionField ::= OCTET STRING

ExportLegsType ::= SEQUENCE SIZE (1..numOfLegs) OF SEQUENCE {

sourceLeg [0] DTLegID,

newLeg [1] DTLegID

}

--ExtensionField ::= SEQUENCE {

-- type EXTENSION.&id ({SupportedExtensions}),

-- -- shall identify the value of an EXTENSION type

-- criticality CriticalityType DEFAULT ignore,

-- value [1] EXTENSION.&ExtensionType

-- ({OurSupportedExtensions}{@type})

-- }

Integer4 ::= INTEGER(0..2147483647)

InvokeId ::= INTEGER (-128..127)

-- From Remote-Operations-Information-Objects

Code ::= CHOICE

{

local INTEGER,

global OBJECT IDENTIFIER

}

--Types which are defined in Q.1238.3

Annex A: incs3-dt

```
CollectedDigits ::= SEQUENCE {
    minimumNbOfDigits [0] INTEGER (1..127) DEFAULT 1,
    maximumNbOfDigits [1] INTEGER (1..127),
    endOfReplyDigit [2] OCTET STRING (SIZE (1..2)) OPTIONAL,
    cancelDigit [3] OCTET STRING (SIZE (1..2)) OPTIONAL,
    startDigit [4] OCTET STRING (SIZE (1..2)) OPTIONAL,
    firstDigitTimeOut [5] INTEGER (1..127) OPTIONAL,
    interDigitTimeOut [6] INTEGER (1..127) OPTIONAL,
    errorTreatment [7] ErrorTreatment, -- DEFAULT reportErrorToScf,
    interruptableAnnInd [8] BOOLEAN DEFAULT TRUE,
    voiceInformation [9] BOOLEAN DEFAULT FALSE,
    voiceBack [10] BOOLEAN DEFAULT FALSE,
    detectModem [11] BOOLEAN DEFAULT FALSE
}
-- The use of voiceBack is network operator specific.
-- The endOfReplyDigit, cancelDigit, and startDigit parameters have been designa
ted as OCTET STRING,
-- and are to be encoded as BCD, one digit per octet only, contained
-- in the four least significant bits of each OCTET. The usage is service depend
ent.
```

```
CollectedInfo ::= CHOICE {
    collectedDigits [0] CollectedDigits,
    iA5Information [1] BOOLEAN,
    detectModem [2] BOOLEAN
}
```

```
ElementaryMessageID ::= Integer4
```

```
ErrorTreatment ::= ENUMERATED {
    reportErrorToScf(0),
    help(1),
    repeatPrompt(2)
}
-- reportErrorToScf means returning the "ImproperCallerResponse" error in the ev
ent of an error
-- condition during collection of user info.
```

```
GapOnResource ::= Code
```

```
InbandInfo ::= SEQUENCE {
    messageID [0] MessageID,
    numberOfRepetitions [1] INTEGER (1..127) OPTIONAL,
    duration [2] INTEGER (0..32767) OPTIONAL,
    interval [3] INTEGER (0.. 32767) OPTIONAL,
    preferredLanguage [4] Language OPTIONAL
}
-- Interval is the time in seconds between each repeated announcement. Duration
is the total
-- amount of time in seconds, including repetitions and intervals.
-- The end of announcement is either the end of duration or numberOfRepetitions,
whatever comes first.
-- duration with value 0 indicates infinite duration
```

```
InformationToRecord ::= SEQUENCE {
    messageID [0] ElementaryMessageID OPTIONAL,
    messageDeletionTimeOut [1] INTEGER (1..3600) OPTIONAL,
    --Time units = hours
    timeToRecord [3] INTEGER (0..maxRecordingTime) OPTIONAL,
    --Time units = seconds
    controlDigits [4] SEQUENCE {
        endOfRecordingDigit [0] OCTET STRING (SIZE(1..2)) OPTIONAL,
        cancelDigit [1] OCTET STRING (SIZE(1..2)) OPTIONAL,
        replayDigit [2] OCTET STRING (SIZE(1..2)) OPTIONAL,
        restartRecordingDigit [3] OCTET STRING (SIZE(1..2)) OPTIONAL,
        restartAllowed [4] BOOLEAN DEFAULT FALSE,
        replayAllowed [5] BOOLEAN DEFAULT FALSE
    }
}
```

```
InformationToSend ::= CHOICE {
    inbandInfo [0] InbandInfo,
    tone [1] Tone,
```


Annex A: incs3-dt

```

displayInformation [2] DisplayInformation ,
sDSSinformation [3] SDSSinformation
}

```

Language ::= PrintableString (SIZE (3)) -- ISO 639 codes only;

MailBoxID ::= OCTET STRING (SIZE(minMailBoxIDLength..maxMailBoxIDLength))

```

Media ::= ENUMERATED {
    voiceMail (0),
    faxGroup3 (1),
    faxGroup4 (2)
}

```

```

MessageID ::= CHOICE {
    elementaryMessageID [0] Integer4,
    text [1] SEQUENCE {
        messageContent [0] IA5String (SIZE (minMessageContentLength..maxMessageContentLength)),
        attributes [1] OCTET STRING (SIZE (minAttributesLength..maxAttributesLength))
    },
    elementaryMessageIDs [29] SEQUENCE SIZE (1.. numOfMessageIDs) OF Integer4,
    variableMessage [30] SEQUENCE {
        elementaryMessageID[0] Integer4,
        variableParts [1] SEQUENCE SIZE (1..5) OF VariablePart
    }
}
-- OPTIONAL denotes network operator specific use.

```

```

ReceivedStatus ::=ENUMERATED {
    messageComplete (0),
    messageInterrupted (1),
    messageTimeOut (2)
}

```

RecordedMessageID ::= Integer4

```

SRFGapCriteria ::= CHOICE {
    iAddressValue [1] Digits ,
    gapOnResource [2] GapOnResource,
    iAddressAndresource [3] SEQUENCE {
        iAddressValue [1] Digits ,
        gapOnResource [2] GapOnResource
    }
}

```

```

Tone ::= SEQUENCE {
    toneID [0] Integer4,
    duration [1] Integer4 OPTIONAL
}

```

-- The duration specifies the length of the tone in seconds, value 0 indicates infinite duration.

```

VariablePart ::= CHOICE {
    integer [0] Integer4,
    number [1] Digits , -- Generic digits
    time [2] OCTET STRING (SIZE(2)), -- HH:MM, BCD coded
    date [3] OCTET STRING (SIZE(3)), -- YYMMDD, BCD coded
    price [4] OCTET STRING (SIZE(4))
}

```

-- Indicates the variable part of the message.

-- BCD coded variable parts are encoded as described in the examples below.

-- For example, time = 12:15 would be encoded as:

```

-- Bits          HGFE      DCBA
-- leading octet  2      1
--               5      1

```

-- date = 1993 September 30th would be encoded as:

```

-- Bits          HGFE      DCBA
-- leading octet  3      9
--               9      0
--               0      3

```

Q.1238.2 : December 1999

Annex A: incs3-dt

-- For a system operating when or after this Recommendation is released, the 2 digit value representing a Year shall be interpreted as follows :

-- -- If the two-digits value is 00 through 49 inclusive, it shall be interpreted as representing year 2000 through 2049.

-- -- If the two-digits value is 50 through 99 inclusive, it shall be interpreted as representing year 1950 through 1999.

-- End of added definitions.

-- *****

-- The following three definitions are local short-hand notation for convenience

.

-- B1 ::= COMMON-BOUNDS defined in Q.1238.1 (Part 1 of Recommendation Q.1238)

-- B2 ::= SCF-SSF-BOUNDS defined in this Recommendation (Q.1238.2)

-- B3 ::= SCF-SRF-BOUNDS defined in Q.1238.3 (Part 3 of Recommendation Q.1238)

AccessCode ::= LocationNumber

-- An access code from a business group dialling plan attendant access codes, access codes to escape to the public network, access code to access a private facility/network, and feature access codes.

-- Uses the LocationNumber format which is based on the Q.763 Location Number format.

-- The Nature of Address indicator field shall be set to "Spare" (value 00000000).

-- The Numbering Plan Indicator field shall be set to "Spare" (value 000).

-- Of local significance.

ACHBillingChargingCharacteristics ::= OCTET STRING (SIZE

(minACHBillingChargingLength..maxACHBillingChargingLength))

-- The ACHBillingChargingCharacteristics parameter specifies the charging related information

-- to be provided by the SSF and the conditions on which this information has to be reported

-- back to the SCF with the ApplyChargingReport operation.

-- Different set of criteria may be provided in case more than one report is expected.

-- Its content is network operator specific.

-- Examples of charging related information to be provided by the SSF may be: bulk counter

-- values, costs, tariff change and time of charge, time stamps, durations, etc.

-- Examples of conditions on which the charging related information are to be reported may be:

-- threshold value reached, timer expiration, tariff change, end of connection configuration, etc

ActionIndicator ::= ENUMERATED {

activate (1),

deactivate (2),

retrieve (3)

}

-- indicates the action to be performed by the ManageTriggerData operation (activate, deactivate

-- or retrieve the status of a TDP.

ActionOnProfile ::= ENUMERATED {

activate (0),

deactivate (1)

}

ActionPerformed ::= ENUMERATED {

activated (1),

deactivated (2),

alreadyActive (3),

alreadyInactive (4),

Annex A: incs3-dt

```

    isActive      (5),
    isInactive    (6),
    tDPunknown    (7)
}
-- indicates the result of the operation ManageTriggerData
-- activated: response of activate TDP
-- deactivated: response of deactivate TDP
-- alreadyActive: response of activate TDP
-- alreadyInactive: response of deactivate TDP
-- isActive: response of retrieve status of TDP
-- isInactive: response of retrieve status of TDP

```

```

ActivableServices ::= BIT STRING {
    callingLineIdentificationPresentation (1),
    callingLineIdentificationRestriction (2),
    connectedLineIdentificationPresentation (3),
    connectedLineIdentificationRestriction (4),
    callForwardingOnNoReply (5),
    callForwardingUnconditional (6),
    callForwardingOnBusy (7),
    callForwardingOnNotReachable (8),
    reverseCharging (9),
    adviceOfChargeOnStart (10),
    adviceOfChargeAtEnd (11),
    adviceOfChargeDuringCall (12),
    timeDependentRouting (13),
    callingPartingDependentRouting (14),
    outgoingCallBarring (15),
    incomingCallBarring (16)
}

```

AdditionalCallingPartyNumber ::= Digits
 -- Indicates the Additional Calling Party Number. Refer to Q.763 for encoding.

AlertingPattern ::= OCTET STRING (SIZE(3))
 -- Indicates a specific pattern that is used to alert a subscriber (e.g. distinctive ringing, tones, etc.).
 -- Only the trailing OCTET is used, the remaining OCTETS should be sent as NULL (zero)
 -- The receiving side ignores the leading two OCTETS.
 -- Only applies if SSF is the terminating local exchange for the subscriber.
 -- Refer to the Q.931 Signal parameter for encoding.

ApplicationTimer ::= INTEGER (0..2047)
 -- Used by the SCF to set a timer in the SSF. The timer is in seconds.

AssistingSSIPRoutingAddress ::= Digits
 -- Indicates the destination address of the SRF for the assist procedure.

ATMTrafficDescriptor ::= OCTET STRING (SIZE(
 minATMTrafficDescriptorLength..maxATMTrafficDescriptorLength))
 -- Indicates the ATM Traffic Descriptors. Refer to Q.2961 for encoding.

BackwardGVNS ::= OCTET STRING (SIZE(
 minBackwardGVNSLength..maxBackwardGVNSLength))
 -- Indicates the GVNS Backward information. Refer to Q.735, §6 for encoding.

```

BackwardServiceInteractionInd ::= SEQUENCE {
    conferenceTreatmentIndicator [1] OCTET STRING (SIZE(1)) OPTIONAL,
    -- acceptConferenceRequest 'xxxx xx01'B
    -- rejectConferenceRequest 'xxxx xx10'B
    -- network default is accept conference request,

    callCompletionTreatmentIndicator [2] OCTET STRING (SIZE(1)) OPTIONAL
    -- acceptCallCompletionServiceRequest 'xxxx xx01'B,
    -- rejectCallCompletionServiceRequest 'xxxx xx10'B
    -- network default is accept call completion service request
}

```

```

BCSMEvent ::= SEQUENCE {
    eventTypeBCSM [0] EventTypeBCSM,
    monitorMode [1] MonitorMode,
    legID [2] DTLegID OPTIONAL,

```

Annex A: incs3-dt

```
dpSpecificCriteria [30] DpSpecificCriteria OPTIONAL
}
```

-- Indicates the BCSM Event information for monitoring.

```
BearerCapability ::= CHOICE {
  bearerCap [0] OCTET STRING
    (SIZE(2..maxBearerCapabilityLength)),
  tmr [1] OCTET STRING (SIZE(1)),
  broadbandBearerCap [2] OCTET STRING
    (SIZE (minBroadbandBearerCapabilityLength..
      maxBroadbandBearerCapabilityLength))
}
```

-- Indicates the type of bearer capability connection to the user. For narrowband bearerCapability, either

-- DSS 1 (Q.931) or the ISUP User Service Information (Q.763) encoding can be used. Refer

-- to the Q.763 Transmission Medium Requirement parameter for tmr encoding.

-- For broadband ISDN: Indicates the Broadband Bearer Capability. Refer to Q.2961 for encoding.

```
BothwayThroughConnectionInd ::= ENUMERATED {
  bothwayPathRequired (0),
  bothwayPathNotRequired (1)
}
```

```
CalledDirectoryNumber ::= OCTET STRING (SIZE
  (minCalledDirectoryNumberLength..
  maxCalledDirectoryNumberLength))
```

-- Indicates the Called Directory Number. Refer to Q.769.1 'Called Directory Number' for encoding.

```
CalledPartyBusinessGroupID ::= OCTET STRING
```

-- Indicates the business group of the called party. The value of this octet string is network

-- operator specific.

```
CalledPartyNumber ::= OCTET STRING (SIZE
  (minCalledPartyNumberLength..
  maxCalledPartyNumberLength))
```

-- Indicates the Called Party Number. Both N-ISDN and B-ISDN E.164 addresses are supported. Refer to Q.2763 for encoding.

```
CalledPartySubaddress ::= OCTET STRING
  (SIZE ( minCalledPartySubaddressLength..
  maxCalledPartySubaddressLength))
```

-- Indicates the Called Party Subaddress. Both N-ISDN and B-ISDN E.164 addresses are supported. Refer to Q.2763 for encoding.

```
CallingGeodeticLocation ::= OCTET STRING ( SIZE (minCallingGeodeticLocationLength..maxCallingGeodeticLocationLength))
```

-- The coding of this parameter is based on the appropriate mapping with the ISUP parameter Calling Geodetic Location.

-- Refer to Q.763 for encoding.

-- This parameter indicates the geographic coordinate of a calling party. The excessive amount of data possible

-- within this parameter may require segmentation of the INAP operation to be sent to the SCF.

-- The amount of data possible to be conveyed within this parameter from

-- the SSF to the SCF could be limited, for example it may be considered to only support a relevant subset of all the

-- shape descriptions. This is to be considered in the next Capability Set.

```
CallingPartyBusinessGroupID ::= OCTET STRING
```

-- Indicates the business group of the calling party. The value of this octet string is network

-- operator specific.

```
CallingPartyNumber ::= OCTET STRING (SIZE (
  minCallingPartyNumberLength..
```

Annex A: incs3-dt

maxCallingPartyNumberLength))

-- Indicates the Calling Party Number. Both N-ISDN and B-ISDN E.164 addresses are supported. Refer to Q.2763 for encoding.

CallingPartySubaddress ::= OCTET STRING
(SIZE (minCallingPartySubaddressLength..
maxCallingPartySubaddressLength))

-- Indicates the Calling Party Subaddress. Both N-ISDN and B-ISDN E.164 addresses are supported. Refer to Q.2763 for encoding.

CallingPartysCategory ::= OCTET STRING (SIZE(1))
-- Indicates the type of calling party (e.g. operator, payphone, ordinary subscriber). Refer to Q.763
-- for encoding.

CallProcessingOperationCorrelationID ::=ENUMERATED {
aLERTing (1),
sETUP (5),
cONNNect (7),
dISConnect (69),
rELease (77),
rELeaseCOMplete(90),
fACility (98)
}

CallReference ::= OCTET STRING (SIZE(1..maxCallReferenceLength))
-- The coding of this parameter is based on the appropriate mapping with the ISUP parameter.
-- Refer to Q.763 for encoding
-- The Call Reference value is unique within one network. When transit through a private
-- network the uniqueness of the call reference parameter is not maintained

--[editor note: new call reference parameter NOT included in latest version of Q.763 ???]

CallResult ::= OCTET STRING (SIZE (minCallResultLength..
maxCallResultLength))

-- This parameter provides the SCF with the charging related information previously requested
-- using the ApplyCharging operation. This shall include the partyToCharge parameter as
-- received in the related ApplyCharging operation to correlate the result to the request
-- The remaining content is network operator specific.

-- Examples of charging related information to be provided by the SSF may be: bulk counter values,
-- costs, tariff change and time of change, time stamps, durations, etc.
-- Examples of conditions on which the charging related information are to be reported may be:
-- threshold value reached, timer expiration, tariff change, end of connection configuration, etc.

DTCallSegmentID ::= INTEGER (1..numOfCSs)

Carrier ::= OCTET STRING (SIZE (minCarrierLength..
maxCarrierLength))
-- Contains the carrier selection field followed by either Carrier ID information (option 1), or the Transit
--Network selection information (option 2), depending on the network.
-- In both cases, the Carrier selection is one octet and is encoded as:
-- 00000000 No indication
-- 00000001 Selected carrier code pre subscribed and not input by calling party
-- 00000010 Selected carrier identification code pre subscribed and input by calling party
-- 00000011 Selected carrier identification code pre subscribed, no indication of whether input by calling party
-- 00000100 Selected carrier identification code not pre subscribed and input by calling party
-- 00000101

Q.1238.2 : December 1999

Annex A: incs3-dt

```
--      to      Spare
-- 11111110
-- 11111111      Reserved

--
-- For the first option, Carrier ID has a one octet field indicating the number
of digits followed by the digits encoded
-- using BCD. Detailed coding is outside the scope of this capability set. It is
of local significance and carrying
-- it through the ISUP is outside the scope of this capability set
-- For the second option, refer to Q.763 for the TNS encoding.
-- Note that this ASN.1 encoding of this parameter includes 2 possible encodings,
referred to as option 1 and option 2.
-- The encoding that should be used is dependent on the network. It is a hard-coded
decision based
-- on the region in which the switch is located.
```

Cause ::= OCTET STRING (SIZE (minCauseLength..
maxCauseLength))
-- Indicates the cause for interface related information. Refer to the Q.763 Cause
parameter for encoding
-- For the use of cause and location values refer to Q.850.

CCSS::=BOOLEAN
-- Used by the SSF to indicate CCSS (Call Completion on Service Set-up) if set to
"True" to the SCF, i.e. that the current call is due a special procedure (CCBS or CCNR).

CGEncountered ::= ENUMERATED {
noCGEncountered (0),
manualCGEncountered (1),
sCPOverload (2)
}
-- Indicates the type of automatic call gapping encountered, if any.

ChargeNumber ::= LocationNumber
-- Information sent in either direction indicating the chargeable number for the
call and consisting
-- of the odd/even indicator, nature of address indicator, numbering plan indicator,
and address signals.
-- Uses the LocationNumber format which is based on the Q.763 Location Number format
-- For example, the ChargeNumber may be a third party number to which a call is
billed for the 3rd party billing
-- service. In this case, the calling party may request operator assistance to charge
the call to,
-- for example, their home number.

ChargingEvent ::= SEQUENCE {
eventTypeCharging [0] EventTypeCharging ,
monitorMode [1] MonitorMode,
legID [2] DTLegID OPTIONAL
}
-- This parameter indicates the charging event type and corresponding monitor mode
and LegID

Component ::= CHOICE {
componentInfo [0] OCTET STRING (SIZE(1..118)),
-- Contains the operation value (object identifier), error value, etc. within the
UNI APDU, in addition also contain
-- the parameter set/sequence for the operation invocation/return result or return
error/reject on UNI. See Q.932
-- for encoding

relayedComponent [1] OCTET STRING
}
-- If componentInfo is chosen, then it is necessary to use this parameter in
sequence with ComponentType and
-- ComponentCorrelationID
-- If relayedComponent is chosen, then ComponentType and ComponentCorrelationID
may not be used in the
-- sequence

ComponentCorrelationID ::= INTEGER

Annex A: incs3-dt

```

ComponentType ::= ENUMERATED {
    anny          (0),
    invoke        (1),
    rResult       (2),
    rError        (3),
    rReject       (4)
}

```

```

ConnectedNumberTreatmentInd ::= ENUMERATED {
    noINImpact          (0),
    presentationRestricted (1),
    presentCalledINNumber (2),
    presentCalledINNumberRestricted (3)
}

```

```

ControlType ::= ENUMERATED {
    sCPOverloaded (0),
    manuallyInitiated (1),
    destinationOverload (2)
-- other values are outside the scope of this capability set.
}

```

CorrelationID ::= Digits
 -- used by SCF for correlation with a previous operation. Refer to clause 4 for a description of the procedures
 -- associated with this parameter.

```

CounterAndValue ::= SEQUENCE {
    counterID [0] CounterID,
    counterValue [1] Integer4
}

```

CounterID ::= INTEGER (0..9)
 -- Indicates the counters to be incremented The counterIDs can be addressed by using the last digits of the dialed number.

CountersValue ::= SEQUENCE SIZE(0..numOfCounters) OF CounterAndValue

```

CreateOrRemoveIndicator ::= ENUMERATED {
    create (0),
    remove (1)
}

```

CSAID ::= INTEGER (1..numOfCSAs)
 -- Indicates the SSF CSA identifier

CutAndPaste ::= INTEGER (0..2)
 -- Indicates the number of leading digits to be deleted (cut) and to paste remaining dialed digits.

DateAndTime ::= OCTET STRING (SIZE(6))
 -- Indicates, amongst others, the start time for activate service filtering. Coded as YYMMDDHHMMSS

-- with each digit coded BCD
 -- The first octet contains YY and the remaining items are sequenced following
 -- For example, 1993 September 30th, 12:15:01 would be encoded as:

-- Bits HGFE DCBA

-- leading octet 3 9

-- 9 0

-- 0 3

-- 2 1

-- 5 1

-- 1 0

-- For a system operating when or after this Recommendation is released, the 2 digit value

-- representing a Year shall be interpreted as follows

-- If the two-digits value is 00 through 49 inclusive, it shall be interpreted as representing

-- year 2000 through 2004.

-- If the two-digits value is 50 through 99 inclusive, it shall be interpreted as

Annex A: incs3-dt

s representing

-- year 1950 through 1999.

DefaultFaultHandling ::= SEQUENCE

```

{  action      [0]      ENUMERATED {
    resumeCallProcessing (0),
    releaseCall      (1)
  },
  treatment [1] GapTreatment OPTIONAL
}

```

DestinationRoutingAddress ::=

SEQUENCE SIZE(1..numOfAddresses) OF CalledPartyNumber
 -- Indicates the list of Called Party Numbers (primary and alternates).

Digits ::=

OCTET STRING (SIZE (minDigitsLength..maxDigitsLength))
 -- Indicates the address signalling digits. Refer to the Q.763 Generic Number and Generic Digits parameter
 -- for encoding. The coding of the subfield's 'NumberQualifier' in Generic Number and 'TypeOfDigits' in
 -- Generic Digits are irrelevant to the INAP, the ASN.1 tags are sufficient to identify the parameter.
 -- The ISUP format does not allow to exclude these subfields, therefore the value is network operator specific.
 -- The following parameters should use Generic Number
 -- Additional Calling Number, CorrelationID for AssistRequestInstructions, AssistingSSPIPRoutingAddress
 -- for EstablishTemporaryConnection
 -- calledAddressValue for all occurrences, callingAddressValue for all occurrences
 -- The following parameters should use Generic Digits: prefix, all
 -- other CorrelationID occurrences, dialledNumber filtering criteria, callingLineID filtering criteria, lineID for
 -- ResourceID type, digitResponse for ReceivedInformationArg, iNServiceControlLow / iNServiceControlHigh for
 -- MidCallInfoType,, iNServiceControlCode for MidCallInfo.

DisplayInformation ::= IA5String (SIZE (minDisplayInformationLength.. maxDisplayInformationLength))

-- Indicates the display information
 -- Delivery of DisplayInformation parameter to Private Networks cannot be guaranteed due to signalling
 -- interworking problems, solutions are outside the scope of this capability set

DpSpecificCommonParameters ::= SEQUENCE {

```

  serviceAddressInformation [0] ServiceAddressInformation,
  bearerCapability          [1] BearerCapability          OPTIONAL,
  calledPartyNumber         [2] CalledPartyNumber         OPTIONAL,
  callingPartyNumber        [3] CallingPartyNumber        OPTIONAL,
  callingPartysCategory     [4] CallingPartysCategory     OPTIONAL,
  iPSSPCapabilities         [5] IPSSPCapabilities         OPTIONAL,
  iPAvailable               [6] IPAvailable               OPTIONAL,

  iSDNAccessRelatedInformation [7] ISDNAccessRelatedInformation OPTIONAL,
  cGEncountered             [8] CGEncountered             OPTIONAL,
  locationNumber            [9] LocationNumber            OPTIONAL,
  serviceProfileIdentifier   [10] ServiceProfileIdentifier OPTIONAL,
  terminalType              [11] TerminalType              OPTIONAL,
  extensions                [12] Extensions              OPTIONAL,
  chargeNumber              [13] ChargeNumber              OPTIONAL,
  servingAreaID             [14] ServingAreaID            OPTIONAL,
  serviceInteractionIndicators [15] ServiceInteractionIndicators OPTIONAL,
  iNServiceCompatibilityIndication [16] INServiceCompatibilityIndication OPTIONAL,

  serviceInteractionIndicatorsTwo [17] ServiceInteractionIndicatorsTwo OPTIONAL,
  uSIServiceIndicator          [18] USIServiceIndicator          OPTIONAL,
  uSIInformation               [19] USIInformation           OPTIONAL,
  forwardGVNS                  [20] ForwardGVNS              OPTIONAL,
  createdCallSegmentAssociation [21] CSAID                   OPTIONAL
}

```


Annex A: incs3-dt

-- OPTIONAL for iPSSPCapabilities, iPAvailable, and cGEncountered denotes network operator specific use
 -- OPTIONAL for callingPartyNumber, and callingPartysCategory
 -- . bearerCapability should be appropriately coded as speech.

```
DpSpecificCriteria ::= CHOICE {
    numberOfDigits      [0] NumberOfDigits,
    applicationTimer    [1] ApplicationTimer,
    midCallControlInfo  [2] MidCallControlInfo ,
    numberOfDigitsTwo   [3] SEQUENCE {
        requestedNumberOfDigits [0] NumberOfDigits,
        minNumberOfDigits      [1] NumberOfDigits OPTIONAL
    }
}
```

-- The SCF may specify the number of digits to be collected by the SSF for the CollectedInfo event
 -- When all digits are collected, the SSF reports the event to the SCF
 -- The SCF may set a timer in the SSF for the No Answer event. If the user does not answer the call
 -- within the allotted time, the SSF reports the event to the SCF
 -- The SCF may specify the number of digits to be collected by the SSF for the CollectInfo event and hereby specify a minimum number of digits to be collected in case
 -- the exact number of digits is unknown to the SCF, but a report is desired in case of complete number
 -- is determined before the requested number of digits has been collected.

XDuration ::= INTEGER (-2..86400)
 -- Values are seconds

```
Entry ::= CHOICE {
    agreements      [0] OBJECT IDENTIFIER,
    networkSpecific [1] Integer4
}
```

```
EventSpecificInformationBCSM ::= CHOICE {
    collectedInfoSpecificInfo [0] SEQUENCE {
        calledPartyNumber [0] CalledPartyNumber
    },
    analysedInfoSpecificInfo [1] SEQUENCE {
        calledPartyNumber [0] CalledPartyNumber
    },
    routeSelectFailureSpecificInfo [2] SEQUENCE {
        failureCause [0] Cause OPTIONAL
    },
    oCalledPartyBusySpecificInfo [3] SEQUENCE {
        busyCause [0] Cause OPTIONAL
    },
    oNoAnswerSpecificInfo [4] SEQUENCE {
        -- no specific info defined
    },
    oAnswerSpecificInfo [5] SEQUENCE {
        backwardGVNS [0] BackwardGVNS OPTIONAL
    },
    oMidCallSpecificInfo [6] SEQUENCE {
        connectTime [0] Integer4 OPTIONAL,
        oMidCallInfo [1] MidCallInfo OPTIONAL
    },
    oDisconnectSpecificInfo [7] SEQUENCE {
        releaseCause [0] Cause OPTIONAL,
        connectTime [1] Integer4 OPTIONAL
    },
    tBusySpecificInfo [8] SEQUENCE {
        busyCause [0] Cause OPTIONAL
    }
}
```

Annex A: incs3-dt

```

    },
    tNoAnswerSpecificInfo      [9] SEQUENCE {
        -- no specific info defined
    },
    tAnswerSpecificInfo        [10] SEQUENCE {
        -- no specific info defined
    },
    tMidCallSpecificInfo        [11] SEQUENCE {
        connectTime             [0] Integer4 OPTIONAL,
        tMidCallInfo            [1] MidCallInfo OPTIONAL
    },
    tDisconnectSpecificInfo     [12] SEQUENCE {
        releaseCause            [0] Cause OPTIONAL,
        connectTime             [1] Integer4 OPTIONAL
    },
    oTermSeizedSpecificInfo     [13] SEQUENCE {
        -- no specific info defined
    },
    oSuspend                    [14] SEQUENCE {
        -- no specific info defined
    },
    tSuspend                    [15] SEQUENCE {
        -- no specific info defined
    },
    origAttemptAuthorized       [16] SEQUENCE {
        -- no specific info defined
    },
    oReAnswer                   [17] SEQUENCE {
        -- no specific info defined
    },
    tReAnswer                   [18] SEQUENCE {
        -- no specific info defined
    },
    facilitySelectedAndAvailable [19] SEQUENCE {
        -- no specific info defined
    },
    callAccepted                [20] SEQUENCE {
        -- no specific info defined
    },
    oAbandon                    [21] SEQUENCE {
        abandonCause            [0] Cause OPTIONAL
    },
    tAbandon                    [22] SEQUENCE {
        abandonCause            [0] Cause OPTIONAL
    },
    authorizeRouteFailure        [23] SEQUENCE {
        authoriseRouteFailureCause [0] Cause OPTIONAL
    },
    terminationAttemptAuthorized [24] SEQUENCE {
        -- no specific info defined
    },
    orinationDenied             [25] SEQUENCE {
        originationDeniedCause    [0] Cause OPTIONAL
    },
    terminationDenied           [26] SEQUENCE {
        terminationDeniedCause    [0] Cause OPTIONAL
    }

```

Annex A: incs3-dt

```

    }

}

-- Indicates the call related information specific to the event.
-- The unit for the connectTime is 100 milliseconds

EventSpecificInformationCharging ::= OCTET STRING (SIZE
    (minEventSpecificInformationChargingLength..
     maxEventSpecificInformationChargingLength))
-- defined by network operator.
-- Its content is network signalling/operator specific.

-- Indicates the charging related information specific to the event.
-- An example data EmbeddedType definition for this parameter is given below:
--     chargePulses      [0] Integer4,
--     chargeMessages     [1] OCTET STRING (SIZE (min..max))

EventTypeBCSM ::= ENUMERATED {
    origAttemptAuthorized(1),
    collectedInfo(2),
    analysedInformation(3),
    routeSelectFailure(4),
    oCalledPartyBusy(5),
    oNoAnswer(6),
    oAnswer(7),
    oMidCall(8),
    oDisconnect(9),
    oAbandon(10),
    termAttemptAuthorized(12),
    tBusy(13),
    tNoAnswer(14),
    tAnswer(15),
    tMidCall(16),
    tDisconnect(17),
    tAbandon(18),
    oTermSeized(19),
    oSuspend(20),
    tSuspend(21),
    origAttempt(22),
    termAttempt(23),
    oReAnswer(24),
    tReAnswer(25),
    facilitySelectedAndAvailable(26),
    callAccepted(27),
    authorizeRouteFailure(28),
    originationDenied(29),
    terminationDenied(30)
}

-- Indicates the name of the BCSM detection point event.
-- Notice: Values origAttempt and termAttempt can only be used for TDPs

EventTypeCharging ::= OCTET STRING (SIZE
    (minEventTypeChargingLength..
     maxEventTypeChargingLength))
-- This parameter indicates the charging event type.
-- Its content is network signalling / operator specific.

--
-- An example data type definition for this parameter is given below:
-- EventTypeCharging EmbeddedType ::= ENUMERATED {
--     chargePulses (0),
--     chargeMessages (1)
-- }

FacilityGroup ::= CHOICE {
    trunkGroupID [0] INTEGER,
    privateFacilityID [1] INTEGER,
    huntGroup [2] OCTET STRING,
    routeIndex [3] OCTET STRING
}
-- Indicates the particular group of facilities to route the call. huntGroup and
routeIndex are encoded as

```

Annex A: incs3-dt

-- network operator specific.

FacilityGroupMember ::= INTEGER

-- Indicates the specific member of a trunk group or multi-line hunt group.

FCIBillingChargingCharacteristics ::= OCTET STRING (SIZE
(minFCIBillingChargingLength..
maxFCIBillingChargingLength))

-- This parameter indicates the billing and/or charging characteristics.

-- Its content is network operator specific

-- An example datatype definition for this parameter is given below:

```
FCIBillingChargingCharacteristicsEmbeddedType ::= CHOICE {
  completeChargingrecord [0] OCTET STRING (SIZE (min..max)),
  correlationID [1] CorrelationID,
  scenario2Dot3 [2] SEQUENCE {
    chargeParty [0] LegID OPTIONAL,
    chargeLevel [1] OCTET STRING (SIZE (min..max)) OPTIONAL,
    chargeItems [2] SET OF Attribute OPTIONAL
  }
}
```

-- Depending on the applied charging scenario the following information elements can be included

-- (refer to Q.1214 Appendix II):

-- complete charging record (scenario 2.2)

-- charge party (scenario 2.3)

-- charge level (scenario 2.3)

-- charge items (scenario 2.3)

-- correlationID (scenario 2.4)

FeatureCode ::= LocationNumber

-- The two-digit feature code preceded by "*" or "11".

-- Uses the LocationNumber format which is based on the Q.763 Location Number format.

-- The Nature of Address indicator field shall be set to "Spare" (value 00000000).

-- The Numbering Plan Indicator field shall be set to "Spare" (value 000)

-- Used for stimulus signalling (Q.932).

```
FeatureRequestIndicator ::= ENUMERATED {
  hold(0),
  retrieve(1),
  featureActivation(2),
  spare1(3),
  sparen(127)
}
```

-- Indicates the feature activated (e.g. a switch-hook flash, feature activation

). Spare values reserved

-- for future use.

```
FilteredCallTreatment ::= SEQUENCE {
  sFBillingChargingCharacteristics [0] SFBillingChargingCharacteristics,
  informationToSend [1] InformationToSend OPTIONAL,
  maximumNumberOfCounters [2] MaximumNumberOfCounters OPTIONAL,
  releaseCause [3] Cause OPTIONAL
}
```

-- If releaseCause is not present, the default value is the same as the ISUP cause value decimal 31.

-- If informationToSend is present, the call will be released after the end of the announcement

-- with the indicated or default releaseCause.

-- If maximumNumberOfCounters is not present, ServiceFilteringResponse will be sent with

-- CountersValue ::= SEQUENCE SIZE (0) OF CountersAndValue

```
FilteringCharacteristics ::= CHOICE {
  interval [0] INTEGER (-1..32000),
  numberOfCalls [1] Integer4
}
```

-- Indicates the severity of the filtering and the point in time when the ServiceFilteringResponse is to be sent.

Annex A: incs3-dt

-- If = interval, every interval of time the next call leads to an InitialDP and a ServiceFilteringResponse is sent to the SCF.
 -- An interval of 0 indicates that all calls matching the filtering criteria will result in sending of an "InitialDP" or a DP-specific operation and no filtering will be applied (i.e. no "ServiceFilteringResponse" will be sent).
 -- An interval of -1 indicates that none of the calls matching the filtering criteria will either result in sending of an "InitialDP" or a DP-specific operation or a "ServiceFilteringResponse" operation.
 -- Other values of Interval indicate duration in seconds.

-- If = NumberOfCalls, every N calls the Nth call leads to an InitialDP and a ServiceFilteringResponse is sent to the SCF.
 -- If ActivateServiceFiltering implies several counters – filtering on several dialled numbers –, the numberOfCalls would include calls to all the dialled numbers.

```
FilteringCriteria ::= CHOICE {
    dialledNumber    [0] Digits ,
    callingLineID    [1] Digits ,
    serviceKey       [2] ServiceKey,
    addressAndService [30] SEQUENCE {
        calledAddressValue [0] Digits ,
        serviceKey          [1] ServiceKey,
        callingAddressValue [2] Digits  OPTIONAL,
        locationNumber      [3] LocationNumber OPTIONAL
    }
}
```

-- In case calledAddressValue is specified, the numbers to be filtered are from calledAddressValue up to and including calledAddressValue + maximumNumberOfCounters-1.
 -- The last two digits of calledAddressvalue can not exceed 100-maximumNumberOfCounters.

```
FilteringTimeOut ::= CHOICE {
    duration    [0] XDuration,
    stopTime    [1] DateAndTime
}
```

-- Indicates the maximum duration of the filtering. When the timer expires, a ServiceFilteringResponse is sent to the SCF.

ForwardCallIndicators ::= OCTET STRING (SIZE(2))
 -- Indicates the Forward Call Indicators. Refer to Q.763 for encoding

ForwardGVNS ::= OCTET STRING (SIZE(minForwardGVNSLength..maxForwardGVNSLength))
 -- Indicates the GVNS Forward information. Refer to Q.735, §6 for encoding.

```
ForwardingCondition ::= ENUMERATED {
    busy      (0),
    noanswer  (1),
    anny      (2)
}
```

-- Indicates the condition that must be met to complete the connect.

```
ForwardServiceInteractionInd ::= SEQUENCE {
    conferenceTreatmentIndicator [1] OCTET STRING (SIZE(1))  OPTIONAL,
    -- acceptConferenceRequest    'xxxx xx01'B
    -- rejectConferenceRequest    'xxxx xx10'B
    -- network default is accept conference request
    callDiversionTreatmentIndicator [2] OCTET STRING (SIZE(1))  OPTIONAL,
    -- callDiversionAllowed       'xxxx xx01'B
    -- callDiversionNotAllowed    'xxxx xx10'B
    -- network default is Call Diversion allowed
    callOfferingTreatmentIndicator [3] OCTET STRING (SIZE(1))  OPTIONAL ,
    -- indicates if call offering is "allowed", "not allowed" or "no impact by IN"
    -- the value 'no impact by IN,' has only local significance in SSF as a request to SSF
    -- not to modify the value of the call offering treatment indicator conveyed in
```

Annex A: incs3-dt

signaling.

```
-- callOfferingNotAllowed      'xxxx xx01'B,
-- callOfferingAllowed        'xxxx xx10'B
-- callOfferingNoINImpact     'xxxx x100'B
-- indicates if call offering is "allowed"," not allowed" or "no impact by IN".

-- network default is Call Offering not allowed
callWaitingTreatmentIndicator [5] OCTET STRING (SIZE(1))  OPTIONAL
-- callWaitingAllowed         'xxxx xx01'B,
-- callWaitingNotAllowed      'xxxx xx10'B
-- network default is Call Waiting allowed

-- Notice Tag value 4 is otherwise reserved ,
```

```
}

--The forwardServiceInteractionInd parameter is applicable to IDP, CON, CWA and
ICA operations.
--This parameter can be received either in the O_BCSM or in the T_BCSM.
```

```
GapCriteria ::= CHOICE {
    calledAddressValue [0] Digits ,
    gapOnService       [2] GapOnService,
    gapAllInTraffic    [3] NULL,
    calledAddressAndService [29] SEQUENCE {
        calledAddressValue [0] Digits ,
        serviceKey         [1] ServiceKey
    },
    callingAddressAndService [30] SEQUENCE {
        callingAddressValue [0] Digits ,
        serviceKey          [1] ServiceKey,
        locationNumber      [2] LocationNumber
    }
    OPTIONAL
}
```

```
-- Both calledAddressValue and callingAddressValue can be
-- incomplete numbers, in the sense that a limited amount of digits can be given
.
--
-- For the handling of numbers starting with the same digit string refer to the
detailed procedure
-- of the CallGap operation in 3.3
```

```
GapOnService ::= SEQUENCE {
    serviceKey [0] ServiceKey,
    dpCriteria [1] EventTypeBCSM  OPTIONAL
}
```

```
GapIndicators ::= SEQUENCE {
    duration [0] XDuration,
    gapInterval [1] Interval
}
```

```
-- Indicates the gapping characteristics.
--No gapping when gapInterval equals 0, and
--gap all calls when gapInterval equals -1.
```

```
GapTreatment ::= CHOICE {
    informationToSend [0] InformationToSend ,
    releaseCause      [1] Cause ,
    both              [2] SEQUENCE {
        informationToSend [0] InformationToSend ,
        releaseCause      [1] Cause
    }
}
```

```
-- The default value for Cause is the same as in ISUP.
```

```
GenericName ::= OCTET STRING (SIZE(
```

Annex A: incs3-dt

minGenericNameLength..
maxGenericNameLength))

-- Refer to Q.931 Display Information for encoding.

GenericNumber ::= OCTET STRING (SIZE(
minGenericNumberLength..
maxGenericNumberLength))

-- Refer to Q.763 Generic Number for encoding.

GenericNumbers ::=
SET SIZE(1..numOfGenericNumbers) OF GenericNumber

HighLayerCompatibility ::= OCTET STRING (SIZE (highLayerCompatibilityLength))
-- Indicates the teleservice. For encoding, DSS1 (Q.931) is used.

HoldCause ::= OCTET STRING
-- defined by network operator.
-- Indicates the cause for holding the call.
-- Its content is network operator specific

initialCallSegment INTEGER ::= 1
-- the initial call segment represents the call segment that was there when the
CSA was created, ie. the CS where
-- the trigger took place or the CS that was created by an InitiateCallAttempt wi
thin a TC-BEGIN message.

INprofile ::=SEQUENCE {
actionOnProfile [0] ActionOnProfile,
tDPIIdentifier [1] TDPIIdentifier ,
dpName [2] EventTypeBCSM OPTIONAL,
extensions [3] Extensions OPTIONAL
}

INServiceCompatibilityIndication ::=
SEQUENCE SIZE (1..numOfInServiceCompatibilityIndLength) OF Entry

INServiceCompatibilityResponse ::= Entry

Interval ::= INTEGER (-1..60000)
-- Units are milliseconds. A -1 value denotes infinite.

IPAvailable ::= OCTET STRING (SIZE (
minIPAvailableLength..
maxIPAvailableLength))
-- defined by network operator.
-- Indicates that the resource is available.
-- Its content is network operator specific

IPRoutingAddress ::= CalledPartyNumber
-- Indicates the routing address for the IP.

IPSSPCapabilities ::= OCTET STRING (SIZE (
minIPSSPCapabilitiesLength..
maxIPSSPCapabilitiesLength))
-- defined by network operator.
-- Indicates the SRF resources available at the SSP.
-- Its content is network operator specific

ISDNAccessRelatedInformation ::= OCTET STRING (SIZE
(minISDNAccessRelatedInfoLength..
maxISDNAccessRelatedInfoLength))
-- Indicates the destination user network interface related information. Refer t
o the Q.763 Access
-- Transport parameter for encoding.

DTLegID ::= CHOICE {
sendingSideID [0] DTLegType,
receivingSideID [1] DTLegType
}

Annex A: incs3-dt

-- Indicates a reference to a specific party in a call. OPTIONAL denotes network operator specific use
 -- with a choice of unilateral ID assignment or bilateral ID assignment.
 -- OPTIONAL for LegID also denotes the following:
 -- when only one party exists in the call, this parameter is not needed (as no ambiguity exists);
 -- when more than one party exists in the call, one of the following alternatives applies:
 -- 1. LegID is present and indicates which party is concerned.
 -- 2. LegID is not present and a default value is assumed (e.g. calling party in the case of the ApplyCharging operation).
 -- Choice between these two alternatives is kept a network operator option.
 -- sendingSideID is used where legID is sent from the SCF to the SSF and
 -- receivingSideID is used where SCF receives legID from the SSF.

DTLegType ::= OCTET STRING (SIZE(1))

dtleg1 DTLegType ::= '01'H

dtleg2 DTLegType ::= '02'H

LocationNumber ::= OCTET STRING (SIZE (minLocationNumberLength..maxLocationNumberLength))

-- Indicates the Location Number for the calling party. Refer to Q.763 (White book) for encoding.

MaximumNumberOfCounters ::= INTEGER (1..numOfCounters)

MidCallControlInfo ::= SEQUENCE SIZE (minMidCallControlInfoNum .. maxMidCallControlInfoNum)
 OF SEQUENCE {
 midCallInfoType [0] MidCallInfoType ,
 midCallReportType [1] ENUMERATED {
 inMonitoringState(0),
 inAnyState(1)
 } DEFAULT inMonitoringState
 }

MidCallInfo ::= SEQUENCE {
 iNServiceControlCode [0] Digits
 }

MidCallInfoType ::= SEQUENCE {
 iNServiceControlCodeLow [0] Digits ,
 iNServiceControlCodeHigh [1] Digits OPTIONAL
 }

DTMiscCallInfo ::= SEQUENCE {
 messageType [0] ENUMERATED {
 request(0),
 notification(1)
 },
 dpAssignment [1] ENUMERATED {
 individualLine(0),
 groupBased(1),
 switchBased(2)
 } OPTIONAL
 }

-- Indicates detection point related information.

-- Note: "Switch based "is used with the same meaning as the previous used term "office based"

MonitorMode ::= ENUMERATED {
 interrupted(0),
 notifyAndContinue(1),
 transparent(2)
 }

-- Indicates the event is relayed and/or processed by the SSP.

Annex A: incs3-dt

NumberingPlan ::= OCTET STRING (SIZE(1))

-- Indicates the numbering plan for collecting the user information. Refer to the Q.763 Numbering Plan
 -- Indicator field for encoding.

NumberOfDigits ::= INTEGER (1..255)

-- Indicates the number of digits to be collected

OriginalCalledPartyID ::= OCTET STRING (SIZE
 (minOriginalCalledPartyIDLength..
 maxOriginalCalledPartyIDLength))

-- Indicates the original called number. Both N-ISDN and B-ISDN E.164 addresses are supported Refer to the Q.2763 Original Called Number for encoding.

ProfileIdentifier ::= CHOICE {
 access [0] CalledPartyNumber ,
 group [1] FacilityGroup
 }

-- Please note that 'CalledPartyNumber' is used to address a subscriber access line.

--The data type was reused from the existing types to avoid the definition of a new one.

QoSParameter ::= OCTET STRING (SIZE (
 minQoSParameterLength..
 maxQoSParameterLength))

-- Indicates the QoS. Refer to Q.2961 for encoding.

Reason ::= OCTET STRING (SIZE(
 minReasonLength..
 maxReasonLength))

-- Its content is network operator specific

RedirectingPartyID ::= OCTET STRING (SIZE (
 minRedirectingPartyIDLength..
 maxRedirectingPartyIDLength))

-- Indicates redirecting number. Refer to the Q.763 Redirecting number for encoding.

RedirectionInformation ::= OCTET STRING (SIZE(2))

-- Indicates redirection information. Refer to the Q.763 Redirection Information for encoding.

RedirectReason ::= OCTET STRING (SIZE(1))

-- Indicates redirection reason information. Refer to the "Invoking Pivot Reason" parameter of Q.763 for encoding.

RegistrarIdentifier ::= OCTET STRING

-- Its content is network operator specific

ReportCondition ::= ENUMERATED {
 statusReport (0),
 timerExpired (1),
 canceled (2)
 }

-- ReportCondition specifies the cause of sending "StatusReport" operation to the SCF

RequestedInformationList ::= SEQUENCE SIZE (1..numOfInfoItems) OF RequestedInformation

RequestedInformationTypeList ::= SEQUENCE SIZE (1..numOfInfoItems) OF RequestedInformationType

RequestedInformation ::= SEQUENCE {
 requestedInformationType [0] RequestedInformationType,

Annex A: incs3-dt

```

requestedInformationValue  [1] RequestedInformationValue
}

```

```

RequestedInformationType ::= ENUMERATED {
    callAttemptElapsedTime(0),
    callStopTime(1),
    callConnectedElapsedTime(2),
    calledAddress(3),
    releaseCause(30)
}

```

```

RequestedInformationValue ::= CHOICE {
    callAttemptElapsedTimeValue  [0] INTEGER (0..255),
    callStopTimeValue           [1] DateAndTime,
    callConnectedElapsedTimeValue [2] Integer4,
    calledAddressValue           [3] Digits ,
    releaseCauseValue            [30] Cause
}

```

-- The callAttemptElapsedTimeValue is specified in seconds.
 --The unit for the callConnectedElapsedTimeValue is 100 milliseconds

```

RequestedUTSI ::= SEQUENCE {
    uSIServiceIndicator  [0] USIServiceIndicator ,
    uSIMonitorMode       [1] USIMonitorMode
}

```

```

RequestedUTSIList ::= SEQUENCE SIZE
    (minRequestedUTSINum..
     maxRequestedUTSINum) OF RequestedUTSI

```

```

ResourceID ::= CHOICE {
    lineID           [0] Digits ,
    facilityGroupID  [1] FacilityGroup,
    facilityGroupMemberID [2] INTEGER,
    trunkGroupID     [3] INTEGER
}

```

-- Indicates a logical identifier for the physical termination resource.

```

ResourceStatus ::= ENUMERATED {
    busy(0),
    idle(1)
}

```

```

ResponseCondition ::= ENUMERATED {
    intermediateResponse(0),
    lastResponse(1)
}

```

-- additional values are outside the scope of this capability set

-- ResponseCondition is used to identify the reason why ServiceFilteringResponse operation is sent.

```

RouteList ::=
    SEQUENCE SIZE(1..3) OF OCTET STRING (SIZE
        (minRouteListLength..
         maxRouteListLength))

```

-- Indicates a list of trunk groups or a route index. .

-- Its content is network operator specific

```

RoutingNumber ::= OCTET STRING (SIZE
    (minRoutingNumberLength..
     maxRoutingNumberLength))

```

-- Indicates the Routing Number. Refer to Q.769.1 parameter Network Routing Number for encoding.

```

ScfID ::= OCTET STRING (SIZE
    (minScfIDLength..
     maxScfIDLength))

```

Q.1238.2 : December 1999

Annex A: incs3-dt

-- defined by network operator.
 -- Indicates the SCF identity.
 -- Used to derive the INAP address of the SCF to establish a connection between a requesting FE
 -- and the specified SCF.
 -- When ScfID is used in an operation which may cross an internetwork boundary, its encoding must
 -- be understood in both networks; this requires bilateral agreement on the encoding.
 -- Refer to 3.5/Q.713 "calling party address" parameter for encoding.
 -- It indicates the SCCP address e.g. Global Title of the SCF..
 -- Other encoding schemes are also possible as a operator specific option..

SCIBillingChargingCharacteristics ::= OCTET STRING (SIZE
 (minSCIBillingChargingLength..
 maxSCIBillingChargingLength))

-- This parameter indicates the billing and/or charging characteristics.
 -- Its content is network signalling / operator specific

-- An example datatype definition for this parameter is given below:

SCIBillingChargingCharacteristicsEmbeddedType ::= CHOICE {
 -- chargeLevel [0] OCTET STRING (SIZE (min..max),
 -- chargePulses [1] Integer4,
 -- chargeMessages [2] OCTET STRING (SIZE (min..max))
 -- }

-- Depending on the applied charging scenario the following information elements
 -- can be included (refer to Q.1214 Appendix II):

-- chargeLevel (scenario 3.2)
 -- chargePulses (scenario 3.2)
 -- chargeMessages (scenario 3.2)

SDSSinformation ::= OCTET STRING (SIZE
 (minSDSSinformationLength..
 maxSDSSinformationLength))

-- Its content is network operator specific

ServiceAddressInformation ::= SEQUENCE {
 serviceKey [0] ServiceKey OPTIONAL,
 miscCallInfo [1] DTMiscCallInfo,
 triggerType [2] TriggerType OPTIONAL
 }

-- Information that represents the result of trigger analysis and allows the SCF to choose the appropriate service logic

ServiceInteractionIndicators ::= OCTET STRING (SIZE (
 minServiceInteractionIndicatorsLength..
 maxServiceInteractionIndicatorsLength))

-- Indicators which are exchanged between SSP and SCP to resolve interactions between IN based services

-- and network based services, respectively between different IN based services.

-- Its content is network signalling/operator specific

-- Note this parameter is kept in this Recommendation for backward compatibility to CS-1R,
 -- for this Recommendation see parameter ServiceInteractionIndicatorsTwo

ServiceInteractionIndicatorsTwo ::= SEQUENCE {
 forwardServiceInteractionInd [0] ForwardServiceInteractionInd OPTIONAL,
 -- applicable to operations IDP, CON, ICA, CWA.

 backwardServiceInteractionInd [1] BackwardServiceInteractionInd OPTIONAL,
 -- applicable to operations IDP, CON, CTR, ETC, CWA.

 bothwayThroughConnectionInd [2] BothwayThroughConnectionInd OPTIONAL,
 -- applicable to operations CTR, ETC.

 suspendTimer [3] SuspendTimer OPTIONAL,
 -- applicable to operations CON, ICA CWA.

Annex A: incs3-dt

connectedNumberTreatmentInd [4] ConnectedNumberTreatmentInd OPTIONAL,
 --applicable to operations CON, CTR, ETC, CWA .

suppressCallDiversionNotification [5] BOOLEAN OPTIONAL,
 -- applicable to CON, ICA, CWA

suppressCallTransferNotification [6] BOOLEAN OPTIONAL,
 -- applicable to CON, ICA, CWA

allowCdINNoPresentationInd [7] BOOLEAN OPTIONAL,
 -- applicable to CON, ICA CWA
 -- indicates whether the Address Presentation restricted indicator of the ISUP
 -- "called IN number" shall be set to presentation allowed (TRUE)
 -- or presentation restricted (FALSE)

userDialogueDurationInd [8] BOOLEAN DEFAULT TRUE,
 -- applicable to operations CTR, ETC.
 -- applicable when interaction with the user is required during call set-up
 -- The interaction TRUE means the user interaction may last longer than 90 seconds.
 -- Otherwise the indicator should be set to FALSE. Used for delaying ISUP T9 timer.

overrideLineRestrictions [9] BOOLEAN DEFAULT FALSE,
 -- only applicable to operations (e.g. Connect) which lead to a transition to a PIC before
 -- the AuthorizeCallSetup PIC.
 -- When set to TRUE, this parameter indicates that some facility restrictions
 -- should not be checked when the authority to place a call is verified in the
 -- Authorize_Call_Setup PIC.
 -- Which restrictions are actually overridden is network specific.

suppressVPNAPP [10] BOOLEAN DEFAULT FALSE,
 -- applicable to CWA, CON, ICA.
 -- indicates whether to allow or stop (suppress) the forward transmission of the
 -- VPN PSS1 capability.
 -- When set to TRUE, the exchange, on receipt of this parameter, will not transmit for this call
 -- any ISUP Application transport parameter with Application Context Identifier
 -- set to « PSS1 ASE (VPN) »
 -- This indicator is populated by the SCF, where the SCF and SSF in conjunction
 -- have provided the
 -- outgoing gateway PINX functionality as required by PSS1.

calledINNumberOverriding [11] BOOLEAN OPTIONAL,
 -- applicable to CON and CWA
 -- indicates whether the generation/override of the ISUP
 -- "called IN number" is allowed (TRUE) or not allowed (FALSE)
 -- If set to FALSE, the ISUP shall not generate a "called IN number" or override
 -- an already existing "called IN number".
 -- if absent, the default will be "generation/overriding allowed" (TRUE).

redirectServiceTreatmentInd [12] SEQUENCE {
 redirectReason [0] RedirectReason OPTIONAL
 } OPTIONAL
 }
 --applicable to operation CON .
 --if absent, call redirection service is not allowed
 --Existence of this parameter requests Pivot Routing or Redirection supplementary service to be
 --allowed for the new routing address specified in the Connect operation.
 --If absent, neither Pivot Routing nor Redirection service is allowed.
 -- The redirectReason Parameter indicates the reason for invoking Pivot Routing / Redirection service
 -- Whether the service is actually invoked depends only on SSP conditions.
 --In this capability set the SCP will not know all the SSP conditions.
 -- To send that kind of conditions is out of scope of this capability set.
 --If the service is allowed, the parameter must be sent in the ISUP-FAC message (Pivot Routing case)
 -- or ISUP-REL message (Redirection case) as the service is invoked,

-- ServiceInteractionIndicatorsTwo contains Indicators which are exchanged between

Q.1238.2 : December 1999

Annex A: incs3-dt

en SSP and SCP to resolve interactions

--between IN based services and network based services, respectively between different IN based services.

ServiceKey ::= Integer4

-- Information that allows the SCF to choose the appropriate service logic.

ServiceProfileIdentifier ::= OCTET STRING

-- Indicates a particular ISDN terminal. Refer to Q.932 for encoding.

ServingAreaID ::= LocationNumber

-- Identifies the local serving area where a network provider operates. Uses the LocationNumber

-- format which is based on the Q.763 Location Number format.

-- The Nature of Address indicator field shall be set to "Spare" (value 00000000).

-- The Numbering Plan Indicator field shall be set to "Spare" (value 000).

-- Defined by the network operator.

SFBillingChargingCharacteristics ::= OCTET STRING (SIZE
(minSFBillingChargingLength..
maxSFBillingChargingLength))

-- This parameter indicates the billing and/or charging characteristics for filtered calls.

-- Its content is network operator specific

-- Its content is network signalling/operator specific.

-- SupportedTriggers TRIGGER ::= {}

SuspendTimer ::= INTEGER (0..120)

--value in seconds

TDPIIdentifier ::= CHOICE

```
{
  oneTrigger  INTEGER,
  triggers    [1] Triggers
}
```

TerminalType ::= ENUMERATED {

```
  unknown(0),
  dialPulse(1),
  dtmf(2),
  isdn(3),
  isdnNoDtmf(4),
  spare(16)
}
```

-- Identifies the terminal type so that the SCF can specify, to the SRF, the appropriate type of capability

-- (voice recognition, DTMF, display capability, etc.). Since present signalling systems do not convey

-- terminal type, this parameter applies only at originating or terminating local exchanges.

XTimerID ::= ENUMERATED {

```
  tssf(0)
  -- others ffs
}
```

-- Indicates the timer to be reset.

TimerValue ::= Integer4

-- Indicates the timer value (in seconds).

TraceInformation ::= SEQUENCE OF TraceItem

TraceItem ::= SET {

```
  scf [0] ScfID
```

```
}
```

TravellingClassMark ::= LocationNumber

-- Indicates travelling class mark information.

-- Uses the LocationNumber format which is based on the Q.763 Location Number format

Annex A: incs3-dt

rmat.

-- The Nature of Address indicator field shall be set to "Spare" (value 00000000).
 -- The Numbering Plan Indicator field shall be set to "Spare" (value 000).
 -- Maximum 2 digits.

```
-- TriggerData ::= SEQUENCE {
--   triggerId      [0] TRIGGER.&id      ({SupportedTriggers}),
--   triggerPar     [1] TRIGGER.&Parameter ({SupportedTriggers}{@triggerId})
-- }

```

```
TriggerDataIdentifier ::= SEQUENCE {
  triggerID      [0] EventTypeBCSM,
  profileIdentifier [1] ProfileIdentifier,
  extensions     [2] Extensions OPTIONAL
}

```

-- It is outside the scope of this capability set whether all TDP types really apply

```
TriggerDPTType ::= ENUMERATED {tdp-r(0), tdp-n(1)}

```

```
TriggerResults ::=
  SEQUENCE SIZE (1..numOfTriggers) OF TriggerResult

```

```
TriggerResult ::= SEQUENCE {
  tDPIdentifier [0] INTEGER,
  actionPerformed [1] ActionPerformed,
  dpName        [2] EventTypeBCSM OPTIONAL
}

```

```
Triggers ::= SEQUENCE SIZE (1..numOfTriggers) OF Trigger

```

```
Trigger ::= SEQUENCE {
  tDPIdentifier [0] INTEGER,
  dpName       [1] EventTypeBCSM OPTIONAL
}

```

```
TriggerStatus ::= ENUMERATED {
  created (0),
  alreadyExist (1),
  deleted (2),
  unknownTrigger(3)
}

```

```
TriggerType ::= ENUMERATED {
  featureActivation(0),
  verticalServiceCode(1),
  customizedAccess(2),
  customizedIntercom(3),
  emergencyService(12),
  aFR(13),
  sharedIOTrunk(14),
  offHookDelay(17),
  channelSetupPRI(18),
  tNoAnswer(25),
  tBusy(26),
  oCalledPartyBusy(27),
  oNoAnswer(29),
  originationAttemptAuthorized(30),
  oAnswer(31),
  oDisconnect(32),
  termAttemptAuthorized(33),
  tAnswer(34),
  tDisconnect(35)
}

```

Annex A: incs3-dt

```

    }
-- The type of trigger which caused call suspension
-- 4-11: Reserved; 15,16: Reserved; 19-24: Reserved

```

```

USIInformation ::= OCTET STRING (SIZE
    (minUSIInformationLength..
    maxUSIInformationLength))
-- Its content is network signalling/operator specific

```

```

USIMonitorMode ::= ENUMERATED {
    monitoringActive (0),
    monitoringInactive (1)
}

```

-- Indicates if the monitoring relationship for the specified UTSI IE should be activated or deactivated.

```

USIServiceIndicator ::= CHOICE {
    global OBJECT IDENTIFIER,
    local OCTET STRING (SIZE (
        minUSIServiceIndicatorLength..
        maxUSIServiceIndicatorLength))
}

```

-- In case of local its content is network signalling/operator specific

```

VPNIndicator ::= BOOLEAN

```

-- This parameter is set to TRUE if the originating call part supports VPN with PSS1 information flows

```

-- The Definition of range of constants Follows
highLayerCompatibilityLength INTEGER ::= 2
minCauseLength INTEGER ::= 1
numOfCounters INTEGER ::= 100
numOfInfoItems INTEGER ::= 5

```

```

    minAChBillingChargingLength INTEGER ::= 1 -- example value
    maxAChBillingChargingLength INTEGER ::= 5 -- example value
    minATMTrafficDescriptorLength INTEGER ::= 1 -- example value
    maxATMTrafficDescriptorLength INTEGER ::= 5 -- example value
    minBackwardGVNSLength INTEGER ::= 1 -- example value
    maxBackwardGVNSLength INTEGER ::= 5 -- example value
    maxBearerCapabilityLength INTEGER ::= 5 -- example value
    minBroadbandBearerCapabilityLength INTEGER ::= 1 -- example value
    maxBroadbandBearerCapabilityLength INTEGER ::= 5 -- example value
    minCalledDirectoryNumberLength INTEGER ::= 1 -- example value
    maxCalledDirectoryNumberLength INTEGER ::= 5 -- example value
    minCalledPartyNumberLength INTEGER ::= 1 -- example value
    maxCalledPartyNumberLength INTEGER ::= 5 -- example value
    minCalledPartySubaddressLength INTEGER ::= 1 -- example value
    maxCalledPartySubaddressLength INTEGER ::= 5 -- example value
    minCallingGeodeticLocationLength INTEGER ::= 1 -- example value
    maxCallingGeodeticLocationLength INTEGER ::= 10 -- example value
    minCallingPartyNumberLength INTEGER ::= 1 -- example value
    maxCallingPartyNumberLength INTEGER ::= 5 -- example value
    minCallingPartySubaddressLength INTEGER ::= 1 -- example value
    maxCallingPartySubaddressLength INTEGER ::= 5 -- example value

```

Annex A: incs3-dt

```

minCallResultLength      INTEGER ::= 1  -- example value
maxCallResultLength      INTEGER ::= 5  -- example value
minCarrierLength         INTEGER ::= 3   -- example value
maxCarrierLength         INTEGER ::= 10  -- example value
maxCauseLength           INTEGER ::= 4   -- example value
minDigitsLength          INTEGER ::= 1   -- example value
maxDigitsLength          INTEGER ::= 5   -- example value
minDisplayInformationLength INTEGER ::= 1   -- example value
maxDisplayInformationLength INTEGER ::= 5   -- example value
minEventSpecificInformationChargingLength INTEGER ::= 1   -- example value
maxEventSpecificInformationChargingLength INTEGER ::= 5   -- example value
minEventTypeChargingLength INTEGER ::= 1   -- example value
maxEventTypeChargingLength INTEGER ::= 5   -- example value
minFCIBillingChargingLength INTEGER ::= 1   -- example value
maxFCIBillingChargingLength INTEGER ::= 5   -- example value
minForwardGVNSLength     INTEGER ::= 1   -- example value
maxForwardGVNSLength     INTEGER ::= 5   -- example value
minGenericNameLength     INTEGER ::= 1   -- example value
maxGenericNameLength     INTEGER ::= 5   -- example value
minGenericNumberLength   INTEGER ::= 1   -- example value
maxGenericNumberLength   INTEGER ::= 5   -- example value
maxInitialTimeInterval   INTEGER ::= 5   -- example value
maxINServiceCompatibilityIndLength INTEGER ::= 5   -- example value
minIPAvailableLength     INTEGER ::= 1   -- example value
maxIPAvailableLength     INTEGER ::= 5   -- example value
minIPSSPCapabilitiesLength INTEGER ::= 1   -- example value
maxIPSSPCapabilitiesLength INTEGER ::= 5   -- example value
minISDNAccessRelatedInfoLength INTEGER ::= 1   -- example value
maxISDNAccessRelatedInfoLength INTEGER ::= 10  -- example value
minLocationNumberLength  INTEGER ::= 1   -- example value
maxLocationNumberLength  INTEGER ::= 5   -- example value

minMidCallControlInfoNum INTEGER ::= 1  -- example value
maxMidCallControlInfoNum INTEGER ::= 5  -- example value
minOriginalCalledPartyIDLength INTEGER ::= 1  -- example value
maxOriginalCalledPartyIDLength INTEGER ::= 5  -- example value
minQoSParameterLength     INTEGER ::= 1  -- example value
maxQoSParameterLength     INTEGER ::= 5  -- example value

minReasonLength           INTEGER ::= 1  -- example value
maxReasonLength           INTEGER ::= 5  -- example value
minRedirectingPartyIDLength INTEGER ::= 1  -- example value
maxRedirectingPartyIDLength INTEGER ::= 5  -- example value
minRequestedUTSINum       INTEGER ::= 1  -- example value
maxRequestedUTSINum       INTEGER ::= 5  -- example value
minRouteListLength        INTEGER ::= 1  -- example value
maxRouteListLength        INTEGER ::= 5  -- example value
minRoutingNumberLength    INTEGER ::= 1  -- example value
maxRoutingNumberLength    INTEGER ::= 5  -- example value
minScfIDLength            INTEGER ::= 1  -- example value
maxScfIDLength            INTEGER ::= 5  -- example value

minSCIBillingChargingLength INTEGER ::= 1  -- example value
maxSCIBillingChargingLength INTEGER ::= 5  -- example value
minSDSSinformationLength   INTEGER ::= 1  -- example value
maxSDSSinformationLength   INTEGER ::= 5  -- example value
minServiceInteractionIndicatorsLength INTEGER ::= 1  -- example value
maxServiceInteractionIndicatorsLength INTEGER ::= 5  -- example value
minSFBillingChargingLength INTEGER ::= 1  -- example value
maxSFBillingChargingLength INTEGER ::= 5  -- example value
minUSIInformationLength    INTEGER ::= 1  -- example value
maxUSIInformationLength    INTEGER ::= 5  -- example value
minUSIServiceIndicatorLength INTEGER ::= 1  -- example value
maxUSIServiceIndicatorLength INTEGER ::= 5  -- example value
numOfBCSMEvents           INTEGER ::= 4  -- example value
numOfBCUSMEvents          INTEGER ::= 4  -- example value
numOfChargingEvents        INTEGER ::= 4  -- example value
numOfCSAs                 INTEGER ::= 10  -- example value
numOfCSs                  INTEGER ::= 10  -- example value
numOfGenericNumbers        INTEGER ::= 2  -- example value
numOfINProfile             INTEGER ::= 1  -- example value
numOfTriggers              INTEGER ::= 5  -- example value
numOfInServiceCompatibilityIndLength INTEGER ::= 2  -- example value
numOfLegs                  INTEGER ::= 10  -- example value

```


Annex A: incs3-dt

```

maxAmount          INTEGER ::= 10000  -- example value
maxCallReferenceLength  INTEGER ::= 5   -- example value
maxInitialUnitIncrement  INTEGER ::= 2   -- example value
maxScalingFactor      INTEGER ::= 2   -- example value
maxSegmentsPerDataInterval  INTEGER ::= 5   -- example value

ub-nbCall           INTEGER ::= 5   -- example value
numOfAddresses       INTEGER ::= 5   -- example value
numOfServiceKeys     INTEGER ::= 3

```

```

minAttributesLength  INTEGER ::= 1
maxAttributesLength  INTEGER ::= 5
minMailBoxIDLength   INTEGER ::= 1
maxMailBoxIDLength   INTEGER ::= 5
minMessageContentLength  INTEGER ::= 1
maxMessageContentLength  INTEGER ::= 5
minReceivedInformationLength  INTEGER ::= 1
maxReceivedInformationLength  INTEGER ::= 5
maxRecordingTime      INTEGER ::= 5
numOfMessageIDs       INTEGER ::= 2

```

```

numOfExtensions      INTEGER ::= 1
maxRecordedMessageUnits  INTEGER ::= 5

```

-- Note: Definition of the cause values (informative)

```

noIndication Cause ::= '00'H
originationDenied Cause ::= '01'H
collectInformationFailure Cause ::= '02'H
invalidCollectedInformation Cause ::= '03'H
authorisationFailure Cause ::= '04'H
routeSelectFailure Cause ::= '05'H
calledPartyBusy Cause ::= '06'H
oNoAnswer Cause ::= '07'H
ss7Failure Cause ::= '08'H
bptyNoAnswer Cause ::= '09'H
tException Cause ::= '0A'H
routeFailure1 Cause ::= '0B'H
routeFailure2 Cause ::= '0C'H
bPtyBusyUDUB Cause ::= '0D'H
bPtyBusyNDUB Cause ::= '0E'H
aPtyAbandon Cause ::= '0F'H
aPtyDisc Cause ::= '10'H
bPtyDisc Cause ::= '11'H
terminationDenied Cause ::= '12'H
presentationFailure Cause ::= '13'H
calledPartyCallRejected Cause ::= '14'H
failureOfConnection Cause ::= '15'H
releaseCall Cause ::= '16'H
bPtyAlerted Cause ::= '17'H
accessOutOfOrder Cause ::= '18'H
accessMaintenanceBusy Cause ::= '19'H
accessCustomerBusy Cause ::= '1A'H
pstnLineBusy Cause ::= '1B'H
isdnLineBusy Cause ::= '1C'H
trunkGroupBusy Cause ::= '1D'H
presentCallFailure Cause ::= '1E'H
collectDigits Cause ::= '1F'H
notReachable Cause ::= '20'H

```

END

ASN.1 Text INCS3Internals

INCS3Internals

```
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
```

```
IMPORTS
```

```
    CalledPartyNumber,
    CallingPartyNumber,
    CallingPartysCategory,
    Cause,
    DestinationRoutingAddress,
    Digits,
    DTLegType,
    EventTypeBCSM,
    EventTypeCharging,
    FacilityGroup,
    ForwardCallIndicators,
    GenericNumbers,
    DTLegID,
    MonitorMode,
    OriginalCalledPartyID,
    RedirectingPartyID,
    RedirectionInformation,
    RouteList,
    ServiceKey,
    USIServiceIndicator,
    USIInformation
```

```
FROM INCS3datatypes
```

```
;
```

```
-- DEFINITION OF SIGNALLING CONTROL PRIMITIVES
```

```
CallFlag ::= ENUMERATED { -- Note: Used on the signals between BCSMs.
```

```
    senderRef(0),
    receiverRef(1)
}
```

```
CallRef ::= INTEGER
```

```
InfoDirection ::= ENUMERATED {
```

```
    utsi(0),
    stui(1)
}
```

```
AddressEndType ::= SEQUENCE {
```

```
    callRef [0] CallRef,
    digits [1] Digits OPTIONAL
}
```

```
CallProgressType ::= SEQUENCE {
```

```
    callRef [0] CallRef,
    cause [1] Cause
}
```

```
FailureType ::= SEQUENCE {
```

```
    callRef [0] CallRef,
    cause [1] Cause
}
```

```
NetworkSRType ::= SEQUENCE {
```

```
    callRef [0] CallRef
}
```

```
ReleaseType ::= SEQUENCE {
```

```
    callRef [0] CallRef,
    cause [1] Cause
}
```

```
ServiceFeatureType ::= SEQUENCE {
```

```
    callRef [0] CallRef
}
```

```
SetupCRTType ::= SEQUENCE {
```

```
    callRef [0] CallRef,
```

Annex A: cs3i

```
connectedNumber [1] CalledPartyNumber OPTIONAL
}
```

```
SetupIRType ::= SEQUENCE {
  callRef          [0] CallRef,
  calledPartyNumber [1] CalledPartyNumber,
  callingPartyNumber [2] CallingPartyNumber OPTIONAL,
  callingPartysCategory [3] CallingPartysCategory OPTIONAL,
  locationNumber [4] OCTET STRING OPTIONAL,
  originalCalledNumber [5] OriginalCalledPartyID OPTIONAL,
  userTeleserviceInformation [6] OCTET STRING OPTIONAL,
  genericNumber [7] GenericNumbers OPTIONAL,
  forwardCallIndicators [8] ForwardCallIndicators OPTIONAL,
  redirectingNumber [9] RedirectingPartyID OPTIONAL,
  redirectingInformation [10] RedirectionInformation OPTIONAL,
  uSIServiceIndicator [11] USIServiceIndicator OPTIONAL,
  uSIInformation [12] USIInformation OPTIONAL
}
```

```
SubsequentAddressType ::= SEQUENCE {
  callRef [0] CallRef,
  digits [1] Digits
}
```

```
UserDataTypes ::= SEQUENCE {
  callRef [0] CallRef,
  uSIServiceIndicator [1] USIServiceIndicator,
  uSIInformation [2] USIInformation,
  infoDirection [3] InfoDirection
}
```

-- DEFINITION OF THE INFORMATIVE MANAGEMENT OPERATION ARGUMENTS

```
MGTSetTriggerTableArg ::= SEQUENCE OF SEQUENCE {
  legID [0] DTLegType,
  serviceKey [1] ServiceKey,
  eventTypeBCSM [2] EventTypeBCSM,
  monitorMode [3] MonitorMode
}
```

-- DEFINITION OF TYPES USED INTERNALLY IN THE SDL SPECIFICATION

```
BCSMStateType ::= ENUMERATED {
  nullState(0), -- Normal case
  waitOrigAttemptAuthorized(1), -- Used with ICA
  analyseInformation(2), -- Used with Connect, AnalyseInformation
  waitMidCall(3), -- Used with Connect
  selectRoute(4), -- Used with Connect, SelectRoute
  origAttempt(5), -- Used with Connect
  collectInfo(6) -- Used with CollectInformation
}
```

```
DPFacilityArg ::= SEQUENCE {
  legID DTLegType,
  party PartyType
}
```

```
PartyType ::= ENUMERATED {
  localParty(0),
  remoteParty(1),
  surrogateParty(2)
}
```

```
DPAArg ::= SEQUENCE {
  legID [0] DTLegType,
  bcsmEvent [1] EventTypeBCSM,
  party [2] PartyType OPTIONAL,
  cgPN [3] CallingPartyNumber OPTIONAL,
  cdPN [4] CalledPartyNumber OPTIONAL,
}
```

Annex A: cs3i

```

callingPartysCategory [5] CallingPartysCategory OPTIONAL,
locationNumber [6] OCTET STRING OPTIONAL,
originalCalledNumber [7] OriginalCalledPartyID OPTIONAL,
userTeleserviceInformation [8] OCTET STRING OPTIONAL,
genericNumber [9] GenericNumbers OPTIONAL,
forwardCallIndicators [10] ForwardCallIndicators OPTIONAL,
redirectingNumber [11] RedirectingPartyID OPTIONAL,
redirectingInformation [12] RedirectionInformation OPTIONAL,
uSIServiceIndicator [13] USIServiceIndicator OPTIONAL,
uSIIInformation [14] USIIInformation OPTIONAL,
digits [15] Digits OPTIONAL,
cause [16] Cause OPTIONAL
}

```

```

DPUTSIArg ::= SEQUENCE {
  legID [0] DTLegType,
  uSIServiceIndicator [1] USIServiceIndicator,
  uSIIInformation [2] USIIInformation,
  party [3] PartyType
}

```

```
emptyString OCTET STRING ::= ''H
```

```

oct0 OCTET STRING ::= '00'H
oct1 OCTET STRING ::= '01'H
oct2 OCTET STRING ::= '02'H
oct3 OCTET STRING ::= '03'H
oct4 OCTET STRING ::= '04'H
oct5 OCTET STRING ::= '05'H
oct6 OCTET STRING ::= '06'H
oct7 OCTET STRING ::= '07'H
oct8 OCTET STRING ::= '08'H
oct9 OCTET STRING ::= '09'H
octa OCTET STRING ::= '0A'H
octb OCTET STRING ::= '0B'H

```

```

OBCSMPars ::= SEQUENCE {
  startState [0] BCSMStateType,
  legID [1] DTLegType,
  cgPN [2] CallingPartyNumber OPTIONAL,
  routingInfo [3] CHOICE {
    cdPN [0] CalledPartyNumber,
    routingAddress [1] DestinationRoutingAddress
  },
  routeList [4] RouteList OPTIONAL,
  callingPartysCategory [5] CallingPartysCategory OPTIONAL,
  locationNumber [6] OCTET STRING OPTIONAL,
  originalCalledNumber [7] OriginalCalledPartyID OPTIONAL,
  userTeleserviceInformation [8] OCTET STRING OPTIONAL,
  genericNumber [9] GenericNumbers OPTIONAL,
  forwardCallIndicators [10] ForwardCallIndicators OPTIONAL,
  redirectingNumber [11] RedirectingPartyID OPTIONAL,
  redirectingInformation [12] RedirectionInformation OPTIONAL,
  uSIServiceIndicator [13] USIServiceIndicator OPTIONAL,
  uSIIInformation [14] USIIInformation OPTIONAL
}

```

```

PointInCall ::= ENUMERATED {
  analyseInformation(0),
  authoriseOriginationAttempt(1),
  collectInformation(2),
  collectNDigits(3),
  oActive(4),
  oAlerting(5),
  oException(6),
  oNull(7),
  selectRoute(8),
  sendCall(9),
  authoriseTerminationAttempt(10),
  presentCall(11),
  selectFacility(12),
  tActive(13),
  tAlerting(14),
  tNull(15),
}

```

Annex A: cs3i

```

releaseCall(16),      -- Used with the ReleaseCall operation

disconnectLeg(17),    -- Note: IN CS-2 PIC
sendSTUI(18),         -- Note: IN CS-2 PIC
sendUTSI(19),         -- Note: IN CS-2 PIC
authorizeTermination(20), -- Note: IN CS-2 PIC
reconnect(21)         -- Note: IN CS-3
}

ChargingEventIndArg ::= SEQUENCE {
    callRef          [0] CallRef,
    eventTypeCharging [1] EventTypeCharging
}

PICArg ::= SEQUENCE {
    legID          [0] DTLegType,
    pic            [1] PointInCall,
    routingInfo    [2] CHOICE {
        cdPN      [0] CalledPartyNumber,
        routingAddress [1] DestinationRoutingAddress
    },
    routeList      [3] RouteList OPTIONAL,
    callingPartyNumber [4] CallingPartyNumber OPTIONAL,
    callingPartysCategory [5] CallingPartysCategory OPTIONAL,
    locationNumber    [6] OCTET STRING OPTIONAL,
    originalCalledNumber [7] OriginalCalledPartyID OPTIONAL,
    userTeleserviceInformation [8] OCTET STRING OPTIONAL,
    genericNumber      [9] GenericNumbers OPTIONAL,
    forwardCallIndicators [10] ForwardCallIndicators OPTIONAL,
    redirectingNumber    [11] RedirectingPartyID OPTIONAL,
    redirectionInformation [12] RedirectionInformation OPTIONAL,
    uSIServiceIndicator  [13] USIServiceIndicator OPTIONAL,
    uSIInformation       [14] USIInformation OPTIONAL,
    infoDirection        [15] InfoDirection OPTIONAL,
    party                [16] PartyType OPTIONAL,
    facilityGroup        [18] FacilityGroup OPTIONAL,
    cause                [17] Cause OPTIONAL
}

```

END

Annex A: CS3_INAP

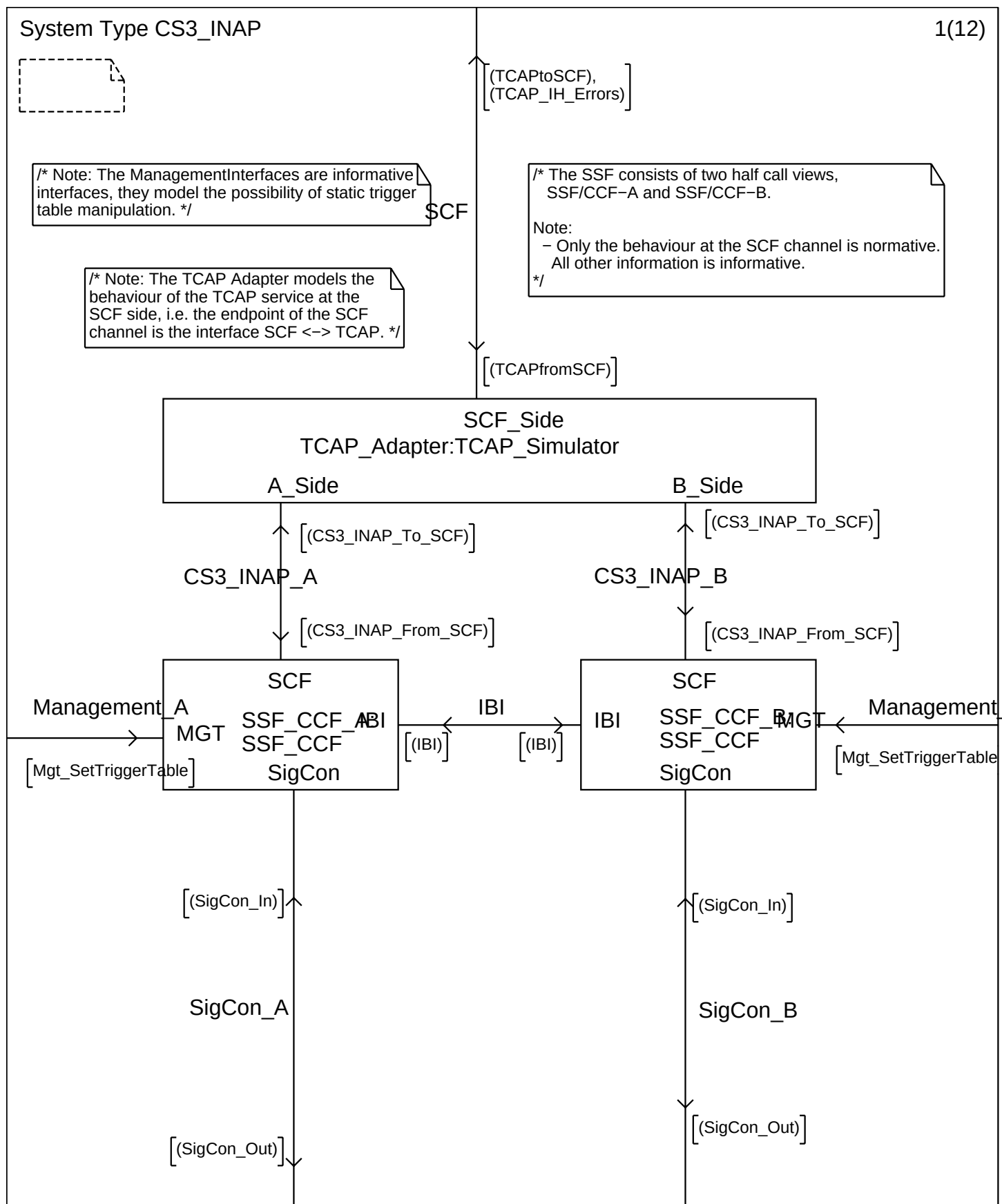
```
USE INCS3Internals;  
USE INCS3datatypes;  
USE INCS3opsargs;  
USE INCS3BundleArg;
```

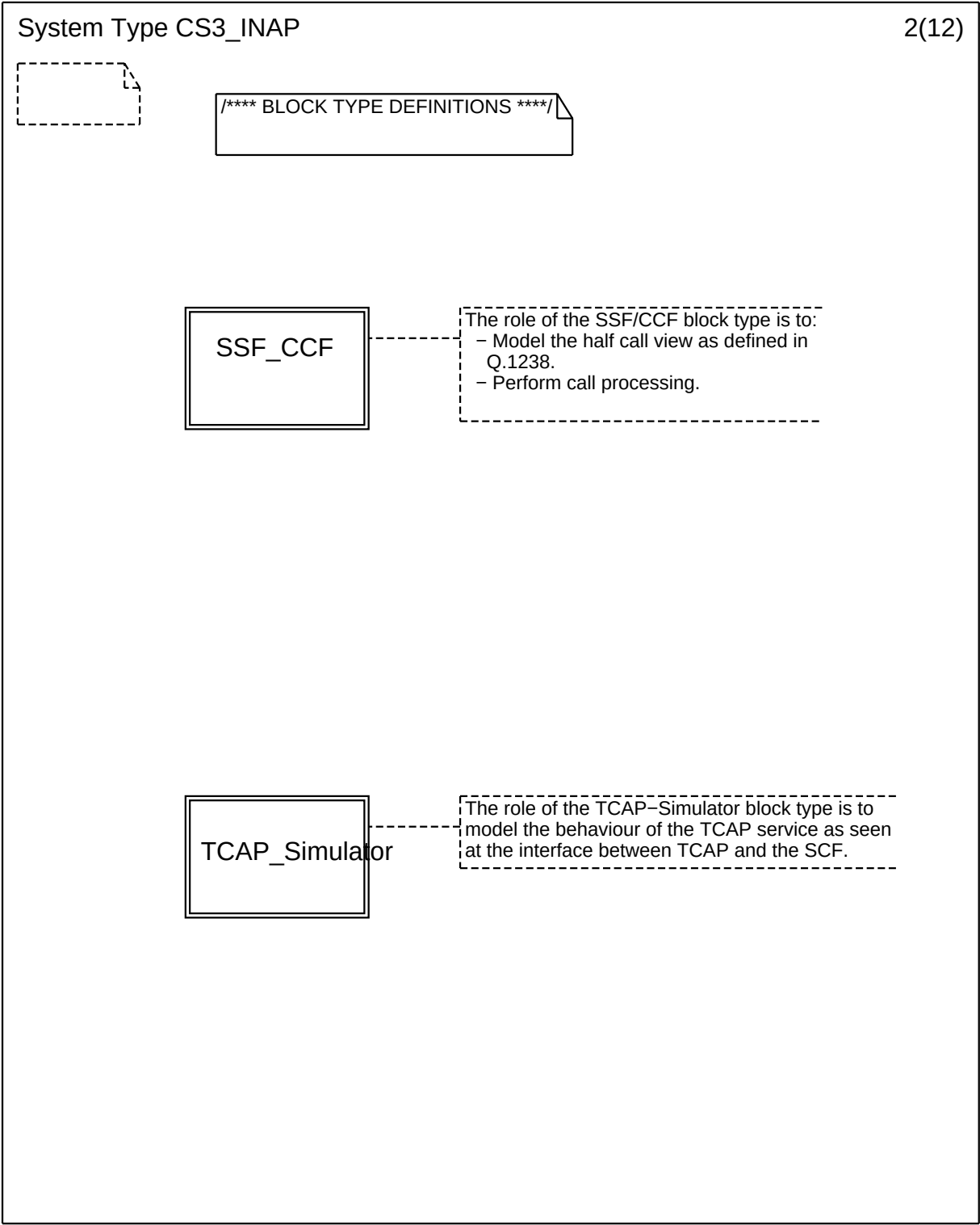
package CS3_INAP

1(1)



system
CS3_INAP





System Type CS3_INAP

3(12)

**** SIGNAL DEFINITIONS FOR THE SCF-SSF OPERATIONS. ****

/* Note: – Connect is defined as Connnect because
Connect is a reserved word in SDL.
*/

SIGNAL

/* SCF – SSF operations */

ActivateServiceFiltering(InvokeID,ActivateServiceFilteringArg),
ActivityTest(InvokeID, CSAID, DialogIDType),
ActivityTestResult(InvokeID, DialogIDType),
AnalysedInformation(CSAID, AnalysedInformationArg),
AnalyseInformation(InvokeID, CSAID, AnalyseInformationArg),
ApplyCharging(InvokeID, CSAID, ApplyChargingArg),
ApplyChargingReport(CSAID, ApplyChargingReportArg, Boolean),
AssistRequestInstructions(CSAID, AssistRequestInstructionsArg),
CallGap(InvokeID, CallGapArg),
CallInformationReport(CSAID, CallInformationReportArg, Boolean),
CallInformationRequest(InvokeID, CSAID, CallInformationRequestArg),
Cancel(InvokeID, CSAID, CancelArg),
CancelStatusReportRequest(InvokeID, CancelStatusReportRequestArg),
CollectInformation(InvokeID, CSAID, CollectInformationArg),
CollectedInformation(CSAID, CollectedInformationArg),
Connnect(InvokeID, CSAID, ConnectArg),
ConnectToResource(InvokeID, CSAID, ConnectToResourceArg),
Continue(InvokeID, CSAID),
DisconnectForwardConnection(InvokeID, CSAID),
EstablishTemporaryConnection(InvokeID, CSAID, EstablishTemporaryConnectionArg),
EventNotificationCharging(CSAID, EventNotificationChargingArg),
EventReportBCSM(CSAID, EventReportBCSMArg),
FurnishChargingInformation(InvokeID, CSAID, FurnishChargingInformationArg),
HoldCallInNetwork(InvokeID, CSAID, HoldCallInNetworkArg),
InitialDP(CSAID, InitialDPArg),
InitiateCallAttempt(InvokeID, CSAID, InitiateCallAttemptArg),
OAbandon(CSAID, OAbandonArg),
OAnswer(CSAID, OAnswerArg),
OCalledPartyBusy(CSAID, OCalledPartyBusyArg),
ODisconnect(CSAID, ODisconnectArg),
OMidCall(CSAID, MidCallArg),
ONoAnswer(CSAID, ONoAnswerArg),
OriginationAttemptAuthorized(CSAID, OriginationAttemptAuthorizedArg),
ReleaseCall(InvokeID, CSAID, ReleaseCallArg),
RequestCurrentStatusReport(InvokeID, RequestCurrentStatusReportArg),
RequestEveryStatusChangeReport(InvokeID, RequestEveryStatusChangeReportArg),
RequestFirstStatusMatchReport(InvokeID, RequestFirstStatusMatchReportArg),
RequestNotificationChargingEvent(InvokeID, CSAID, RequestNotificationChargingEventArg),
RequestReportBCSMEvent(InvokeID, CSAID, RequestReportBCSMEventArg),
ResetTimer(InvokeID, CSAID, ResetTimerArg),
RouteSelectFailure(CSAID, RouteSelectFailureArg),
SelectFacility(InvokeID, CSAID, SelectFacilityArg),
SelectRoute(InvokeID, CSAID, SelectRouteArg),
SendChargingInformation(InvokeID, CSAID, SendChargingInformationArg),
ServiceFilteringResponse(CSAID, ServiceFilteringResponseArg),
StatusReport(InvokeID, StatusReportArg),
TAnswer(CSAID, TAnswerArg),
TBusy(CSAID, TBusyArg),
TDisconnect(CSAID, TDisconnectArg),
TermAttemptAuthorized(CSAID, TermAttemptAuthorizedArg),
TMidCall(CSAID, MidCallArg),
TNoAnswer(CSAID, TNoAnswerArg);

System Type CS3_INAP

4(12)

**** SIGNAL DEFINITIONS FOR THE IN SCF-SSF OPERATIONS. ****

SIGNAL

/* SCF – SRF operations */

PlayAnnouncement(InvokeID, PlayAnnouncementArg),
PromptAndCollectUserInfo(InvokeID, CSAID, PromptAndCollectUserInfoArg),
PromptAndCollectUserInfoResult(CSAID, ReceivedInformationArg),
SpecializedResourceReport(CSAID, SpecializedResourceReportArg),

/* Informative operations */

Mgt_SetTriggerTable(MGTSetTriggerTableArg); /* Informative signal, used to model the management of the trigger table. */

SIGNAL

AuthorizeTermination(InvokeID, CSAID, AuthorizeTerminationArg),
ContinueWithArgument(InvokeID, CSAID, ContinueWithArgumentArg),
CreateCallSegmentAssociation(InvokeID, CSAID, CreateCallSegmentAssociationArg),
CreateCallSegmentAssociationResult(InvokeID, CSAID, CreateCallSegmentAssociationResultArg),
CreateOrRemoveTriggerData(InvokeID, CSAID, CreateOrRemoveTriggerDataArg),
CreateOrRemoveTriggerDataResult(CSAID, CreateOrRemoveTriggerDataArg),
DisconnectLeg(InvokeID, CSAID, DisconnectLegArg),
DisconnectLegResult(InvokeID, CSAID),
DisconnectForwardConnectionWithArgument(InvokeID, CSAID, DisconnectForwardConnectionWithArgumentArg),
EntityReleased(CSAID, EntityReleasedArg),
EventReportFacility(CSAID, EventReportFacilityArg),
FacilitySelectedAndAvailable(CSAID, FacilitySelectedAndAvailableArg),
ManageTriggerData(InvokeID, DialogIDtype, ManageTriggerDataArg),
ManageTriggerDataResult(InvokeID, DialogIDtype, ManageTriggerDataResultArg),
MergeCallSegments(InvokeID, CSAID, MergeCallSegmentsArg),
MergeCallSegmentsResult(InvokeID, CSAID),
MoveCallSegments(InvokeID, CSAID, MoveCallSegmentsArg),
MoveCallSegmentsResult(InvokeID, CSAID),
MoveLeg(InvokeID, CSAID, MoveLegArg),
MoveLegResult(InvokeID, CSAID),
OriginationAttempt(CSAID, OriginationAttemptArg),
OSuspended(CSAID, OSuspendedArg),
Reconnect(InvokeID, CSAID, ReconnectArg),
ReportUTSI(CSAID, ReportUTSIArg),
RequestReportFacilityEvent(InvokeID, CSAID, RequestReportFacilityEventArg),
RequestReportUTSI(InvokeID, CSAID, RequestReportUTSIArg),
SetServiceProfile(InvokeID, CSAID, SetServiceProfileArg),
SendFacilityInformation(InvokeID, CSAID, SendFacilityInformationArg),
SendSTUI(InvokeID, CSAID, SendSTUIArg),
SplitLeg(InvokeID, CSAID, SplitLegArg),
SplitLegResult(InvokeID, CSAID),
TerminationAttempt(CSAID, TerminationAttemptArg),
TSuspended(CSAID, TSuspendedArg),

PromptAndReceiveMessage(InvokeID, PromptAndReceiveMessageArg),
PromptAndReceiveMessageResult(InvokeID, CSAID, MessageReceivedArg),
ScriptClose(InvokeID, ScriptCloseArg),
ScriptEvent(InvokeID, ScriptEventArg),
ScriptInformation(InvokeID, ScriptInformationArg),
ScriptRun(InvokeID, ScriptRunArg);

System Type CS3_INAP

/* Data type definitions used by the TCAP Simulator */

5(12)

```
/* Needed for the initialization of the TCAP Simulator */
NEWTYPE IHroleType
  LITERALS
    A_Side, B_Side;
ENDNEWTYPE;

/* Dialog IDs */
/* - Total valid numbers 1 - 100 */
/* - Direction SCF -> SSF: 1 - 50 */
/* - Direction SSF -> SCF: 51 - 100 */
/* - 0 is used as flag, if something strange happened, e.g., for a given csaID no dialogID is found */
SYNONYM maxDialogIDtotal Integer = 100;
SYNONYM maxDialogIDtoSSF Integer = 50;
SYNTYPE DialogIDtype = Integer CONSTANTS 0:maxDialogIDtotal
ENDSYNTYPE;

/* Invoke IDs */
/* - Total valid numbers 1 - 200 */
/* - Direction SCF -> SSF: 1 - 100 */
/* - Direction SSF -> SCF: 101 - 200 */
/* - 0 is used as dummy */
SYNONYM maxInvokeIDtotal Integer = 200;
SYNONYM maxInvokeIDtoSSF Integer = 100;
SYNTYPE InvokeIDtype = Integer CONSTANTS 0:maxInvokeIDtotal
ENDSYNTYPE;

/* For indicating the origin of TCAP messages */
NEWTYPE TCoriginType
  LITERALS
    oSSF, oSCF;
ENDNEWTYPE;

/* refers to basic and prearranged end in TC_EndReq/Ind primitives */
NEWTYPE TCAPterminationType
  LITERALS
    basic, prearranged;
ENDNEWTYPE;

/* For Timeout values for operations, mandatory parameter in TC_InvokeReq primitives */
NEWTYPE TimeoutValType
  LITERALS
    short, medium, long;
ENDNEWTYPE;
```

System Type CS3_INAP

6(12)



/* Data type definitions used by the TCAP Simulator (Continued)*/

```
/* Reasons for a TCAP failure */
NEWTYPE TCAPfailReasonType
LITERALS
  wrongDialogID, /* incorrect dialog ID: incorrect range or no dialog for ID present/available */
  wrongInvokeID, /* incorrect invoke ID */
  wrongCSaID, /* incorrect CSA ID */
  beginRequired, /* TC_BeginReq is required */
  noComp, /* TC_BeginReq without components */
  unknownOP, /* Unknown operation code (within TC Invoke) */
  noDialogAvail, /* No dialogID available, 'maxDialogIDtoSCF' Dialogs are established */
  cSAnotAvail, /* the provide CSAid is not available */
  unexpSignal, /* the TCAP IH received an unexpected signal */
  signalNotProcessed, /* signal cannot be processed by the TCAP signal handler */
  preArrangedEndReq; /* End signal with wrong termination parameter */
ENDNEWTYPE;
```

```
/* Operation Codes */
NEWTYPE OpCodeType
LITERALS
  /* No Operation */ NoOperation,
  /* CLASS 1 */ CSA, DL, MTD, MC, MCS, ML, RCS, RES, SL,
  /* CLASS 2 */ ASF, AC, CIRQ, CAN, CSR, CI, CON, CTR, CWA,
    DFC, DFCWA, ETC, FCI, ICA, RE, RFS, RNC, RRB, RRU, RT, SCI, SS,
    PA, PAC, PAR, SC, SR, SI, SE, SRR,
  /* CLASS 4 */ AT, CG, CUE, RC,
  /* FROM SSF */ ACR, ARI, CIR, IDP, ER, ENC, ERB, RU, SFR, SRP,
  /* RETURN RESULTS FROM SSF */ AT_R, ASF_R, SL_R, DL_R, MCS_R, MC_R,
    ML_R, RCS_R, RES_R, MTD_R, CSA_R;
ENDNEWTYPE;

/* Operation Classes */
SYNTYPE OpClassType = Integer CONSTANTS 0:4
ENDSYNTYPE;
/*- 0 means no class */
```

System Type CS3_INAP

7(12)

/* SIGNAL Definitions for the TCAP Adapter*/

/* No Operations, but additional Signals exchanged between TCAP Adapter, SSF and SCF */

SIGNAL

CSaIDreq(DialogIDtype),
CSaIDresp(DialogIDtype,CSAID),
RegisterSSFreq(IHroleType),
RegisterSSFresp(IHroleType),
TCAPFailureInd(TCAPfailReasonType);

/* TCAP Primitives */

SIGNAL

/* TC_nameReq: Direction SCF->TCAP */

TC_InvokeReq(InvokeIDtype,DialogIDtype,OpClassType,OpCodeType,TimeoutValType,ArgType),
TC_BeginReq(DialogIDtype,TCoriginType),
TC_ContinueReq(DialogIDtype,TCoriginType),
TC_EndReq(DialogIDtype,TCAPterminationType),
TC_AbortReq(DialogIDtype),

/* TC_nameInd: Direction TCAP->SCF */

TC_BeginInd(DialogIDtype,TCoriginType,Boolean),
TC_ContinueInd(DialogIDtype,TCoriginType,Boolean),
TC_InvokeInd(InvokeIDtype,DialogIDtype,OpCodeType,Boolean,ArgType),
TC_EndInd(DialogIDtype,TCAPterminationType,Boolean),
TC_AbortInd(DialogIDtype),
TC_ErrorInd(InvokeIDtype,DialogIDtype,Boolean),
TC_ReturnResultInd(InvokeIDtype,DialogIDtype,Boolean,opCodeType,ArgType);

/***** SIGNAL DEFINITIONS FOR THE ERROR AND DIALOUGE HANDLING *****/

/* Note: Used between the HalfCalls (SSF/CCF_A and SSF/CCF_B) and the TCAP Adaptor. */

/* Type definition used for the return error. The values
correspond to the error codes defined in the ASN.1 */

SYNTYPE ErrorArg = NATURAL
CONSTANTS 0:23
ENDSYNTYPE;

SIGNAL

Error(InvokeID,CSAID,ErrorArg), /* Return error. */
ApplicationBegin(CSAID), /* Begin a dialouge. */
ApplicationContinue(CSAID),
ApplicationAbort(CSAID), /* Abort a dialouge. */
ApplicationEnd(Boolean,CSAID); /* End a dialouge
indicating whether prearranged end or not. */

System Type CS3_INAP

8(12)



/* SCF to TCAP */

```
SIGNALLIST TCAPfromSCF =  
  TC_InvokeReq,  
  TC_BeginReq,  
  TC_ContinueReq,  
  TC_EndReq,  
  TC_AbortReq;
```

/* TCAP to SCF */

```
SIGNALLIST TCAPtoSCF =  
  TCAPFailureInd,  
  TC_EndInd,  
  TC_AbortInd,  
  TC_BeginInd,  
  TC_ContinueInd,  
  TC_InvokeInd,  
  TC_ErrorInd,  
  TC_ReturnResultInd;
```

/* Signallists for the gate definitions */

```
SIGNALLIST S1 = (TCAPtoSCF), (TCAP_IH_Errors);  
SIGNALLIST S2 = (TCAPfromSCF);  
SIGNALLIST A1 = (CS3_INAP_From_SCF), (TC_IH_To_SSF);  
SIGNALLIST A2 = (CS3_INAP_To_SCF);  
SIGNALLIST B1 = (A1);  
SIGNALLIST B2 = (A2);
```

/* TCAP IH error Messages (to SCF) */

```
SIGNALLIST TCAP_IH_Errors = TCAPFailureInd, TC_AbortInd;
```

/* TCAP interface to SSF */

```
SIGNALLIST TC_IH_To_SSF =  
  RegisterSSFreq, /* SSF side registration request */  
  CSaIDreq; /* request for a CSA ID */
```

System Type CS3_INAP

9(12)



***** SIGNAL/PRIMITIVE LIST DEFINITIONS FOR THE SCF-SSF INTERFACE *****

/* SSF -> SCF */

SIGNALLIST CS1_INAP_To_SCF =

AnalysedInformation,
ApplyChargingReport,
AssistRequestInstructions,
ActivityTestResult,
CallInformationReport,
CollectedInformation,
EventNotificationCharging,
EventReportBCSM,
InitialDP,
OAbandon,
OAnswer,
OCalledPartyBusy,
ODisconnect,
OMidCall,
ONoAnswer,
OriginationAttemptAuthorized,
RouteSelectFailure,
StatusReport,
TAnswer,
TBusy,
TDisconnect,
TermAttemptAuthorized,
TMidCall,
TNoAnswer,
ServiceFilteringResponse,

PromptAndCollectUserInformationResult,
SpecializedResourceReport,

Error,
ApplicationBegin,
ApplicationContinue,
ApplicationAbort,
ApplicationEnd,

/* Signals used to pass information to/from TCAP. */

RegisterSSFresp,
CSalDresp;

/* SCF -> SSF */

SIGNALLIST CS1_INAP_From_SCF =

ActivateServiceFiltering,
ActivityTest,
AnalyseInformation,
ApplyCharging,
CallGap,
CallInformationRequest,
Cancel,
CancelStatusReportRequest,
CollectInformation,
Connect,
ConnectToResource,
Continue,
DisconnectForwardConnection,
EstablishTemporaryConnection,
FurnishChargingInformation,
HoldCallInNetwork,
InitiateCallAttempt,
ReleaseCall,
RequestCurrentStatusReport,
RequestEveryStatusChangeReport,
RequestFirstStatusMatchReport,
RequestNotificationChargingEvent,
RequestReportBCSMEvent,
ResetTimer,
SelectFacility,
SelectRoute,
SendChargingInformation,

PlayAnnouncement,
PromptAndCollectUserInformation,

Mgt_SetTriggerTable, /* Operation on the management interface, informative

ApplicationBegin,
ApplicationContinue,
ApplicationAbort,
ApplicationEnd,

/* Extensions due to the needs of the TCAP Adapter */

RegisterSSFreq,
CSalDreq;

System Type CS3_INAP

10(12)



/* Signal/Primitive List Definitions for the IN SCF-SSF Interface */

```
/* SSF -> SCF */
SIGNALLIST CS3_INAP_To_SCF =
(CS1_INAP_To_SCF),
CreateCallSegmentAssociationResult,
CollectedInformation,
DisconnectLegResult,
EntityReleased,
EventReportFacility,
FacilitySelectedAndAvailable,
ManageTriggerDataResult,
MergeCallSegmentsResult,
MoveCallSegmentsResult,
MoveLegResult,
OriginationAttempt,
OSuspended,
SendFacilityInformation,
TerminationAttempt,
TSuspended,

ReportUTSI,
SplitLegResult,

PromptAndReceiveMessageResult,
ScriptEvent;

/* SCF -> SSF */
SIGNALLIST CS3_INAP_From_SCF =
(CS1_INAP_From_SCF),
AuthorizeTermination,
ContinueWithArgument,
CreateCallSegmentAssociation,
CreateOrRemoveTriggerData,
DisconnectForwardConnectionWithArgument,
DisconnectLeg,
ManageTriggerData,
MergeCallSegments,
MoveCallSegments,
MoveLeg,
Reconnect,
RequestReportFacilityEvent,
RequestReportUTSI,
SetServiceProfile,
SendSTUI,
SendFacilityInformation,
SplitLeg,

PromptAndReceiveMessage,
ScriptClose,
ScriptInformation,
ScriptRun;
```


System Type CS3_INAP

11(12)



**** DEFINITION OF THE PRIMITIVES FOR THE SIGNALLING CONTROL INTERFACE ****

/* Note: The signalling control interface is a generic interface that can be mapped to e.g. ISUP and DSS.1 */

/* Signal and signal list definitions for the CS3 extension to the SigCon interfaces */

SIGNAL

NetworkSuspendInd(NetworkSRType),
NetworkSuspendReq(NetworkSRType),
NetworkResumeInd(NetworkSRType),
NetworkResumeReq(NetworkSRType),
DataReq(UserDataType, CallFlag),
DataInd(UserDataType);

SIGNALLIST SigCon2_Out =

NetworkSuspendReq,
NetworkResumeReq,
DataReq;

SIGNALLIST SigCon2_In =

NetworkSuspendInd,
NetworkResumeInd,
DataInd;

/* Definition of primitives for the extension to the internal interface between BCSMs (IBI). */

SIGNAL

NetworkSuspendReqInd(NetworkSRType, CSAID, LegType),
NetworkResumeReqInd(NetworkSRType, CSAID, LegType),
DataReqInd(UserDataType, CSAID, LegType, CallFlag);

SIGNALLIST IBI2 =

NetworkSuspendReqInd,
NetworkResumeReqInd,
DataReqInd;

System Type CS3_INAP

12(12)

**** DEFINITION OF THE TYPES AND PRIMITIVES FOR THE SIGNALLING CONTROL INTERFACE ****

/* Note: The signalling control interface is a generic interface that can be mapped to e.g. ISUP and DSS.1 */

/* Definition of primitives for the Signalling Control (SigCon) interfaces. */

SIGNAL

AddressEndInd(AddressEndType),
CallProgressInd(CallProgressType),
CallProgressReq(CallProgressType),
FailureInd(FailureType),
ReleaseInd(ReleaseType),
ReleaseReq(ReleaseType, CallFlag),
ServiceFeatureInd(ServiceFeatureType),
SetupConf(SetupCRType),
SetupInd(SetupIRType),
SetupReq(SetupIRType),
SetupResp(SetupCRType),
SubsequentAddressInd(SubsequentAddressType),
SubsequentAddressReq(SubsequentAddressType);

SIGNALLIST SigCon_In =
(O_SigCon_In),
(T_SigCon_In),
(SigCon2_In);

SIGNALLIST SigCon_Out =
(O_SigCon_Out),
(T_SigCon_Out),
(SigCon2_Out);

/* From SigCon to O-BCSM */

SIGNALLIST O_SigCon_In =
AddressEndInd,
FailureInd,
ReleaseInd,
ServiceFeatureInd,
SetupInd,
SubsequentAddressInd;

/* from O-BCSM to SigCon */

SIGNALLIST O_SigCon_Out =
CallProgressReq,
ReleaseReq,
SetupResp,
SubsequentAddressReq;

/* from T-BCSM to SigCon */

SIGNALLIST T_SigCon_Out =
ReleaseReq,
SetupReq,
SubsequentAddressReq;

/* from called agent to T-BCSM */

SIGNALLIST T_SigCon_In =
CallProgressInd,
FailureInd,
ReleaseInd,
ServiceFeatureInd,
SetupConf;

/* Definition of primitives for the internal interface between BCSMs (IBI). */

SIGNAL

CallProgressReqInd(CallProgressType, CSAID, LegType),
ReleaseReqInd(ReleaseType, CSAID, LegType, CallFlag),
SetupReqInd(SetupIRType, CSAID, LegType),
SetupRespConf(SetupCRType, CSAID, LegType),
SubsequentAddressReqInd(SubsequentAddressType, CSAID, LegType);

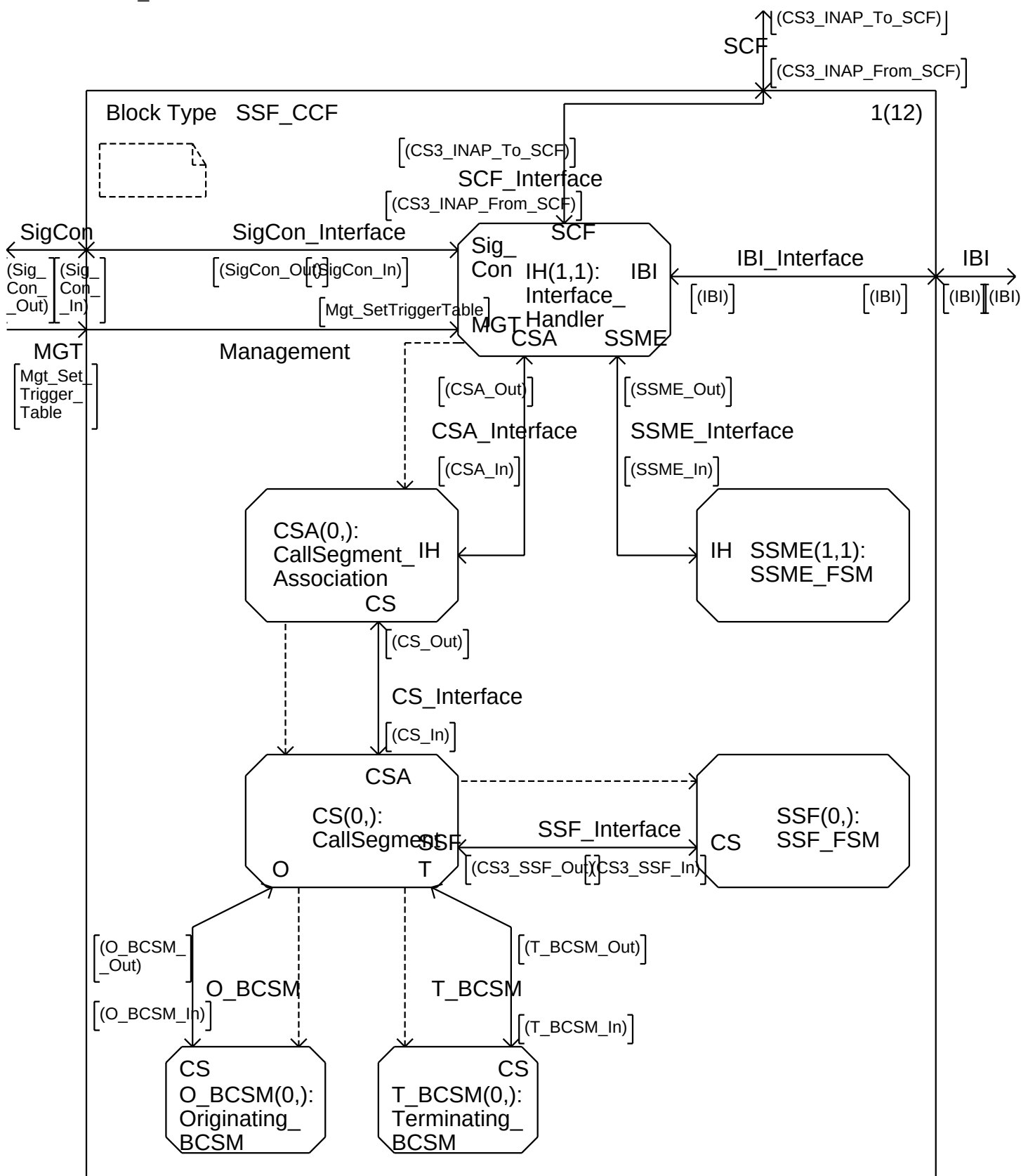
/* Terminating to Originating BCSM */

SIGNALLIST IBI_In =
CallProgressReqInd,
ReleaseReqInd,
SetupRespConf;

/* Originating to Terminating BCSM */

SIGNALLIST IBI_Out =
ReleaseReqInd,
SetupReqInd,
SubsequentAddressReqInd;

SIGNALLIST IBI =
(IBI_In),
(IBI_Out),
(IBI2);

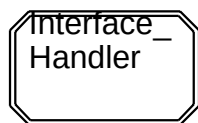


Block Type SSF_CCF

2(12)



**** PROCESS TYPE DEFINITIONS ****



The InterFaceHandler (IH) process type manages:

- The creation of CallSegmentAssociations (CSAs).
- Processing of the operations MoveCallSegments and CreateCSA.
- The dialouge with the SCF.
- The passing of primitives between the Signalling Control Interface and the CSAs.
- The passing of primitives between the SSF-SCF Interface and the CSA.

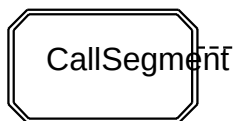


The CallSegmentAssociation (CSA) process type manages:

- The creation of CallSegments
- Processing of call segment related operations.
- The passing of Signalling Control primitives between the InterfaceHandler and the CallSegment.
- The passing of primitives between the InterfaceHandler and the CallSegment.

Block Type SSF_CCF

3(12)



The CallSegment (CS) process type manages:

- The connection view (in terms of legs) of the call segment.
- The creation of O-BCSMs and T-BCSMs.
- The creation of the SSF-FSM.
- Processing of connection view oriented operations.
- Filtering of detection points
- The passing of Signalling Control primitives between the CallSegmentAssociation and the BCSMs.
- The passing of primitives between the CallSegmentAssociation and the SSF-FSM.
- The passing of DPs and PICs between the BCSMs and the SSF-FSM.



The SSF-FSM process type manages:

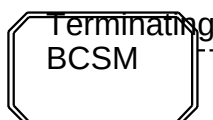
- The processing of IN operations.
- The handling of detection points (EDPs & TDPs).

Note: Only one SSF-FSM/CS can exist.



The OriginatingBCSM (O-BCSM) process type manages:

- The call processing of originating half calls.



The TerminatingBCSM (T-BCSM) process type manages:

- The call processing of terminating half calls.



The SSME-FSM process type manages:

- The processing of IN CS-3 management.
- The handling of static arming of TDPs.
- The processing of ManageTriggerData

Block Type SSF_CCF

4(12)

**** SIGNAL DEFINITIONS FOR DETECTION POINTS ****/

/* Note: Detection points are sent from a BCSM to the SSF-FSM via the CallSegment. */

```
SIGNAL
DP(DPArg),
DPAbandon(DPArg),
DPDisconnect(DPArg),
DPMidCall(DPArg);

/* Signals from O-BCSM to SSF */
SIGNALLIST oDPs =
DP,
DPAbandon,
DPDisconnect,
DPMidCall;

/* Signals from T-BCSM to SSF */
SIGNALLIST tDPs =
DP,
DPAbandon,
DPDisconnect,
DPMidCall;

SIGNALLIST DPs =
(oDPs),
(tDPs);
```

**** SIGNAL DEFINITIONS FOR POINTS IN CALL ****/

/* Note: – Points in call are sent from the SSF-FSM to a BCSM via the CallSegment. */

```
SIGNAL
PIC(PICArg),
PICResume(LegType, Boolean);
/* The second parameter is used in connection with ContinueWithArgument. */

/* from SSF to BCSM */
SIGNALLIST oPICs =
PIC,
PICResume;

SIGNALLIST tPICs =
PIC,
PICResume;

SIGNALLIST PICs =
(oPICs),
(tPICs);
```

Block Type SSF_CCF

5(12)

**** DEFINITIONS COMMON TO THE IH, CSA, CS, BCSM AND SSF-FSM PROCESSES. */

/* Definition of Leg */

```
NEWTYPE LegStatusType
  LITERALS joined, pending, shared, surrogate;
ENDNEWTYPE;
```

```
NEWTYPE BCSMType
  LITERALS Originating, Terminating;
ENDNEWTYPE;
```

```
NEWTYPE LegInfo STRUCT
  Used Boolean; /* Flag indicating whether the leg is in use or not. */
  LegPtr PId; /* Pointer/Address to the BCSM connected to the leg. */
  BCSM BCSMType; /* Type (O or T) of BCSM connected to the (passive) leg. */
  LegStatus LegStatusType; /* Status of the leg. */
DEFAULT (. FALSE, Null, Originating, joined .);
ENDNEWTYPE;
```

```
NEWTYPE LegArray /* Array of legs. The LegID of a leg is the index in the array. */
  ARRAY (LegType, LegInfo)
ENDNEWTYPE;
```

/* An SSF-FSM can start from either the Idle state (initial SSF-FSM) or from the WFI state (additional SSF-FSMs). */

```
NEWTYPE SSFStateType
  LITERALS
    Idle,
    WaitingForInstruction;
ENDNEWTYPE;
```

/* The SSF FSM maintains an Event List for each LegId. The event list contains the BCSM event type and the monitor mode. */

```
SYNTYPE NumberOfBCSMEvents = NATURAL
  CONSTANTS 1:30
ENDSYNTYPE;
```

SYNONYM numOfBCSMEvents NATURAL = 30;

```
NEWTYPE EventListType
  ARRAY (NumberOfBCSMEvents, BCSMEvent)
ENDNEWTYPE;
```

```
SYNTYPE MaxNumOfLegs = NATURAL
  CONSTANTS 1:numOfLegs
ENDSYNTYPE;
```

```
NEWTYPE LegEventListType
  ARRAY (MaxNumOfLegs, EventListType)
ENDNEWTYPE;
```

The SSF-FSM and the SSME maintains an Event List for each LegId. The event list contains the BCSM event type and the monitor mode. */

```
NEWTYPE EventTypeType
  ARRAY (LegType, EventRecordType)
ENDNEWTYPE;
```

```
NEWTYPE EventRecordType
  ARRAY (EventTypeBCSM, ServiceKeyEventType)
ENDNEWTYPE;
```

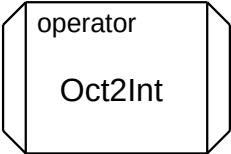
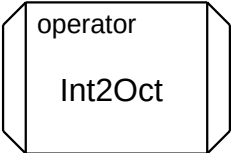
```
NEWTYPE ServiceKeyEventType
  ARRAY (ServiceKey, BCSMEvent)
ENDNEWTYPE;
```

Block Type SSF_CCF

6(12)



```
NEWTYPE bidon
OPERATORS
  Int2Oct  : integer  -> octet_string;
  Oct2Int  : octet_string  -> integer;
OPERATOR Int2Oct REFERENCED;
OPERATOR Oct2Int REFERENCED;
ENDNEWTYPE;
```



Block Type SSF_CCF

7(12)

**** DEFINITIONS OF INTERNAL SIGNALS BETWEEN PROCESSES. */

/* Signals used for passing information between the SSF-FSM and the SSME-FSM. */

SIGNAL

ArmTDPsReq(CSAID,CallSegmentID),
ArmTDPsResp(CSAID,CallSegmentID,EventType),
MSFCReq(CSAID,CallSegmentID),
MSFCResp(CSAID,CallSegmentID,Boolean),
CACGReq(CSAID,CallSegmentID),
CACGResp(CSAID,CallSegmentID,Boolean),
CFReq(CSAID,CallSegmentID),
CFResp(CSAID,CallSegmentID,Boolean);

SIGNALLIST SSME_Reqs =

ArmTDPsReq,
MSFCReq,
CACGReq,
CFReq;

SIGNALLIST SSME_Resps =

ArmTDPsResp,
MSFCResp,
CACGResp,
CFResp;

/* Signals used for passing information between the CSA, CS, BCSM and SSF-FSM. */

SIGNAL

BCSMStop(LegType,PartyType), /* BCSM -> CS: indication that the BCSM has stopped.*/
CSStop(CallSegmentID,Cause), /* CS -> CSA: indication that the CS has stopped. */
SSFStop; /* CS -> SSF-FSM: instruction to stop the SSF-FSM. */

Block Type SSF_CCF

8(12)



**** DATA TYPE DEFINITIONS COMMON TO THE IH, CSA, CS, BCSM AND SSF-FSM PROCESSES. */

/* A Leg can be exported/imported between call segments. */

```
NEWTYPE ExportLegType STRUCT
  sigConId CallRef; /* Used when exporting the controlling leg. */
  csLegData LegInfo;
  ssfLegData EventRecordType;
  controllingLegStatus LegStatusType;
ENDNEWTYPE;
```

/* A call segment can be exported/imported between call segment associations. */

```
SYNTYPE ExportCSType = PId
ENDSYNTYPE;
```

SIGNAL

/* Signals for importing and exporting legs between CS and SSF-FSM. */

```
ExportLegReq(LegType),
ExportLegResp(EventRecordType),
ImportLegReq(LegType,EventRecordType),
ImportLegResp,
```

/* Signals for importing and exporting legs between CSA and CS. */

```
CSExportLegReq(LegType),
CSExportLegResp(ExportLegType),
CSImportLegReq(LegType,ExportLegType),
CSImportLegResp,
```

/* Signal to change the value of the CSA pointer in the CSs when the CS has been moved into another CSA. */

```
SetCSA,
```

/* Signal to change the value of the CS pointer in the BCSMs when the leg has been moved into another call segment. */

```
SetCS,
```

/* Signals for importing and exporting call segments between IH and CSA. */

```
ExportCSReq(CallSegmentID, ExportLegsType),
ExportCSResp(ExportCSType, Boolean),
ImportCSReq(CallSegmentID,ExportCSType, ExportLegsType, Boolean),
ImportCSResp,
SetLegID(LegType,LegType),
```

/* Signal to report to the SSF-FSM that an UTSI event has occurred. */

```
DPUTSI(DPUTSIArg),
DPFacility(DPFacilityArg);
```

Block Type SSF_CCF

9(12)

/* SIGNAL LIST DEFINITIONS */

/* INAP OPERATIONS ON THE CSA/IH INTERFACE */

SIGNALLIST CS1_INAP_CSA_In =

AnalyseInformation,
ApplyCharging,
CallInformationRequest,
Cancel,
CollectInformation,
Connect,
ConnectToResource,
Continue,
DisconnectForwardConnection,
EstablishTemporaryConnection,
FurnishChargingInformation,
HoldCallInNetwork,
InitiateCallAttempt,
ReleaseCall,
RequestNotificationChargingEvent,
RequestReportBCSMEvent,
ResetTimer,
SelectFacility,
SelectRoute,
SendChargingInformation,

PlayAnnouncement,
PromptAndCollectUserInformation,

ApplicationBegin,
ApplicationContinue,
ApplicationAbort,
ApplicationEnd;

SIGNALLIST CS1_INAP_CSA_Out =

AnalysedInformation,
ApplyChargingReport,
AssistRequestInstructions,
CallInformationReport,
CollectedInformation,
EventNotificationCharging,
EventReportBCSM,
InitialDP,
OAbandon,
OAnswer,
OCalledPartyBusy,
ODisconnect,
OMidCall,
ONoAnswer,
OriginationAttemptAuthorized,
RouteSelectFailure,
TAnswer,
TBusy,
TDisconnect,
TermAttemptAuthorized,
TMidCall,
TNoAnswer,

PromptAndCollectUserInformationResult,
SpecializedResourceReport,

Error,
ApplicationBegin,
ApplicationContinue,
ApplicationAbort,
ApplicationEnd;

SIGNALLIST CS3_INAP_CSA_IN =

(CS1_INAP_CSA_IN),
AuthorizeTermination,
ContinueWithArgument,
CreateCallSegmentAssociation,
DisconnectForwardConnectionWithArgument,
DisconnectLeg,
MergeCallSegments,
MoveLeg,
Reconnect,
RequestReportFacilityEvent,
RequestReportUTSI,
SendFacilityInformation,
SendSTUI,
SplitLeg,
SetServiceProfile,

PromptAndReceiveMessage,
ScriptClose,
ScriptInformation,
ScriptRun;

SIGNALLIST CS3_INAP_CSA_OUT =

(CS1_INAP_CSA_OUT),
CreateCallSegmentAssociationResult,
DisconnectLegResult,
EntityReleased,
EventReportFacility,
FacilitySelectedAndAvailable,
MergeCallSegmentsResult,
MoveLegResult,
OriginationAttempt,
OSuspended,
ReportUTSI,
SplitLegResult,
TerminationAttempt,
TSuspended,

ScriptEvent,
PromptAndReceiveMessageResult;

Block Type SSF_CCF

10(12)



/* Signal List Definitions */

/* INAP Operations on the CS/CSA Interface */

SIGNALLIST CS1_INAP_CS_In =
AnalyseInformation,
ApplyCharging,
CallInformationRequest,
Cancel,
CollectInformation,
Connect,
ConnectToResource,
Continue,
DisconnectForwardConnection,
EstablishTemporaryConnection,
FurnishChargingInformation,
HoldCallInNetwork,
InitiateCallAttempt,
ReleaseCall,
RequestNotificationChargingEvent,
RequestReportBCSMEvent,
ResetTimer,
SelectFacility,
SelectRoute,
SendChargingInformation,

PlayAnnouncement,
PromptAndCollectUserInformation,

ApplicationBegin,
ApplicationContinue,
ApplicationAbort,
ApplicationEnd;

SIGNALLIST CS1_INAP_CS_Out =
AnalysedInformation,
ApplyChargingReport,
AssistRequestInstructions,
CallInformationReport,
CollectedInformation,
EventNotificationCharging,
EventReportBCSM,
InitialDP,
OAbandon,
OAnswer,
OCalledPartyBusy,
ODisconnect,
OMidCall,
ONoAnswer,
OriginationAttemptAuthorized,
RouteSelectFailure,
TAnswer,
TBusy,
TDisconnect,
TermAttemptAuthorized,
TMidCall,
TNoAnswer,

PromptAndCollectUserInformationResult,
SpecializedResourceReport,

Error,
ApplicationBegin,
ApplicationContinue,
ApplicationAbort,
ApplicationEnd;

SIGNALLIST CS3_INAP_CS_IN =
(CS1_INAP_CS_IN),
AuthorizeTermination,
ContinueWithArgument,
DisconnectForwardConnectionWithArgument,
DisconnectLeg,
Reconnect,
RequestReportFacilityEvent,
RequestReportUTSI,
SendSTUI,
SendFacilityInformation,

PromptAndReceiveMessage,
ScriptClose,
ScriptInformation,
ScriptRun,
SetServiceProfile;

SIGNALLIST CS3_INAP_CS_OUT =
(CS1_INAP_CS_OUT),
DisconnectLegResult,
EntityReleased,
EventReportFacility,
FacilitySelectedAndAvailable,
OriginationAttempt,
OSuspended,
ReportUTSI,
SendFacilityInformation,
TerminationAttempt,
TSuspended,

ScriptEvent,
PromptAndReceiveMessageResult;

Block Type SSF_CCF

11(12)



/* ** SIGNAL LIST DEFINITIONS ** */

/* INAP OPERATIONS ON THE CS/SSF_FSM INTERFACE */

SIGNALLIST CS1_INAP_SSF_In =

AnalyseInformation,
ApplyCharging,
CallInformationRequest,
Cancel,
CollectInformation,
Connect,
ConnectToResource,
Continue,
DisconnectForwardConnection,
EstablishTemporaryConnection,
FurnishChargingInformation,
HoldCallInNetwork,
InitiateCallAttempt,
ReleaseCall,
RequestNotificationChargingEvent,
RequestReportBCSMEvent,
ResetTimer,
SelectFacility,
SelectRoute,
SendChargingInformation,

PlayAnnouncement,
PromptAndCollectUserInformation,

ApplicationBegin,
ApplicationContinue,
ApplicationAbort,
ApplicationEnd;

SIGNALLIST CS1_INAP_SSF_Out =

AnalysedInformation,
ApplyChargingReport,
AssistRequestInstructions,
CallInformationReport,
CollectedInformation,
EventNotificationCharging,
EventReportBCSM,
InitialDP,
OAbandon,
OAnswer,
OCalledPartyBusy,
ODisconnect,
OMidCall,
ONoAnswer,
OriginationAttemptAuthorized,
RouteSelectFailure,
TAnswer,
TBusy,
TDisconnect,
TermAttemptAuthorized,
TMidCall,
TNoAnswer,

PromptAndCollectUserInformationResult,
SpecializedResourceReport,

Error,
ApplicationBegin,
ApplicationContinue,
ApplicationAbort,
ApplicationEnd;

SIGNALLIST CS3_INAP_SSF_IN =

(CS1_INAP_SSF_IN),
AuthorizeTermination,
ContinueWithArgument,
DisconnectForwardConnectionWithArgument,
DisconnectLeg,
Reconnect,
RequestReportFacilityEvent,
RequestReportUTSI,
SendSTUI,
SendFacilityInformation,

PromptAndReceiveMessage,
ScriptClose,
ScriptInformation,
ScriptRun,
SetServiceProfile;

SIGNALLIST CS3_INAP_SSF_OUT =

(CS1_INAP_SSF_OUT),
DisconnectLegResult,
EntityReleased,
EventReportFacility,
FacilitySelectedAndAvailable,
OriginationAttempt,
OSuspended,
ReportUTSI,
TerminationAttempt,
TSuspended,

ScriptEvent,
PromptAndReceiveMessageResult;

Block Type SSF_CCF

12(12)

**** SIGNAL LIST DEFINITIONS ****

SIGNALLIST CSA_In =
(CS3_INAP_CSA_In),
(SigCon_In),
(IBI),
(SSME_Resps),
ExportCSReq,
ImportCSReq,
ActivityTest;

SIGNALLIST CSA_Out =
(CS3_INAP_CSA_Out),
(SigCon_Out),
(IBI),
(SSME_Reqs),
ExportCSResp,
ImportCSResp,
ActivityTestResult;

SIGNALLIST CS_In =
(CS3_INAP_CS_In),
(SigCon_In),
(IBI),
(SSME_Resps),
CSStop,
CSExportLegReq,
CSImportLegReq,
SetCSA,
SetLegID;

SIGNALLIST CS_Out =
(CS3_INAP_CS_Out),
(SigCon_Out),
(IBI),
(SSME_Reqs),
CSStop,
CSExportLegResp,
CSImportLegResp;

SIGNALLIST CS3_SSF_In =
(DPs),
(CS3_INAP_SSF_In),
(SSME_Resps),
SSFStop,
ExportLegReq,
ImportLegReq,
SetLegID,
DPUTSI,
DPFacility;

SIGNALLIST CS3_SSF_Out =
(PICs),
(CS3_INAP_SSF_Out),
(SSME_Reqs),
ExportLegResp,
ImportLegResp;

SIGNALLIST O_BCSM_In =
(oPICs),
(O_SigCon_In),
(IBI_In),
BCSMStop,
NetworkSuspendReqInd,
NetworkResumeReqInd,
DataReqInd,
DataInd,
SetCS,
SetLegID;

SIGNALLIST O_BCSM_Out =
(oDPs),
(O_SigCon_Out),
(IBI_Out),
BCSMStop,
NetworkSuspendReq,
NetworkResumeReq,
DataReqInd,
DataReq,
DPUTSI,
DPFacility;

SIGNALLIST T_BCSM_In =
(tPICs),
(T_SigCon_In),
(IBI_Out),
BCSMStop,
NetworkSuspendInd,
NetworkResumeInd,
DataInd,
DataReqInd,
SetCS,
SetLegID;

SIGNALLIST T_BCSM_Out =
(tDPs),
(T_SigCon_Out),
(IBI_In),
BCSMStop,
NetworkSuspendReqInd,
NetworkResumeReqInd,
DataReq,
DataReqInd,
DPUTSI,
DPFacility;

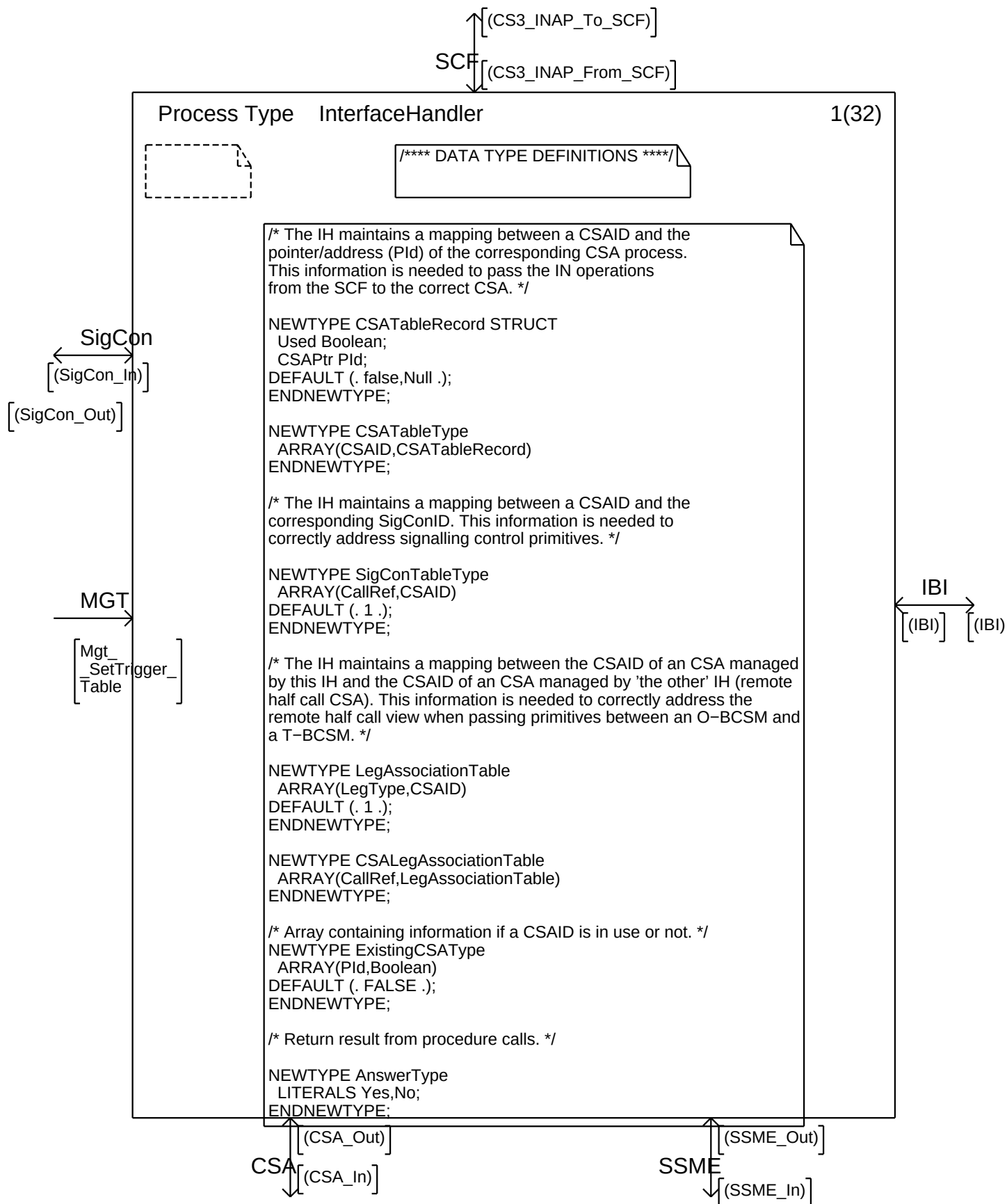
SIGNALLIST CS3_INAP_SSME_IN =
CreateOrRemoveTriggerData,
ManageTriggerData;

SIGNALLIST CS3_INAP_SSME_OUT =
CreateOrRemoveTriggerDataResult,
ManageTriggerDataResult;

SIGNALLIST SSME_In =
ActivateServiceFiltering,
CallGap,
CancelStatusReportRequest,
RequestCurrentStatusReport,
RequestEveryStatusChangeReport,
RequestFirstStatusMatchReport,
(CS3_INAP_SSME_IN),
(SSME_Reqs),

Mgt_SetTriggerTable;

SIGNALLIST SSME_Out =
ServiceFilteringResponse,
StatusReport,
(CS3_INAP_SSME_OUT),
(SSME_Resps);



Process Type InterfaceHandler

2(32)



**** VARIABLE DECLARATIONS ****

DCL

```
/* Routing tables. */
csaTable CSATableType,
sigConTable SigConTableType,
csaLegAssoc CSALegAssociationTable,

/* CSA info. */
existingCSA ExistingCSAType,

/* Other variables */
csa PId,
csaID,newCSAID CSAID,
invokeID InvokeID,
i Integer,
termination Boolean,
csID CallSegmentID,
legID LegType,
halfCallSide IHRoleType,
dialogID DialogIDType,
callFlag CallFlag,
eventTable EventTableType,
r Boolean,
dID DialogIDType;
```

DCL

```
/* Signalling control primitive parameters. */
aeArg AddressEndType,
cpArg CallProgressType,
fArg FailureType,
rArg ReleaseType,
sftArg ServiceFeatureType,
sirArg SetupIRType,
scrArg SetupCRType,
saArg SubsequentAddressType;
```

DCL

```
/* IN operation arguments. */
asfArg ActivateServiceFilteringArg,
aiArg AnalyseInformationArg,
adiArg AnalysedInformationArg,
acArg ApplyChargingArg,
acrArg ApplyChargingReportArg,
ariArg AssistRequestInstructionsArg,
cgArg CallGapArg,
cirArg CallInformationReportArg,
cirqArg CallInformationRequestArg,
cArg CancelArg,
ciArg CollectInformationArg,
cdiArg CollectedInformationArg,
coArg ConnectArg,
ctrArg ConnectToResourceArg,
etcArg EstablishTemporaryConnectionArg,
encArg EventNotificationChargingArg,
erBCSMArg EventReportBCSMArg,
fciArg FurnishChargingInformationArg,
hcinArg HoldCallInNetworkArg,
idpArg InitialDPArg,
icaArg InitiateCallAttemptArg,
oabnArg OAbandonArg,
oansArg OAnswerArg,
ocpbArg OCalledPartyBusyArg,
odisArg ODisconnectArg,
midclArg MidCallArg,
onoansArg ONoAnswerArg,
oaaArg OriginationAttemptAuthorizedArg,
rcArg ReleaseCallArg,
rcsrArg RequestCurrentStatusReportArg,
rescrArg RequestEveryStatusChangeReportArg,
rfsmrArg RequestFirstStatusMatchReportArg,
rnceArg RequestNotificationChargingEventArg,
rrBCSMEArg RequestReportBCSMEEventArg,
rtArg ResetTimerArg,
rsfArg RouteSelectFailureArg,
sciArg SendChargingInformationArg,
sfArg SelectFacilityArg,
srArg SelectRouteArg,
sfrArg ServiceFilteringResponseArg,
srepArg StatusReportArg,
tansArg TAnswerArg,
tbusArg TBusyArg,
tdisArg TDisconnectArg,
taaArg TermAttemptAuthorizedArg,
tnoansArg TNoAnswerArg,

ieArg ErrorArg,

mgt_STTArg MGTSetTriggerTableArg;
```


Process Type InterfaceHandler

3(32)



```
/**** VARIABLE DECLARATIONS ****/
```

```
DCL
/* IN CS-3 operation arguments. */
autArg AuthorizeTerminationArg,
ccsaArg CreateCallSegmentAssociationArg,
ccsarArg CreateCallSegmentAssociationResultArg,
ctwaArg ContinueWithArgumentArg,
ctdArg CreateOrRemoveTriggerDataArg,
ctdrArg CreateOrRemoveTriggerDataResultArg,
dfcwaArg DisconnectForwardConnectionWithArgumentArg,
dlArg DisconnectLegArg,
erArg EntityReleasedArg,
erfArg EventReportFacilityArg,
fsaArg FacilitySelectedAndAvailableArg,
mtdArg ManageTriggerDataArg,
mtdrArg ManageTriggerDataResultArg,
mcsArg MergeCallSegmentsArg,
mocsArg MoveCallSegmentsArg,
mlArg MoveLegArg,
oaArg OriginationAttemptArg,
osusArg OSuspendedArg,
reconArg ReconnectArg,
rutsiArg ReportUTSIArg,
rrutsiArg RequestReportUTSIArg,
rrfeArg RequestReportFacilityEventArg,
sendfArg SendFacilityInformationArg,
sstuiArg SendSTUIArg,
sspArg SetServiceProfileArg,
slArg SplitLegArg,
taArg TerminationAttemptArg,
tsusArg TSuspendedArg
;
```

```
DCL
/* Signalling control primitive parameters. */
nsrArg NetworkSRTYPE,
udArg UserDataType;
```

Process Type InterfaceHandler

4(32)



/* DECLARATION OF OPERATIONS ON THE ROUTING TABLES */

Allocate
CSAId Allocate a new CSA id

Add_
CSAId Add a CSA id

GetCSA
Idfrom_
CSA Retrieve the CSA id
from a CSA pointer.

GetCSA
fromCSA
Id Retrieve the pointer
to the CSA instance
from an CSA id.

GetCSA
fromSig
ConId Retrieve the pointer
to the CSA instance
from a CallRef.

SetSig_
ConAssoc Associates a CallRef with
a pointer (PIId) to a CSA.

SetRemote
Assoc Associate a remote party (identified
by the remote CSA PIId and the remote
leg id) with a local CSA.

GetCSA
from_
RemoteId Retrieves the CSA pointer (PIId) from
a remote party (identified by the
remote CSA PIId and the remote leg id)
with a local CSA.

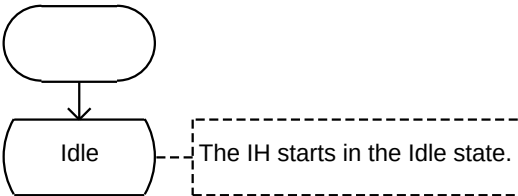
IsCSA Determines if a pointer
is a pointer to a CSA
or not.

/* Procedure for processing of the operation
MoveCallSegments. */

Process_
MoveCall_
Segments

Process Type InterfaceHandler

5(32)

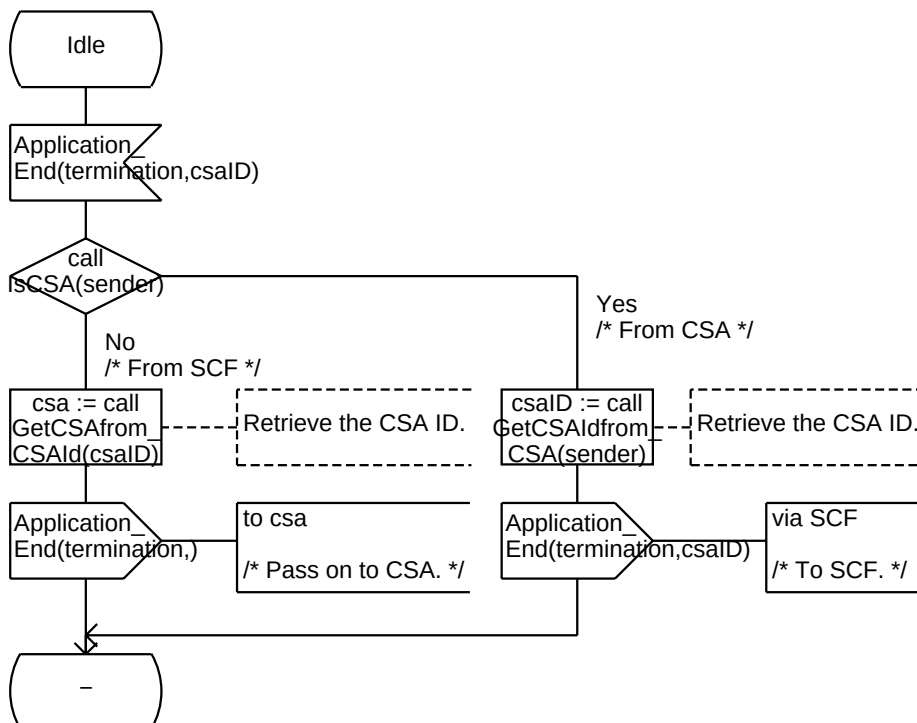
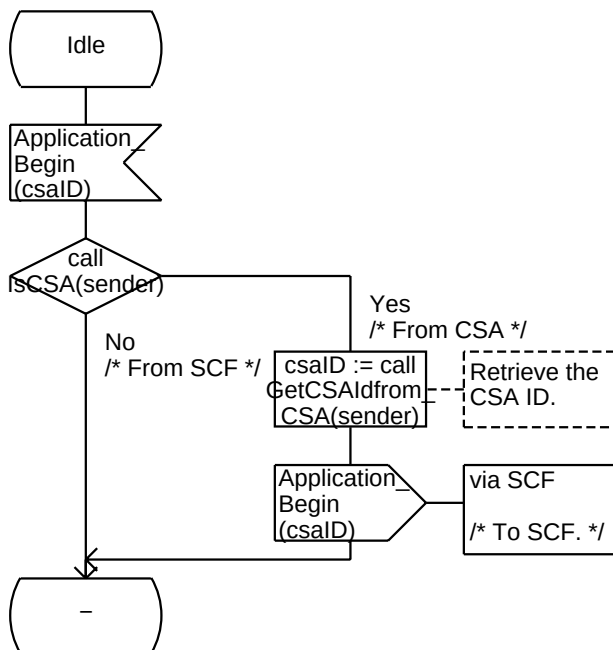


Process Type InterfaceHandler

6(32)

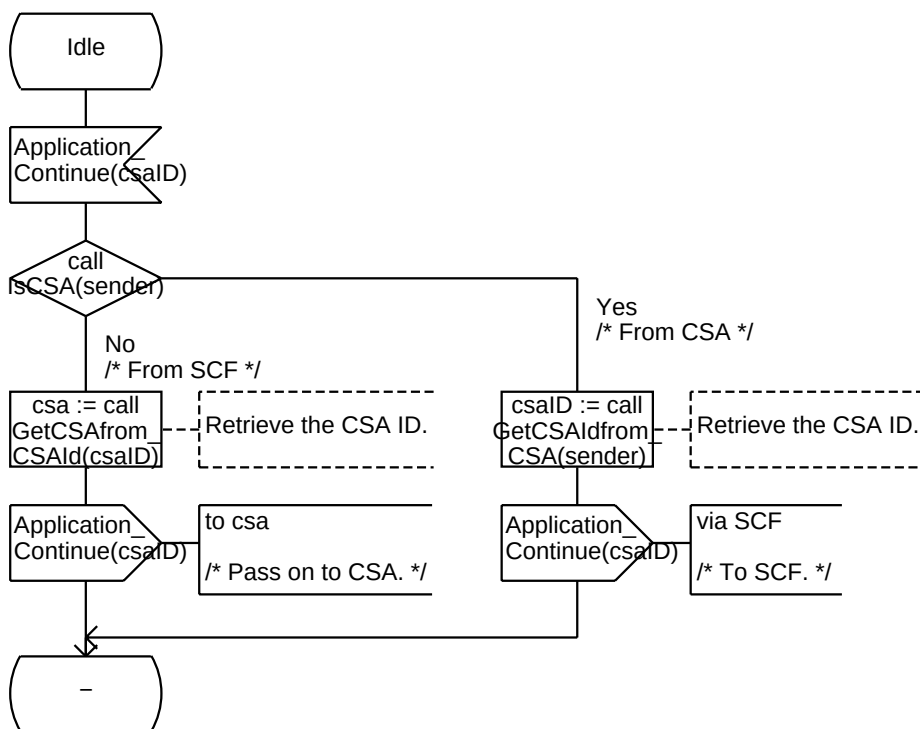
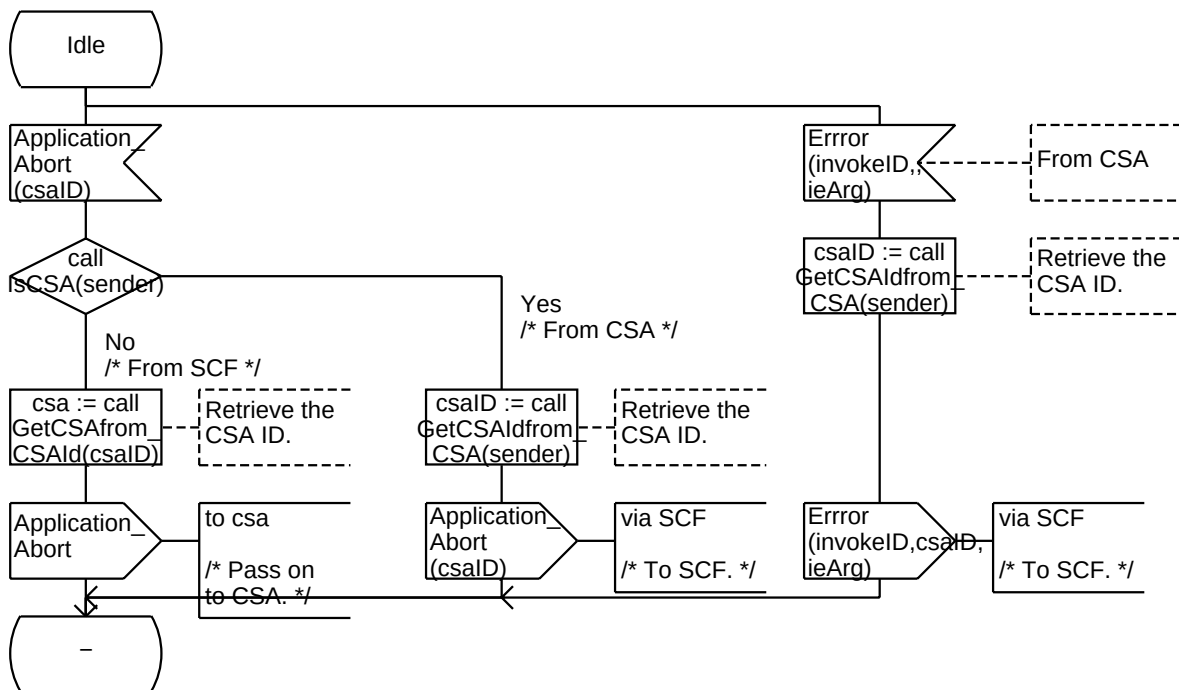


/* Handling of the opening and closing of dialogues with the TCAP. */



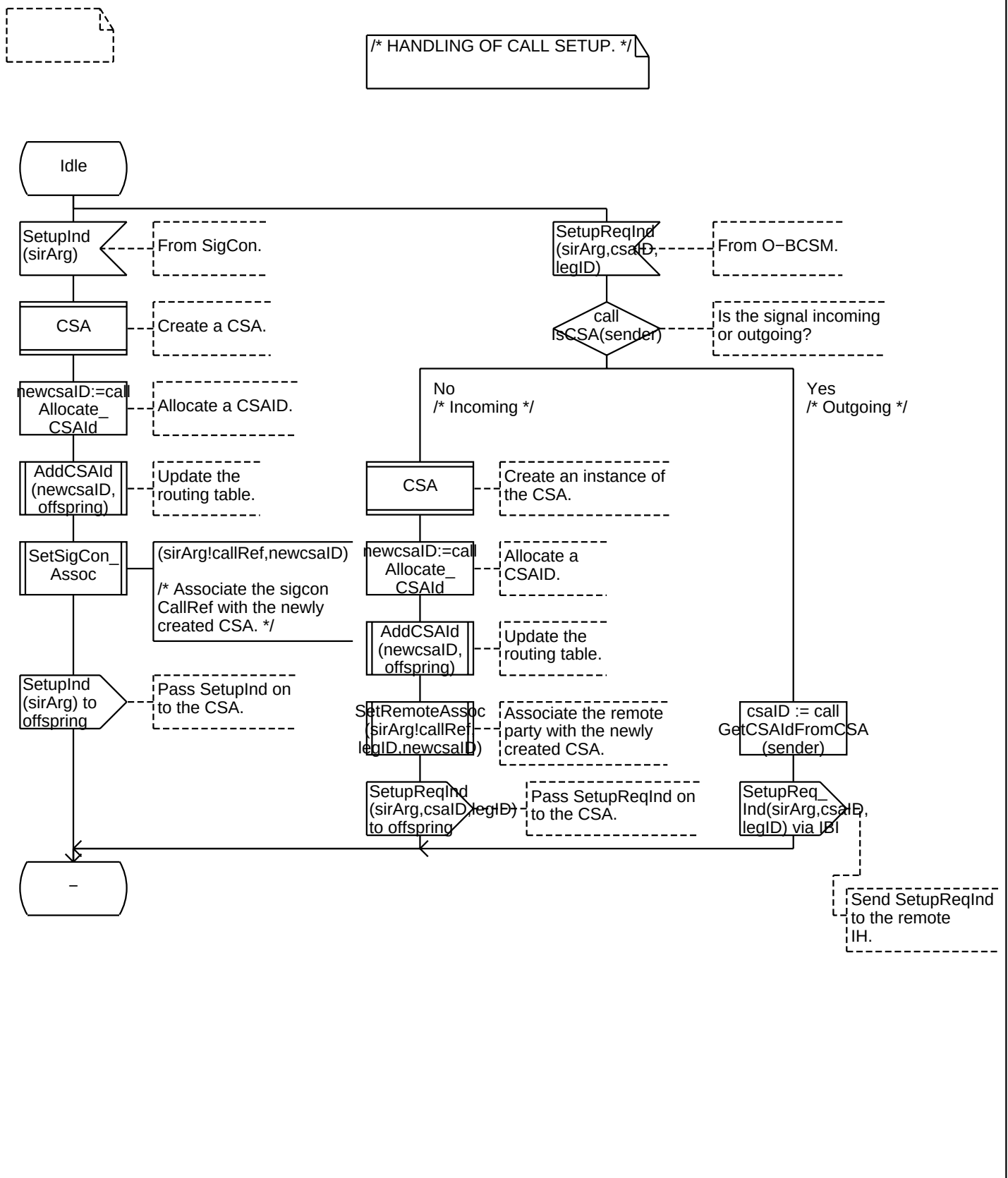
Process Type InterfaceHandler

7(32)



Process Type InterfaceHandler

8(32)

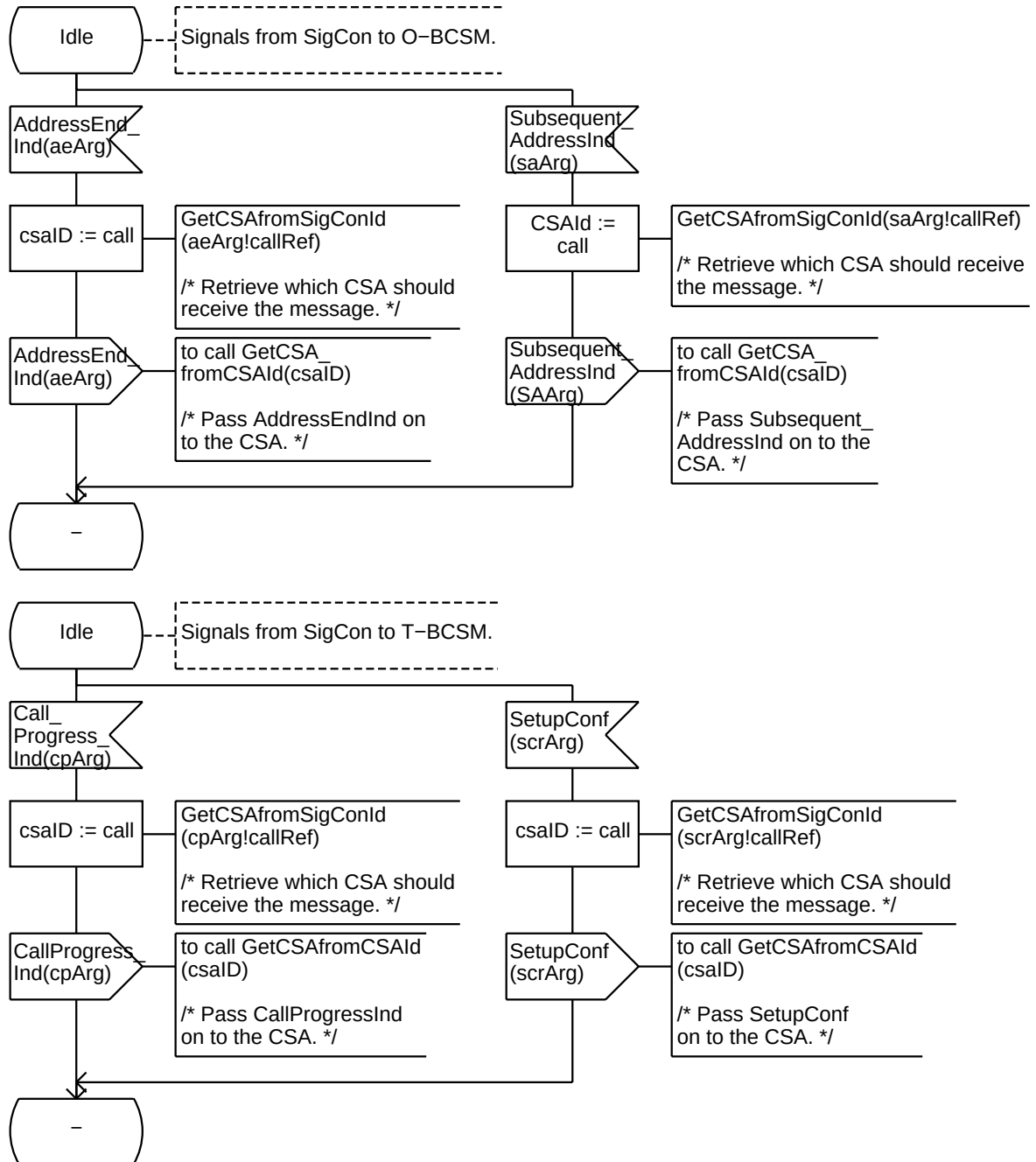


Process Type InterfaceHandler

9(32)

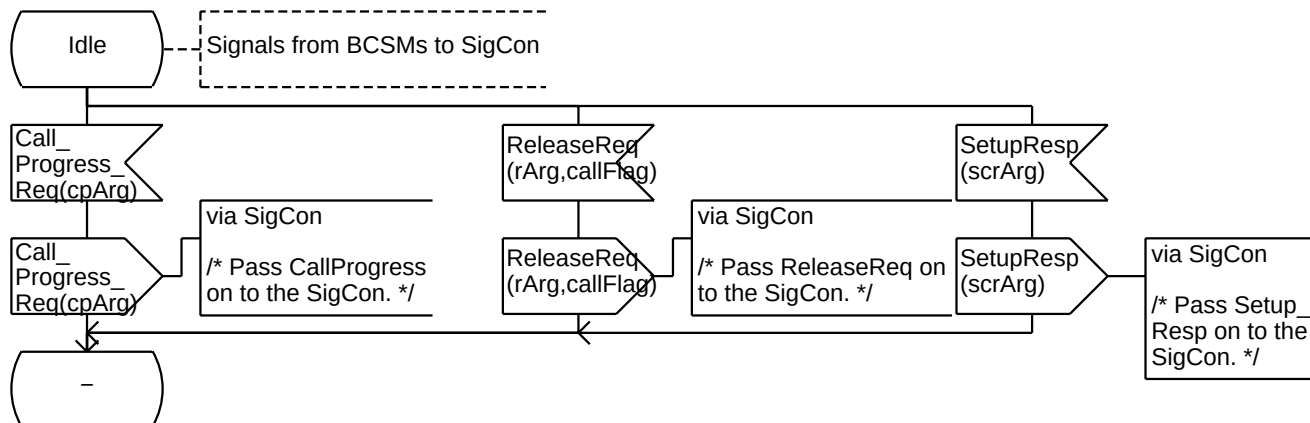
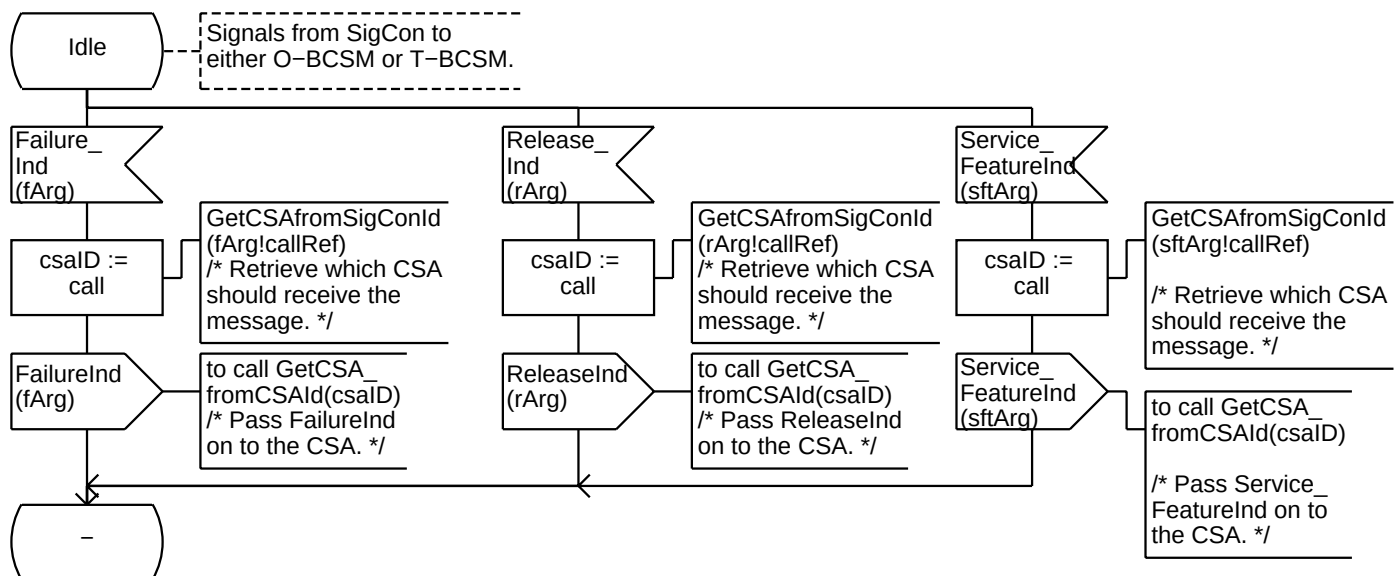


/* ROUTING OF SIGNALS TO/FROM SIGNALLING CONTROL. */



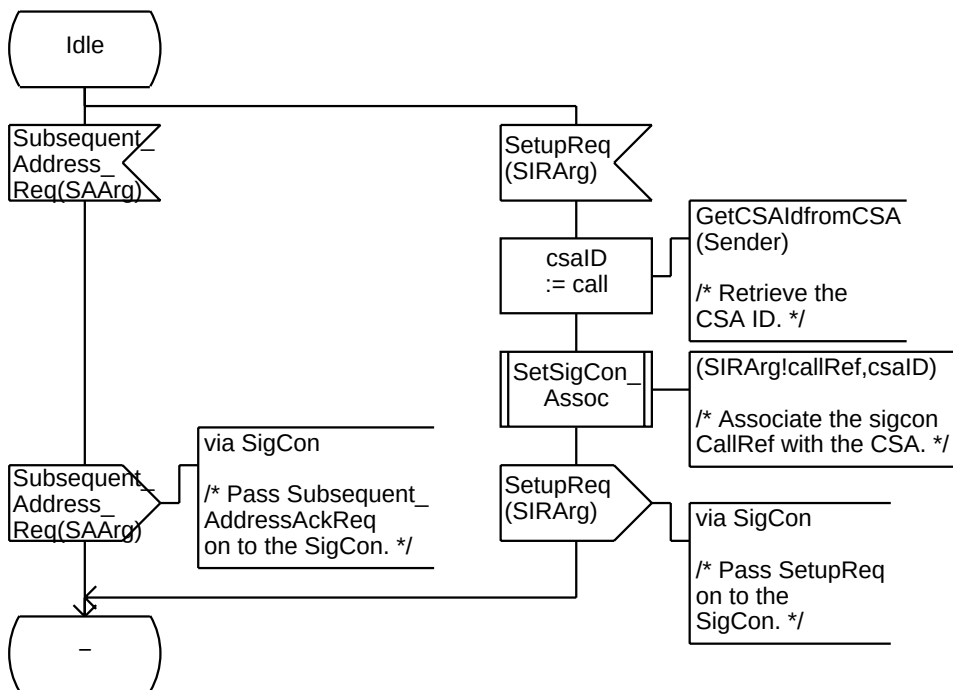
Process Type InterfaceHandler

10(32)

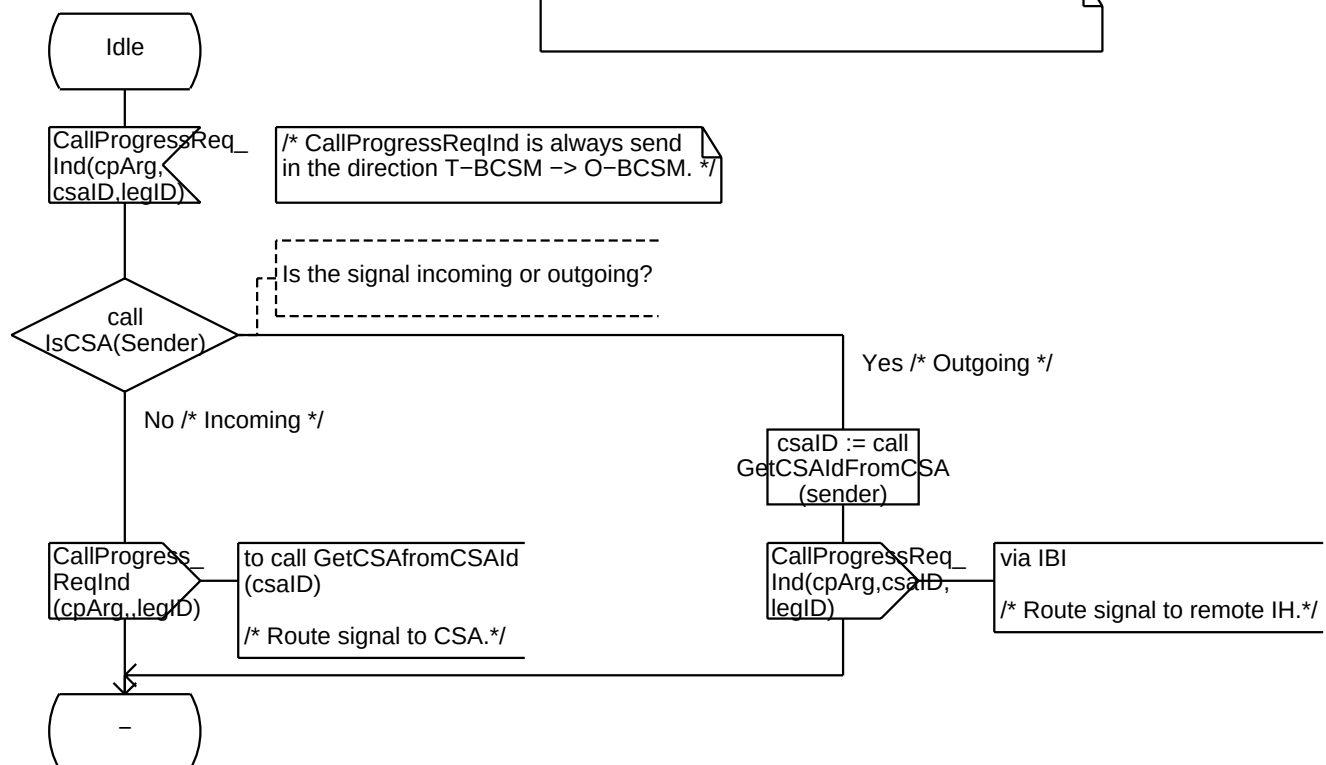


Process Type InterfaceHandler

11(32)

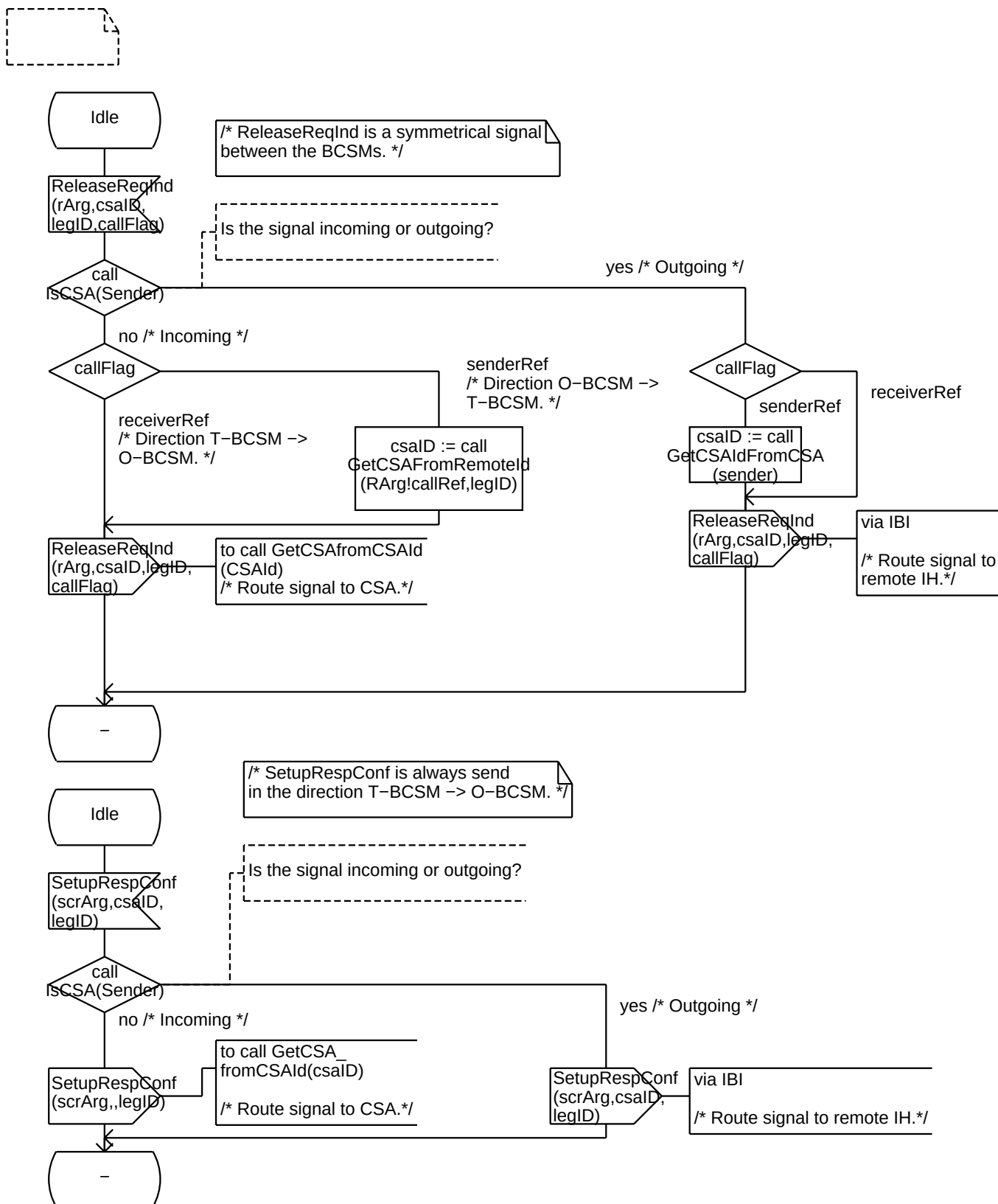


/* ROUTING OF SIGNALS BETWEEN BCSMS. */



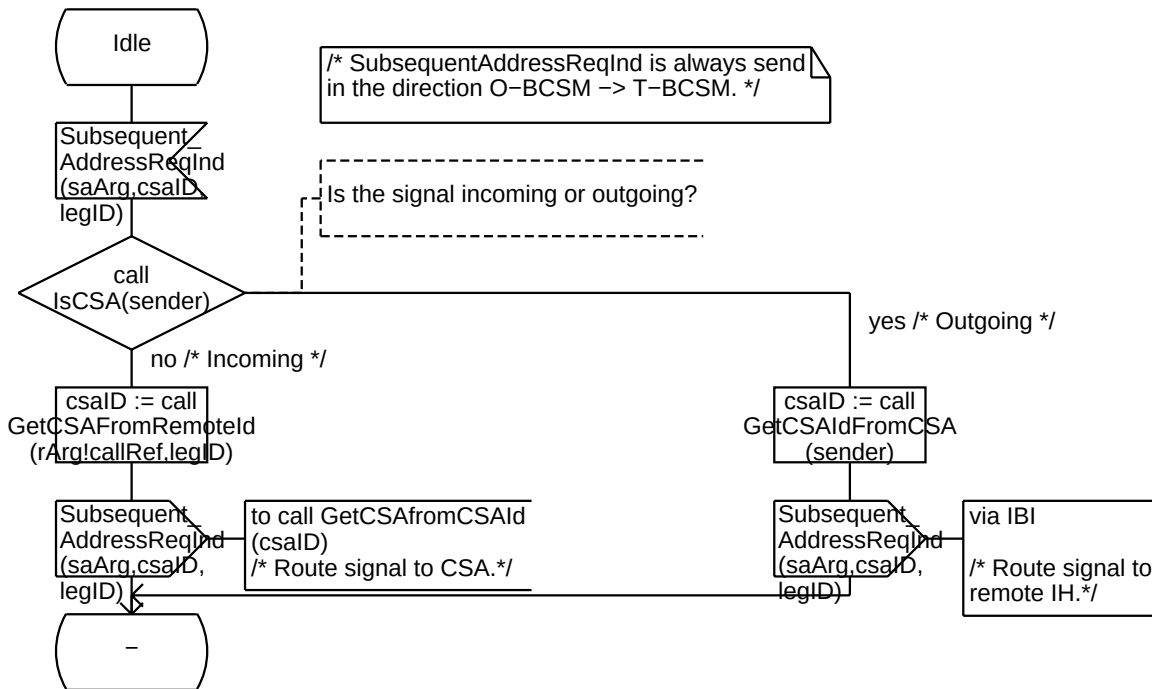
Process Type InterfaceHandler

12(32)



Process Type InterfaceHandler

13(32)

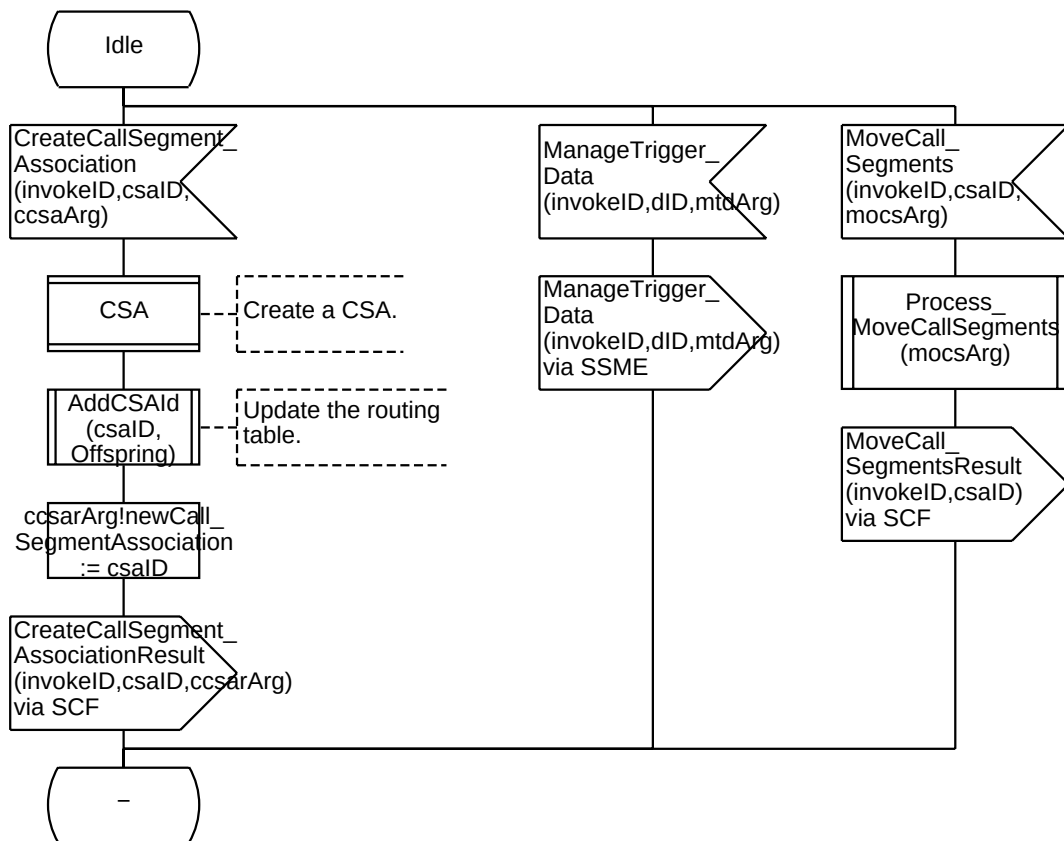


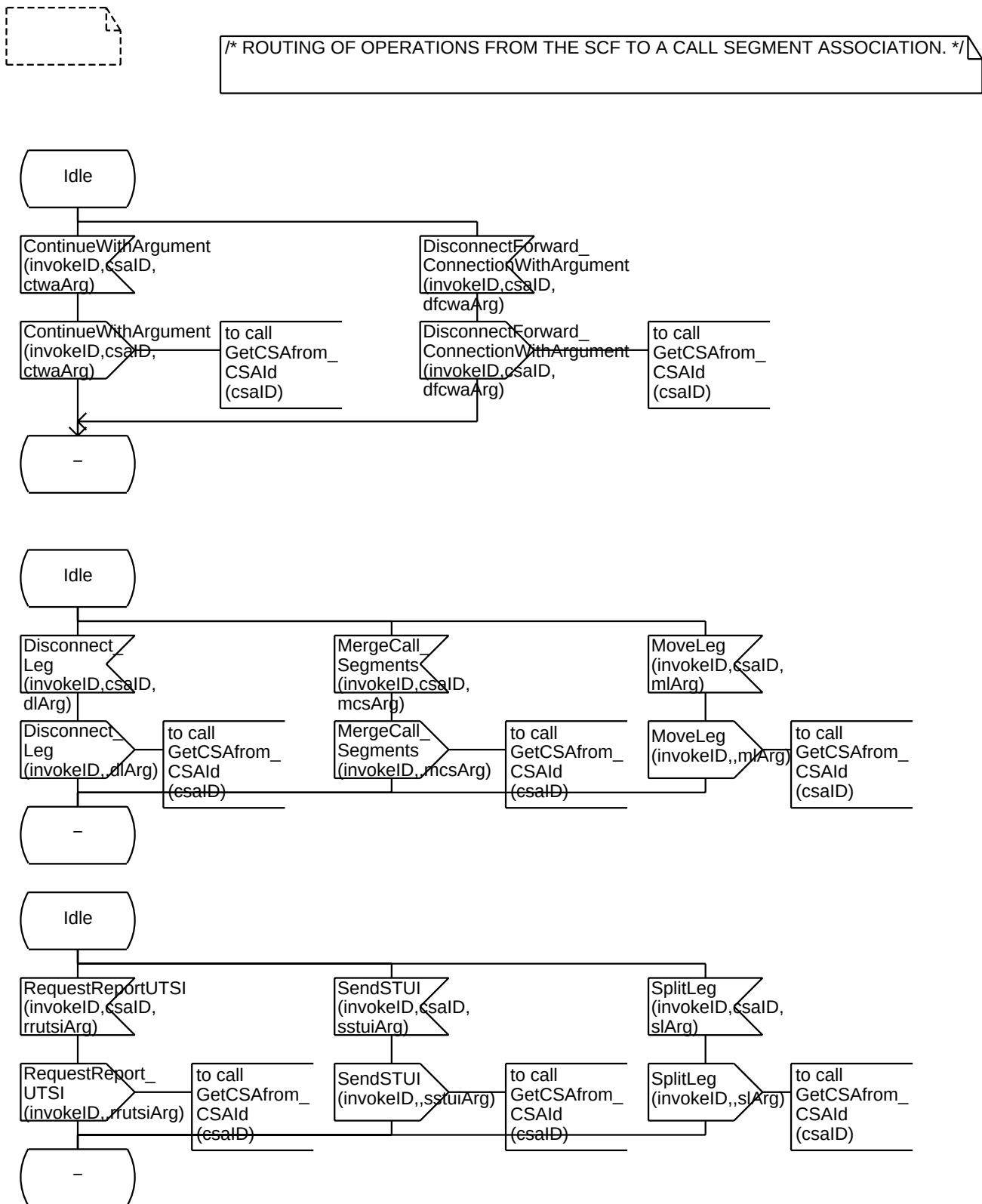
Process Type InterfaceHandler

14(32)



/**** PROCESSING OF INTER-CSA OPERATIONS AND NON-CSA RELATED OPERATIONS. *****/



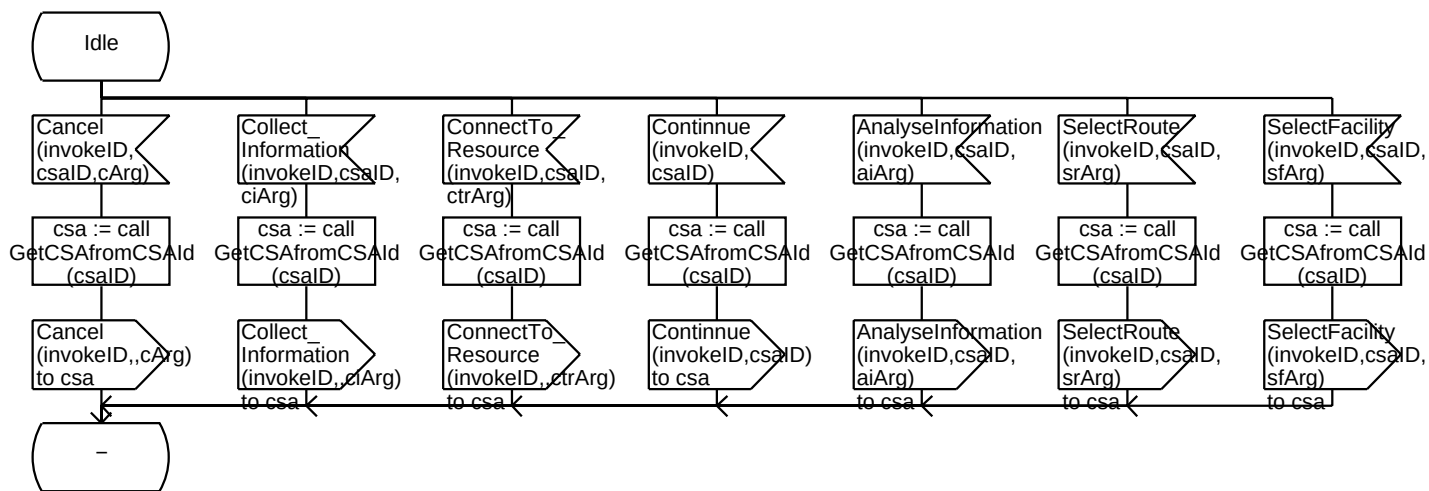
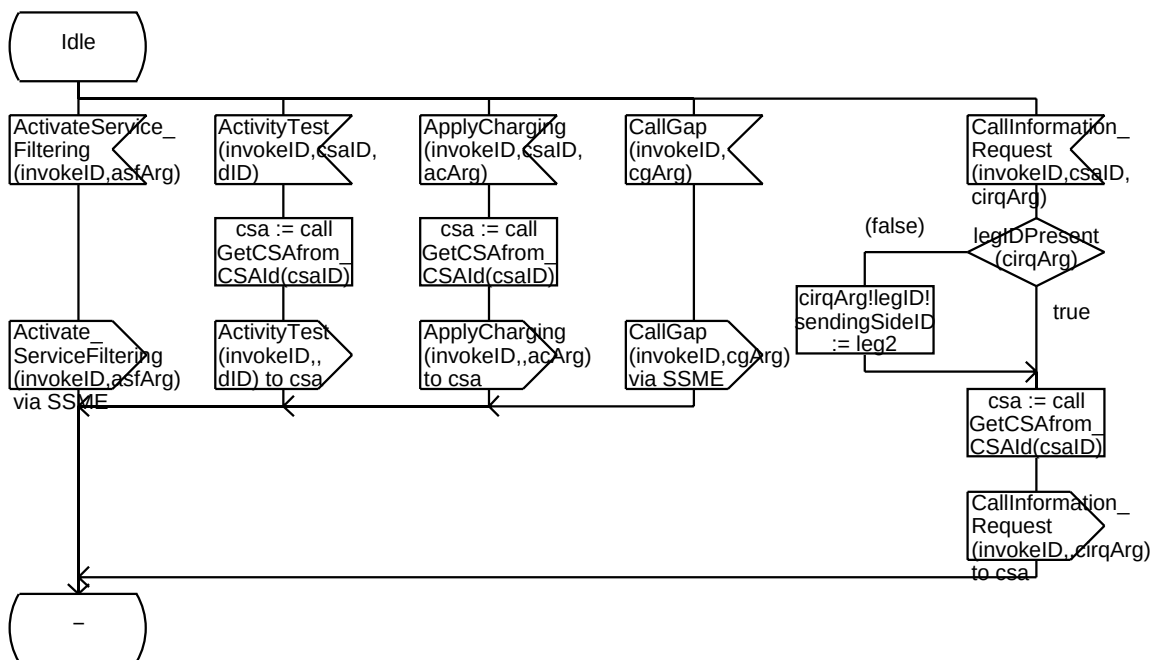


Process Type InterfaceHandler

16(32)

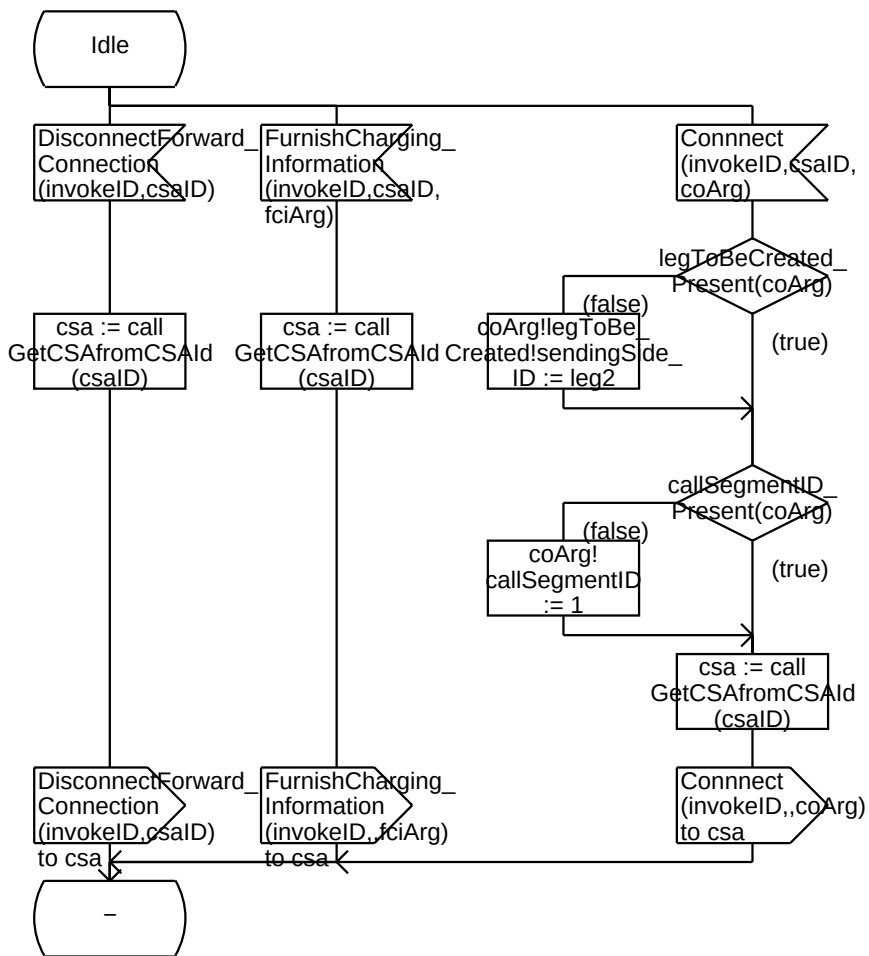


/* ROUTING OF INAP OPERATIONS FROM SCF TO SSF. */



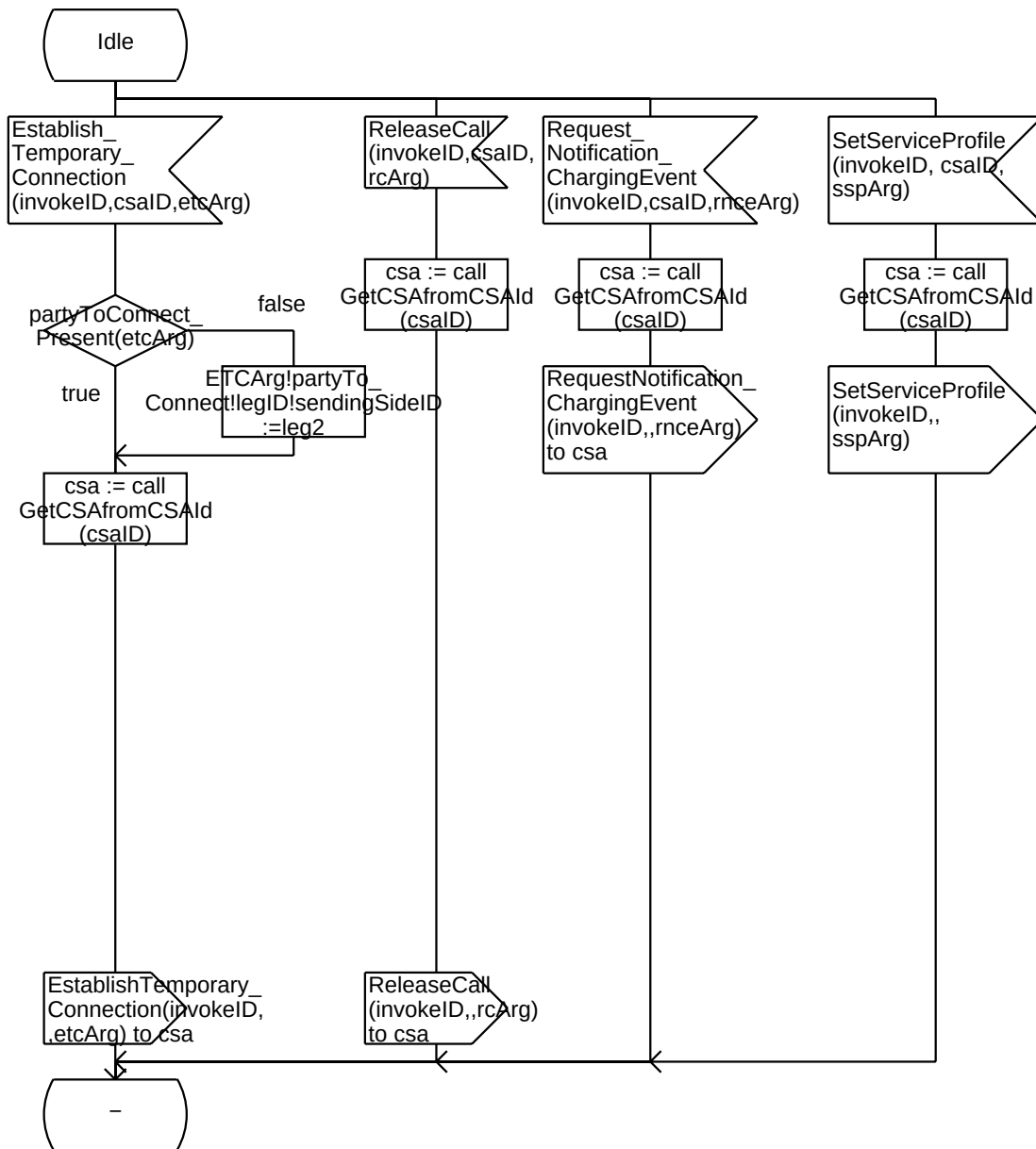
Process Type InterfaceHandler

17(32)



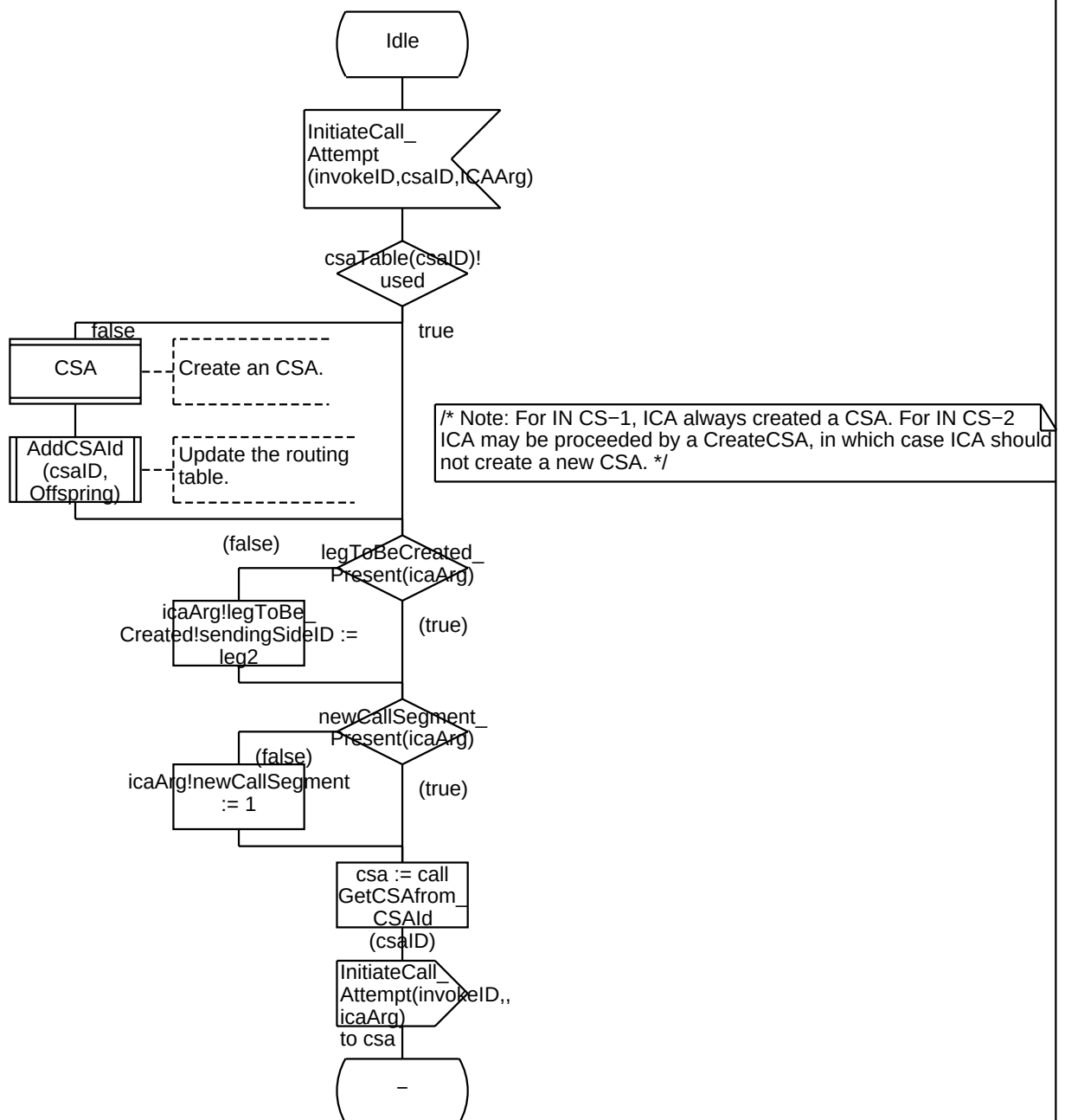
Process Type InterfaceHandler

18(32)



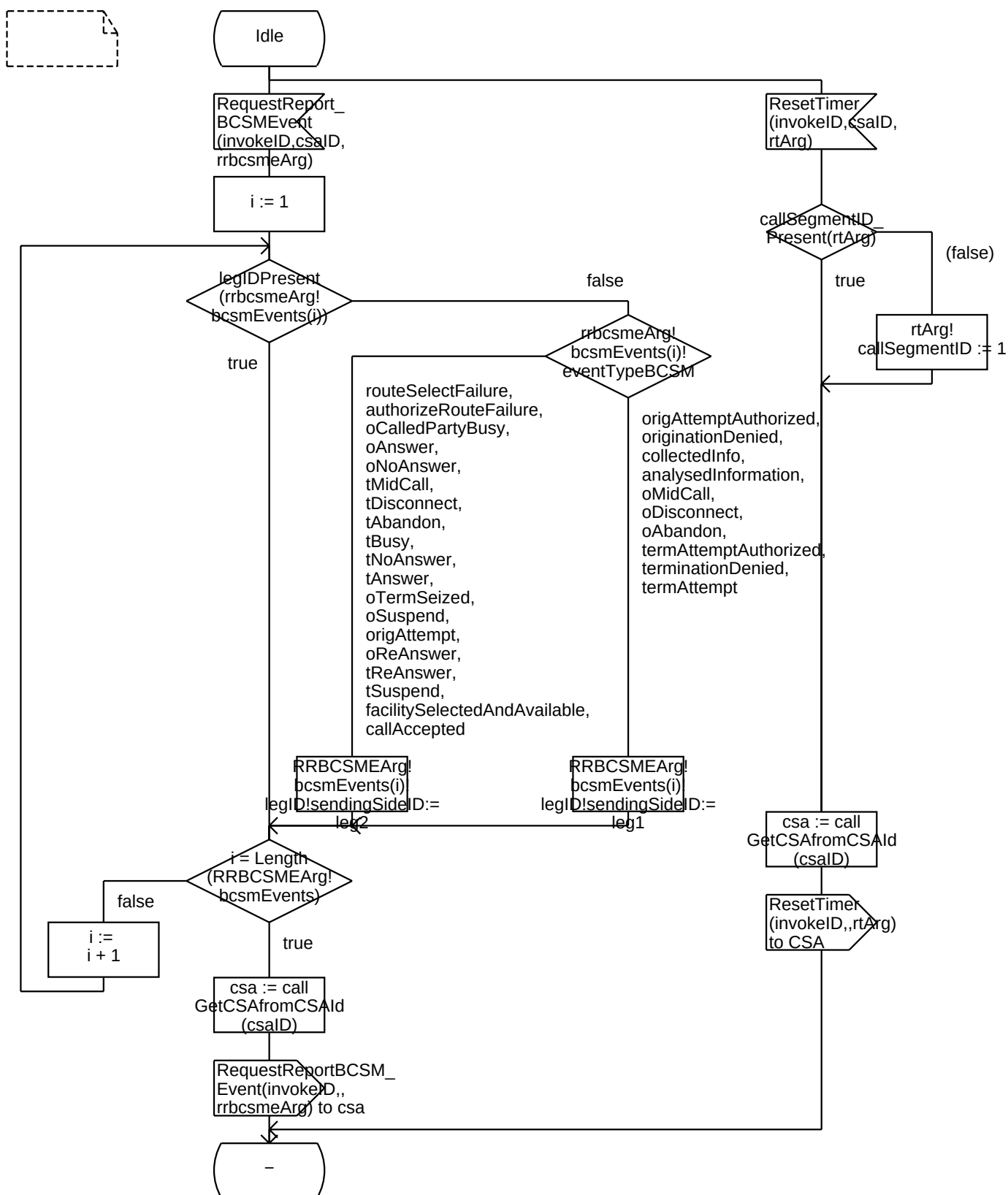
Process Type InterfaceHandler

19(32)



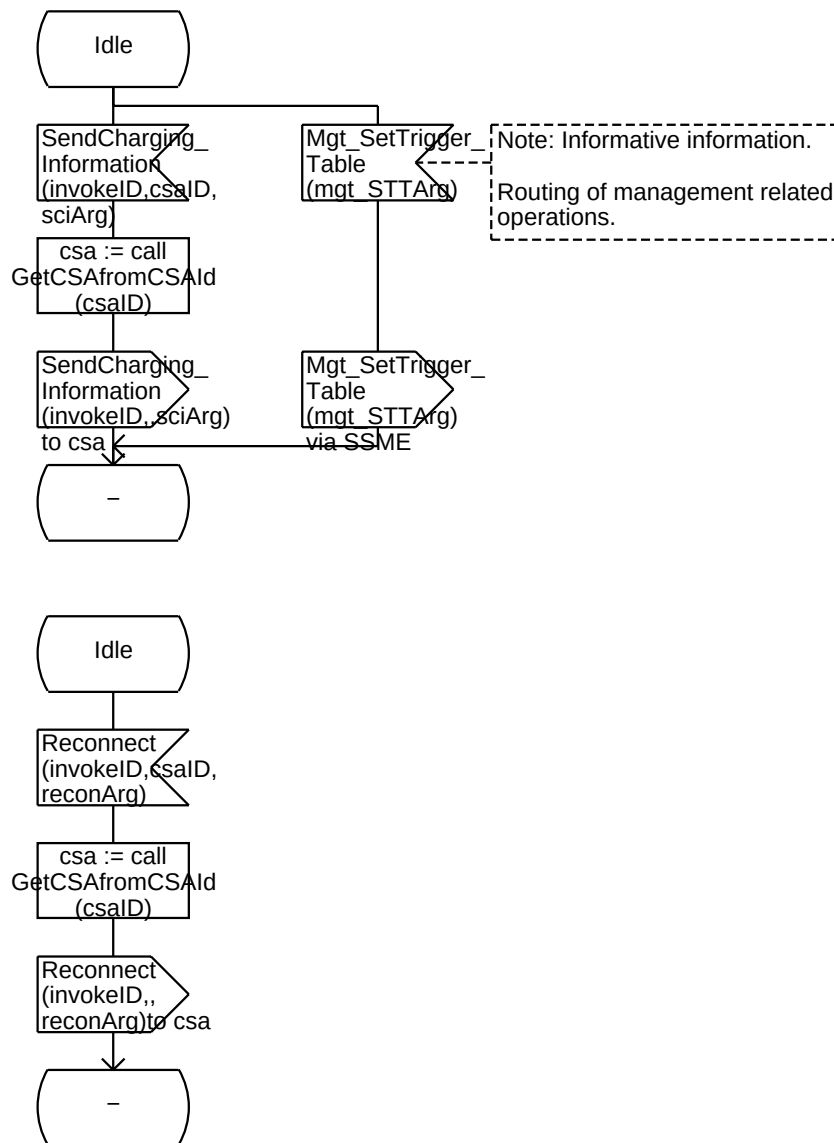
Process Type InterfaceHandler

20(32)



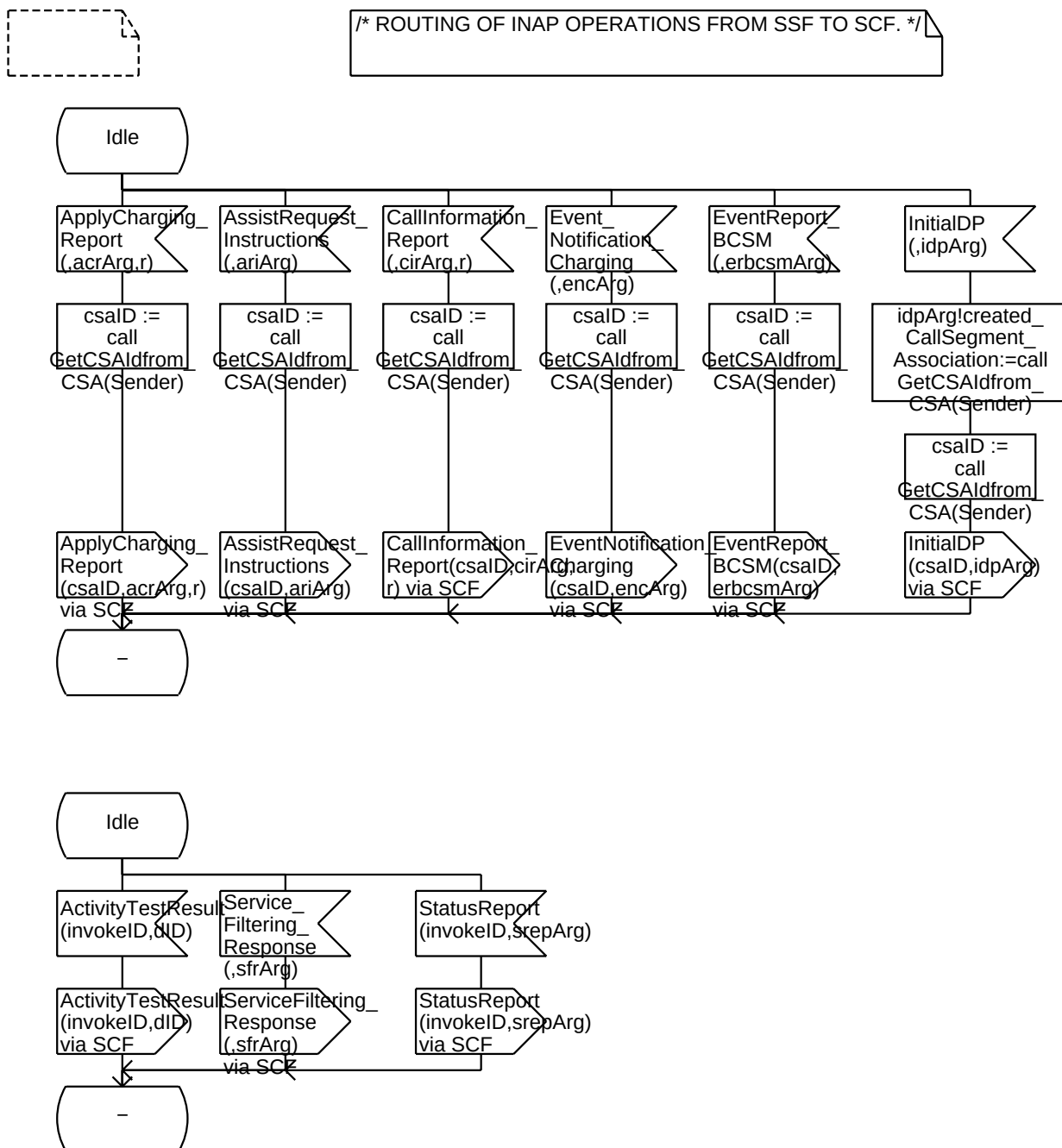
Process Type InterfaceHandler

21(32)



Process Type InterfaceHandler

22(32)

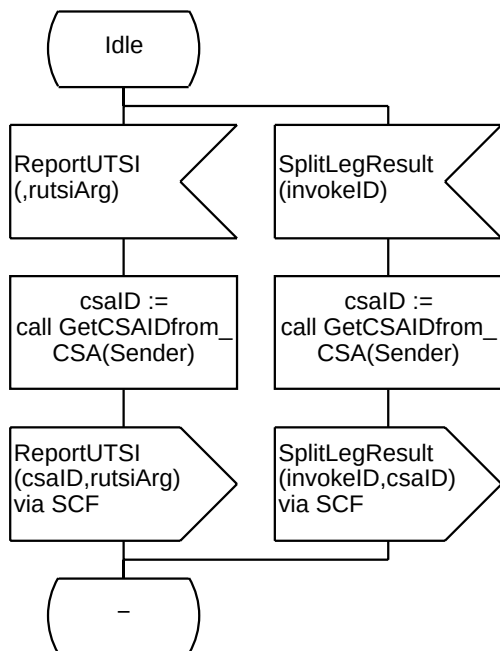
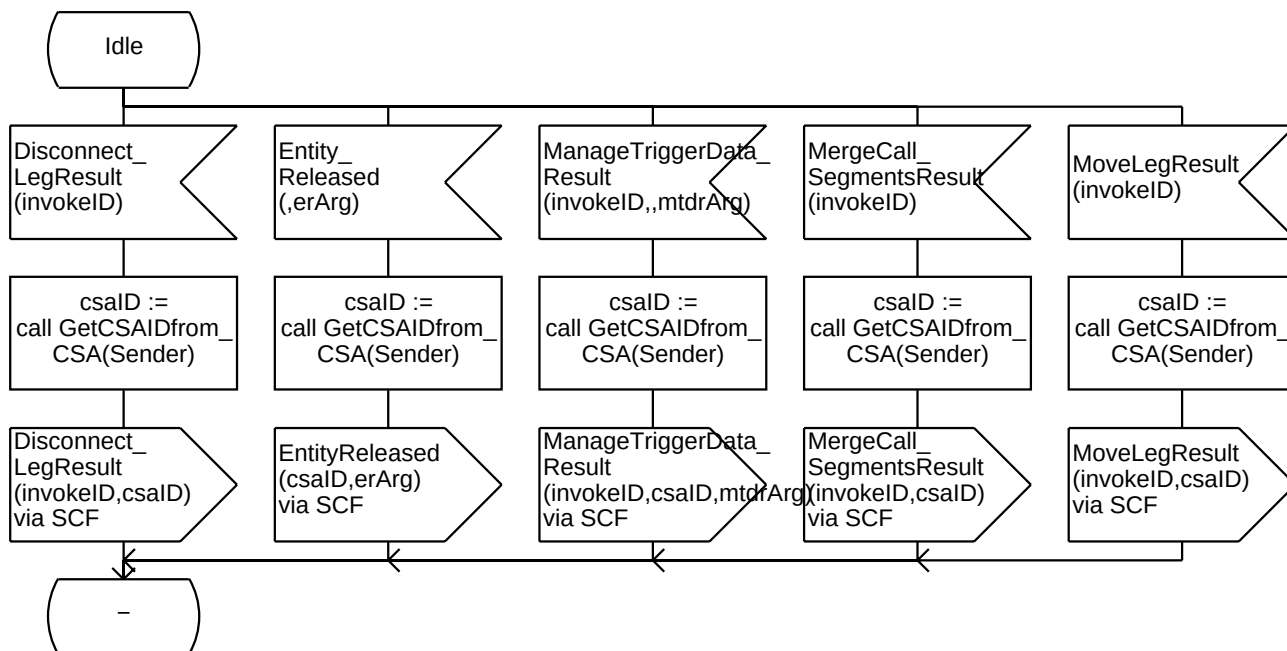


Process Type InterfaceHandler

23(32)



/* ROUTING OF INAP OPERATIONS FROM SSF TO SCF. */

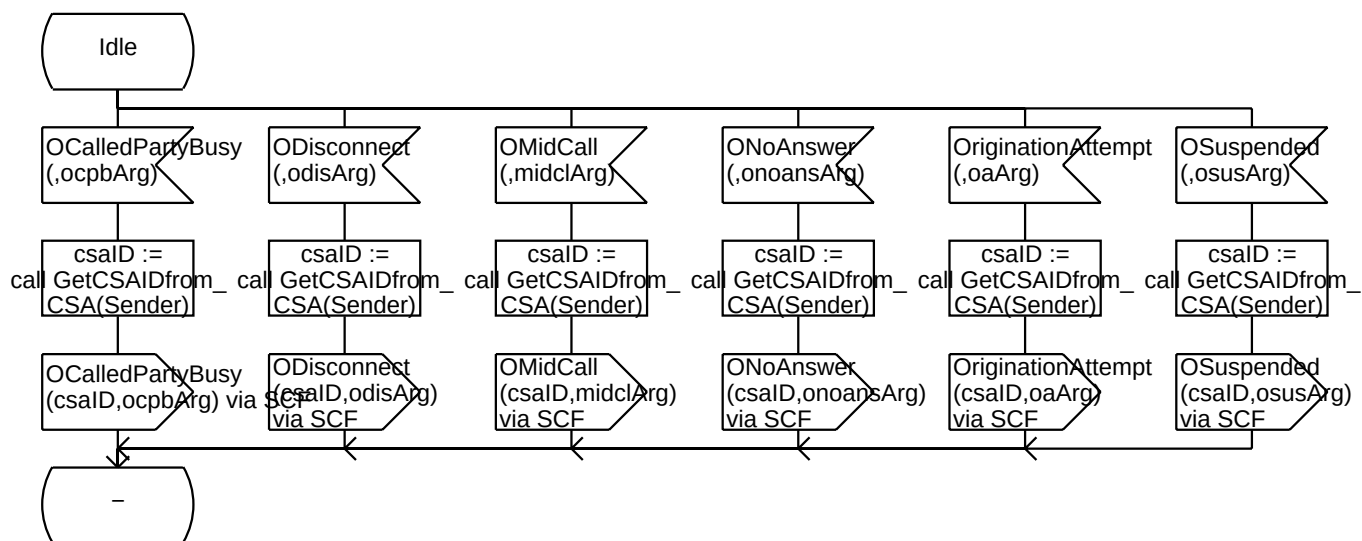
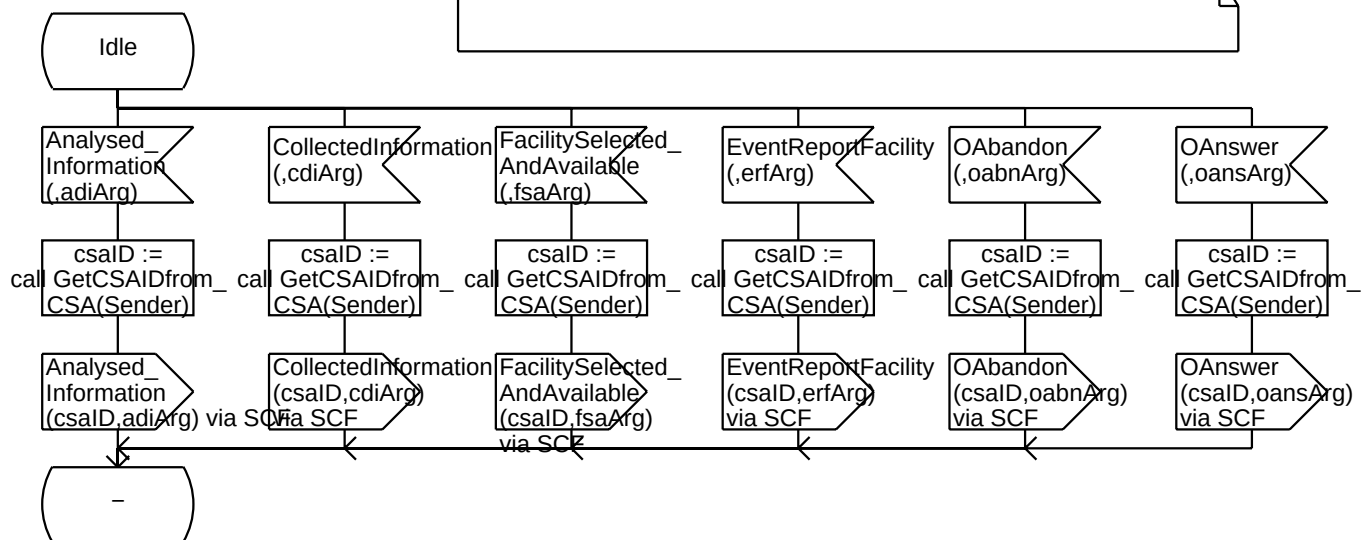


Process Type InterfaceHandler

24(32)



/* ROUTING OF DP SPECIFIC OPERATIONS FROM SSF TO SCF. */

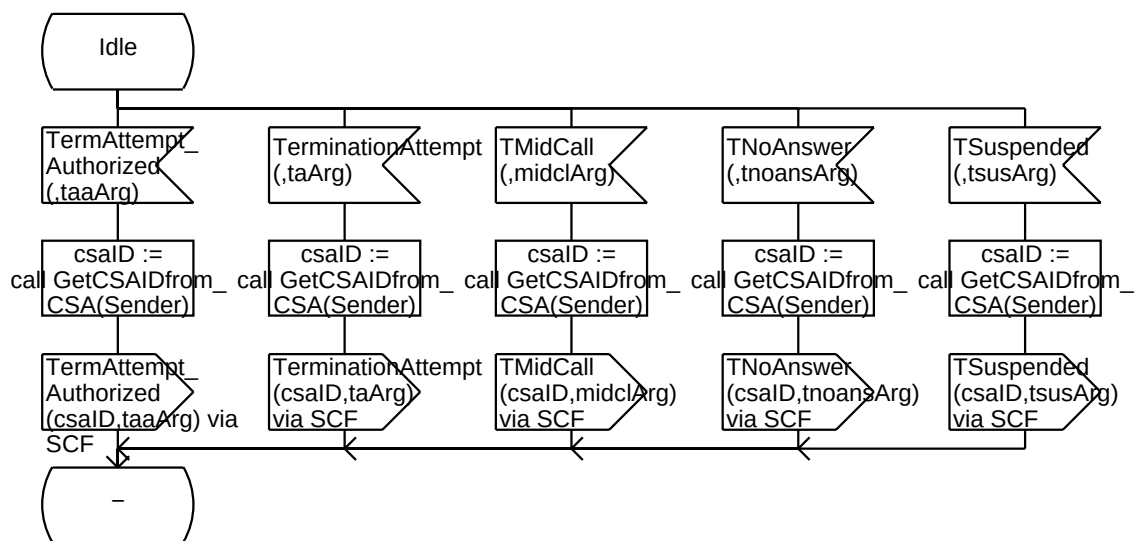
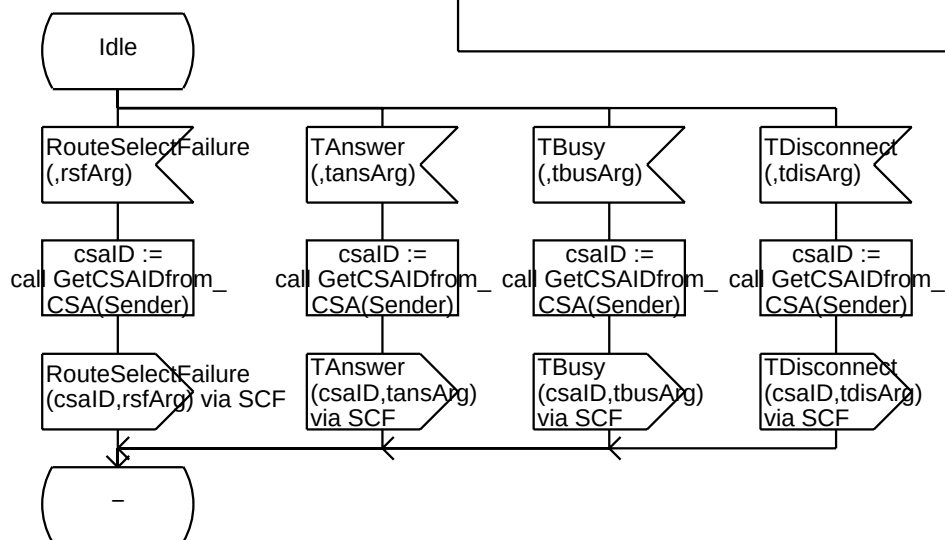


Process Type InterfaceHandler

25(32)



/* ROUTING OF DP SPECIFIC OPERATIONS FROM SSF TO SCF. */

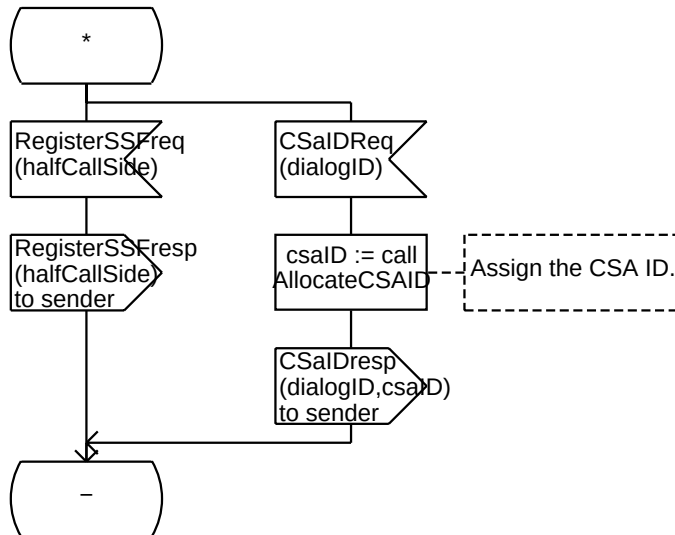


Process Type InterfaceHandler

26(32)



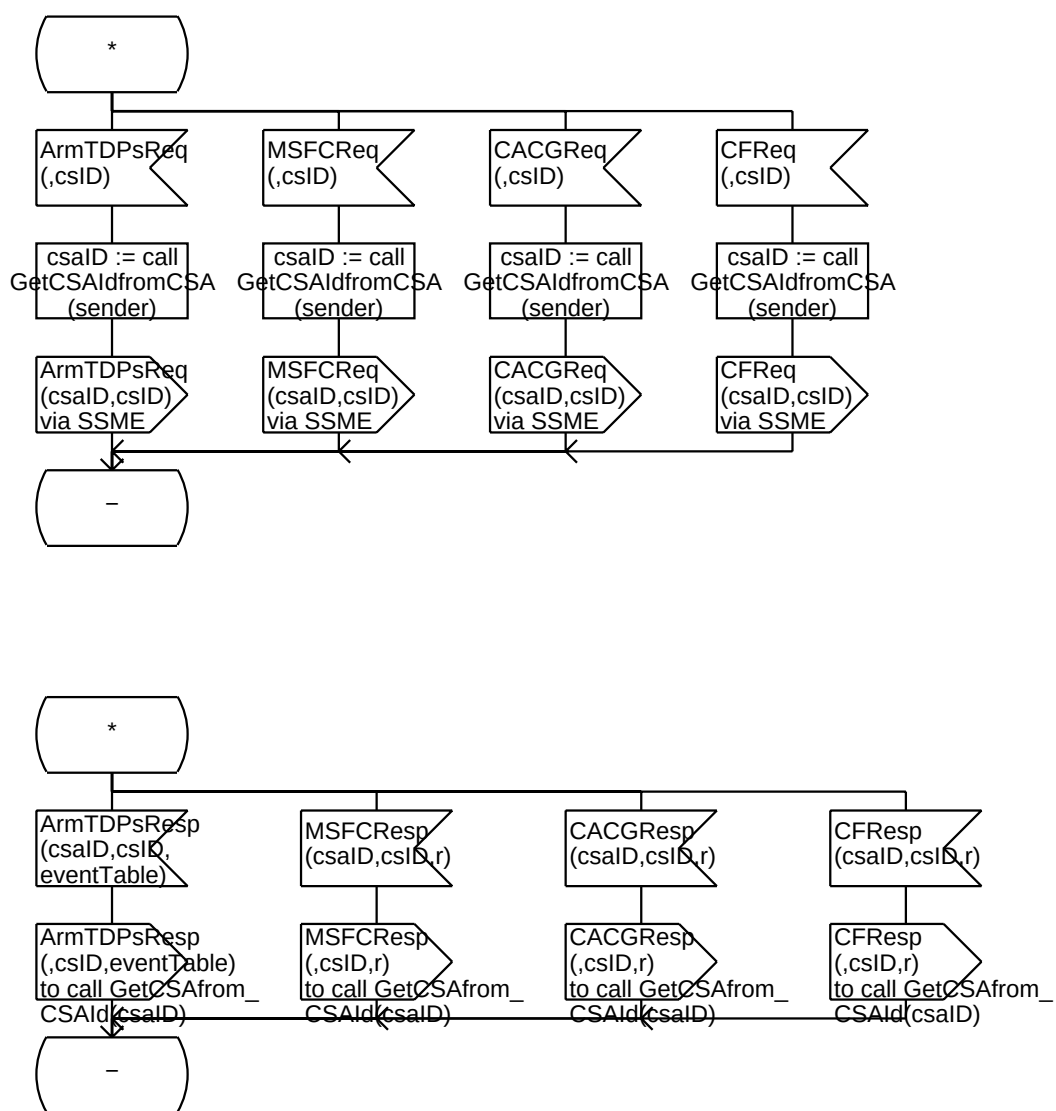
/* Coordination mechanisms with the TCAP adaptor. */



Process Type InterfaceHandler

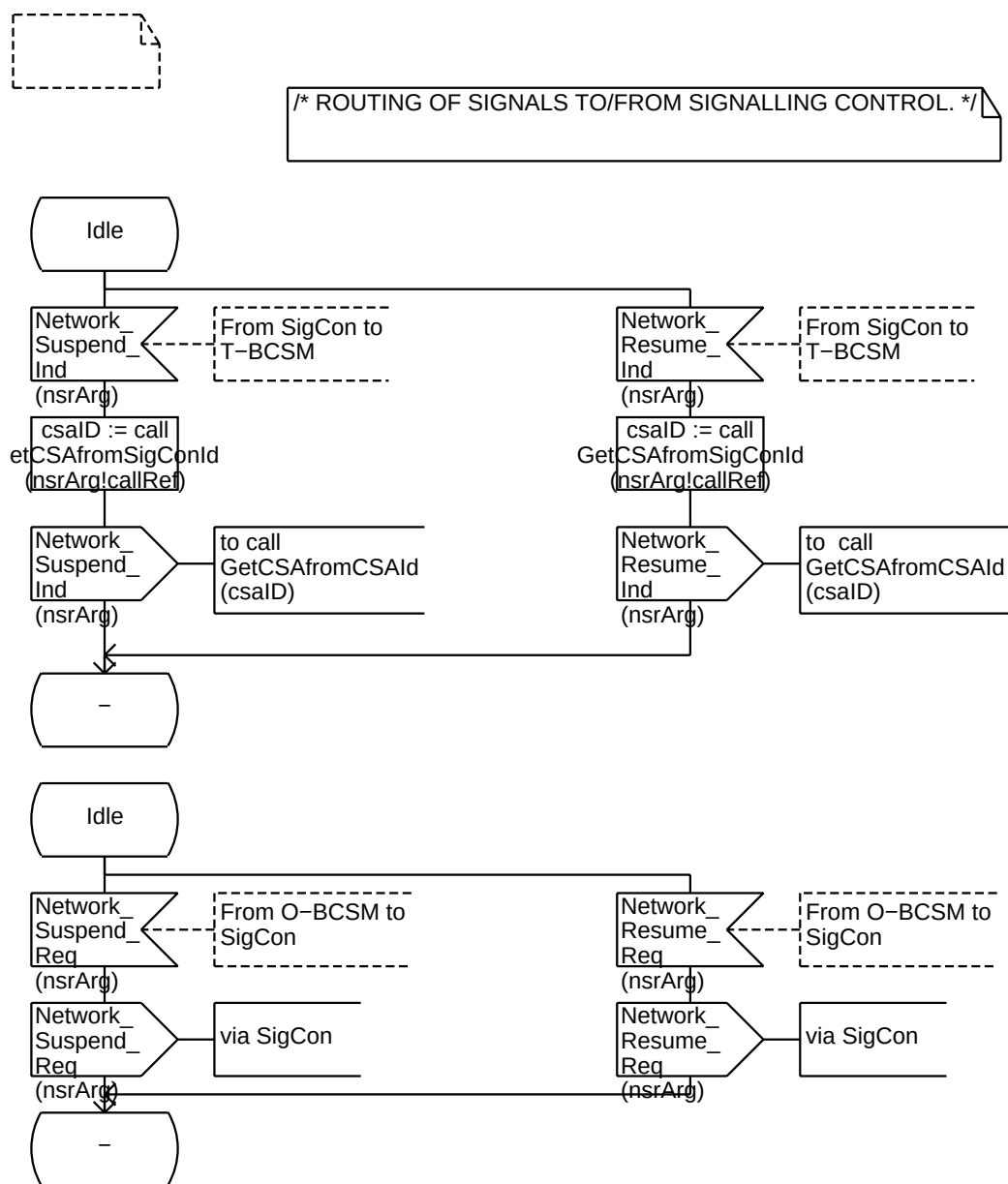
27(32)

/* ROUTING OF SIGNALS BETWEEN THE SSF-FSM AND THE SSME-FSM */



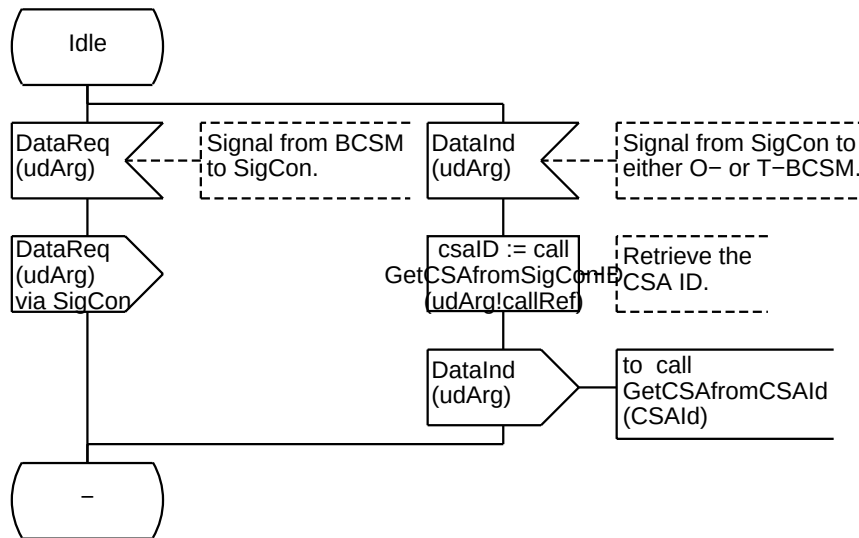
Process Type InterfaceHandler

28(32)



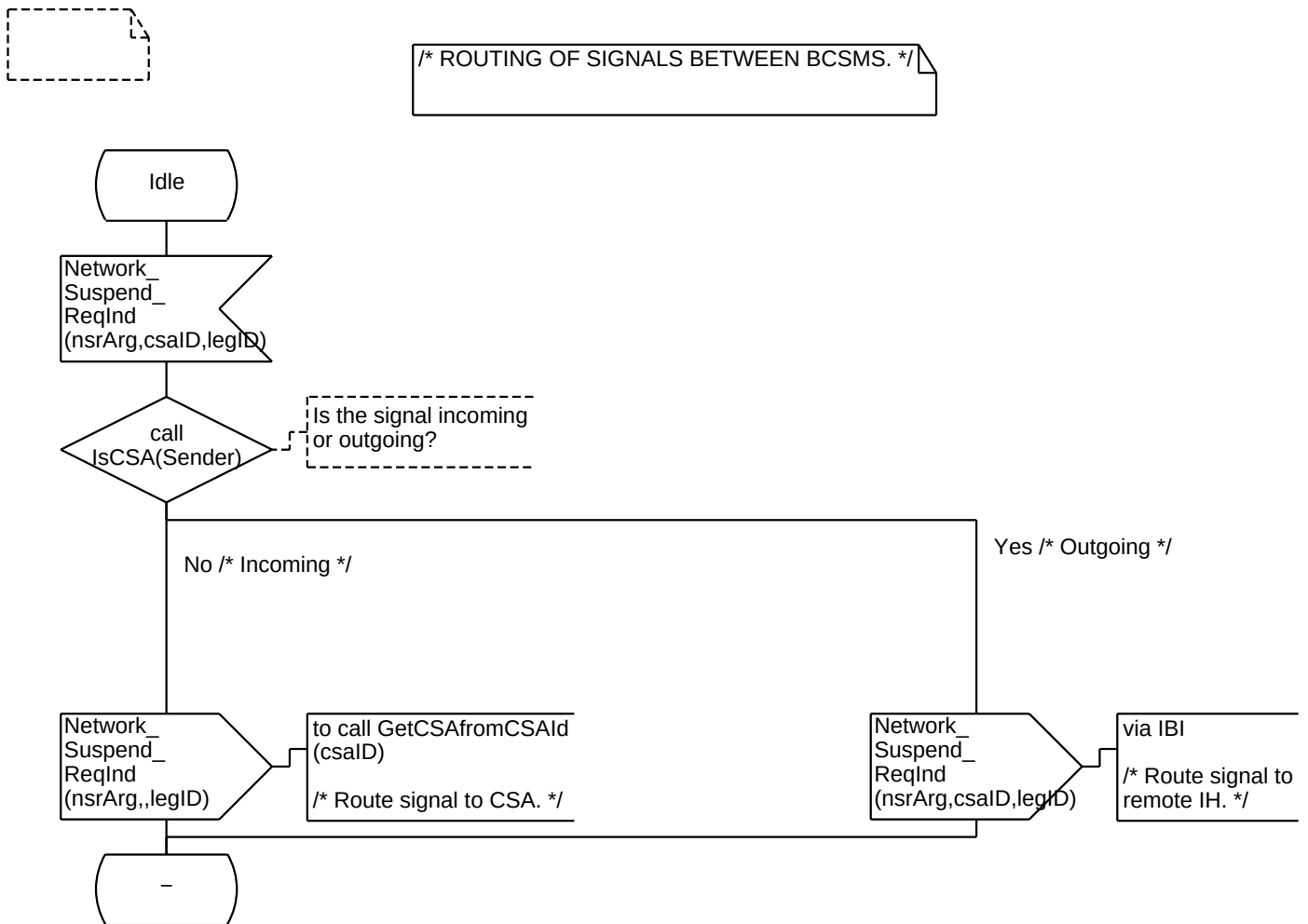
Process Type InterfaceHandler

29(32)



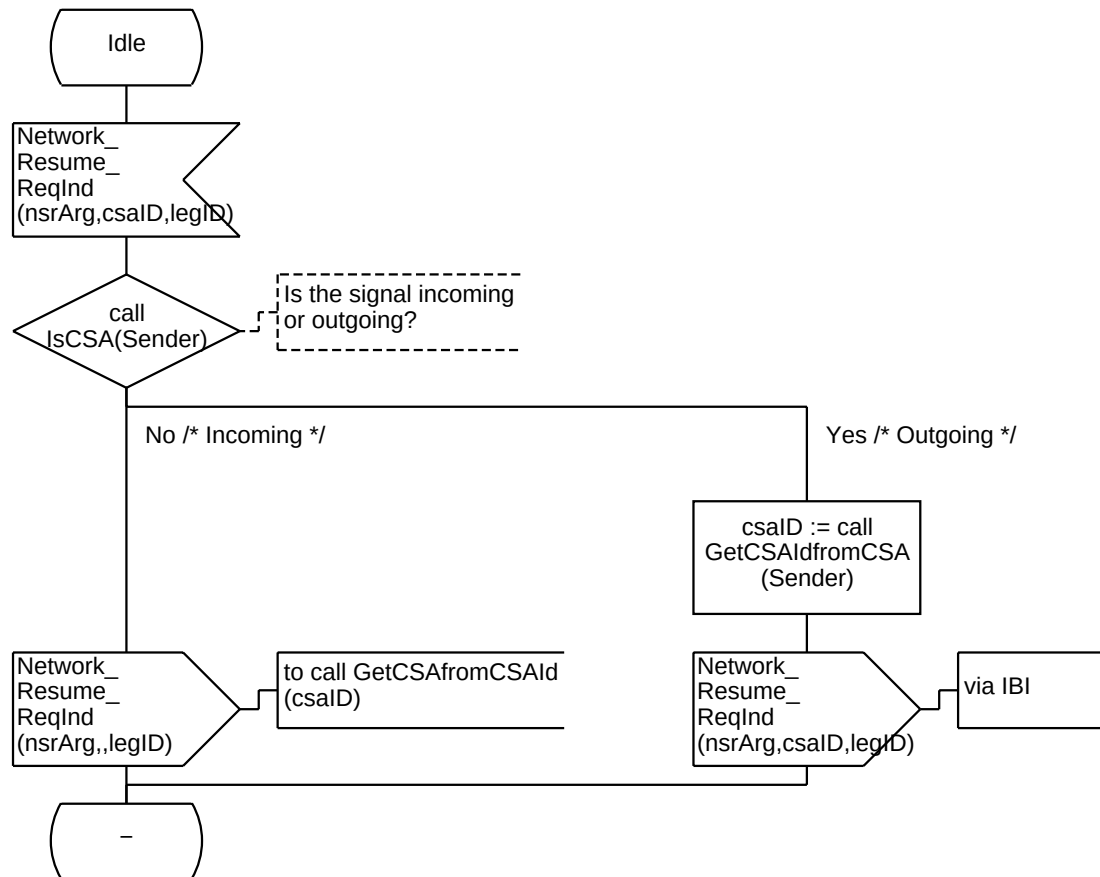
Process Type InterfaceHandler

30(32)



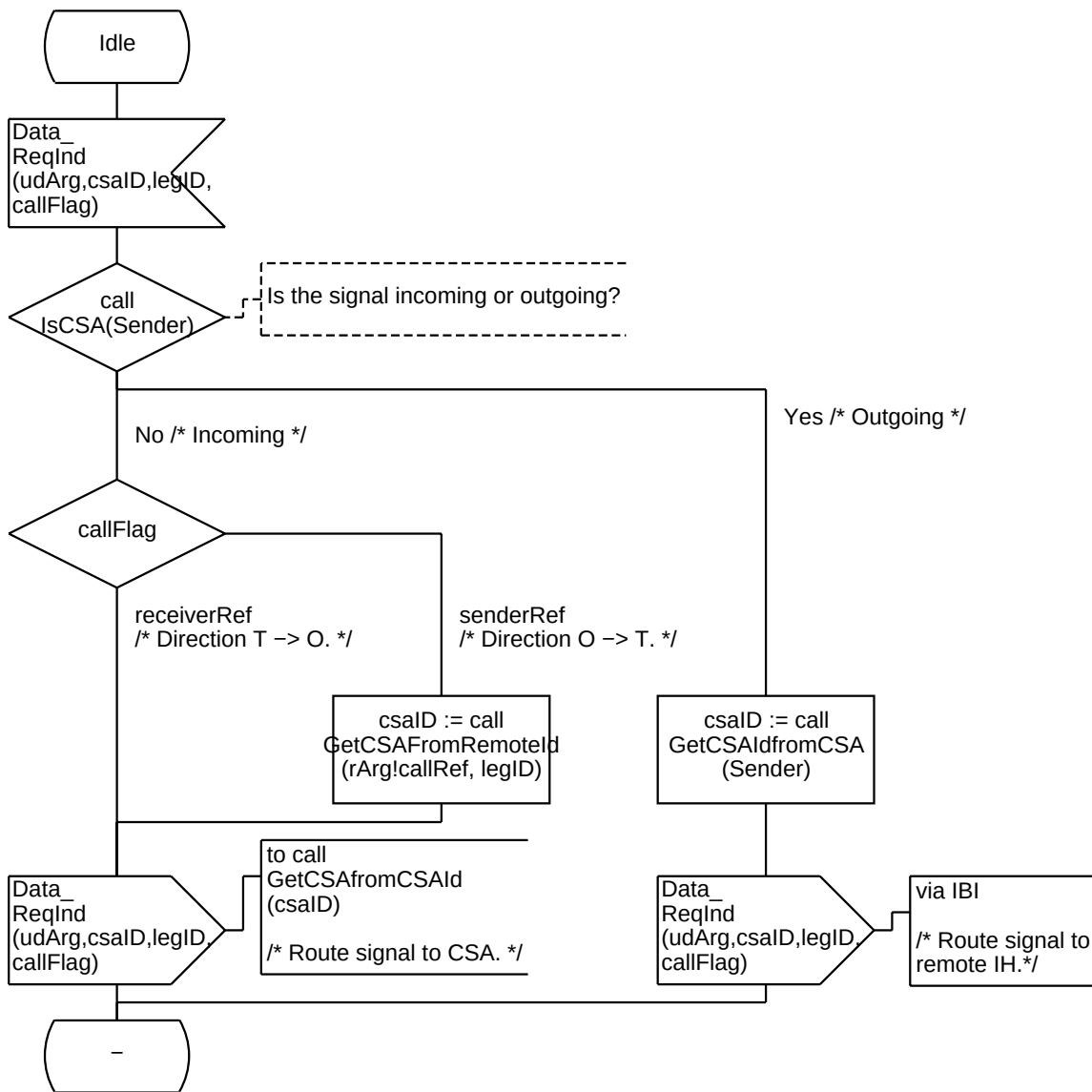
Process Type InterfaceHandler

31(32)



Process Type InterfaceHandler

32(32)

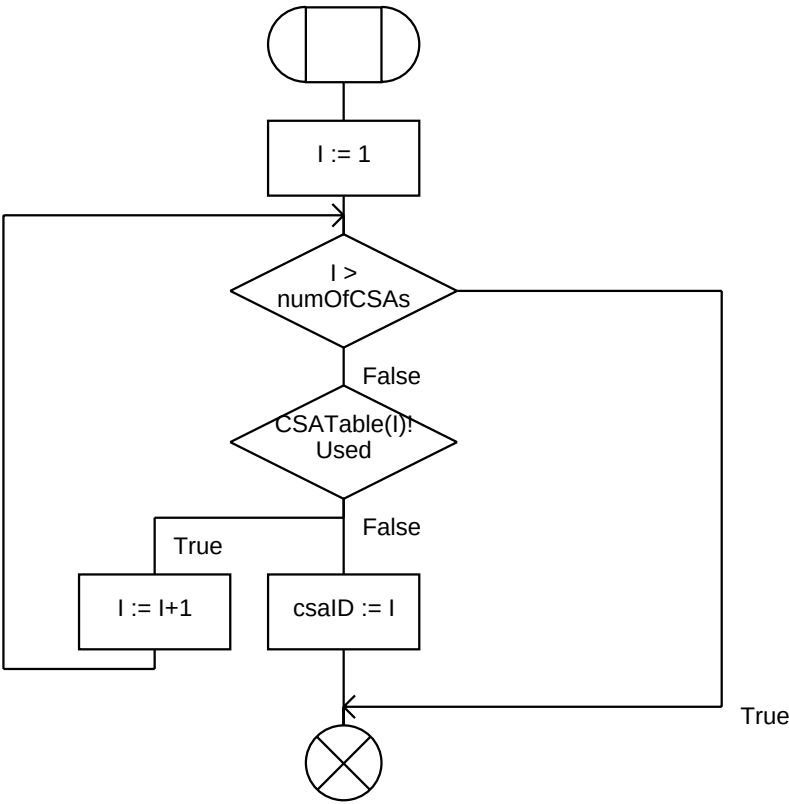


Procedure AllocateCSAId

1(1)

;RETURNS csaID CSAID;

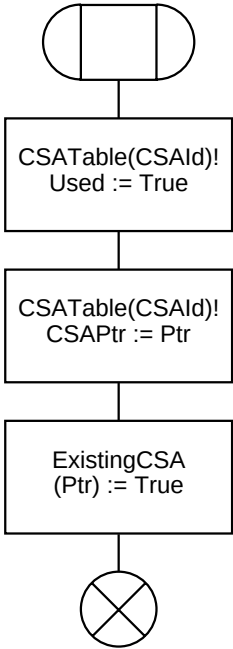
DCL
I Integer;



Procedure AddCSAId

1(1)

;FPAR
IN csalD CSAID, /* The CSA id to be assigned. */
IN Ptr PId; /* The CSA pointer. */

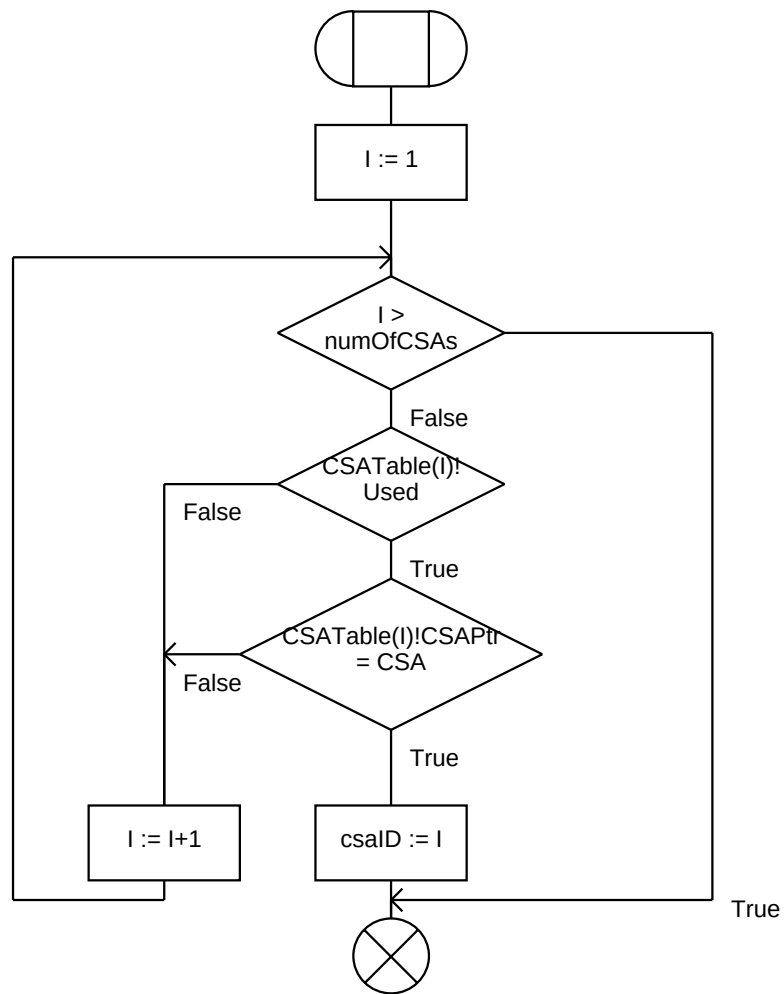


Procedure GetCSAIdfromCSA

1(1)

FPAR
IN CSA PId;
RETURNS csaID CSAID;

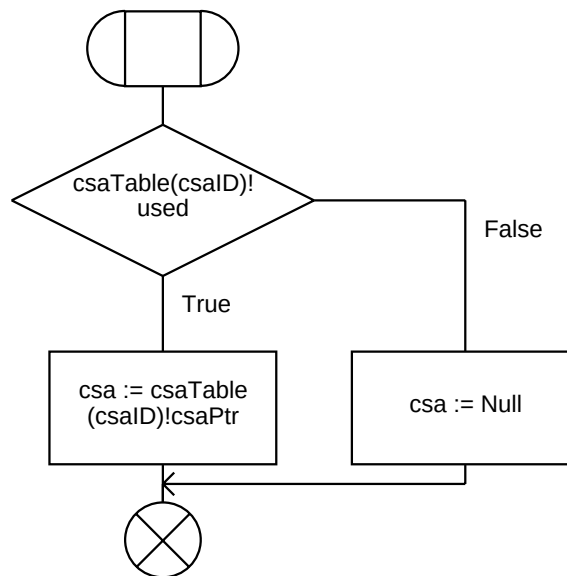
DCL
I Integer;



Procedure GetCSAfromCSAId

1(1)

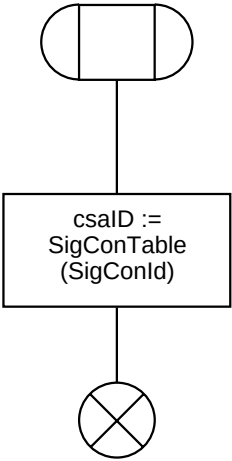
FPAR
IN csaID CSAID;
RETURNS csa PId;



Procedure GetCSAfromSigConId

1(1)

FPAR
IN SigConId CallRef;
RETURNS csaID CSAID;

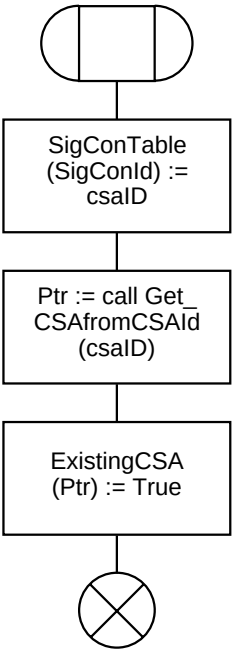


Procedure SetSigConAssoc

1(1)

;FPAR
IN SigConId CallRef;
IN csalD CSAID;

DCL
Ptr PId;

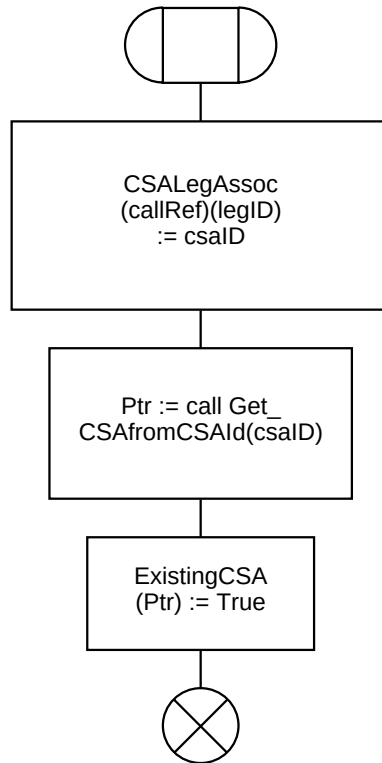


Procedure SetRemoteAssoc

1(1)

FPAR
IN callRef CallRef;
IN legID LegType;
IN csalD CSAID;

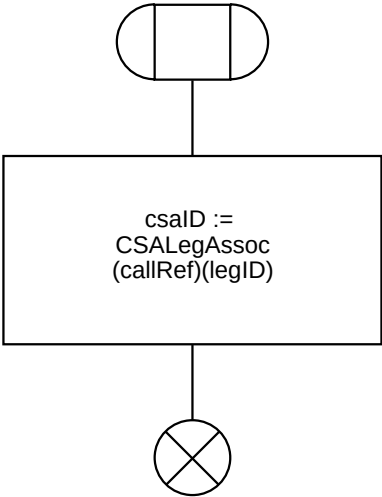
DCL
Ptr PId;



Procedure GetCSAfromRemoteId

1(1)

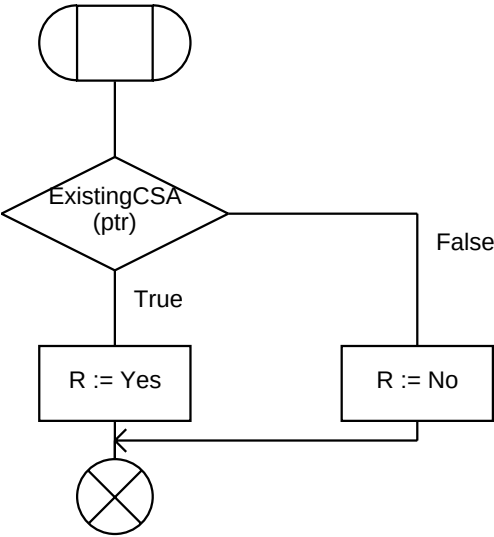
;FPAR
IN callRef CallRef,
IN legID LegType;
RETURNS csaID CSAID;



Procedure IsCSA

1(1)

;FPAR
IN Ptr PId;
RETURNS R AnswerType;

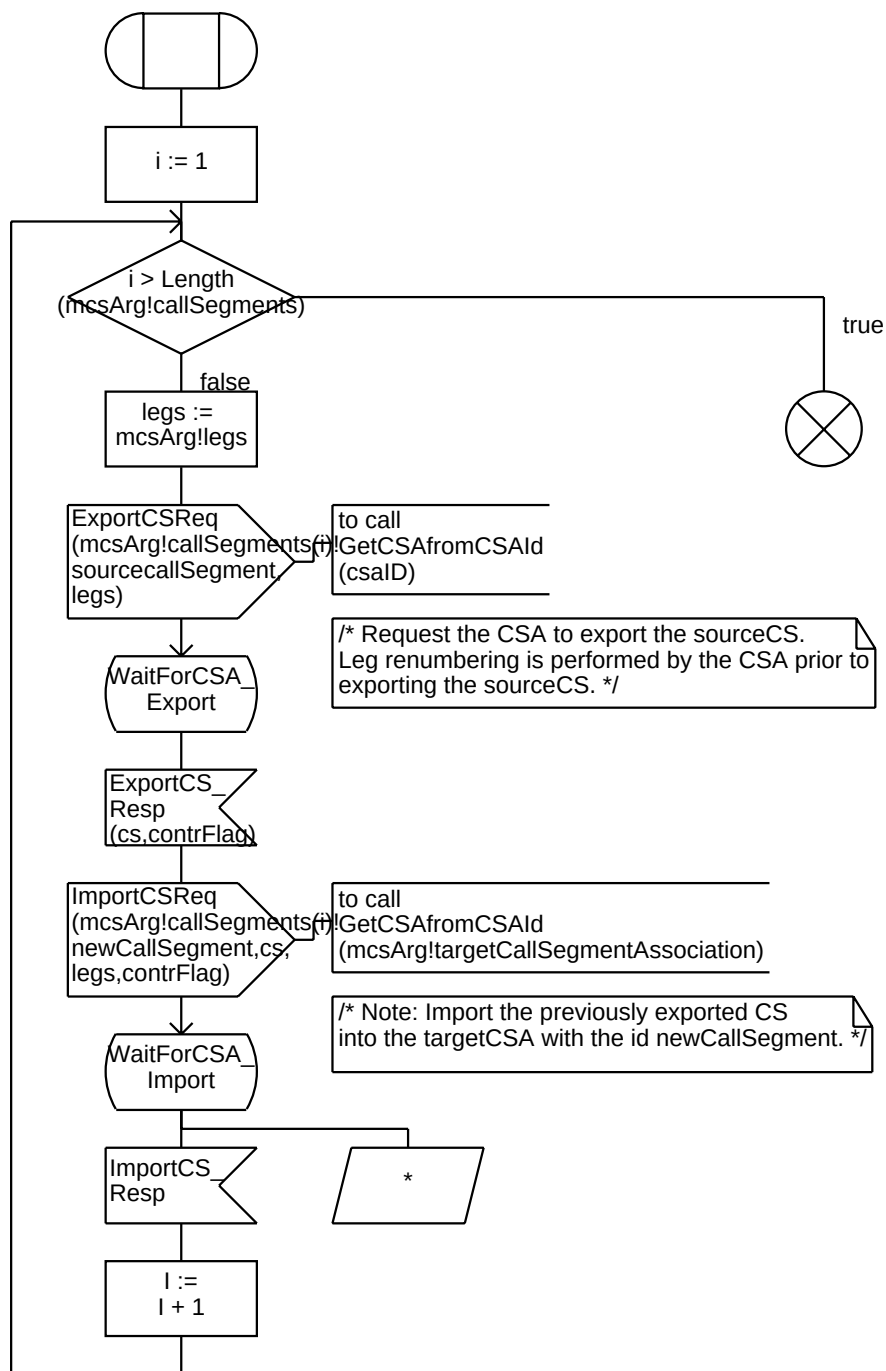


Procedure ProcessMoveCallSegments

1(1)

FPAR
IN mcsArg MoveCallSegmentsArg;

DCL
i Integer,
cs ExportCSType,
contrFlag Boolean,
legs ExportLegsType;



Process Type CallSegmentAssociation

1(42)

**** DATA TYPE DEFINITIONS ****

/* The Call Segment Association (CSA) maintains information about the Call Segments (CS) of the CSA. */

NEWTYPE CS STRUCT

Used Boolean; /* Flag indicating whether the segment exists or not. */

CSPtr PId; /* Pointer/Address of the instance of the CS process associated with the segment. */

DEFAULT (. False, Null .)

ENDNEWTYPE;

NEWTYPE CSAType /* The call segment association. */

ARRAY(CallSegmentID,CS)

ENDNEWTYPE;

/* The CSA maintains a mapping from LegID to CallSegmentID. This information is needed to route primitives to the correct call segment. */

NEWTYPE LegLocationTable

ARRAY(LegType,LegDetails)

ENDNEWTYPE;

NEWTYPE LegDetails STRUCT

Used Boolean;

CSID CallSegmentID;

DEFAULT (. FALSE,1 .)

ENDNEWTYPE;

/* The CSA maintains a leg association table, which associates a 'remote' LegID with a 'local' leg id. This information is needed to correctly address the BCSM when passing primitives between an O-BCSM and a T-BCSM. */

NEWTYPE LegAssociationTable

ARRAY(LegType,LegType)

ENDNEWTYPE;

/* Definition of return results from procedure calls. */

NEWTYPE AnswerType

LITERALS Yes, No;

ENDNEWTYPE;

NEWTYPE ResultType

LITERALS Successful, Failed;

ENDNEWTYPE;

Process Type CallSegmentAssociation

2(42)



/**** DATA TYPE DEFINITIONS ****/

```
/* The CSA must maintain a queue of RRBs for arming of
events of legs that does not yet exist. */

NEWTYPE RRBEQueueSeq STRUCT
  Q Boolean;
  RRBE BCSMEvent;
ENDNEWTYPE;

NEWTYPE RRBEQueueType
  ARRAY(LegType,RRBEQueueSeq)
ENDNEWTYPE;
```

Process Type CallSegmentAssociation

3(42)



/**** VARIABLE DECLARATIONS *****/

DCL

```
/* The CSA */
CSA CSAType,

/* The leg location table. */
LL LegLocationTable,

/* ID of the initial (first created) call segment. */
initialCallSegmentID CallSegmentID,

/* The association of local and remote leg ids. */
LegAssoc LegAssociationTable,

/* The ID of the controlling Leg (1 or 2) */
controllingLegID LegType,

numOfAppls Integer, /* Variable that holds the current number of open
                     applications/dialogues. */

/* Other variables. */
csaID CSAID :=1,
csiID CallSegmentID :=1,
invokeID InvokeID,
termination Boolean,
legID, newLegID LegType,
callFlag CallFlag,
eventTable EventTableType,
dID DialogIDType,
r Boolean;
```

DCL

```
/* Signalling control primitive parameters. */
AEArg AddressEndType,
CPArg CallProgressType,
FArg FailureType,
RArg ReleaseType,
SFtArg ServiceFeatureType,
SIRArg SetupIRType,
SCRArg SetupCRType,
SAArg SubsequentAddressType;
```

DCL

```
/* operation arguments. */
AlArg AnalyseInformationArg,
ADlArg AnalysedInformationArg,
ACArg ApplyChargingArg,
ACRArg ApplyChargingReportArg,
ARlArg AssistRequestInstructionsArg,
ClRArg CallInformationReportArg,
ClRqArg CallInformationRequestArg,
CArg CancelArg,
ClArg CollectInformationArg,
CDlArg CollectedInformationArg,
CoArg ConnectArg,
CTRArg ConnectToResourceArg,
CtArg ContinueWithArgumentArg,
DFCArg DisconnectForwardConnectionWithArgumentArg,
ETCArg EstablishTemporaryConnectionArg,
ENCArg EventNotificationChargingArg,
ERBCSMArg EventReportBCSMArg,
FCIArg FurnishChargingInformationArg,
HCINArg HoldCallInNetworkArg,
IDPArg InitialDPArg,
ICAAArg InitiateCallAttemptArg,
OABNArg OAbandonArg,
OANSArg OAnswerArg,
OCPBArg OCalledPartyBusyArg,
ODISArg ODisconnectArg,
MIDCLArg MidCallArg,
ONOANSArg ONoAnswerArg,
OAAArg OriginationAttemptAuthorizedArg,
RCArg ReleaseCallArg,
RNCEArg RequestNotificationChargingEventArg,
RRBCSMArg RequestReportBCSMEEventArg,
RTArg ResetTimerArg,
RSFArg RouteSelectFailureArg,
SCIArg SendChargingInformationArg,
SFArg SelectFacilityArg,
SRArg SelectRouteArg,
TANSArg TAnswerArg,
TBUSArg TBusyArg,
TDISArg TDisconnectArg,
TAAArg TermAttemptAuthorizedArg,
TNOANSArg TNoAnswerArg,
IEArg ErrorArg
;
```

Process Type CallSegmentAssociation

4(42)



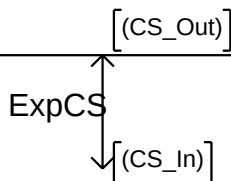
**** VARIABLE DECLARATIONS ****

DCL
/* IN operation arguments. */
autArg AuthorizeTerminationArg,
ctwaArg ContinueWithArgumentArg,
dfcwaArg DisconnectForwardConnectionWithArgumentArg,
dlArg DisconnectLegArg,
erArg EntityReleasedArg,
erfArg EventReportFacilityArg,
fsaArg FacilitySelectedAndAvailableArg,
mtdArg ManageTriggerDataArg,
mtdrArg ManageTriggerDataResultArg,
mcsArg MergeCallSegmentsArg,
mlArg MoveLegArg,
oaArg OriginationAttemptArg,
osusArg OSuspendedArg,
reconArg ReconnectArg,
rutsiArg ReportUTSIArg,
rrutsiArg RequestReportUTSIArg,
rrfeArg RequestReportFacilityEventArg,
sendfArg SendFacilityInformationArg,
sstuiArg SendSTUIArg,
sspArg SetServiceProfileArg,
slArg SplitLegArg,
taArg TerminationAttemptArg,
tsusArg TSuspendedArg
;

DCL
/* Signalling control primitive parameters. */
nsrArg NetworkSRTYPE,
udArg UserDataType;

DCL
rrbeQueue RRBEQueueType, /* The queued RRBES. */

contrFlag Boolean,
leg ExportLegType,
legs ExportLegsType,
expCS ExportCSType,
n Integer;



Process Type CallSegmentAssociation

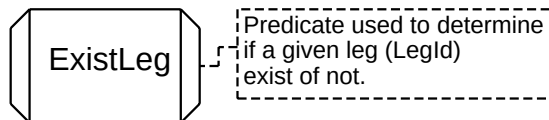
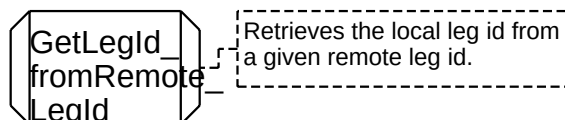
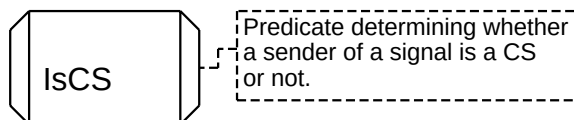
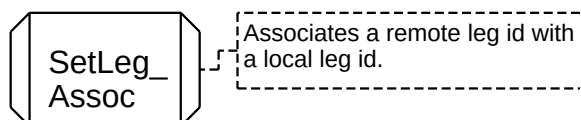
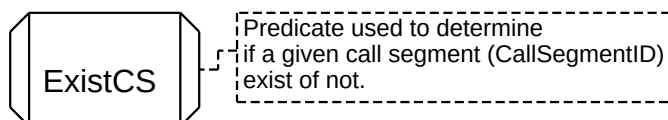
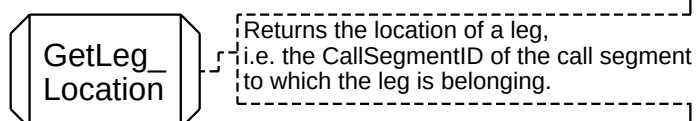
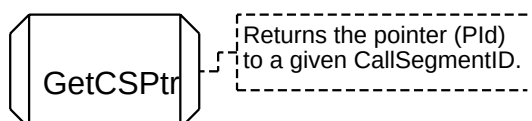
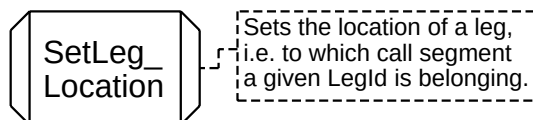
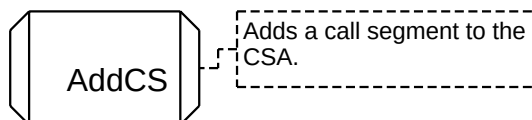
5(42)



/* ** DECLARATION OF OPERATIONS ** */

/* Operations on call segments. */

/* Operations on legs. */



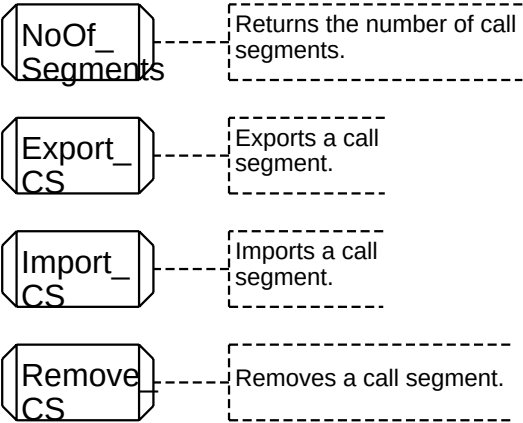
Process Type CallSegmentAssociation

6(42)

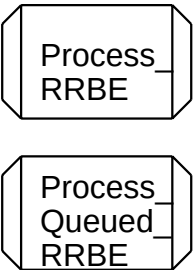


**** DECLARATION OF OPERATIONS ****

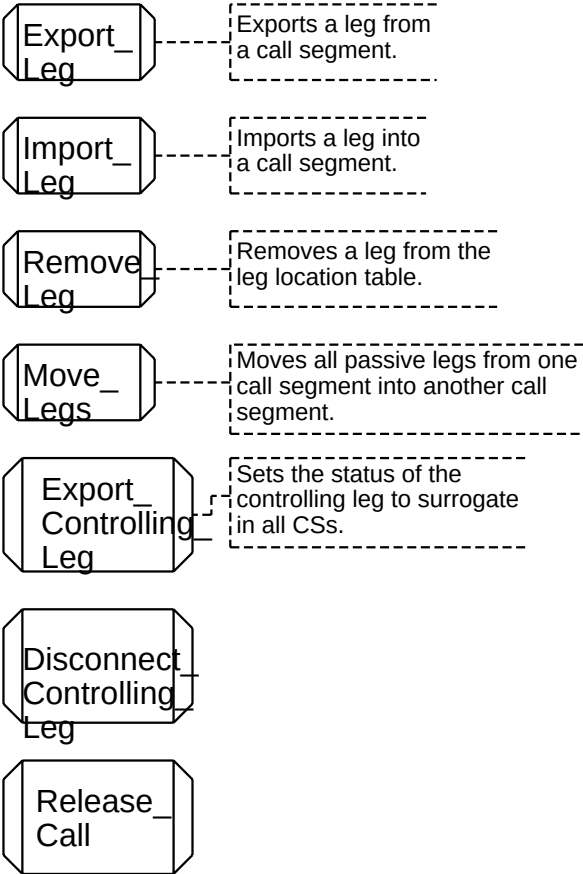
/* Operations on call segments. */

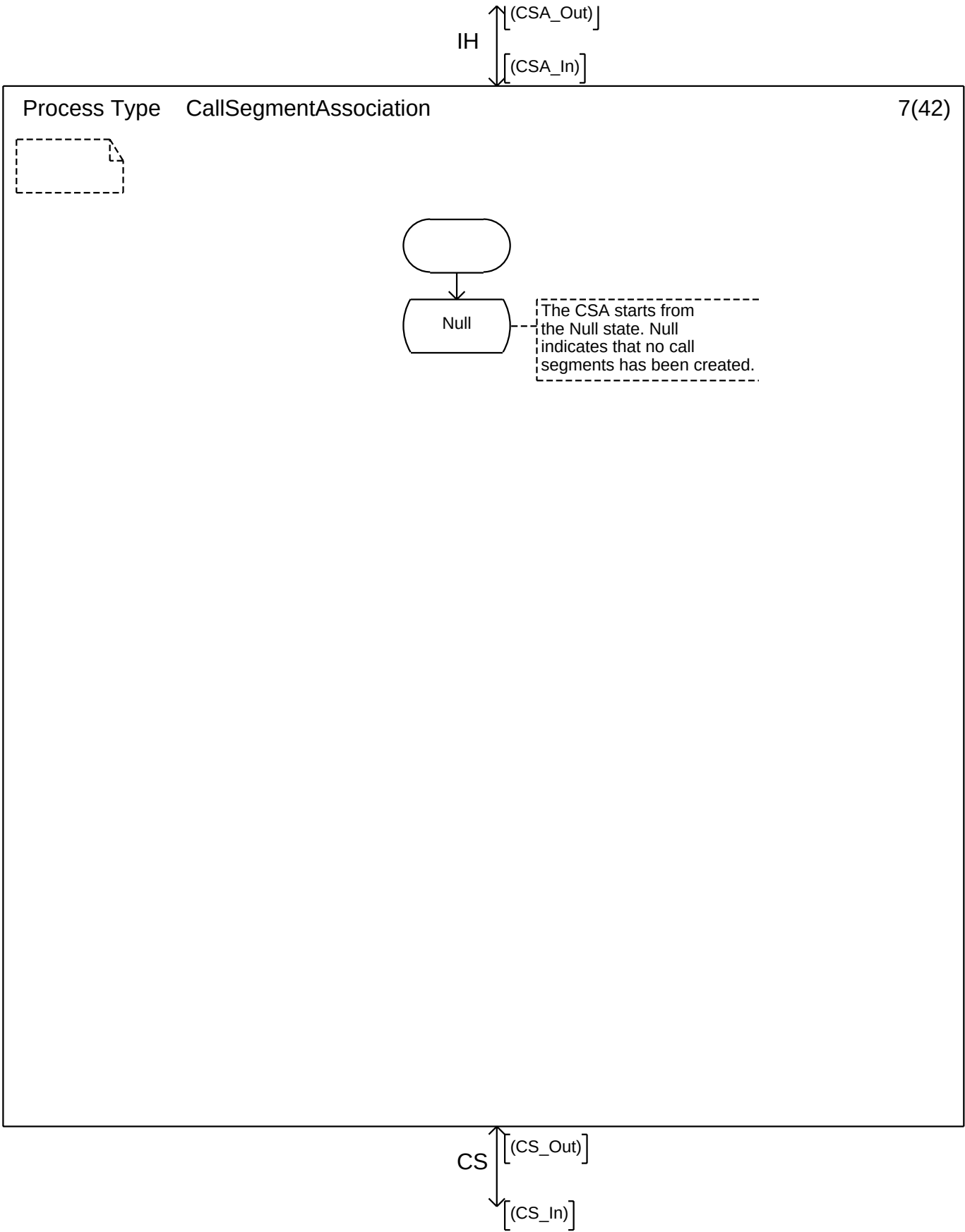


/* Operations on the queue of RRBEs. */



/* Operations on legs. */



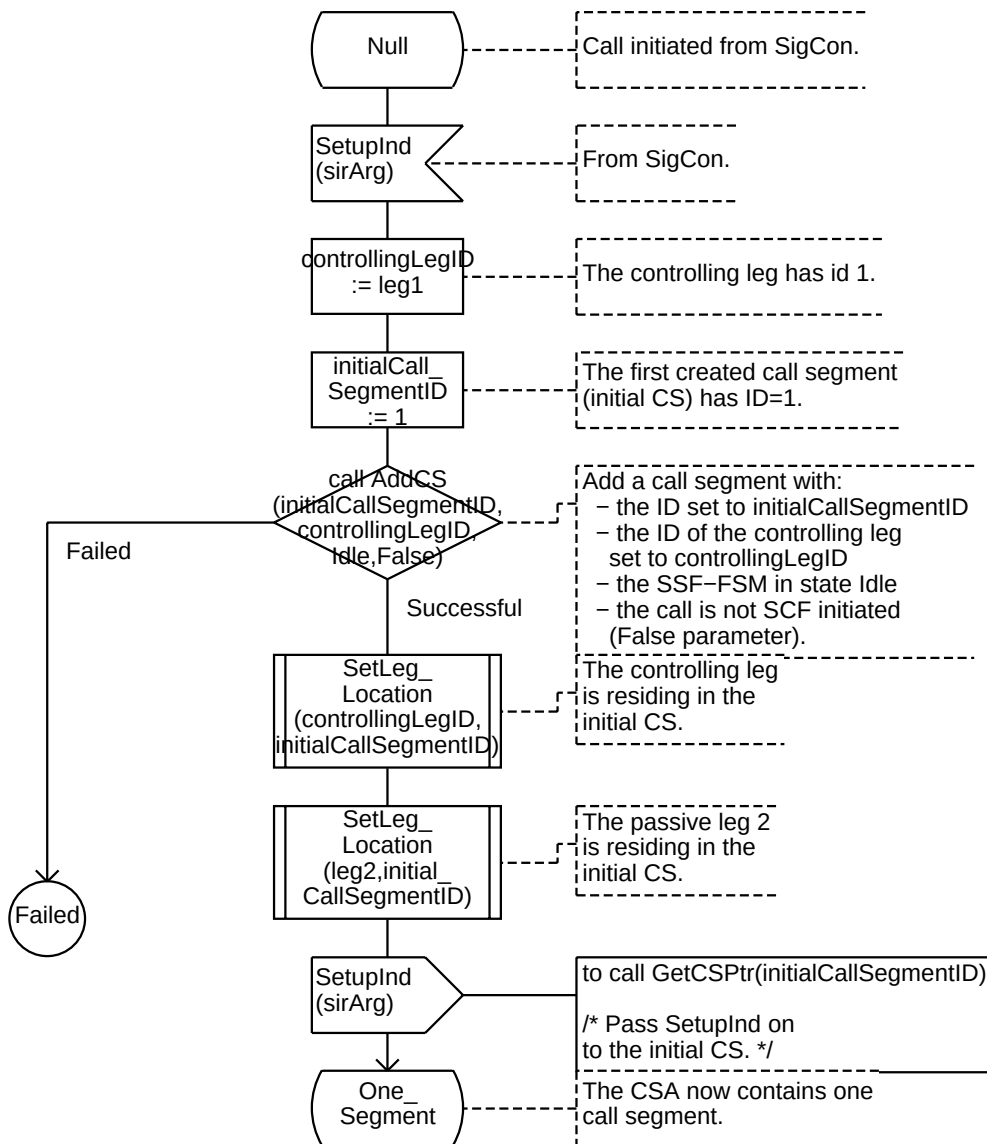


Process Type CallSegmentAssociation

8(42)

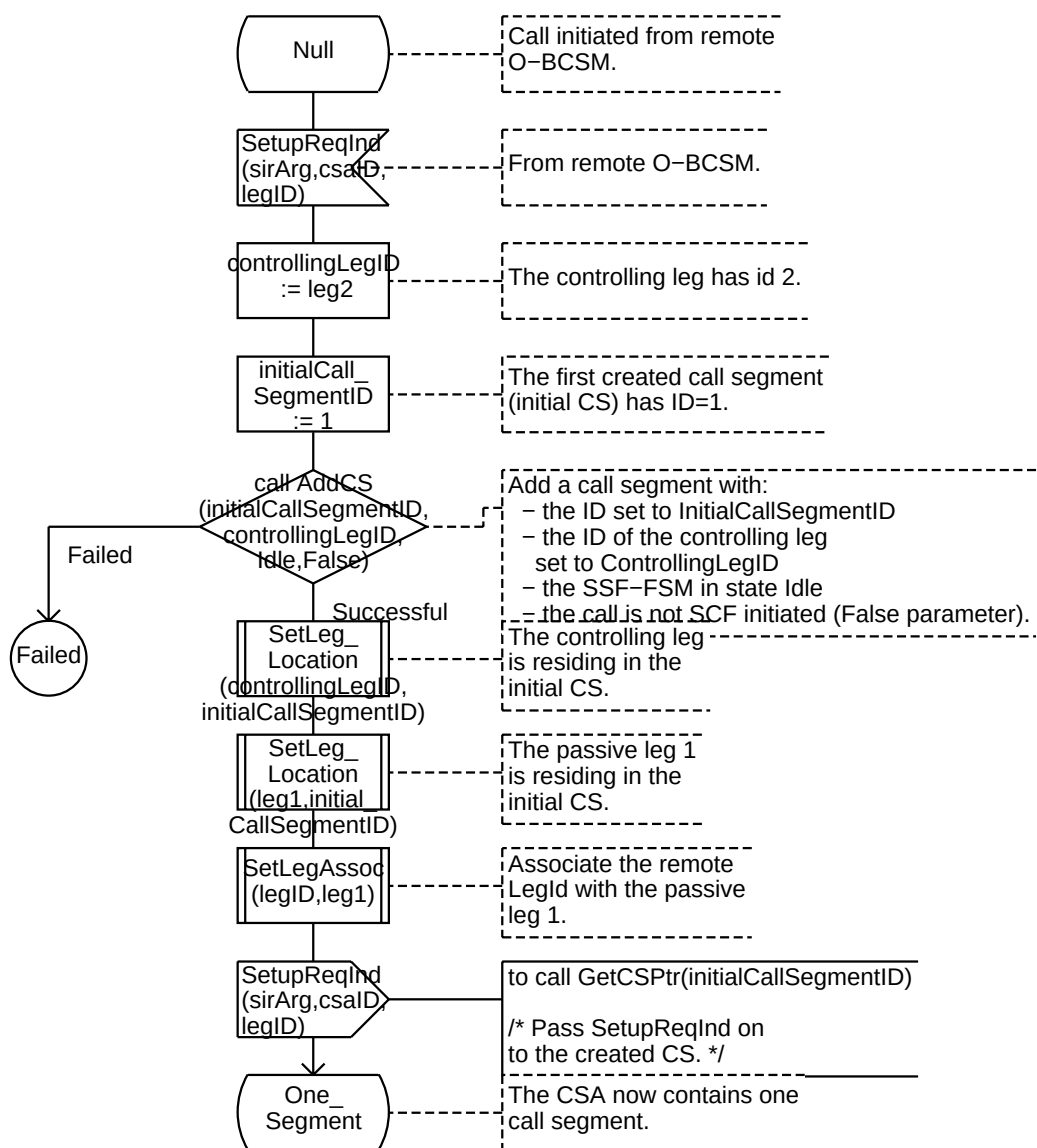


**** HANDLING OF CALL SETUP. ****



Process Type CallSegmentAssociation

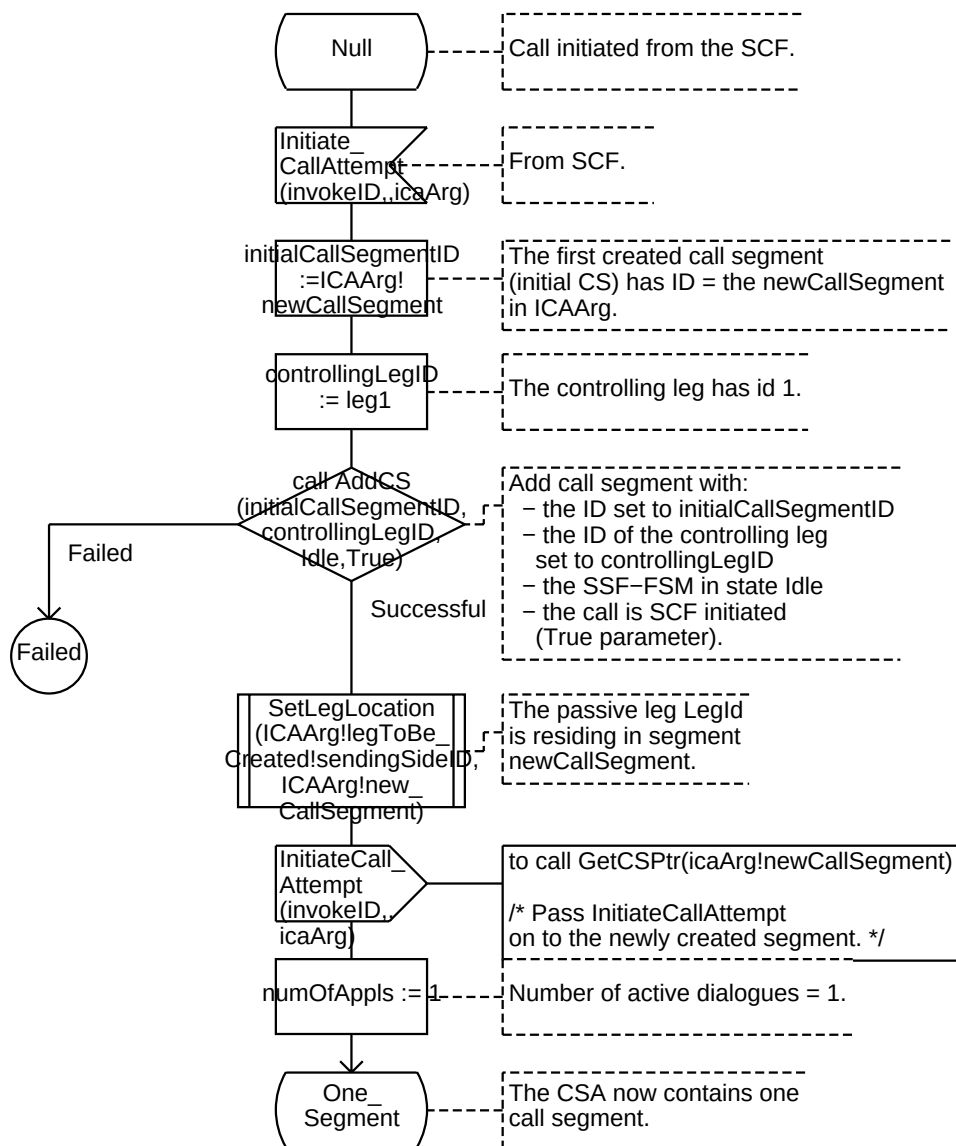
9(42)



Process Type CallSegmentAssociation

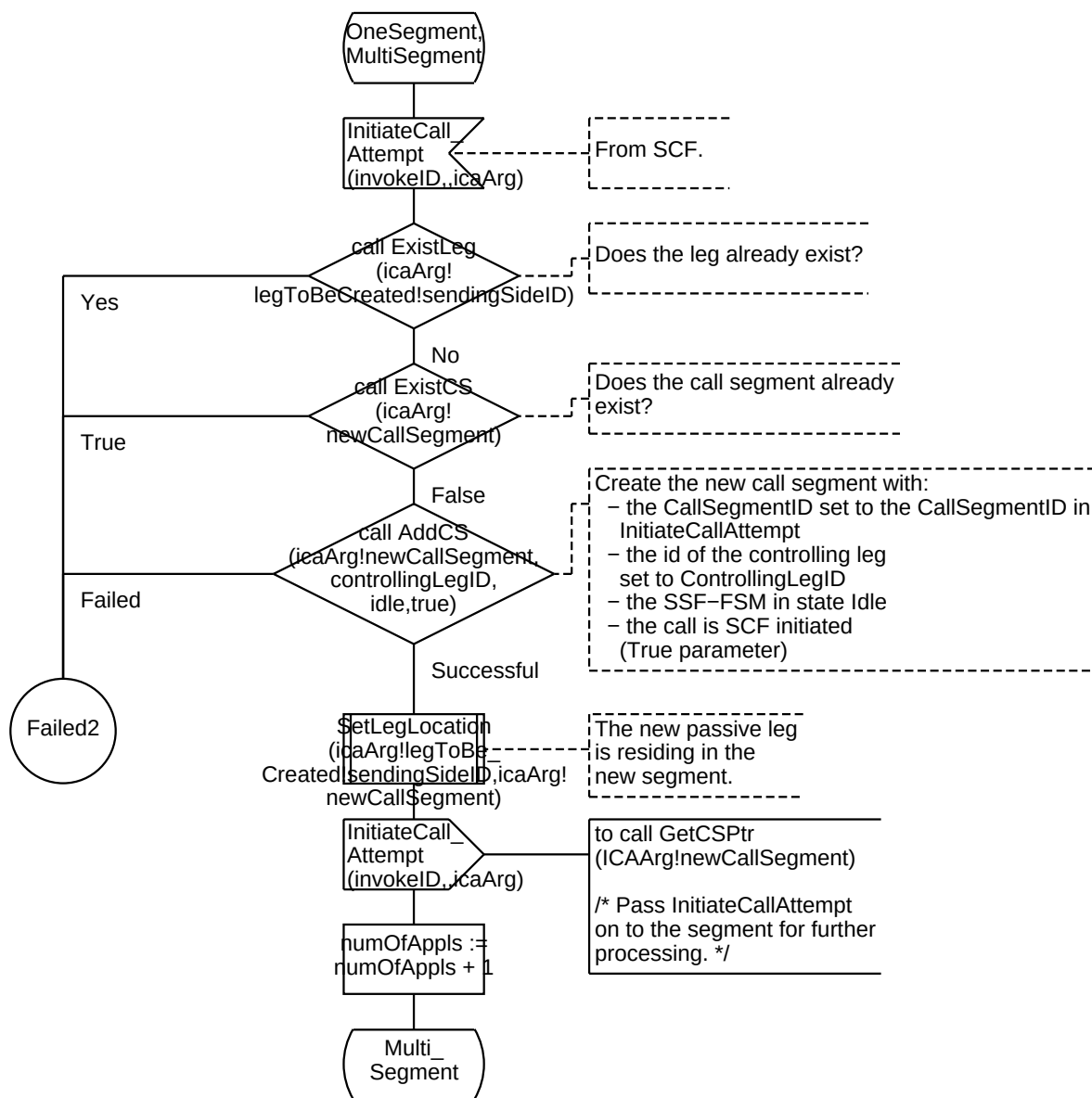
10(42)

/*** PROCESSING OF INAP OPERATIONS THAT CHANGES THE STATE OF THE CALL SEGMENT ASSOCIATION. ***/



Process Type CallSegmentAssociation

11(42)

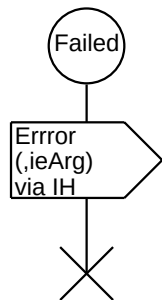
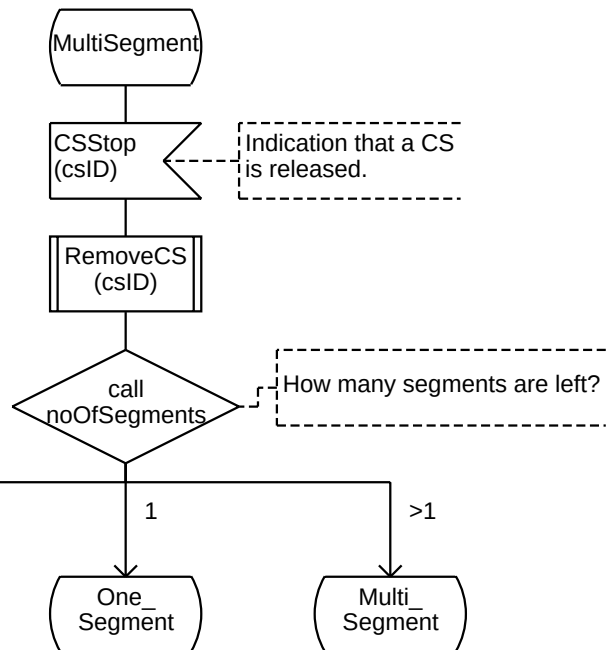
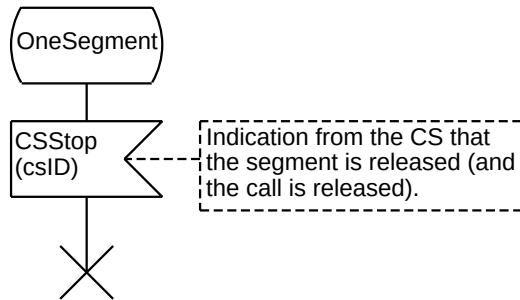
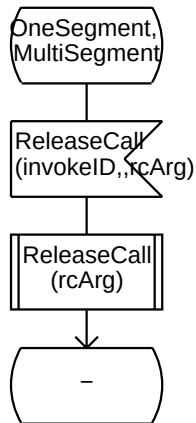


Process Type CallSegmentAssociation

12(42)

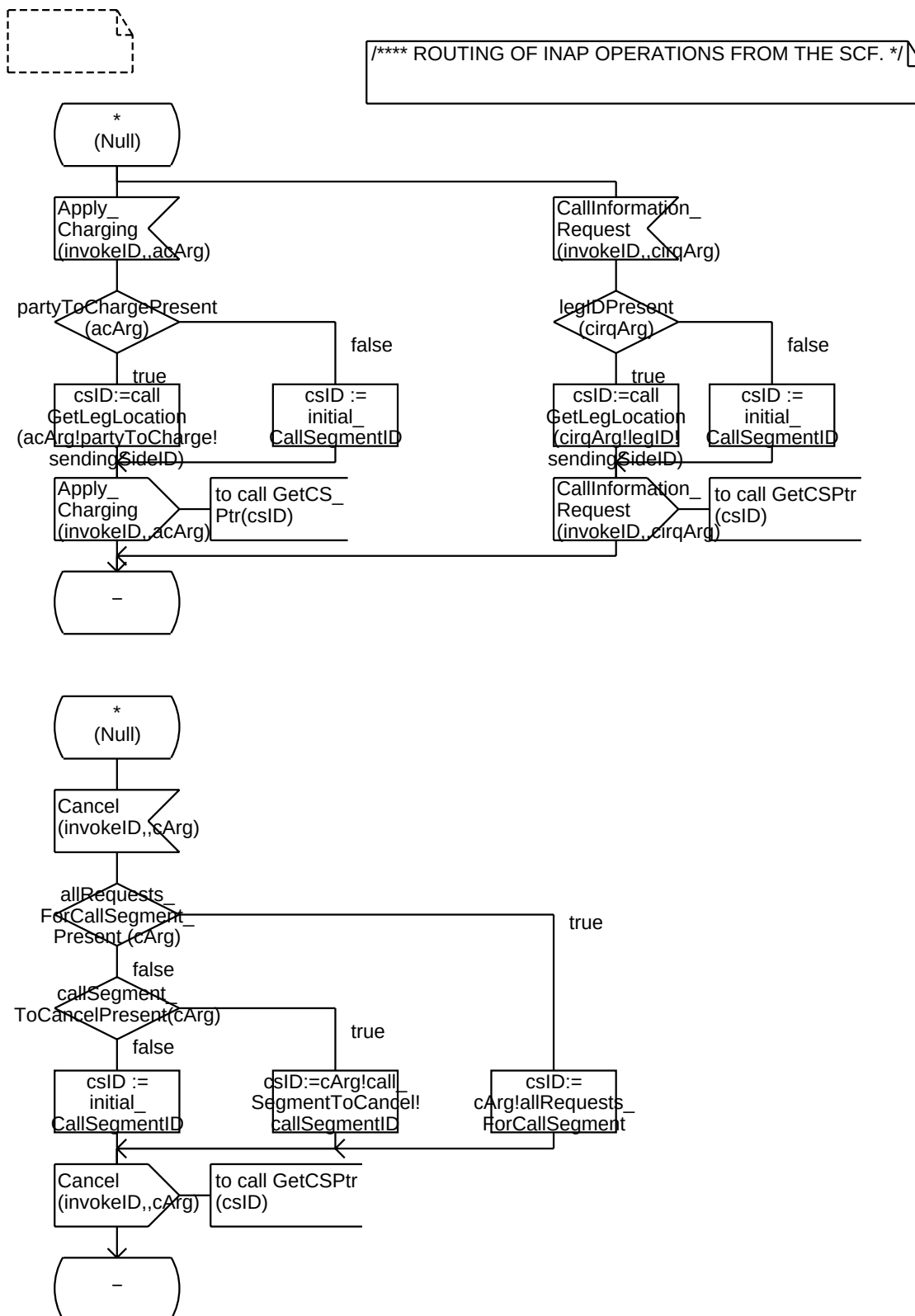


**** HANDLING OF CALL RELEASE ****



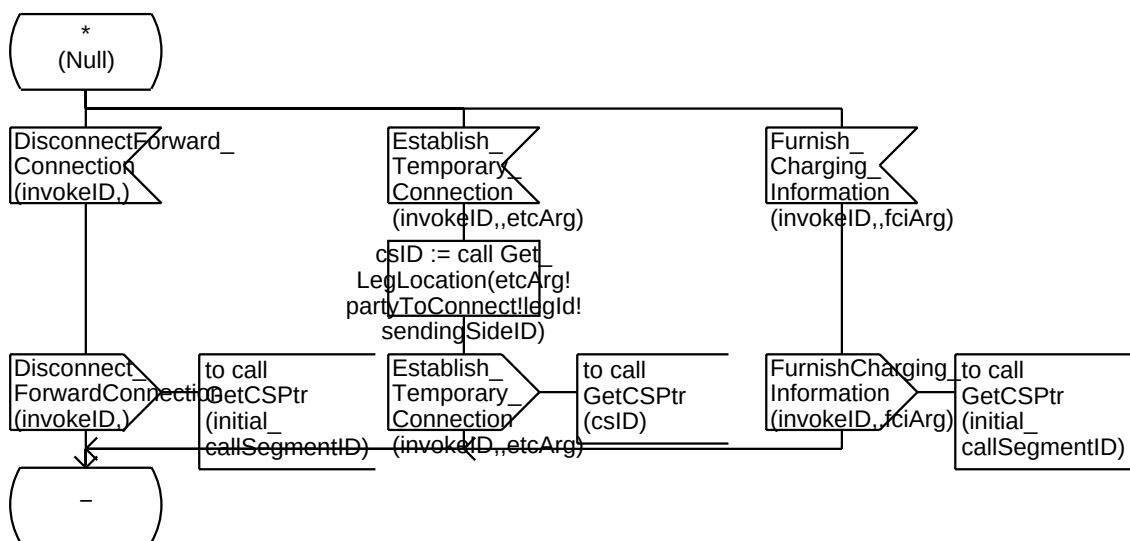
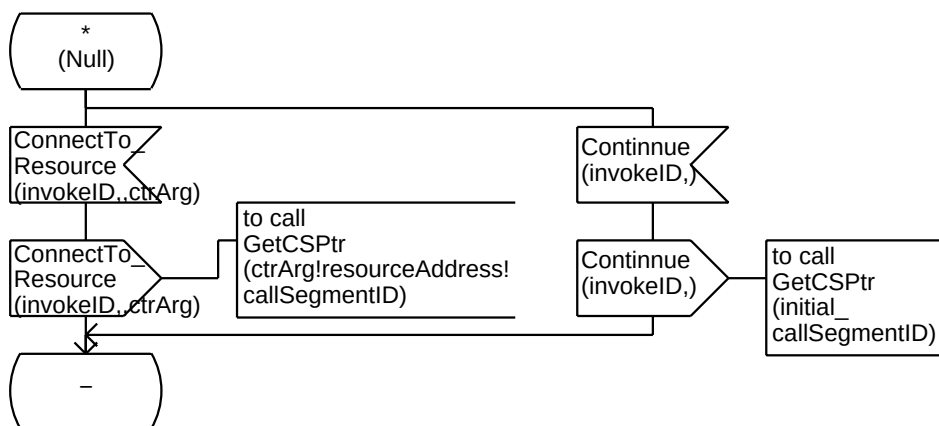
Process Type CallSegmentAssociation

13(42)



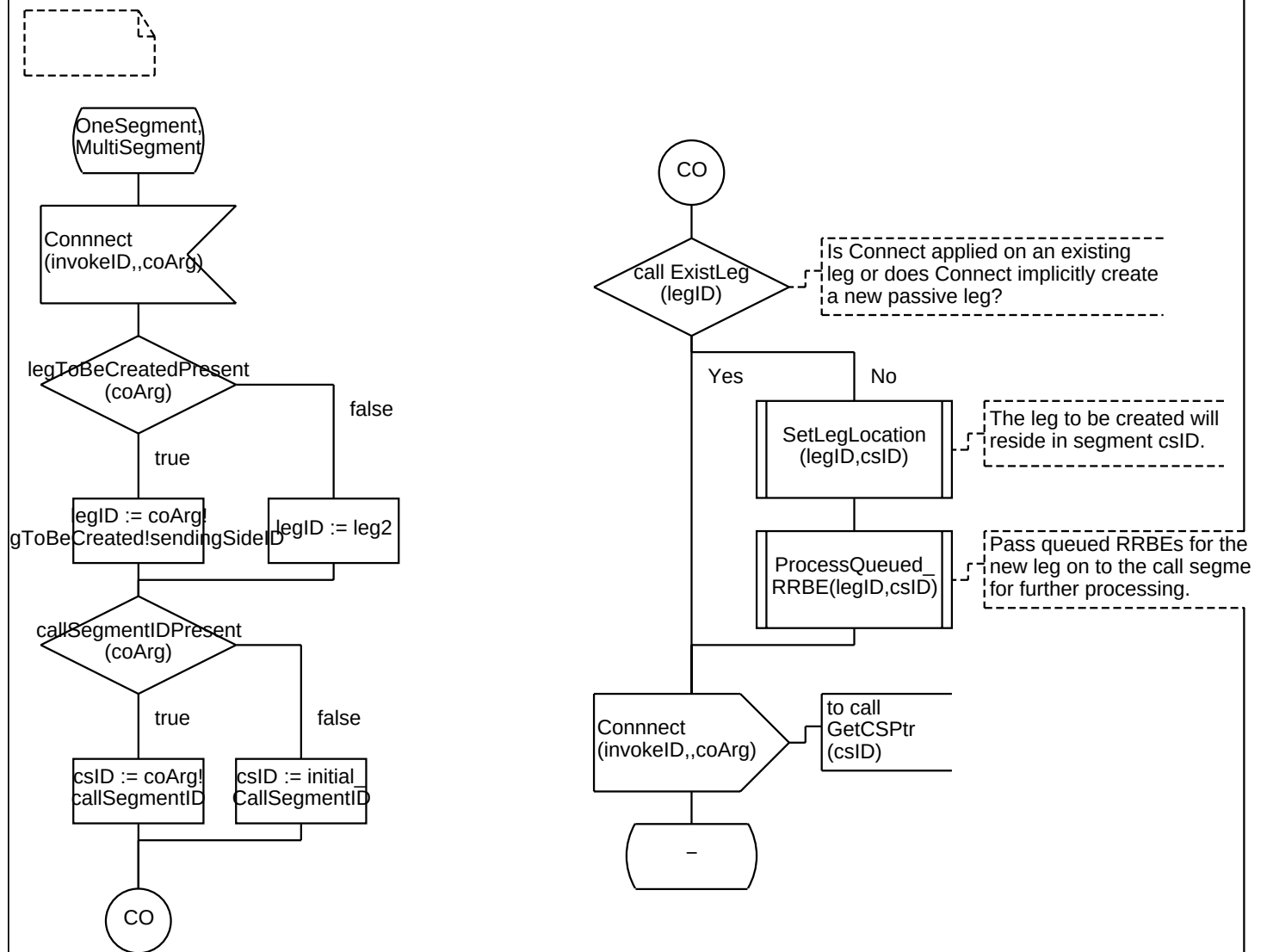
Process Type CallSegmentAssociation

14(42)



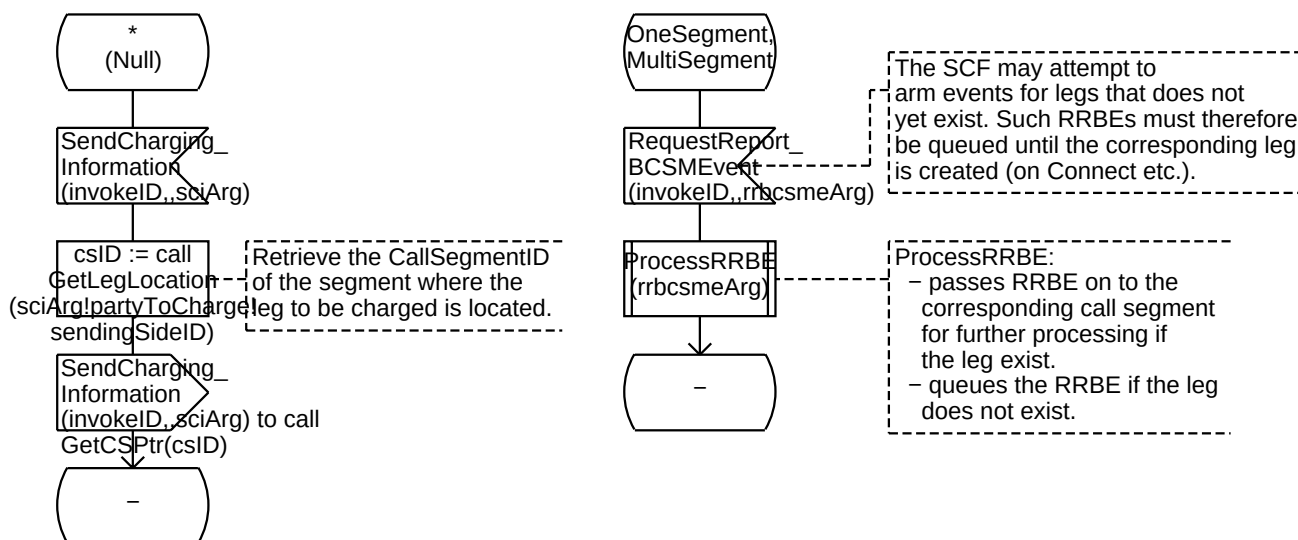
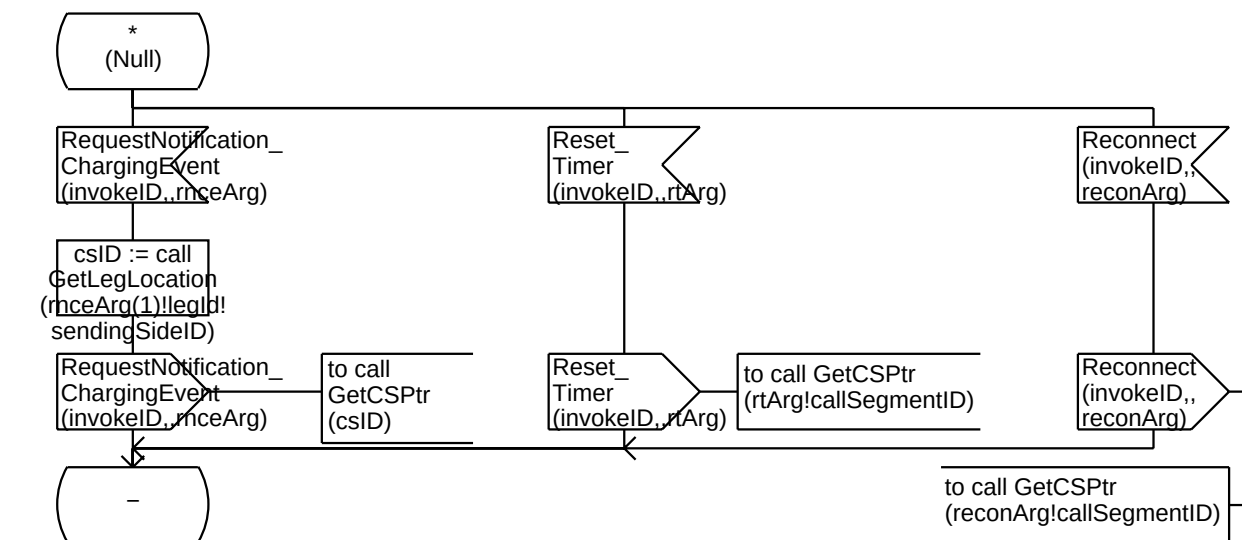
Process Type CallSegmentAssociation

15(42)



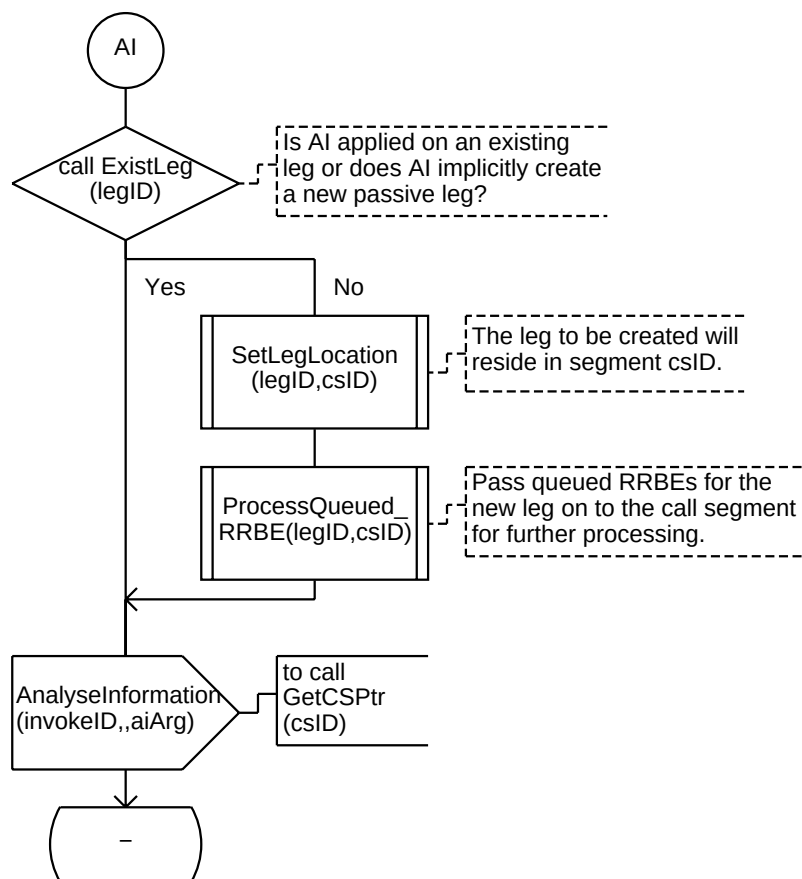
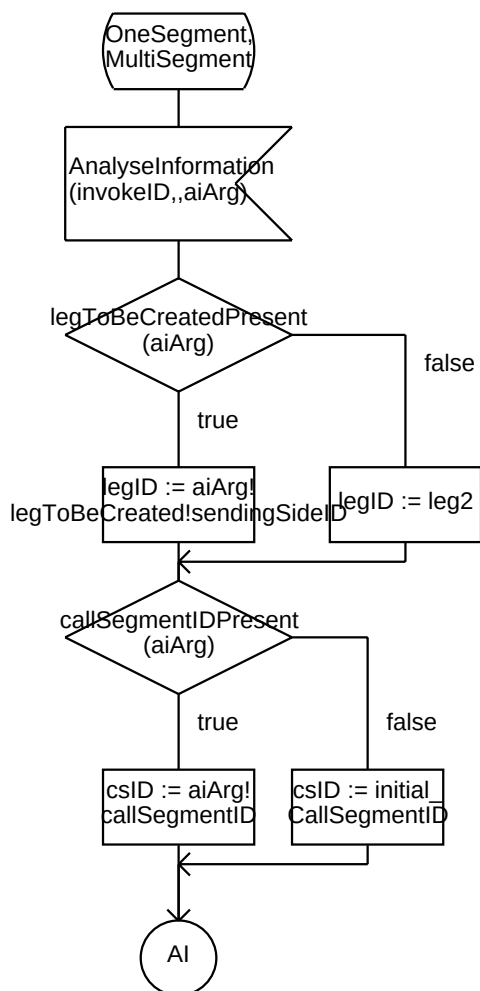
Process Type CallSegmentAssociation

16(42)



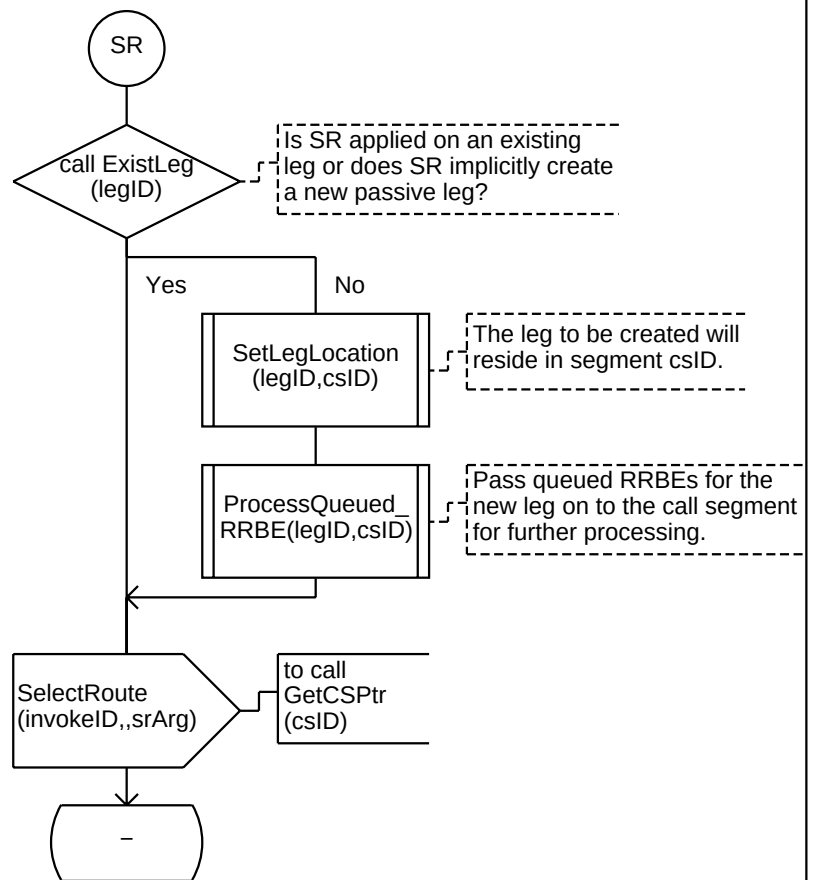
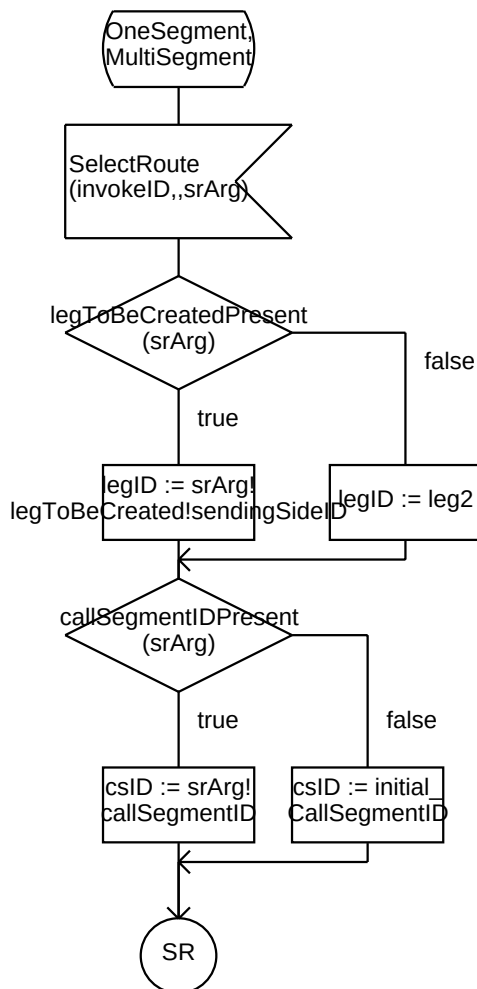
Process Type CallSegmentAssociation

17(42)



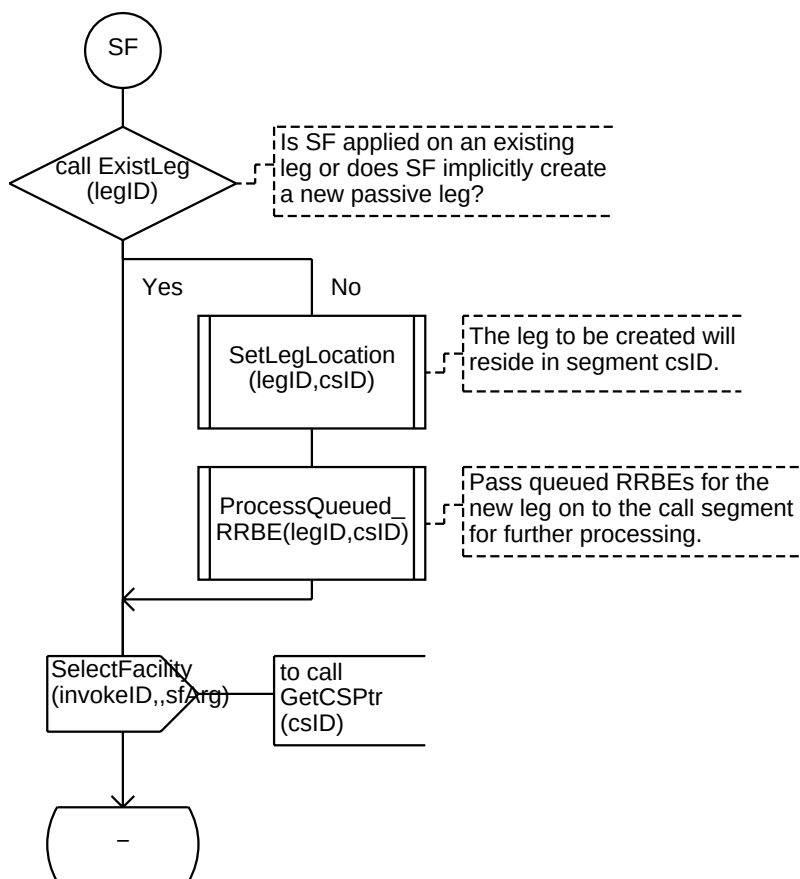
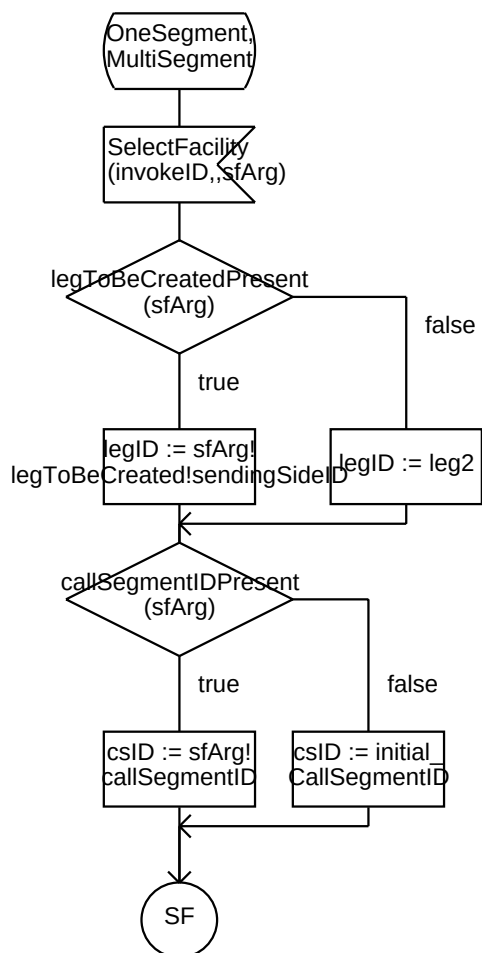
Process Type CallSegmentAssociation

18(42)



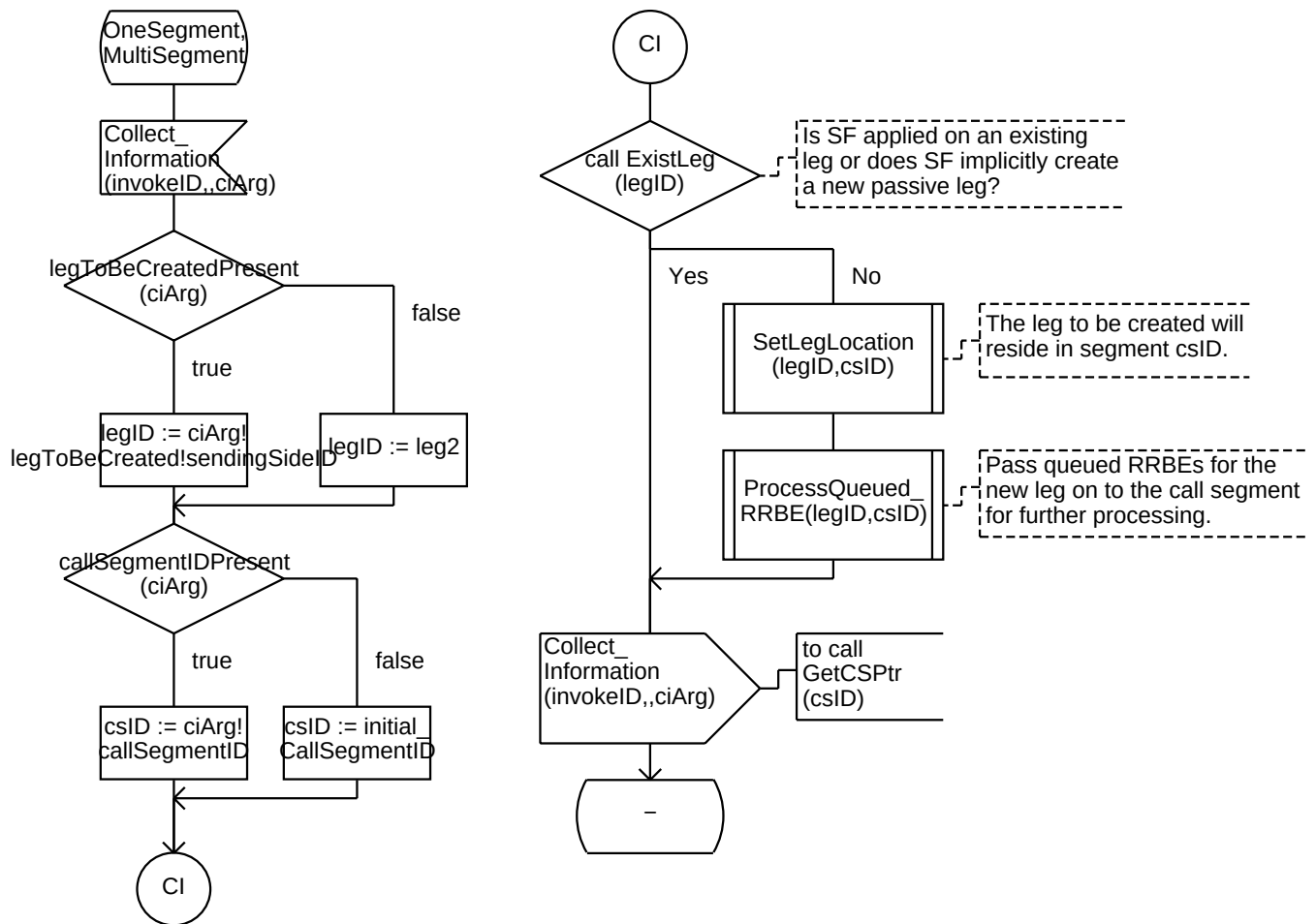
Process Type CallSegmentAssociation

19(42)



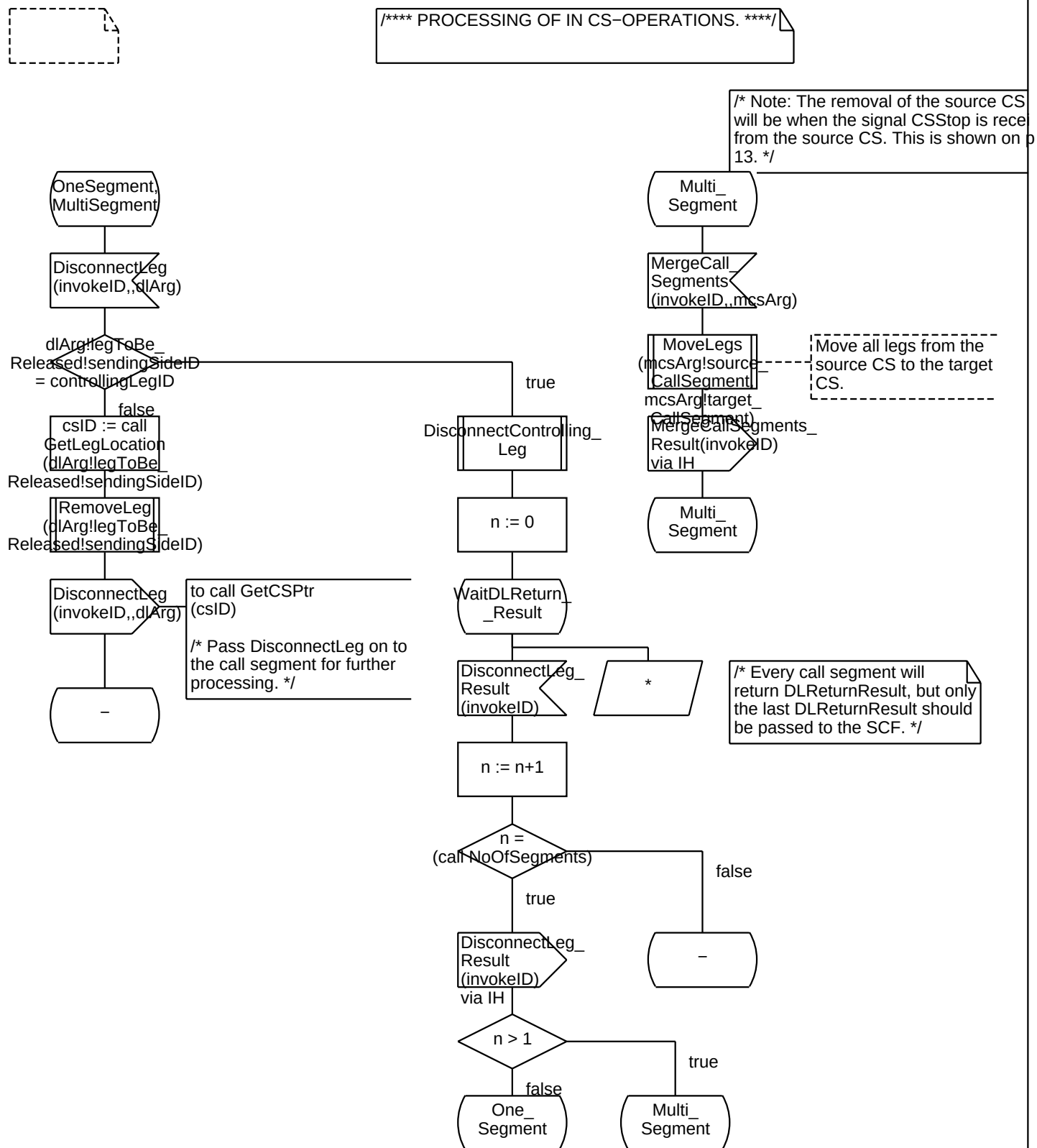
Process Type	CallSegmentAssociation

20(42)



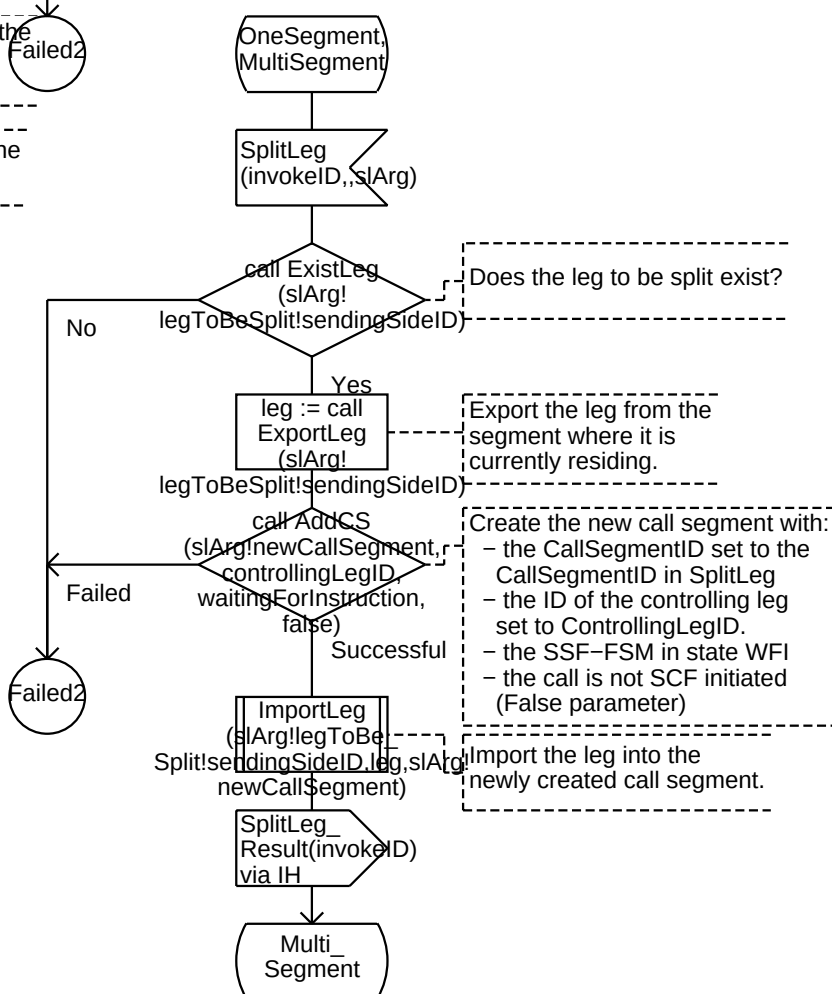
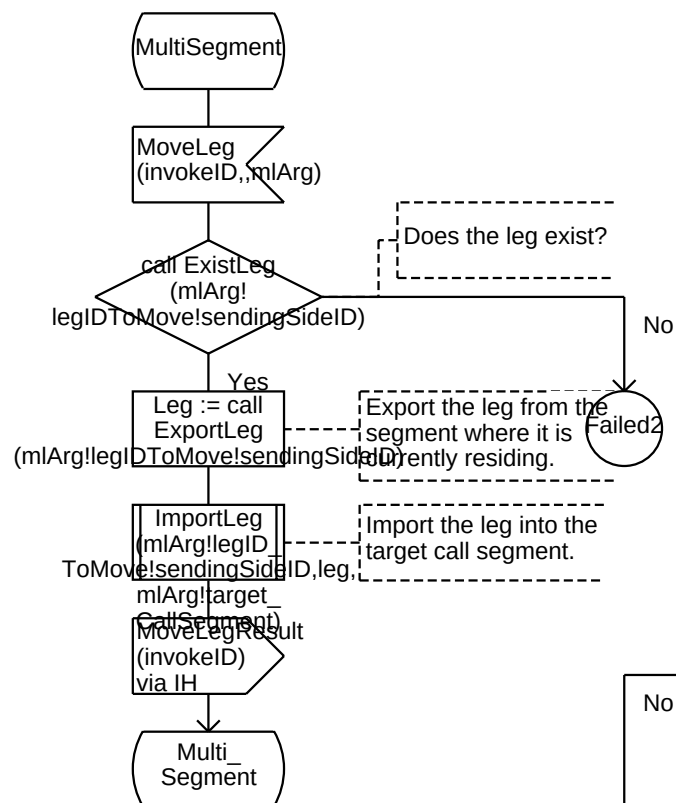
Process Type CallSegmentAssociation

21(42)



Process Type CallSegmentAssociation

22(42)

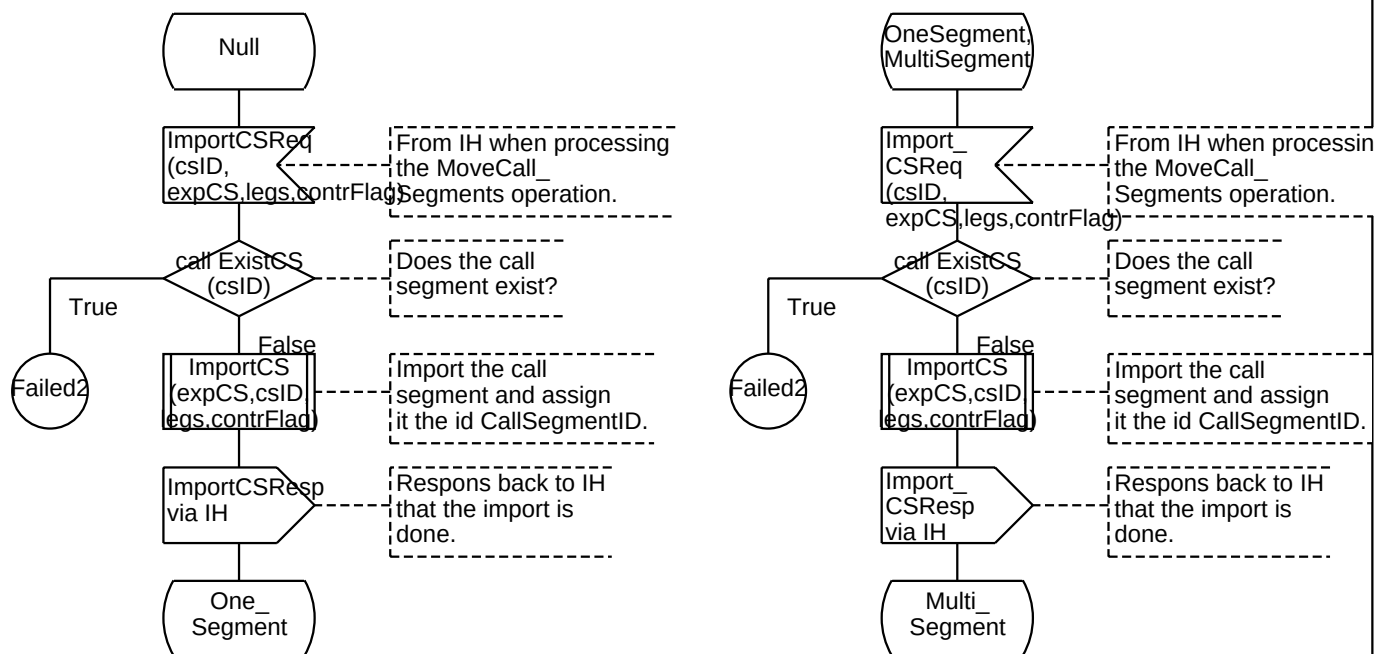


Process Type CallSegmentAssociation

23(42)

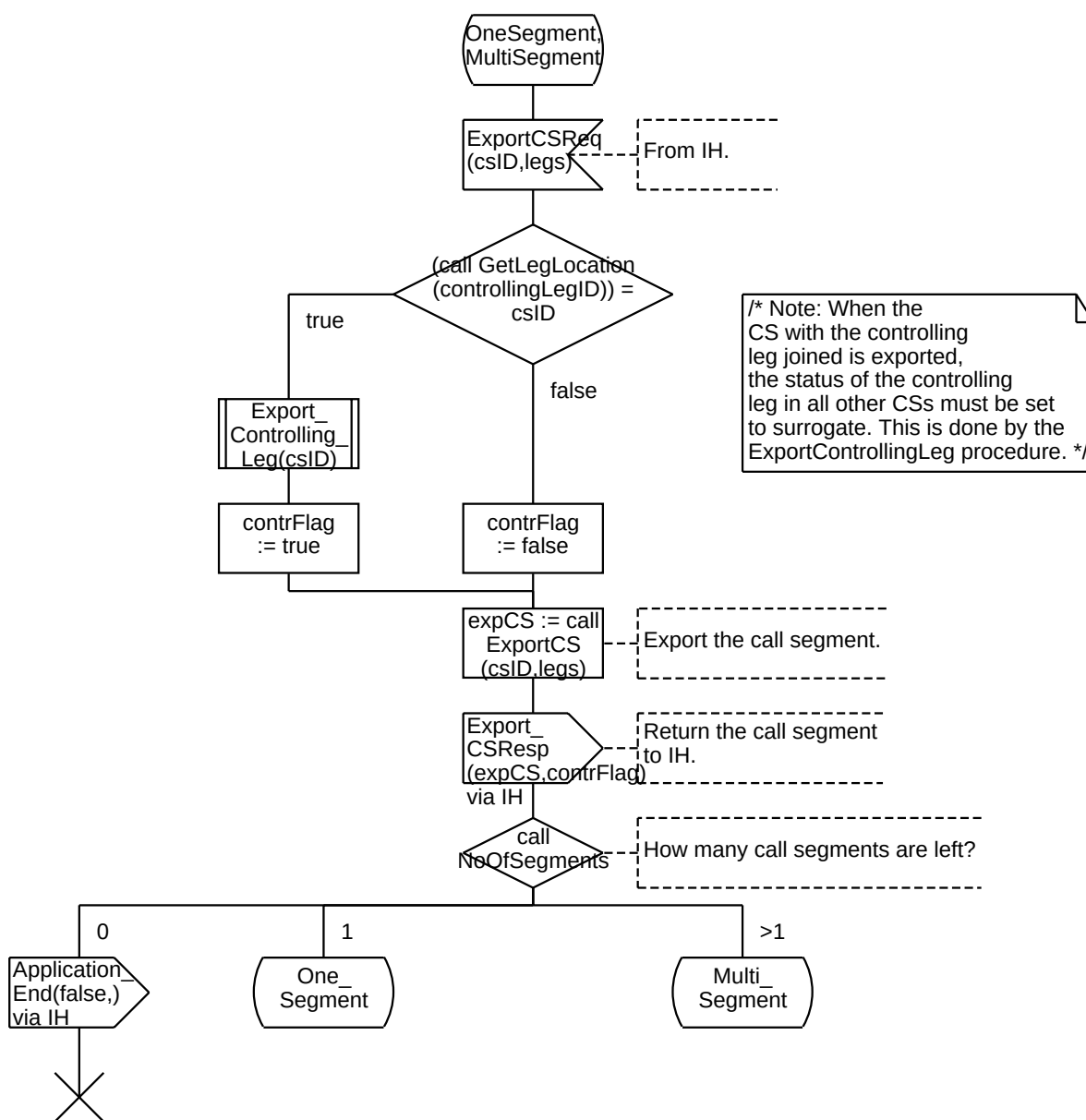


/**** PROCESSING OF ATOMIC OPERATIONS USED BY THE IH. ****/



Process Type CallSegmentAssociation

24(42)

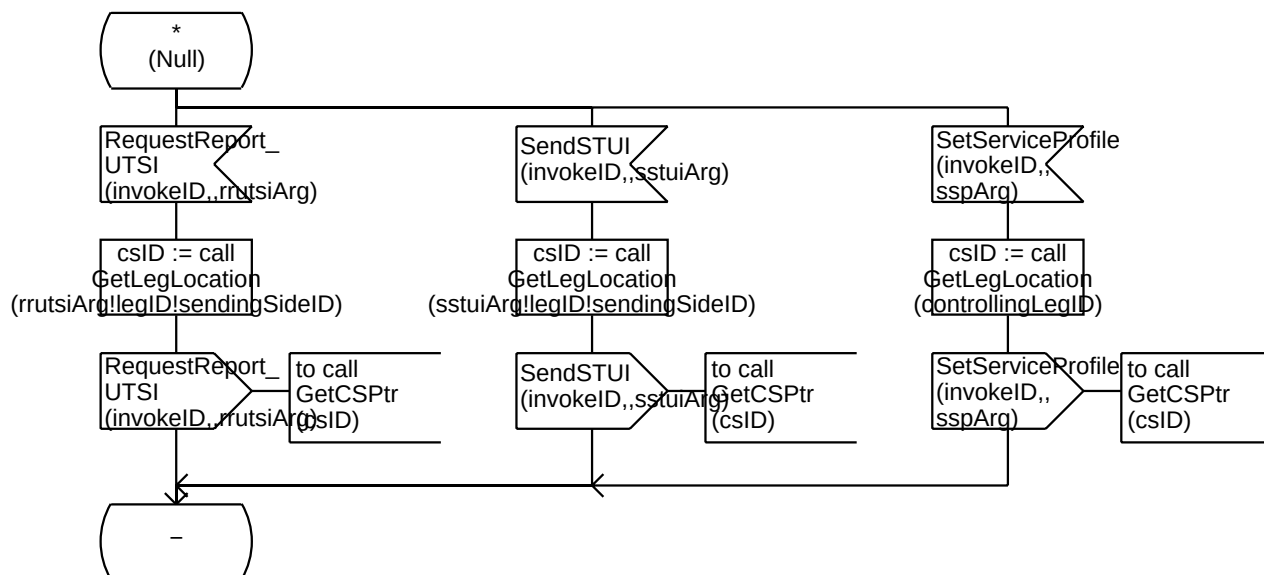
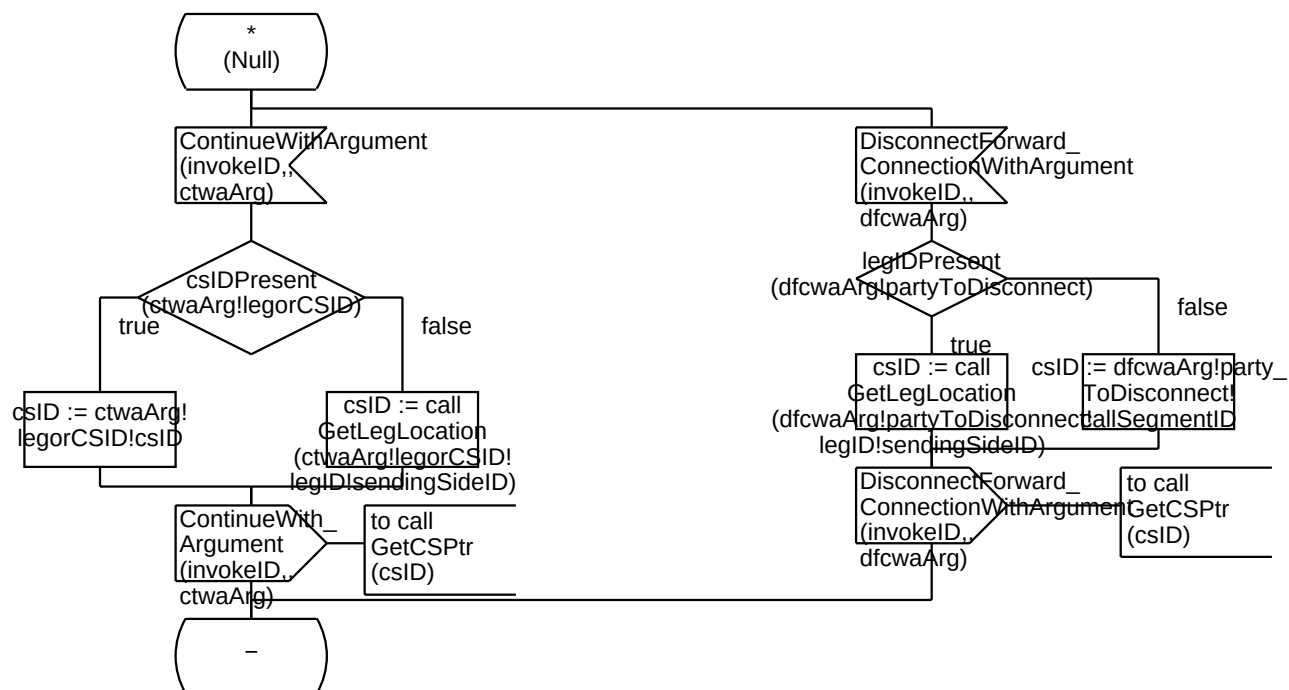


Process Type CallSegmentAssociation

25(42)



**** ROUTING OF IN OPERATIONS FROM THE SCF TO A CALL SEGMENT. ****

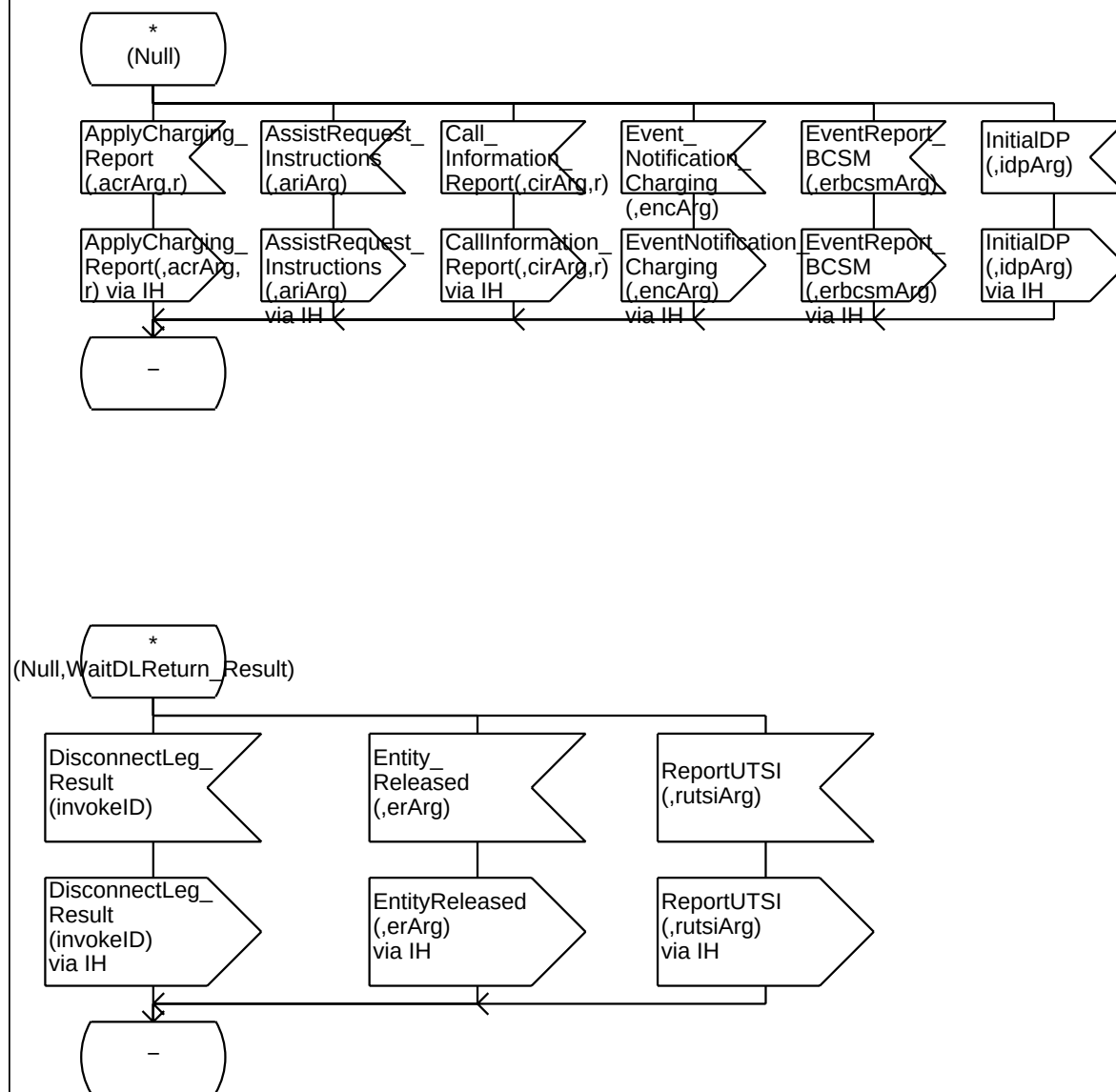


Process Type CallSegmentAssociation

26(42)

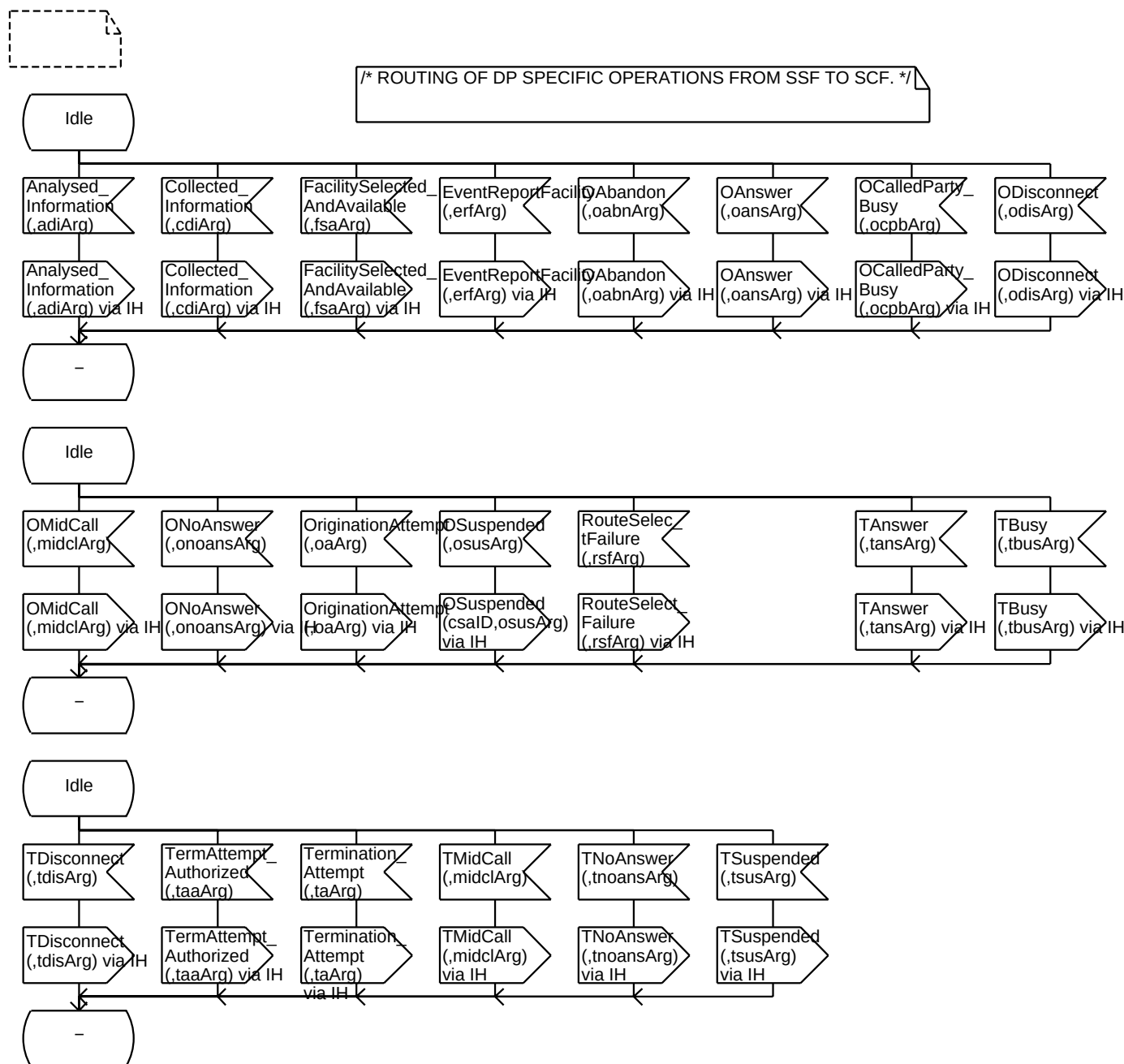


/**** ROUTING OF INAP OPERATIONS TO THE SCF. ****/



Process Type CallSegmentAssociation

27(42)

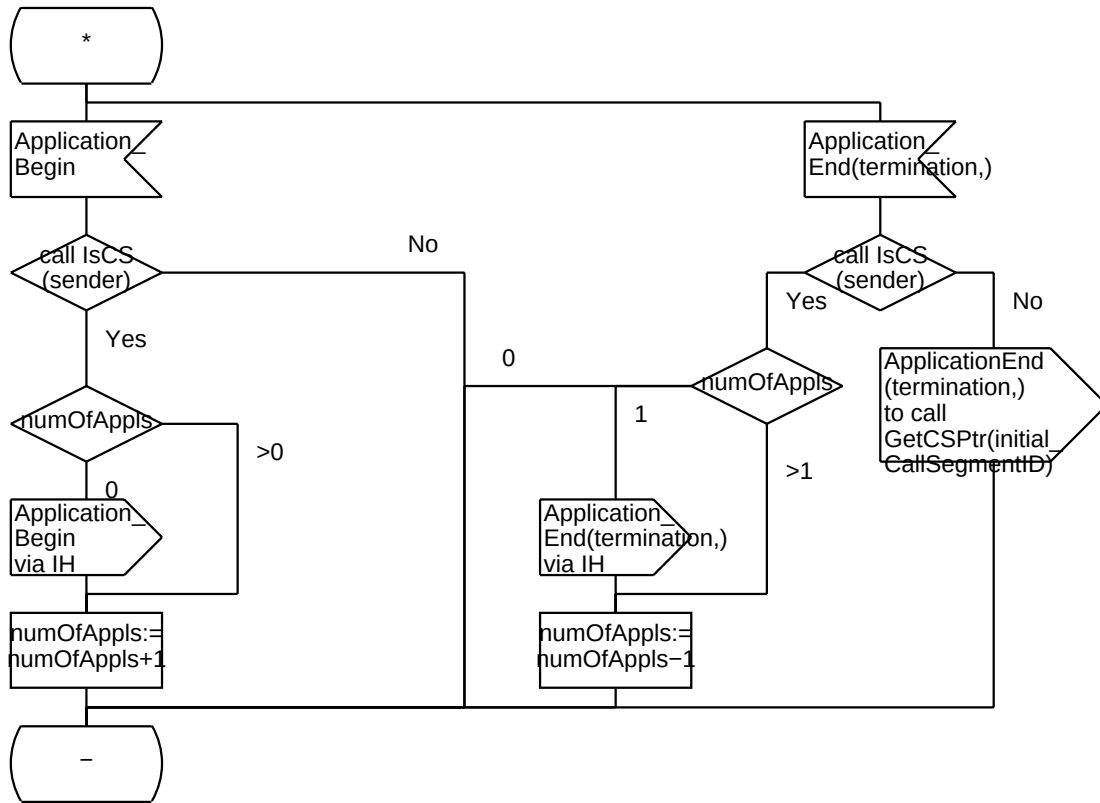


Process Type CallSegmentAssociation

28(42)

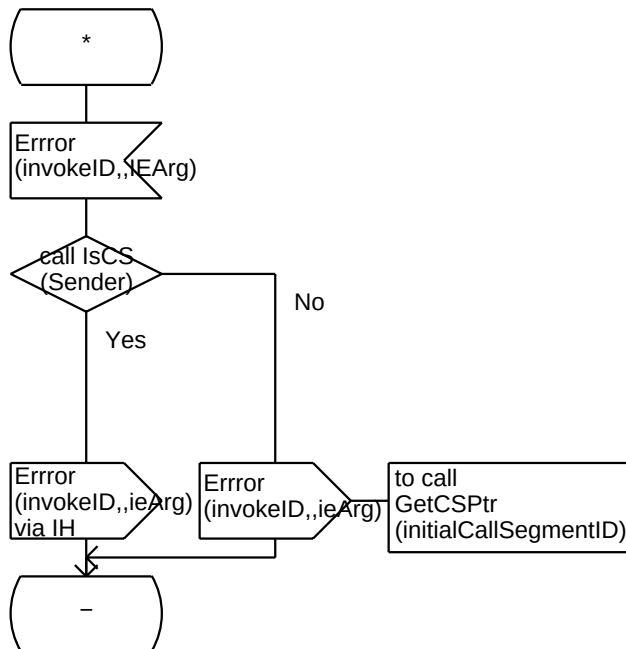
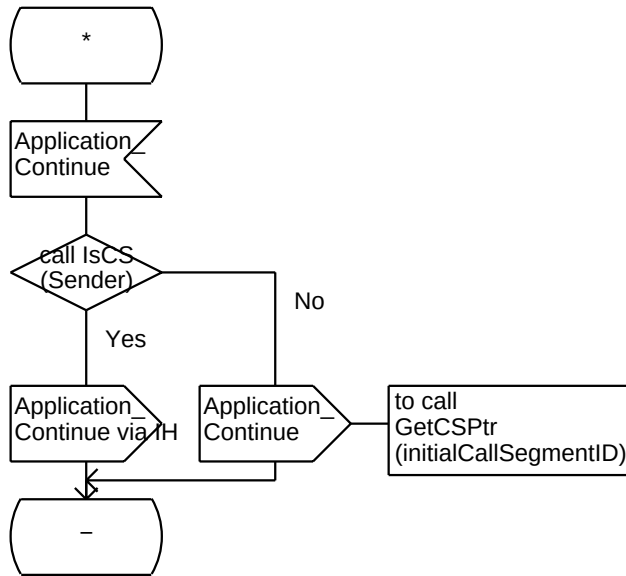


**** ROUTING OF DIALOUGE AND ERROR PRIMITIVES. ****



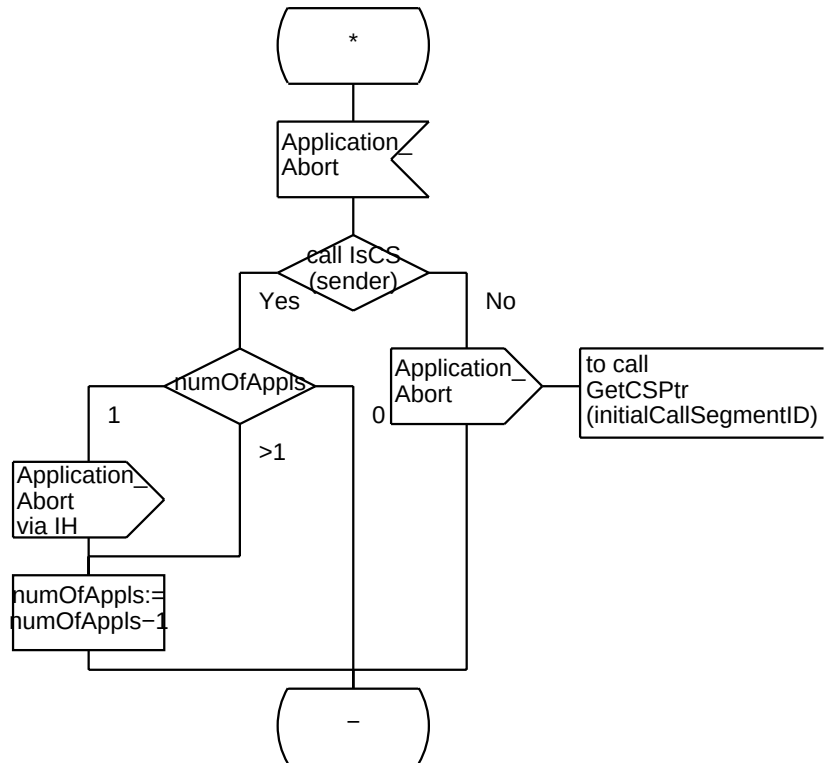
Process Type CallSegmentAssociation

29(42)



Process Type CallSegmentAssociation

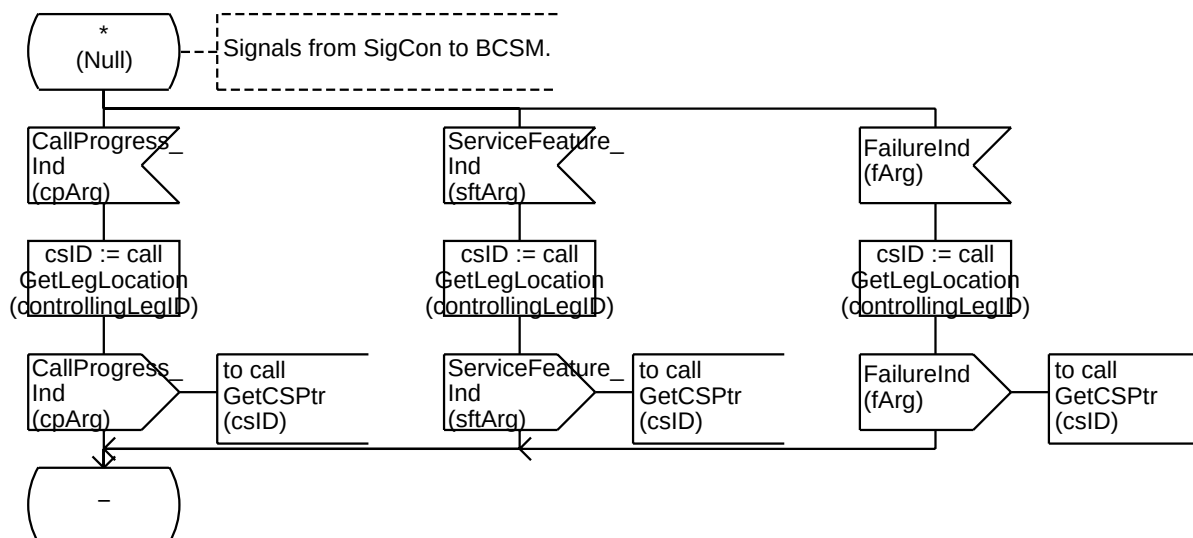
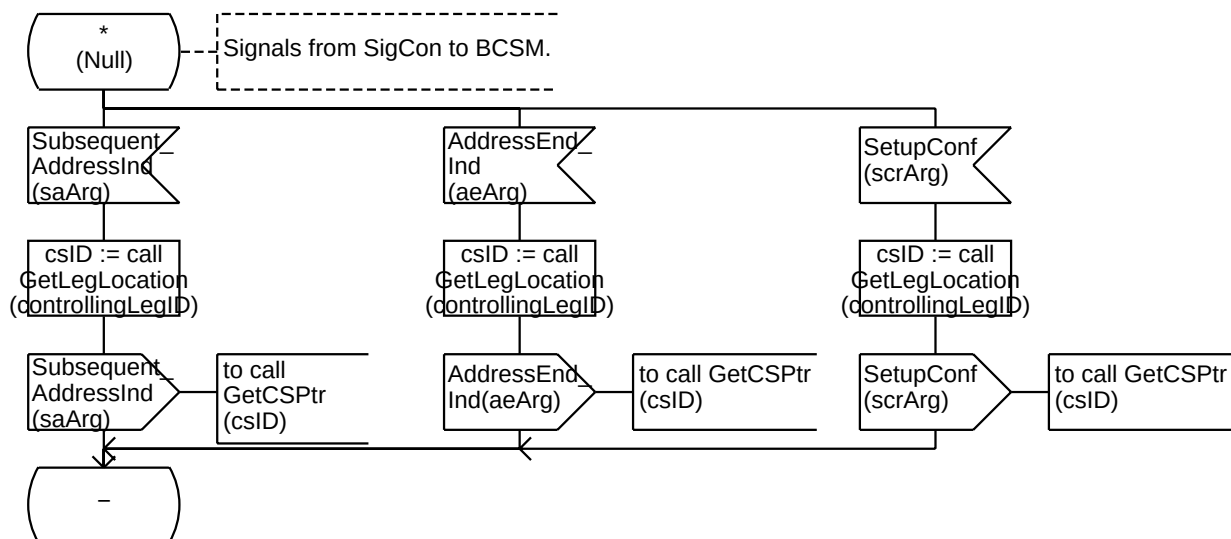
30(42)



Process Type CallSegmentAssociation

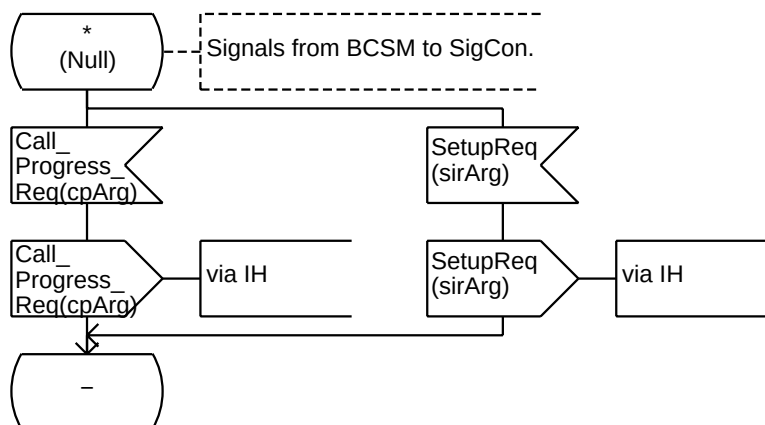
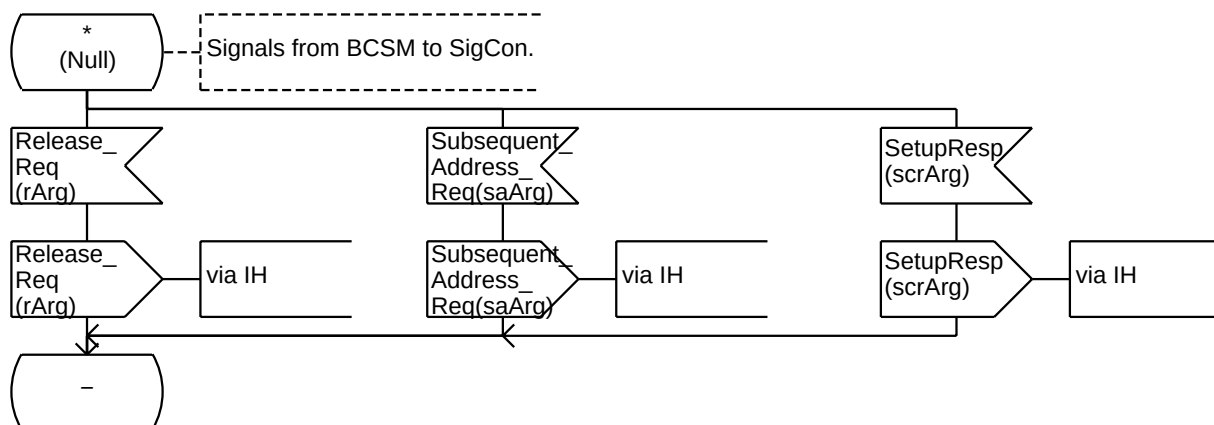
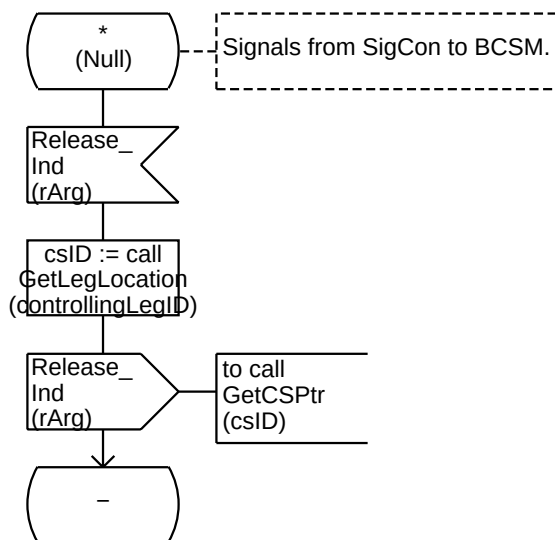
31(42)

/* ROUTING OF SIGNALS TO/FROM SIGCON (VIA THE IH). */



Process Type CallSegmentAssociation

32(42)

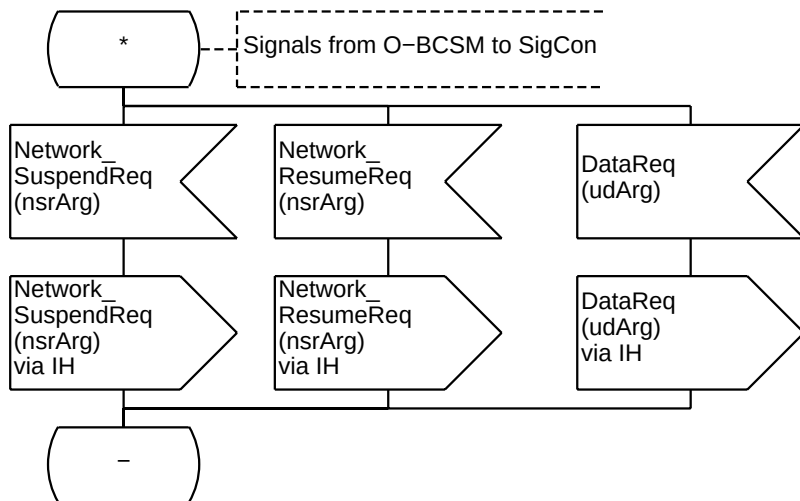
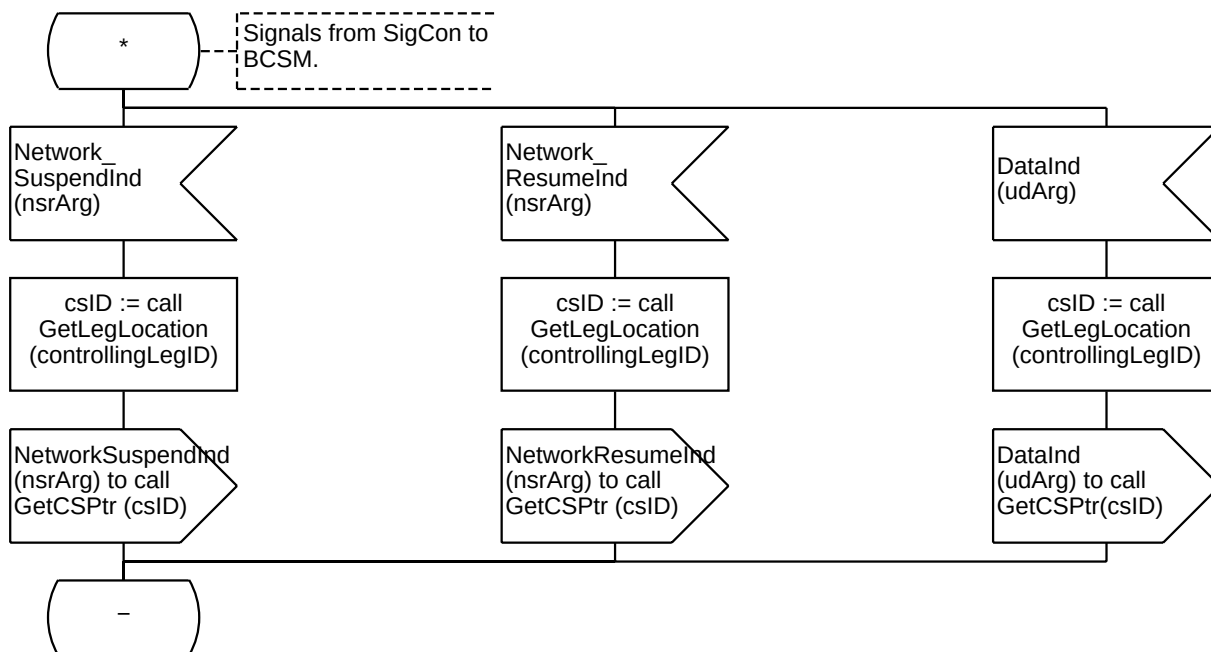


Process Type CallSegmentAssociation

33(42)

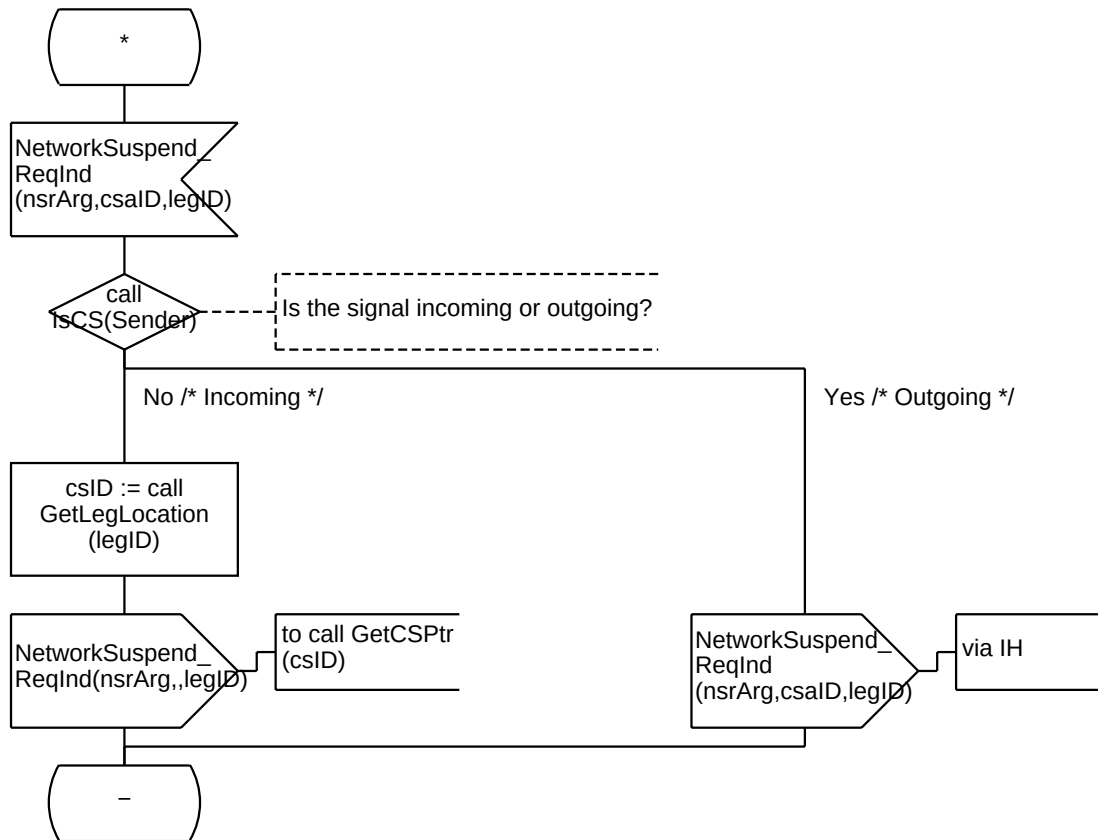


**** ROUTING OF SIGNALLING CONTROL PRIMITIVES TO/FROM SIGCON. ****



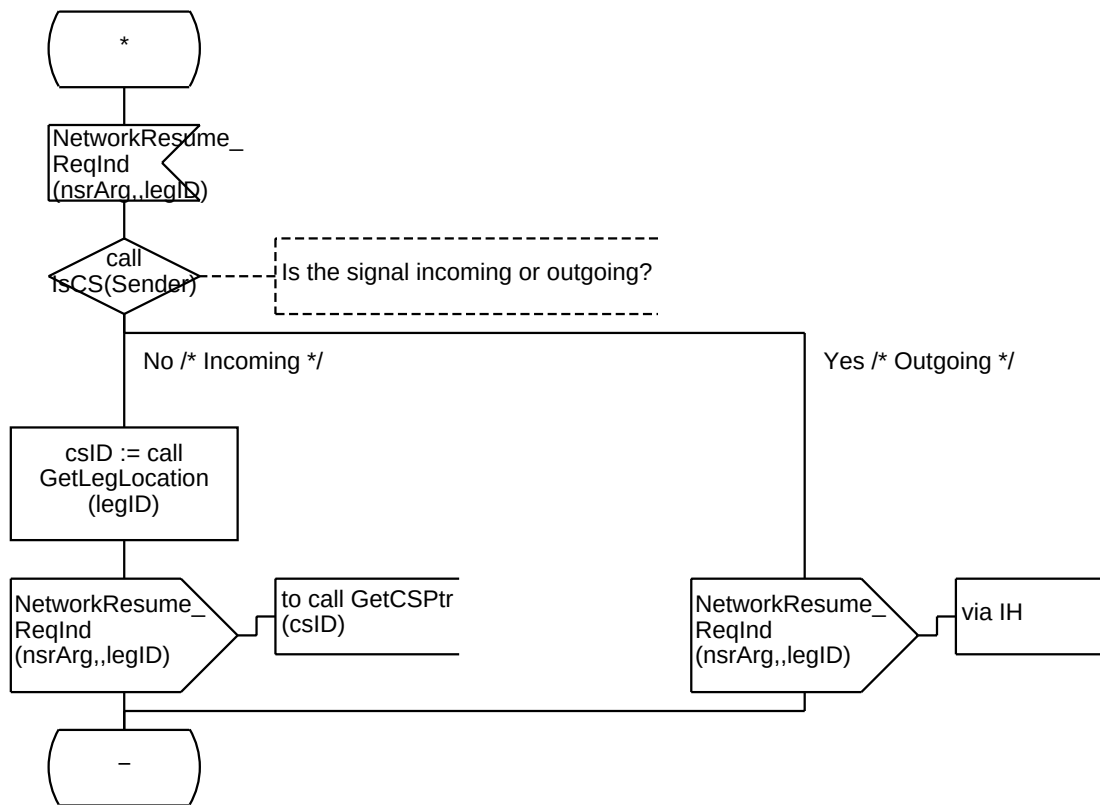
Process Type CallSegmentAssociation

34(42)



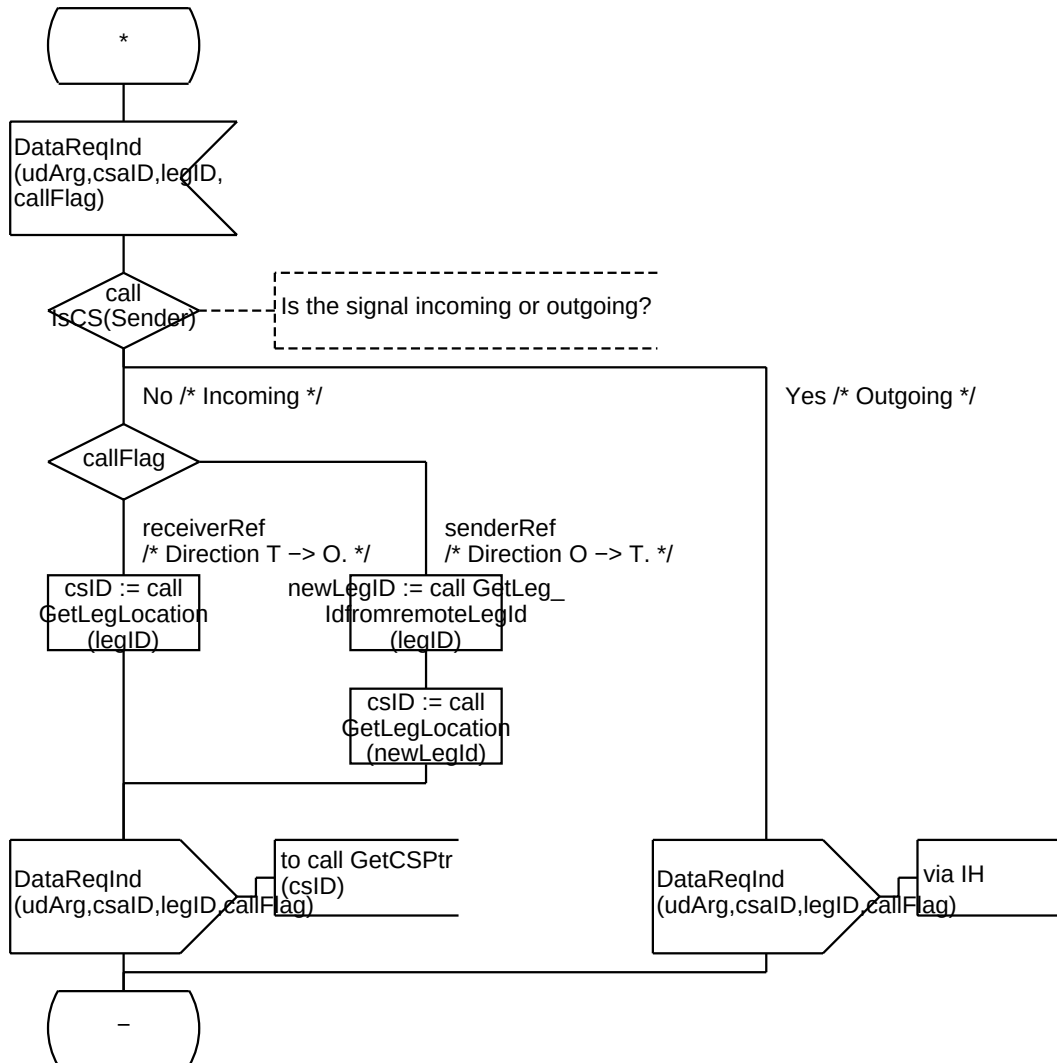
Process Type CallSegmentAssociation

35(42)



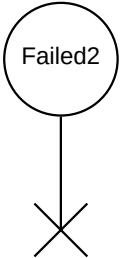
Process Type CallSegmentAssociation

36(42)



Process Type CallSegmentAssociation

37(42)

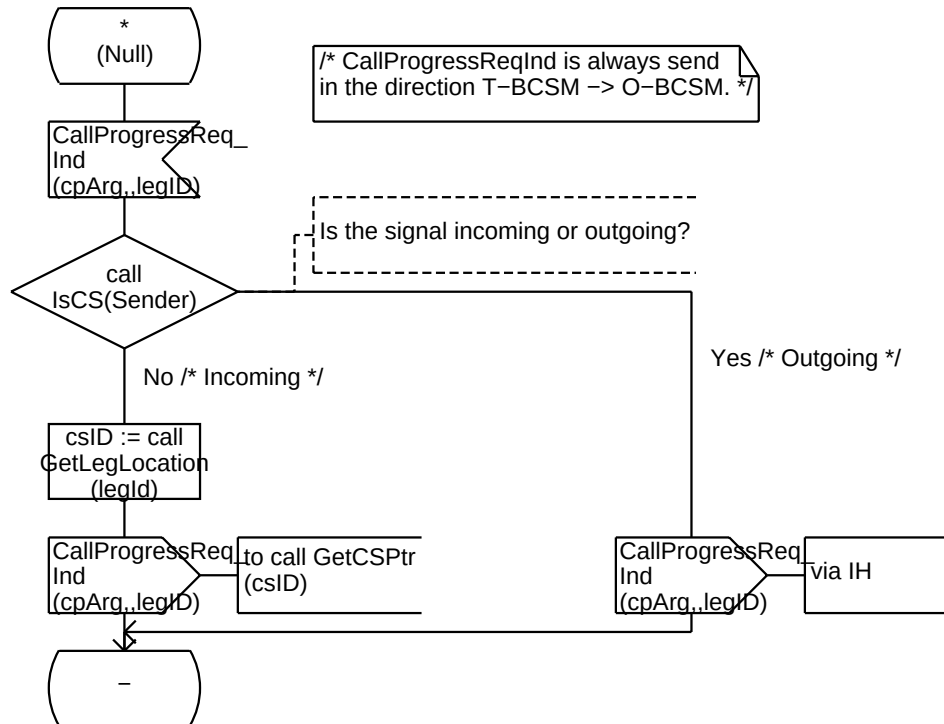


Process Type CallSegmentAssociation

38(42)

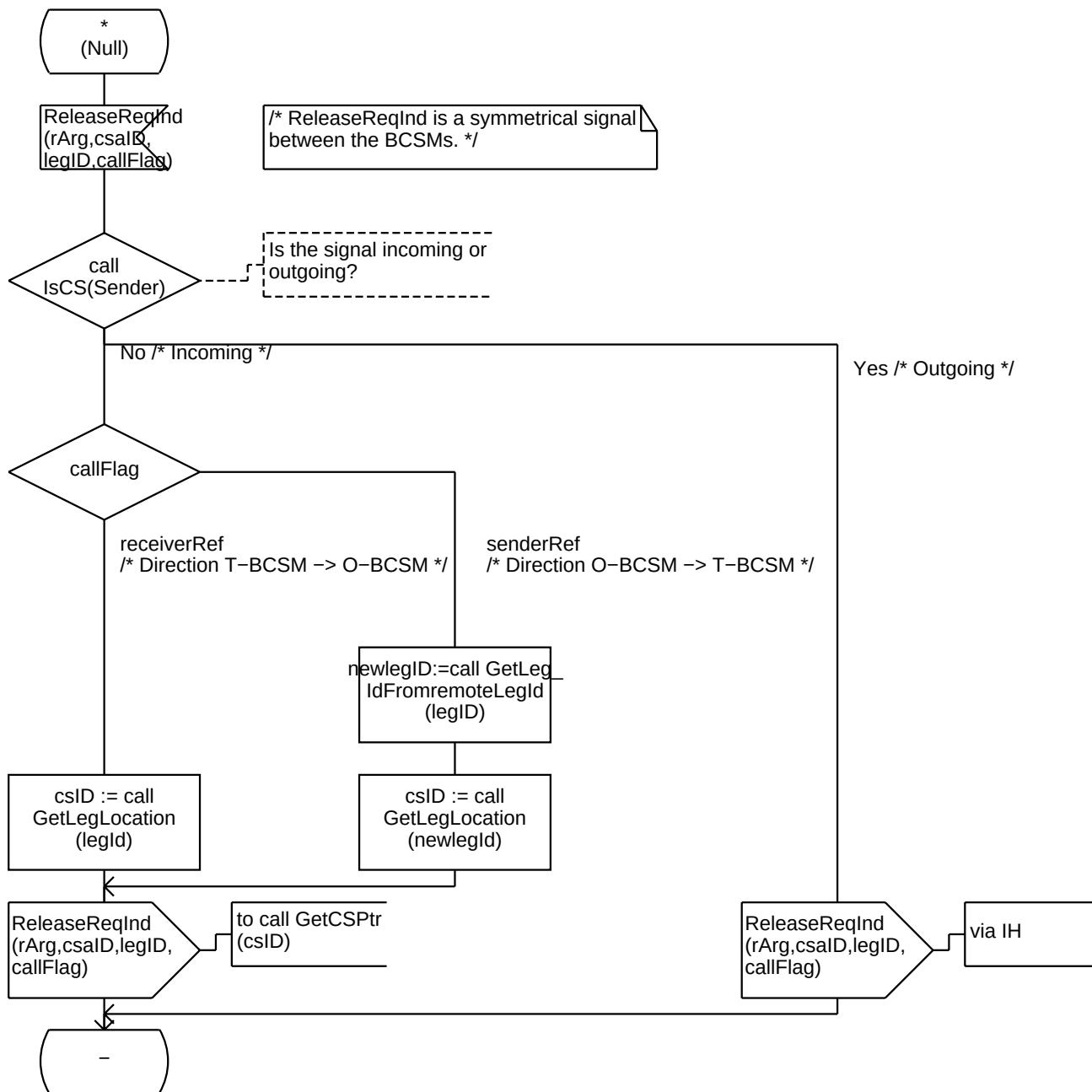


**** ROUTING OF SIGNALS BETWEEN BCSMs ****



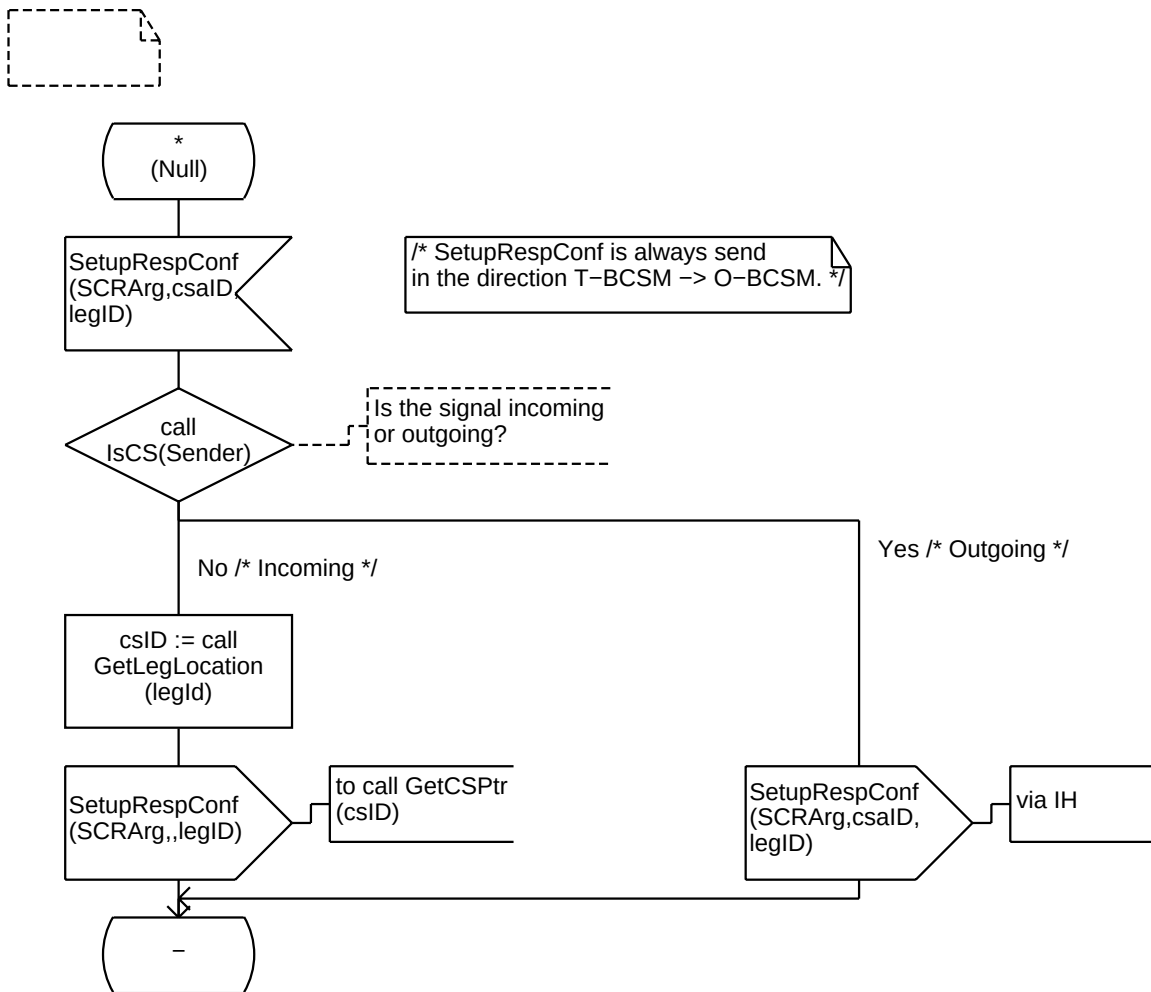
Process Type CallSegmentAssociation

39(42)



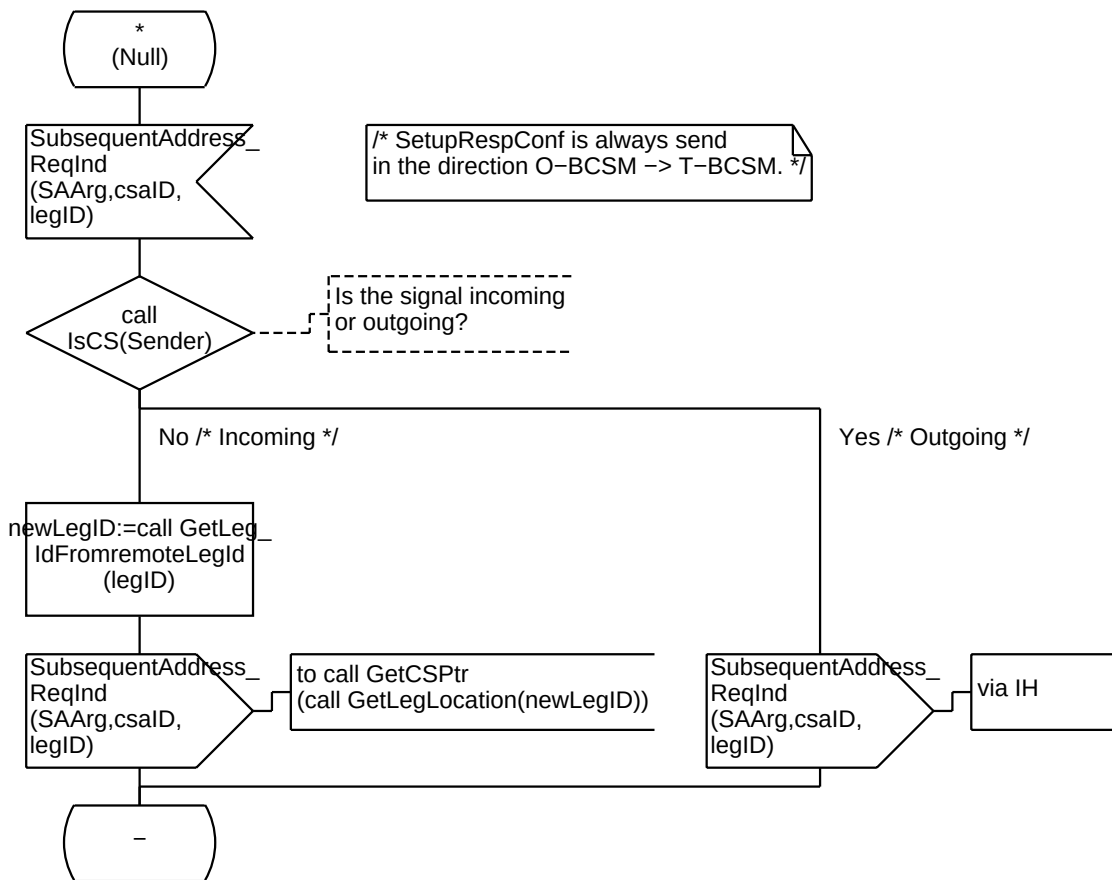
Process Type CallSegmentAssociation

40(42)

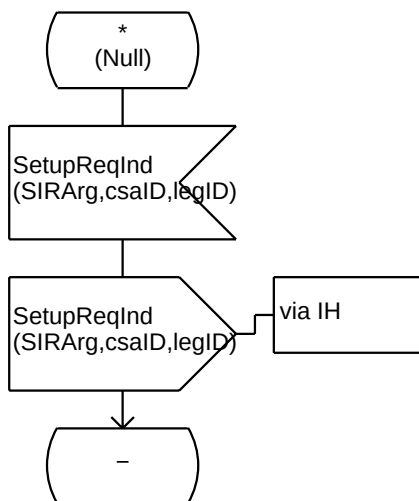


Process Type CallSegmentAssociation

41(42)



/* In all states except Null, SetupReqInd is always outgoing. */

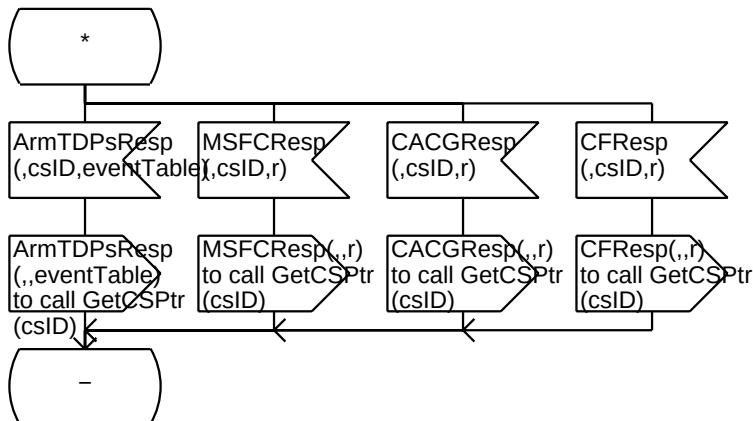
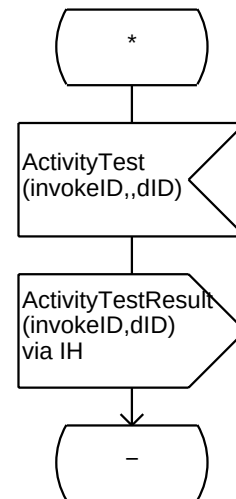
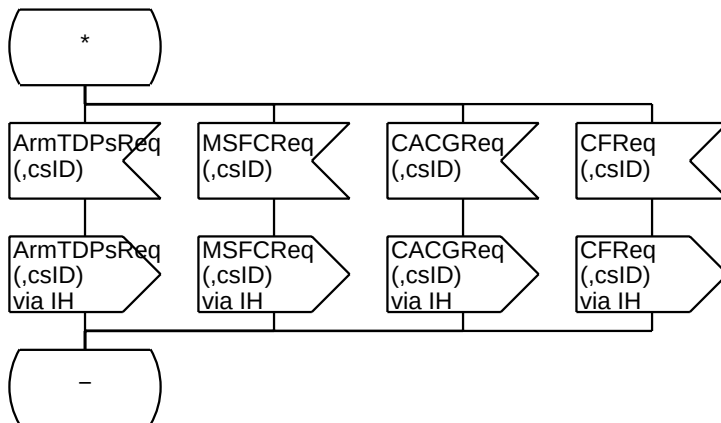


Process Type CallSegmentAssociation

42(42)



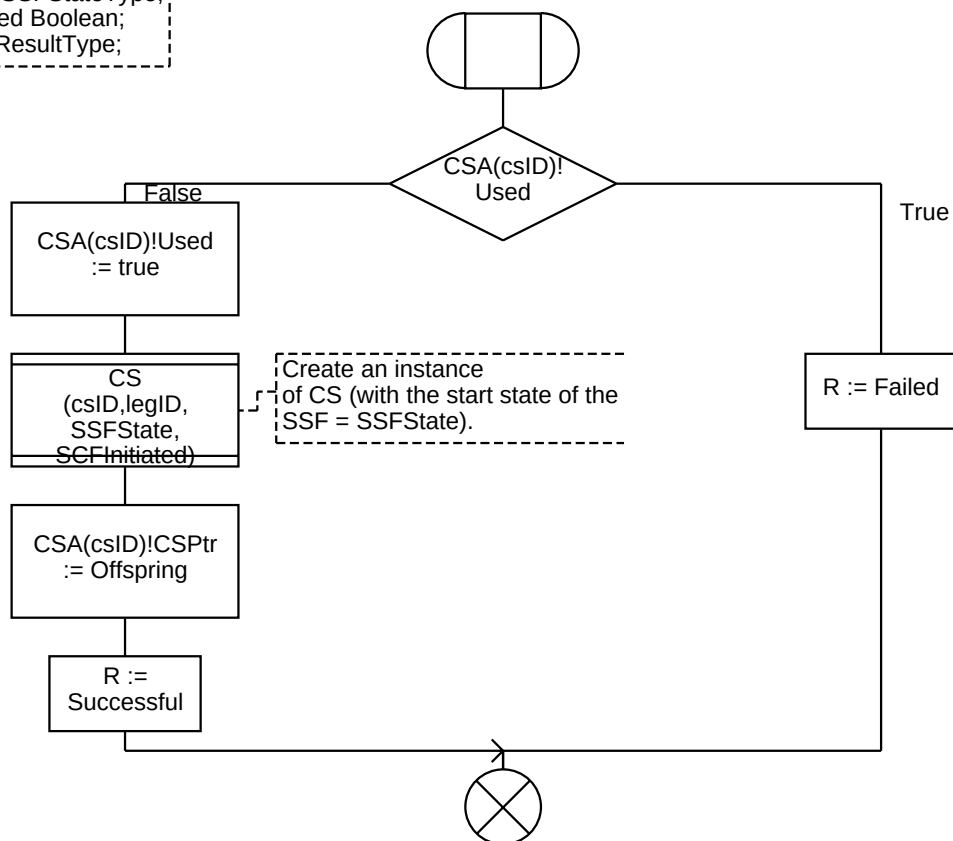
/* ROUTING OF SIGNALS BETWEEN THE SSF-FSM AND THE SSME-FSM */



Procedure <<Block Type SSF_CCF/Process Type CallSegmentAssociation>> AddCS

1(1)

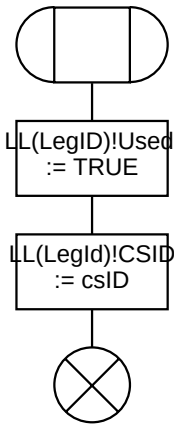
;FPAR
IN csID CallSegmentID,
IN legID LegType,
IN SSFState SSFStateType,
IN SCFInitiated Boolean;
RETURNS R ResultType;



Procedure SetLegLocation

1(1)

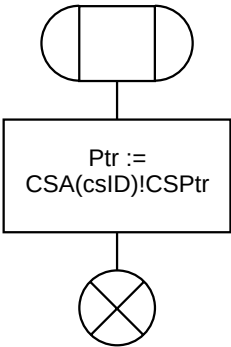
;FPAR
IN LegId LegType,
IN csID CallSegmentID;



Procedure GetCSPtr

1(1)

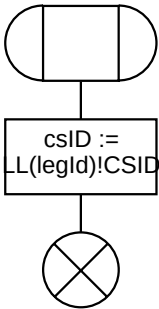
;FPAR
IN csID CallSegmentID;
RETURNS Ptr PId;



Procedure GetLegLocation

1(1)

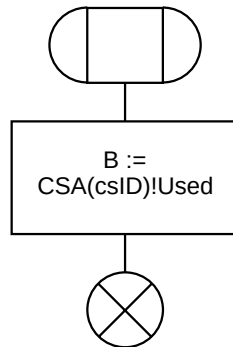
;FPAR
IN legId LegType;
RETURNS csID CallSegmentID;



Procedure <<Block Type SSF_CCF/Process Type CallSegmentAssociation>> ExistCS

1(1)

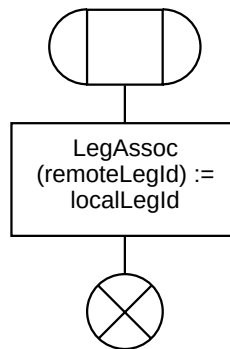
```
;FPAR
IN csID CallSegmentID;
RETURNS B Boolean;
```



Procedure <<Process Type CallSegmentAssociation>> SetLegAssoc

1(1)

```
;FPAR
IN remoteLegId LegType;
IN localLegId LegType;
```

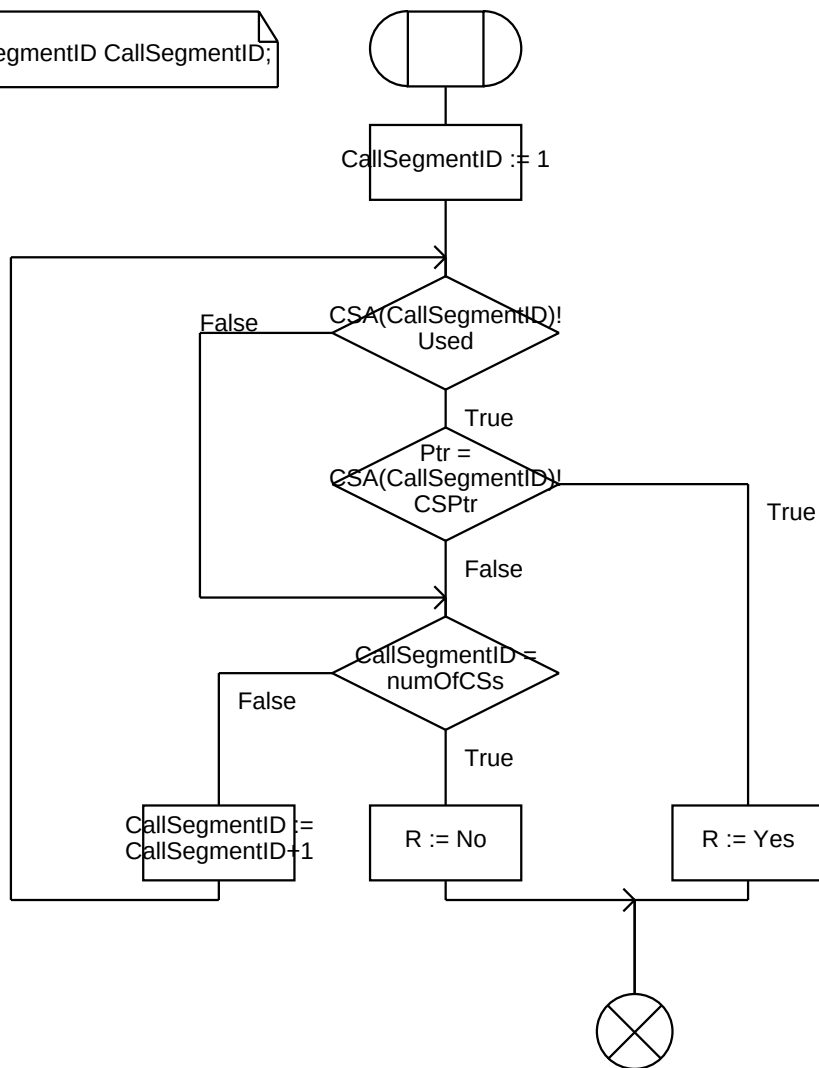


Procedure IsCS

1(1)

;FPAR
IN Ptr Pid;
RETURNS R AnswerType;

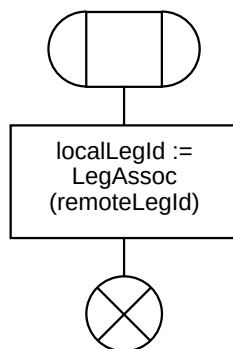
DCL
CallSegmentID CallSegmentID;



Procedure <<Process Type CallSegmentAssociation>> GetLegIdfromRemoteLegId

1(1)

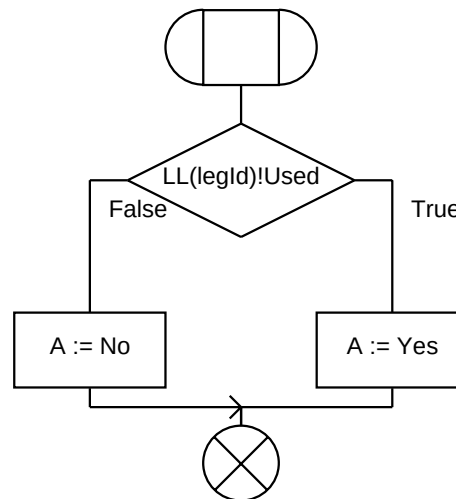
```
;FPAR
IN remoteLegId LegType;
RETURNS localLegId LegType;
```



Procedure <<Process Type CallSegmentAssociation>> ExistLeg

1(1)

```
;FPAR  
IN legId LegType;  
RETURNS A AnswerType;
```

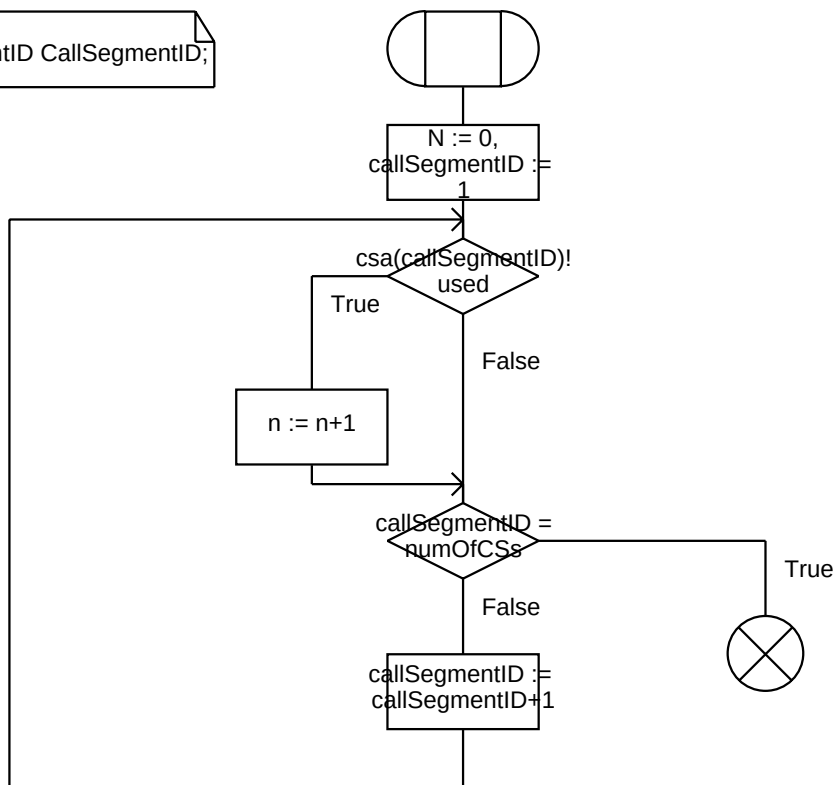


Procedure NoOfSegments

1(1)

RETURNS N Integer;

DCL
callSegmentID CallSegmentID;

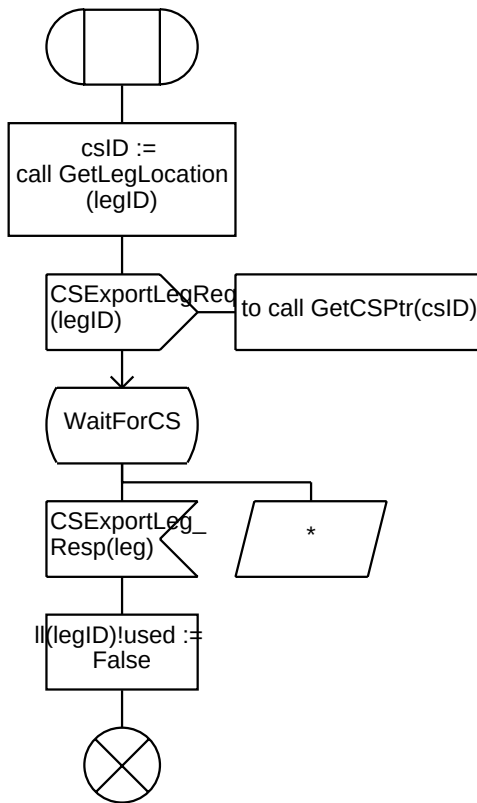


Procedure <<Process Type CallSegmentAssociation>> ExportLeg

1(1)

FPAR
IN legID LegType;
RETURNS leg ExportLegType;

DCL
csID CallSegmentID;



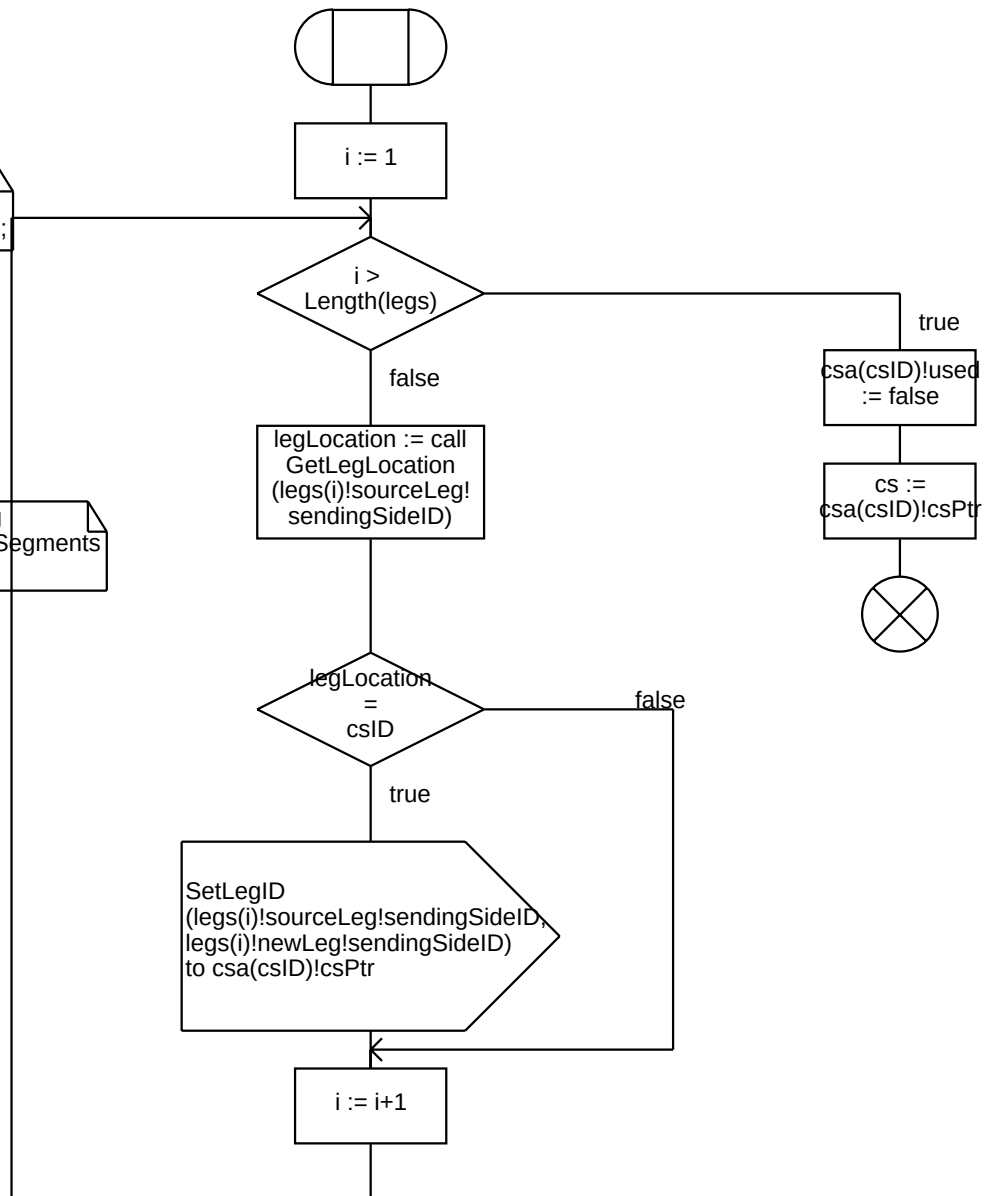
Procedure ExportCS

1(1)

FPAR
IN csID CallSegmentID,
IN legs ExportLegsType;
RETURNS cs ExportCSType;

DCL
i Integer,
legLocation CallSegmentID;

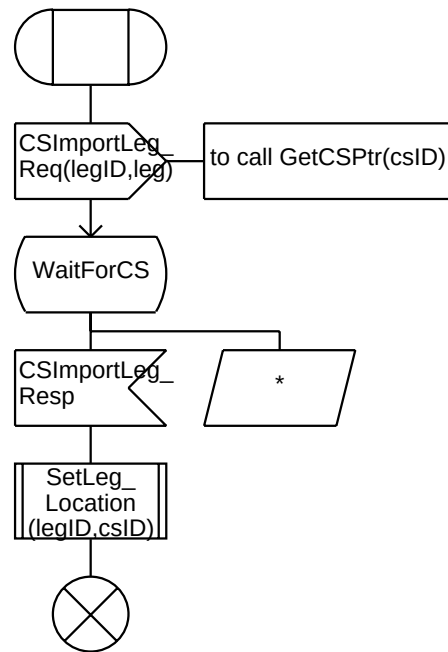
/* Note: Leg renumbering
as required by MoveCallSegments
is performed. */



Procedure <<Process Type CallSegmentAssociation>> ImportLeg

1(1)

FPAR
IN legID LegType,
IN leg ExportLegType,
IN csID CallSegmentID;

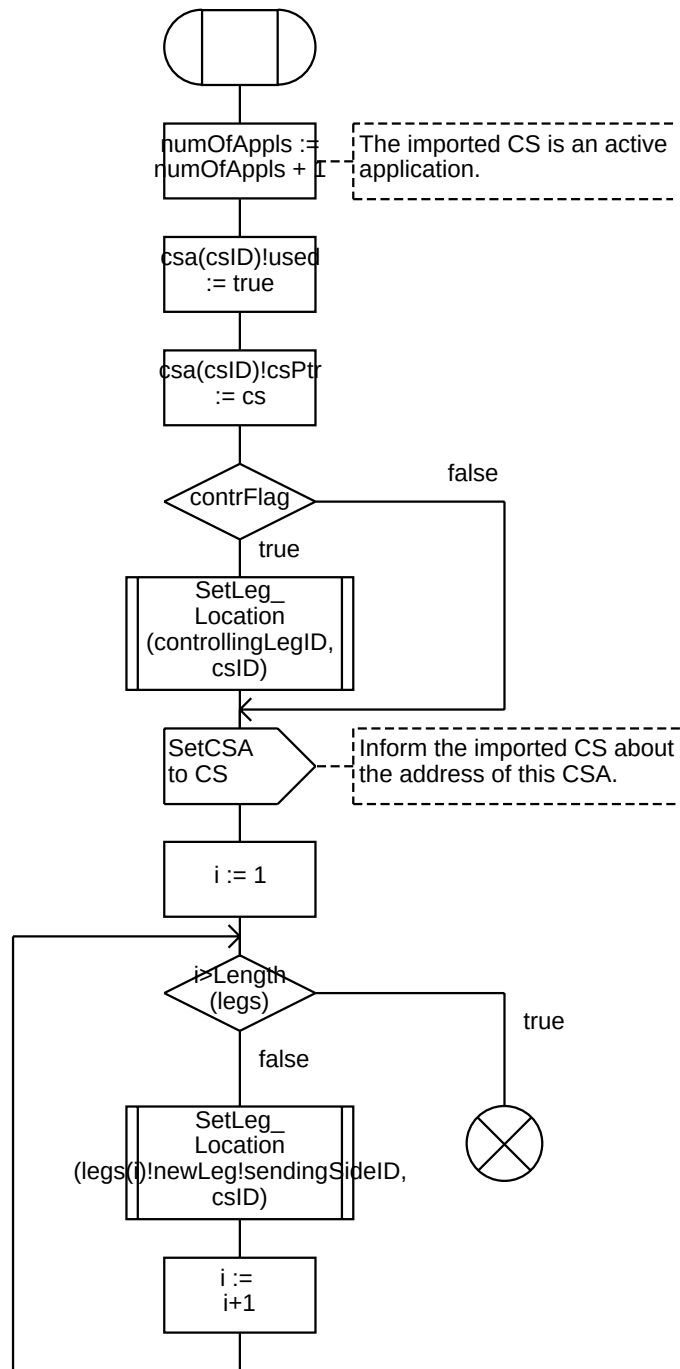


Procedure ImportCS

1(1)

FPAR
 IN cs ExportCSType,
 IN csID CallSegmentID,
 IN legs ExportLegsType,
 IN contrFlag Boolean;

DCL
 i Integer;

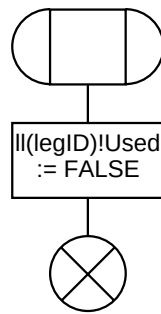


/* Update the leg location table. */

Procedure <<Process Type CallSegmentAssociation>> RemoveLeg

1(1)

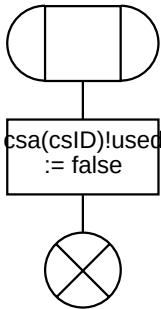
```
;FPAR  
IN legId LegType;
```



Procedure RemoveCS

1(1)

FPAR
IN csID CallSegmentID;

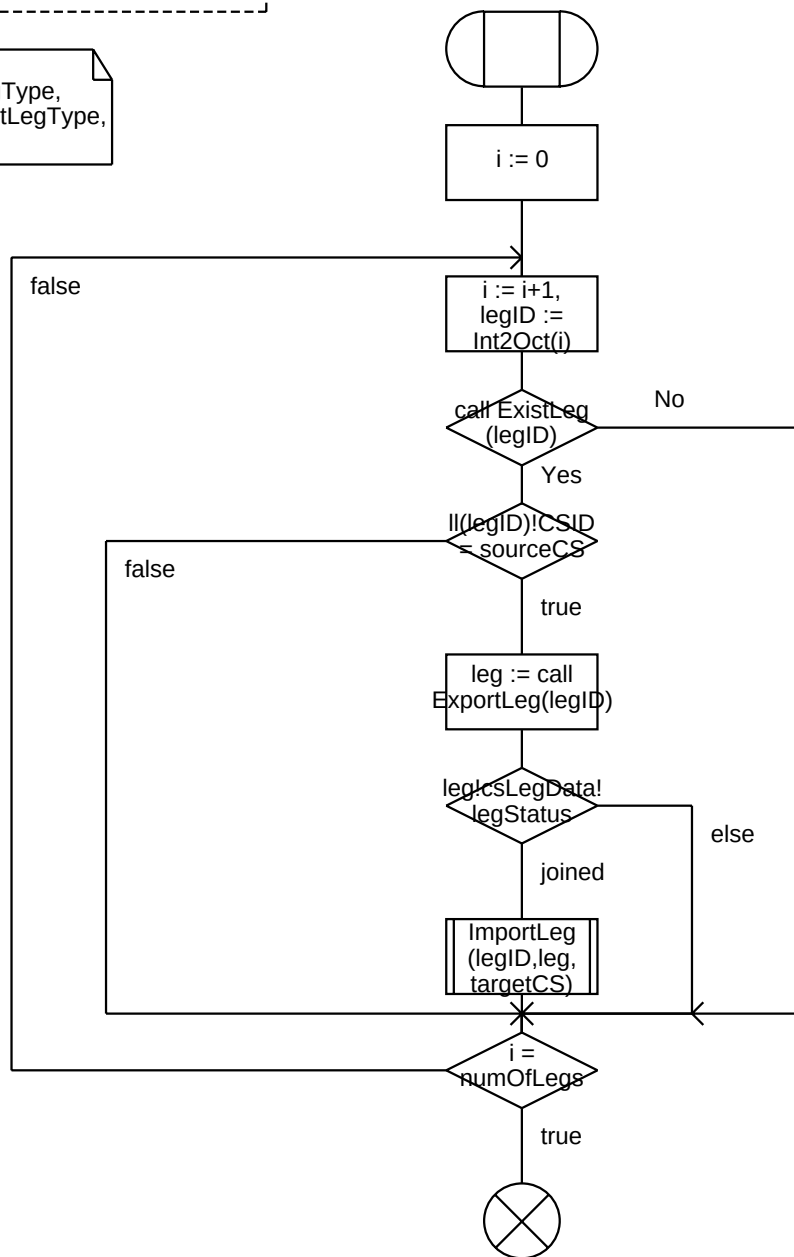


Procedure MoveLegs

1(1)

FPAR
IN sourceCS, targetCS CallSegmentID;

DCL
legID LegType,
leg ExportLegType,
i Integer;

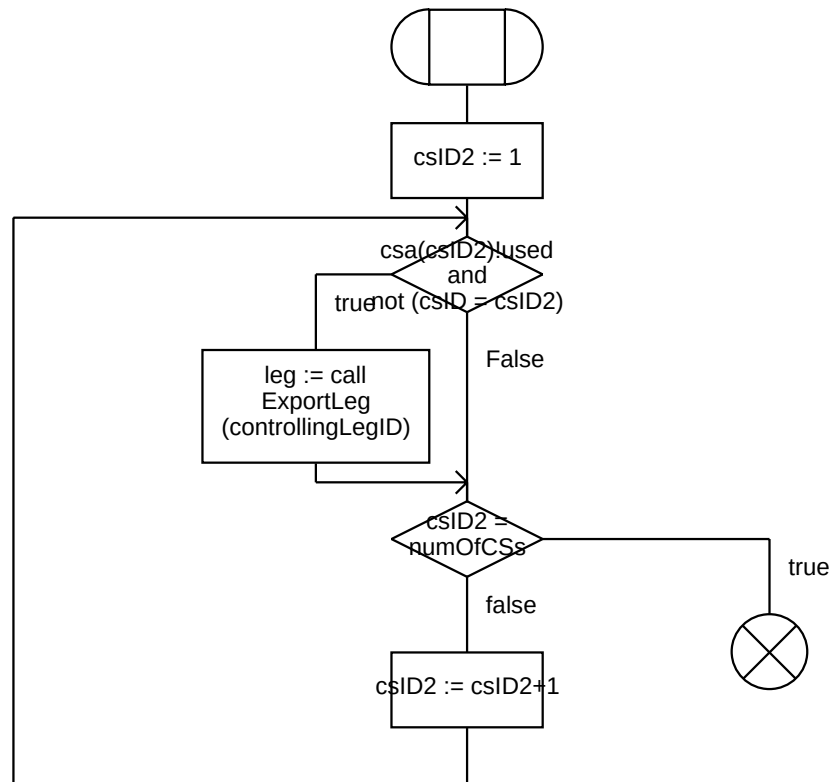


Procedure ExportControllingLeg

1(1)

FPAR
IN csID CallSegmentID;

DCL
csID2 CallSegmentID;

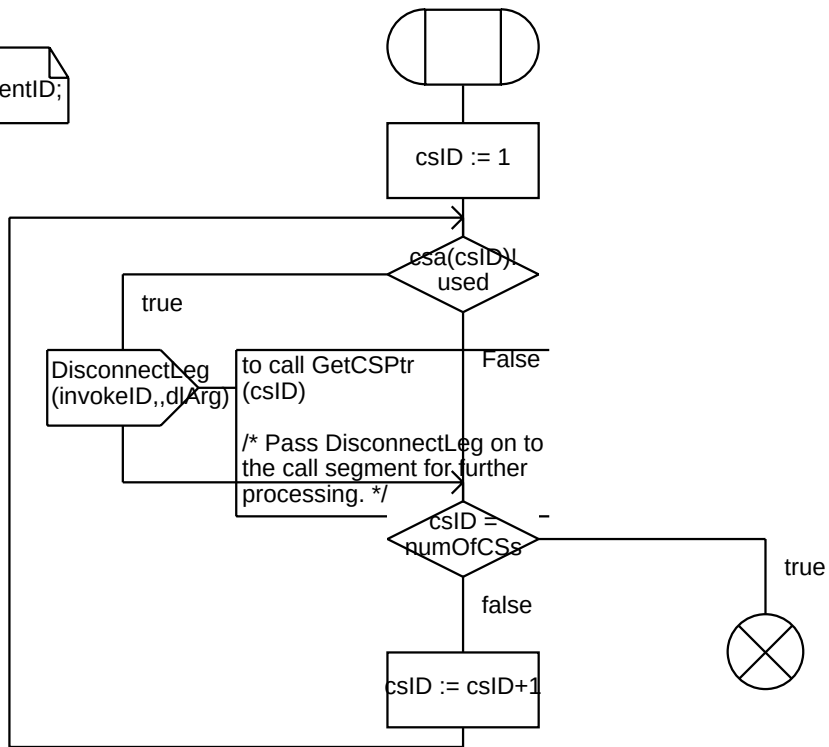


Procedure DisconnectControllingLeg

1(1)



DCL
csID CallSegmentID;

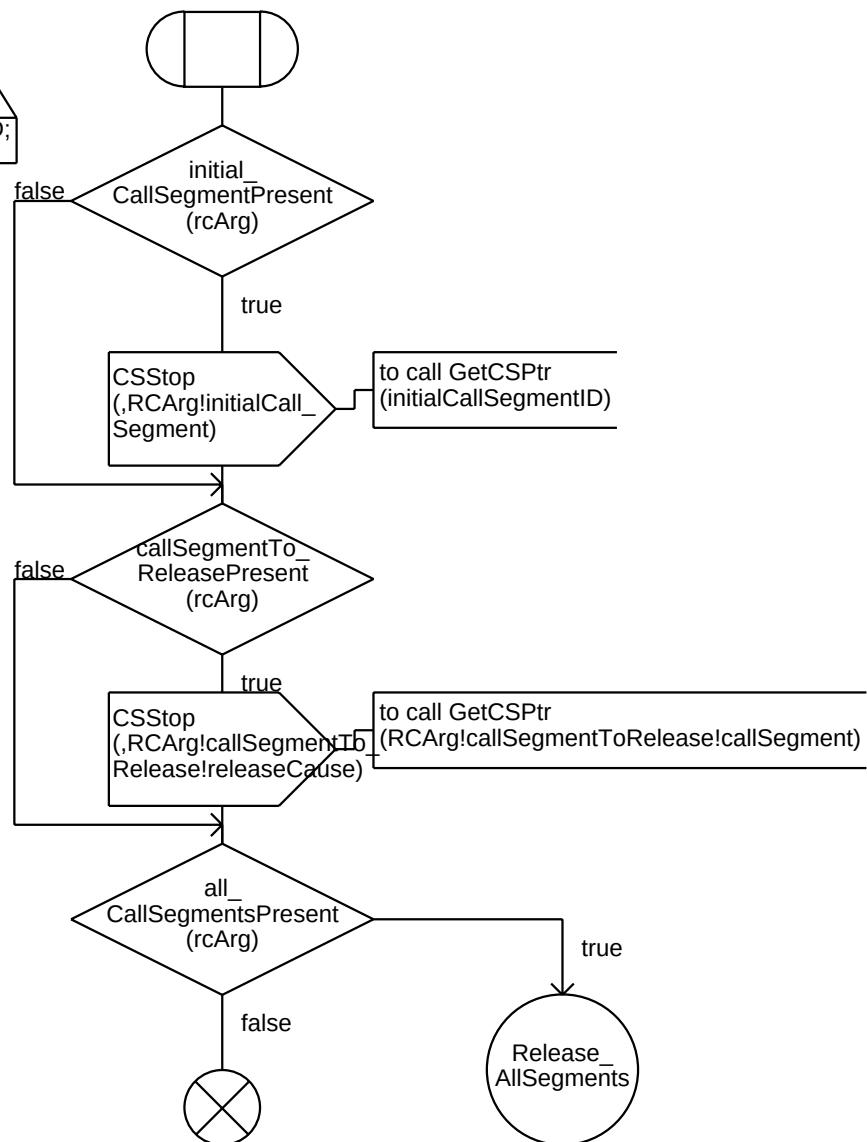


Procedure ReleaseCall

1(2)

FPAR
IN RArg ReleaseCallArg;

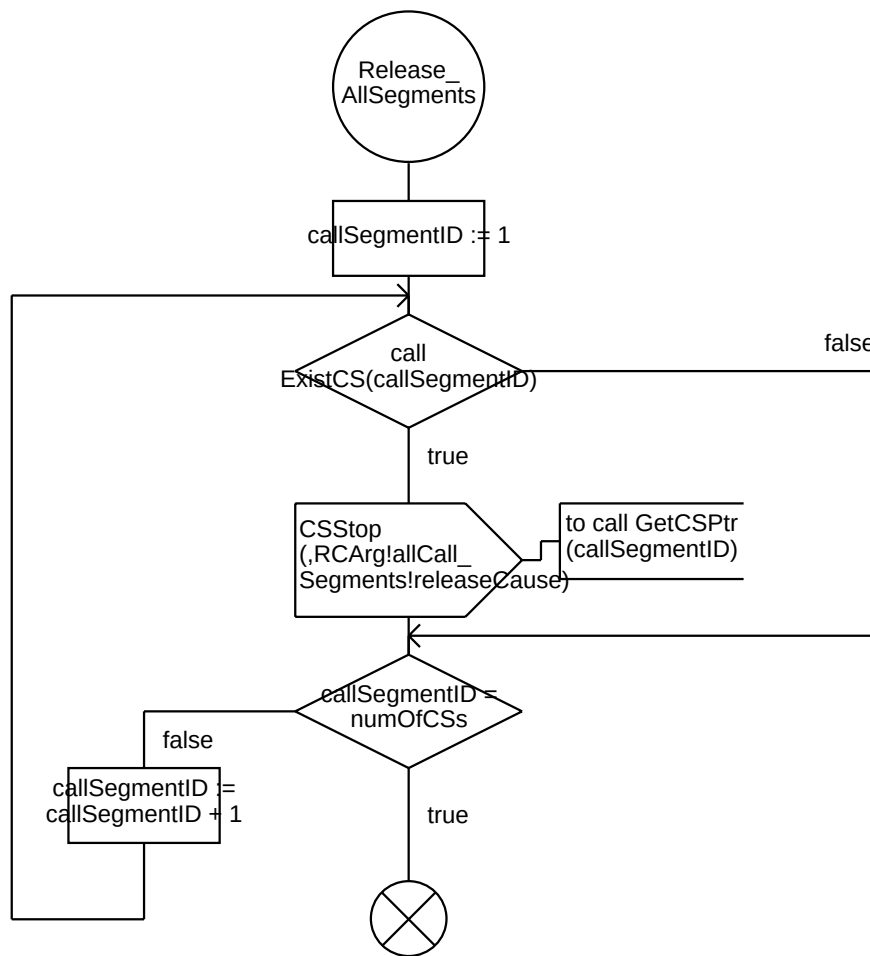
DCL
CallSegmentID CallSegmentID;



Procedure ReleaseCall

2(2)

FPAR
IN RCArg ReleaseCallArg;

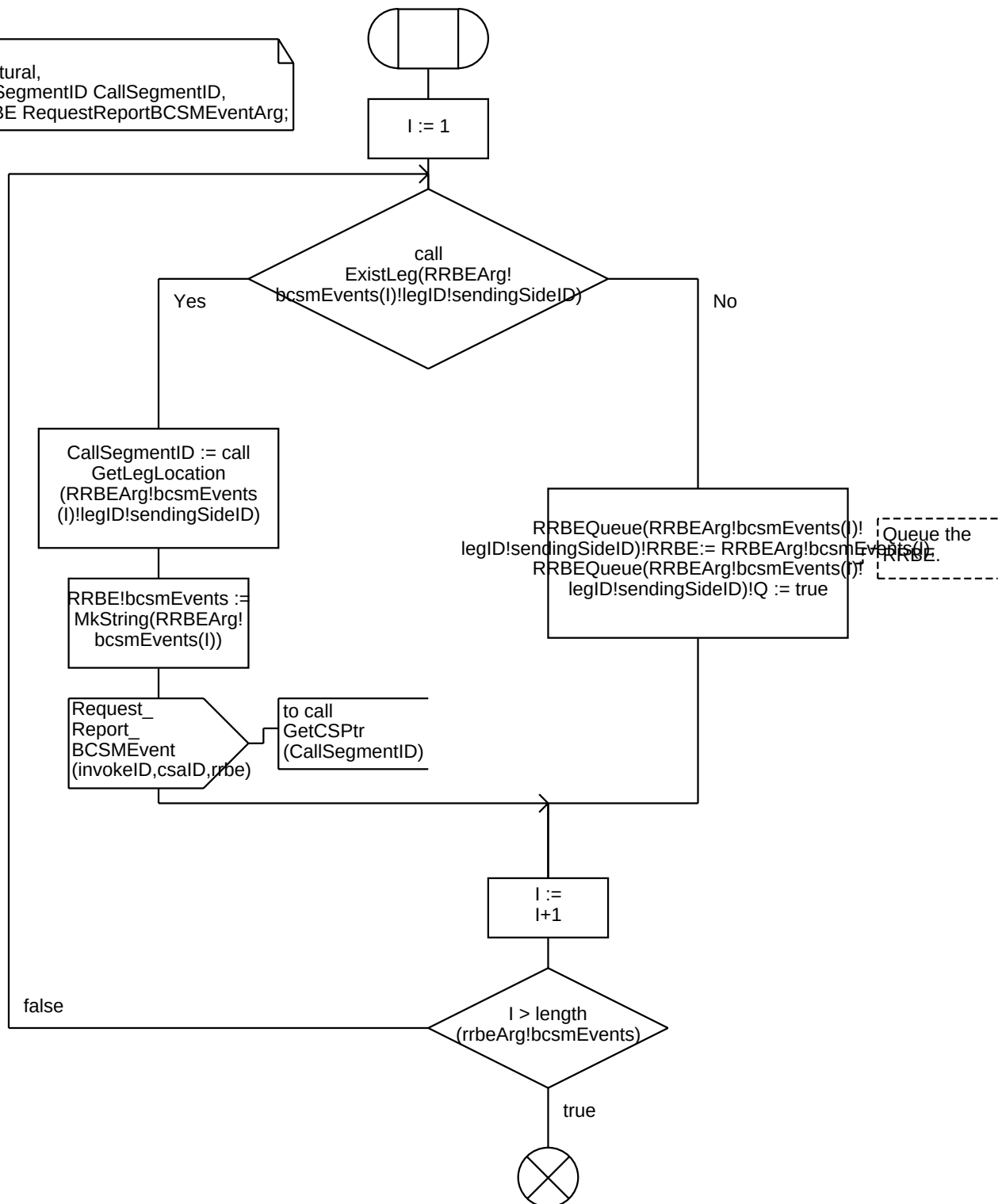


Procedure ProcessRRBE

1(1)

FPAR
IN RRBEArg RequestReportBCSMEventArg;

DCL
I Natural,
callSegmentID CallSegmentID,
RRBE RequestReportBCSMEventArg;

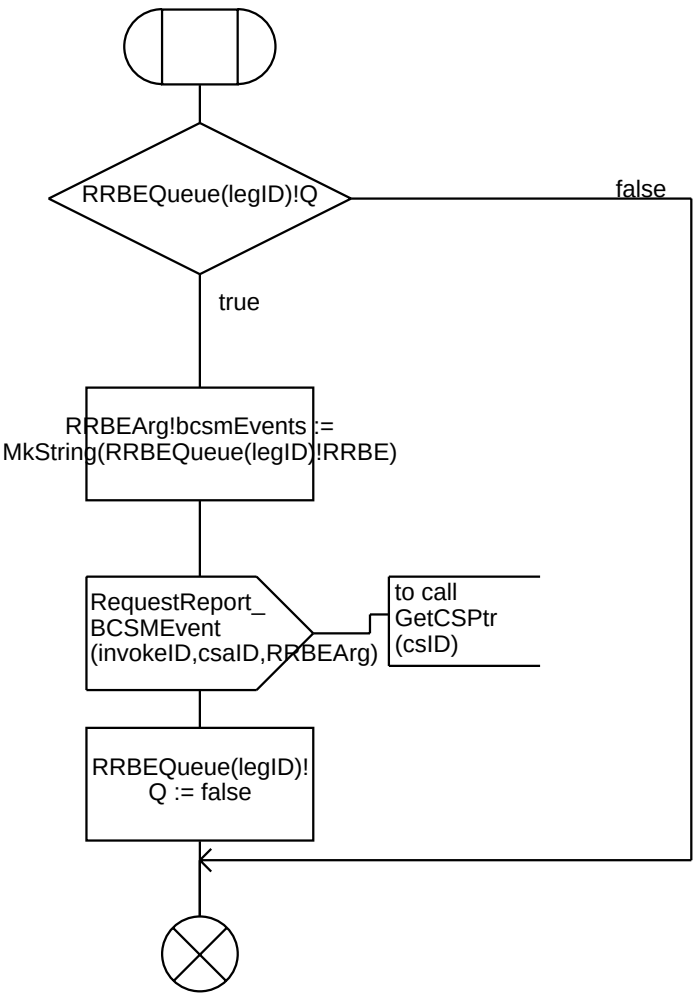


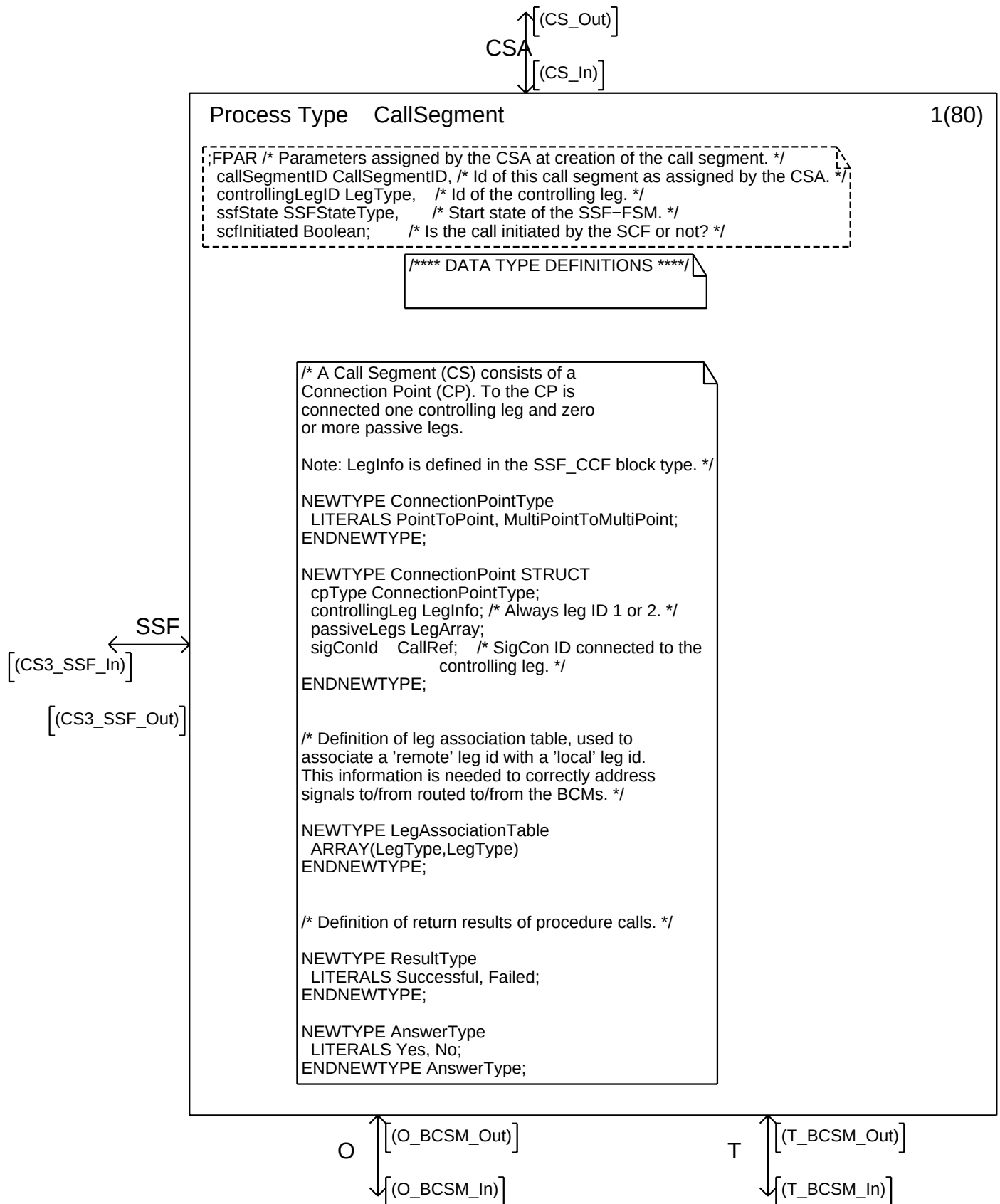
Procedure ProcessQueuedRRBE

1(1)

FPAR
IN legID LegType,
IN csID CallSegmentID;

DCL
rrbeArg RequestReportBCSMEventArg;





Process Type CallSegment

2(80)

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/**** DATA TYPE DEFINITIONS ****/

/* (in case of multi party configurations) the call segment
must broadcast:
- signalling control primitives that can occur in the
active state of the BCSMs, and
- PICs
to all BCSMs of the call segment. */

NEWTYPE BroadcastSignalNames
LITERALS
ServiceFeatureInd,
FailureInd,
ReleaseInd,
DataInd,
NetworkSuspendInd,
NetworkResumeInd,
PICSignal,
PICResume;
ENDNEWTYPE;

Process Type CallSegment

3(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/**** VARIABLE DECLARATIONS ****

```

DCL

```

/* The connection point of this
   call segment. */
CP ConnectionPoint,

/* Pointer to the SSF_FSM associated
   with this segment. */
SSF PId,

/* The association of local and remote
   leg ids. */
LegAssoc LegAssociationTable,

/* Pointer to the CSA in which this call
   segment belongs. */
CSA PId,

/* Dialogue status. */
DialogueActive Boolean := false,

/* Other variables */
csaID CSAID :=1,
invokeID InvokeID,
termination Boolean,
obscmPars OBCSMPars,
bcsStopped Boolean := false,
legID LegType,
newLegID LegType,
pic PICArg,
dp DPArg,
callFlag CallFlag,
eventTable EventType,
r Boolean;

```

DCL

```

/* Signalling control primitive parameters. */
AEArg AddressEndType,
CPArg CallProgressType,
FArg FailureType,
RArg ReleaseType,
SFtArg ServiceFeatureType,
SIRArg SetupIRType,
SCRArg SetupCRType,
SAArg SubsequentAddressType;

```

DCL

```

/* IN operation arguments. */
aiArg AnalyseInformationArg,
adiArg AnalysedInformationArg,
acArg ApplyChargingArg,
acrArg ApplyChargingReportArg,
ariArg AssistRequestInstructionsArg,
cirArg CallInformationReportArg,
cirqArg CallInformationRequestArg,
cArg CancelArg,
ciArg CollectInformationArg,
cdiArg CollectedInformationArg,
coArg ConnectArg,
ctrArg ConnectToResourceArg,
etcArg EstablishTemporaryConnectionArg,
encArg EventNotificationChargingArg,
erBCSMArg EventReportBCSMArg,
fciArg FurnishChargingInformationArg,
hcinArg HoldCallInNetworkArg,
idpArg InitialDPArg,
icaArg InitiateCallAttemptArg,
oabnArg OAbandonArg,
oansArg OAnswerArg,
ocpbArg OCalledPartyBusyArg,
odisArg ODisconnectArg,
midclArg MidCallArg,
onoansArg ONoAnswerArg,
oaaArg OriginationAttemptAuthorizedArg,
rcArg ReleaseCallArg,
rnceArg RequestNotificationChargingEventArg,
rrBCSMEArg RequestReportBCSMEEventArg,
rtArg ResetTimerArg,
rsfArg RouteSelectFailureArg,
sciArg SendChargingInformationArg,
sfArg SelectFacilityArg,
srArg SelectRouteArg,
tansArg TAnswerArg,
tbusArg TBusyArg,
tdisArg TDisconnectArg,
taaArg TermAttemptAuthorizedArg,
tnoansArg TNoAnswerArg,

ieArg ErrorArg;

```

Process Type CallSegment

4(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/**** VARIABLE DECLARATIONS ****/

```

DCL

```

/* IN operation arguments. */
autArg AuthorizeTerminationArg,
ctwaArg ContinueWithArgumentArg,
dfcwaArg DisconnectForwardConnectionWithArgumentArg,
dlArg DisconnectLegArg,
erArg EntityReleasedArg,
erfArg EventReportFacilityArg,
fsaArg FacilitySelectedAndAvailableArg,
oaArg OriginationAttemptArg,
osusArg OSuspendedArg,
reconArg ReconnectArg,
rutsiArg ReportUTSIArg,
rrutsiArg RequestReportUTSIArg,
rrfeArg RequestReportFacilityEventArg,
sendfArg SendFacilityInformationArg,
sstuiArg SendSTUIArg,
sspArg SetServiceProfileArg,
taArg TerminationAttemptArg,
tsusArg TSuspendedArg
;

```

DCL

```

/* Parameters at the signalling control interface. */
udArg UserDataType,
nsrArg NetworkSRType;

```

DCL

```

/* Other variables. */
currentLegID LegType,
expLeg ExportLegType,
dpUTSIArg DPUTSIArg,
noOfMidCallDPs Natural := 0,
noOfDisconnectDPs Natural := 0,
noOfAbandonDPs Natural := 0,
noOfUTSIDPs Natural := 0,
party PartyType,
cwaFlag Boolean := false,
/* Used to indicate the status of the call processing. */
setupSent boolean := false; /* Indicates if SetupReq is sent to the called party */

```

Process Type

CallSegment

5(80)

;

F

P

A

R

/*

P

a

r

a

m

e

t

e

r

s

a

s

s

i

g

n

e

d

b

y

t

h

e

C

S

A

a

t

c

r

e

a

t

i

o

n

o

f

t

h

e

c

a

l

l

s

e

g

m

e

n

t

.

*/

c

a

l

l

S

e

g

m

e

n

t

I

D

C

a

l

l

S

e

g

m

e

n

t

I

D

,

/*

I

d

o

f

t

h

i

s

c

a

l

l

s

e

g

m

e

n

t

a

s

a

s

s

i

g

n

e

d

b

y

t

h

e

C

S

A

.

*/

c

o

n

t

r

o

l

l

i

n

g

i

n

g

L

e

g

I

D

L

e

g

T

y

p

e

,

/*

I

d

o

f

t

h

e

c

o

n

t

r

o

l

l

i

n

g

i

n

g

l

e

g

.

*/

s

s

f

S

t

a

t

e

S

S

F

S

t

a

t

e

T

y

p

e

,

/*

S

t

a

r

t

s

t

a

t

e

o

f

t

h

e

S

S

F

-

F

S

M

.

*/

s

c

f

I

n

i

t

i

a

t

e

d

B

o

o

l

e

a

n

;

/*

I

s

t

h

e

c

a

l

l

i

n

i

t

i

a

t

e

d

b

y

t

h

e

S

C

F

o

r

n

o

t

?

*/

/*

D

E

C

L

A

R

O

F

O

P

E

R

A

T

I

O

N

S

*/

/*

O

p

e

r

a

t

i

o

n

s

o

n

l

e

g

s

*/

AddLeg

Adds a leg to a
call segment (connection
point).

SetLeg
Status

Sets the status of
a leg.

GetLeg
Status

Returns the status of a leg.

SetLeg
Ptr

Connects the leg
to a BCSM instance.

GetLeg
Ptr

Returns the pointer
to a BCSM instance.

IsBCSM

Predicate determining whether
a sender of a signal is a BCSM
or not.

Remove
Leg

Removes a leg.

Release
AllLegs

Removes all legs associated
with this segment.

SetLeg_
Assoc

Associates a remote leg id with
a local leg id.

GetLegId
from_
Remote_
LegId

Retrieves the local leg id
from a given remote leg id.

Get_
Passive
LegId

Returns the id of the passive
leg in the connection point.

/*

P

r

o

c

e

d

u

r

e

s

f

o

r

b

u

i

l

d

i

n

g

t

h

e

i

n

f

o

r

m

a

t

i

o

n

t

o

b

e

p

a

s

s

e

d

t

o

a

n

O

-

B

C

S

M

a

t

c

r

e

a

t

i

o

n

t

i

m

e

.

*/

MapConnect
ToBCSM

MapSI_
ToBCSM

Process Type CallSegment

6(80)

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/**** DECLARATION OF OPERATIONS ****/

/* Operations on Legs. */

Import_
Leg

Imports a leg into the connection point.

Broadcast
ToLegs

Procedure to broadcast a signal from SigCon or a PIC to all the passive legs in the connection point.

Export_
Leg

Exports a leg from the connection point.

NoOf_
Legs

Returns the number of passive legs.

/* Procedures for filtering detection points. */

DP_
Filtering

MapSF_
ToBCSM

MapCI_
ToBCSM

DPFiltering_
UTSI

MapAI_
ToBCSM

MapSR_
ToBCSM

Process Type CallSegment

7(80)

```

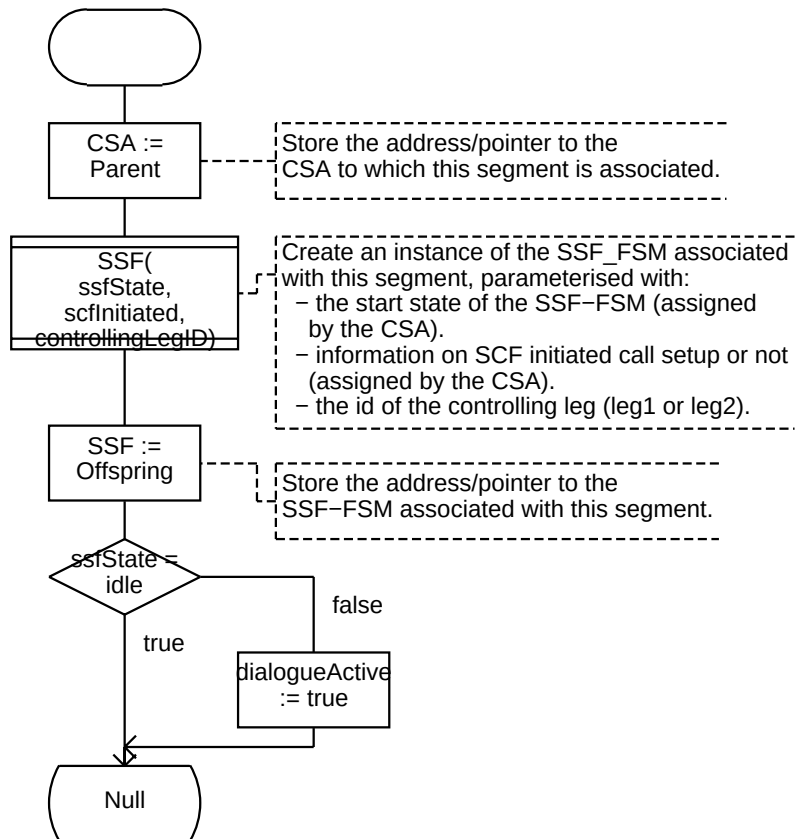
;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/**** INITIALISATION OF THE CALL SEGMENT. ****/

```



Process Type CallSegment

8(80)

```

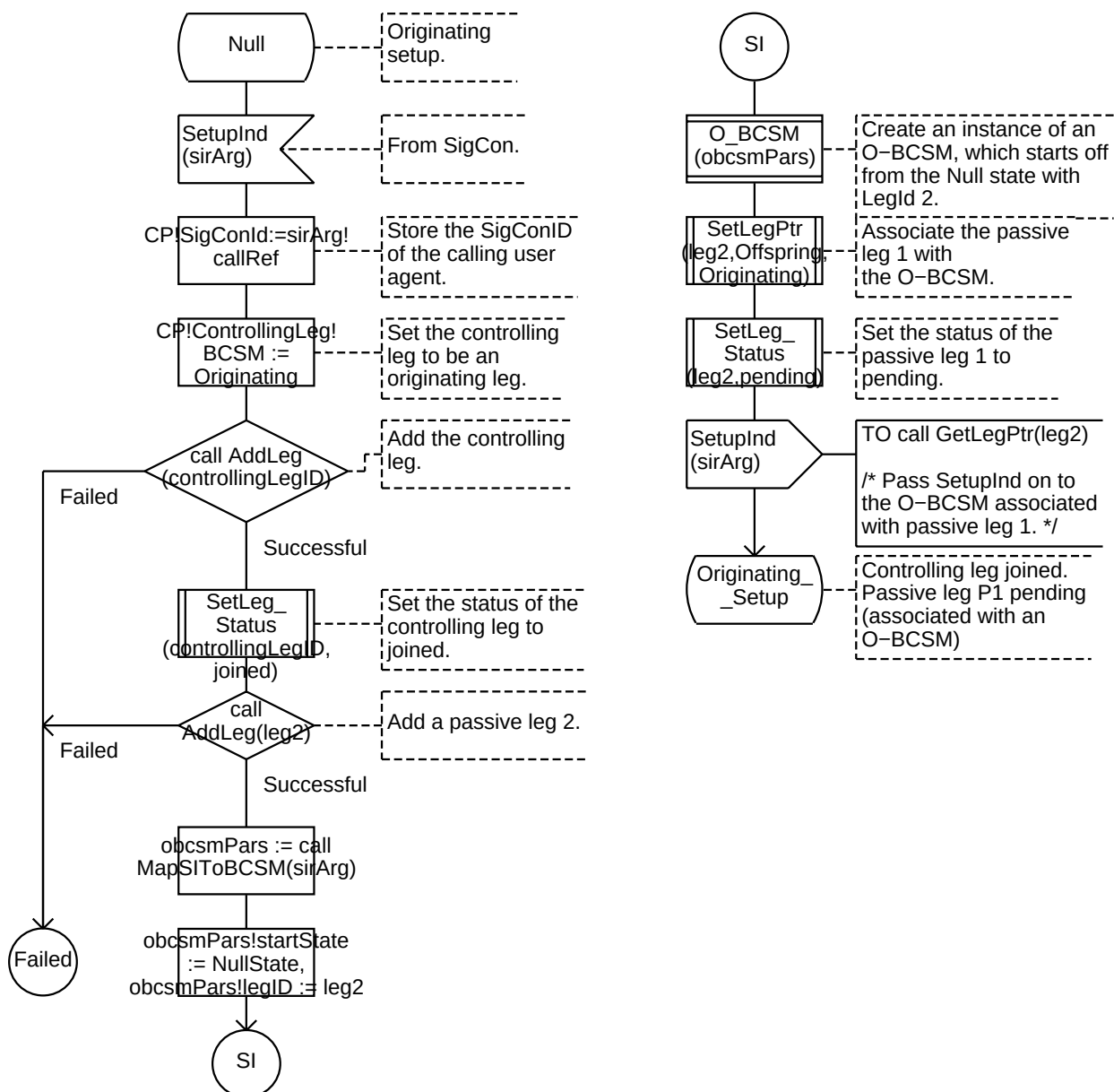
;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/**** HANDLING OF CALL SETUP. ****/

```



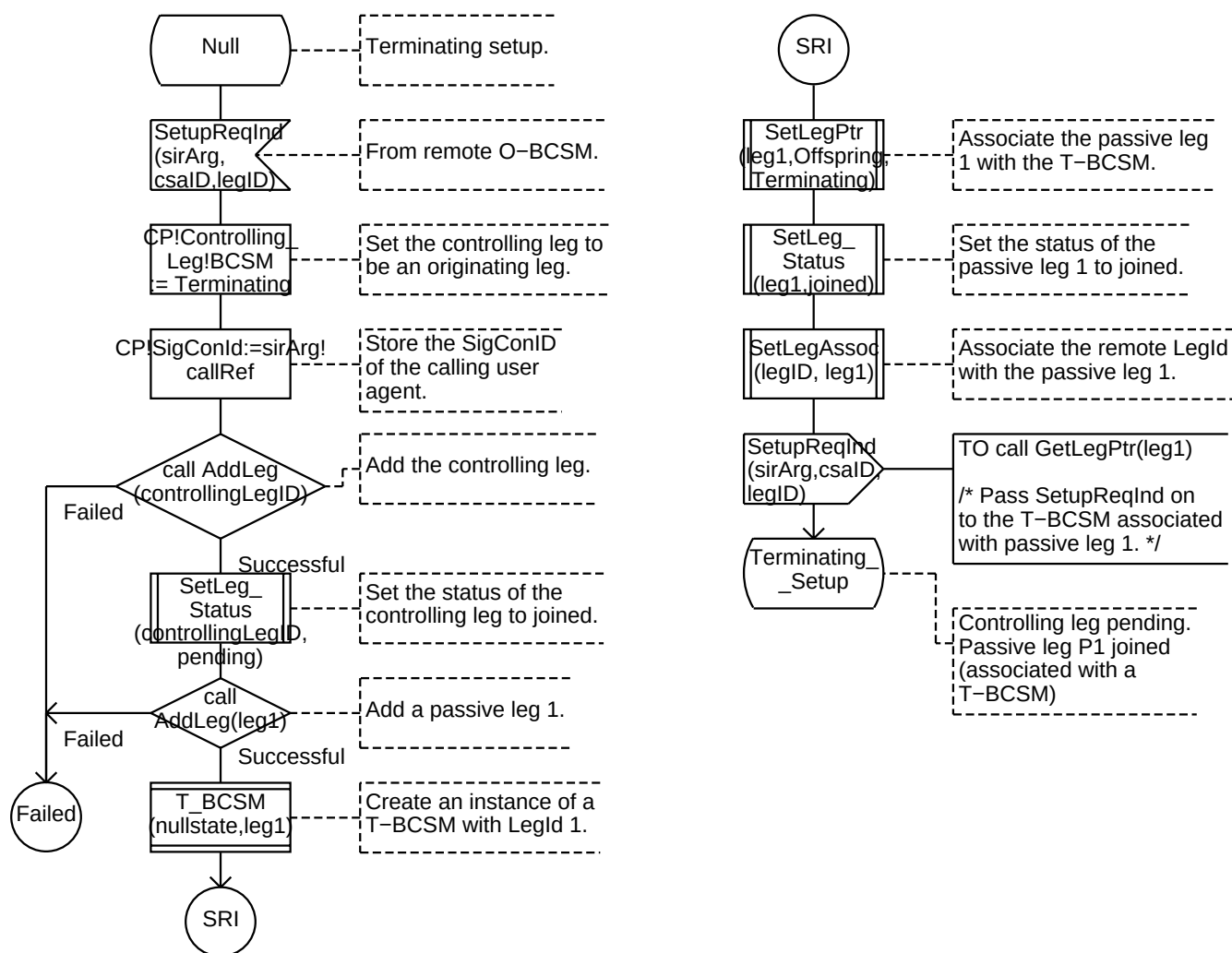
Process Type CallSegment

9(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

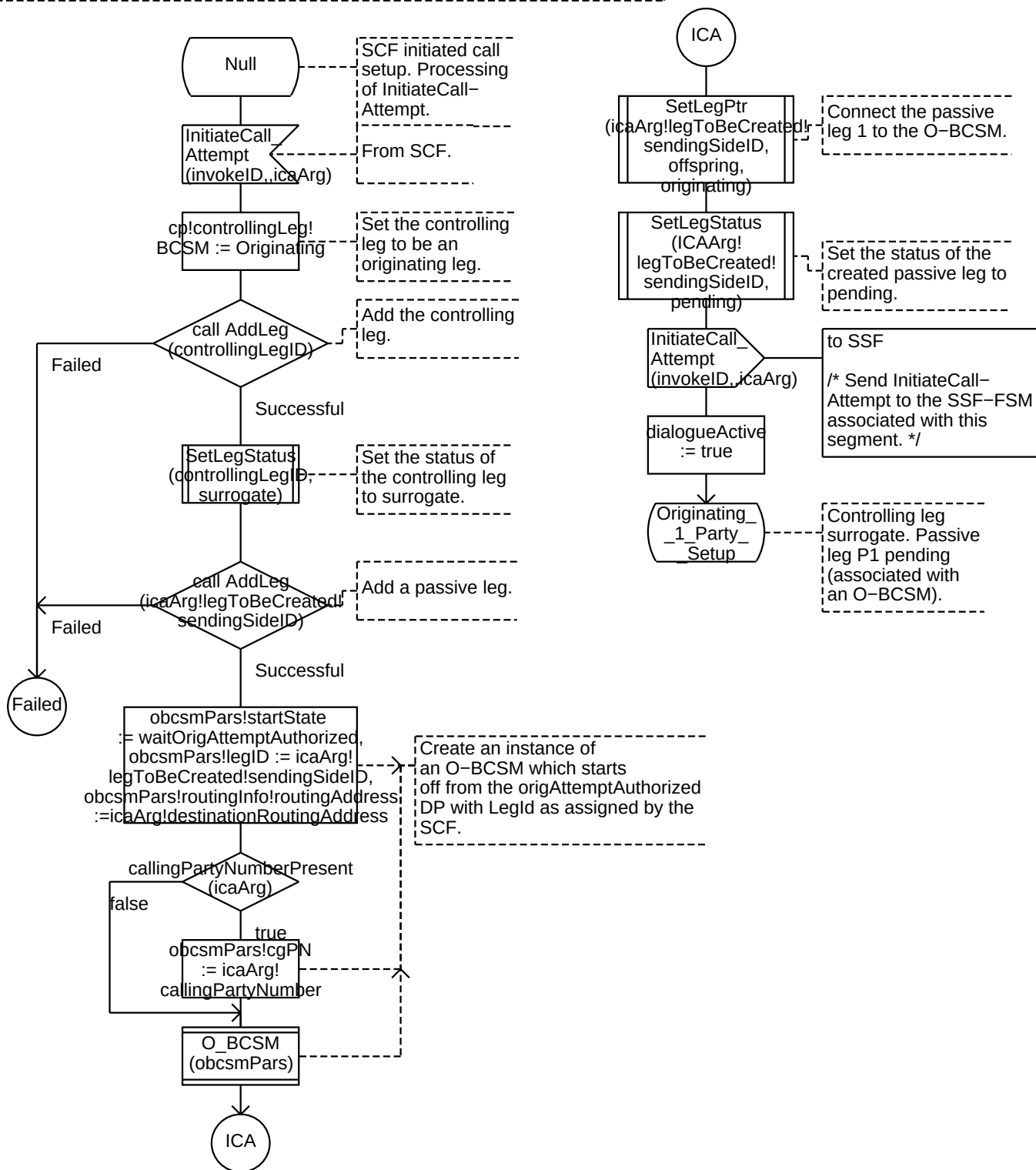
```



Process Type CallSegment

10(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */



Process Type CallSegment

11(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

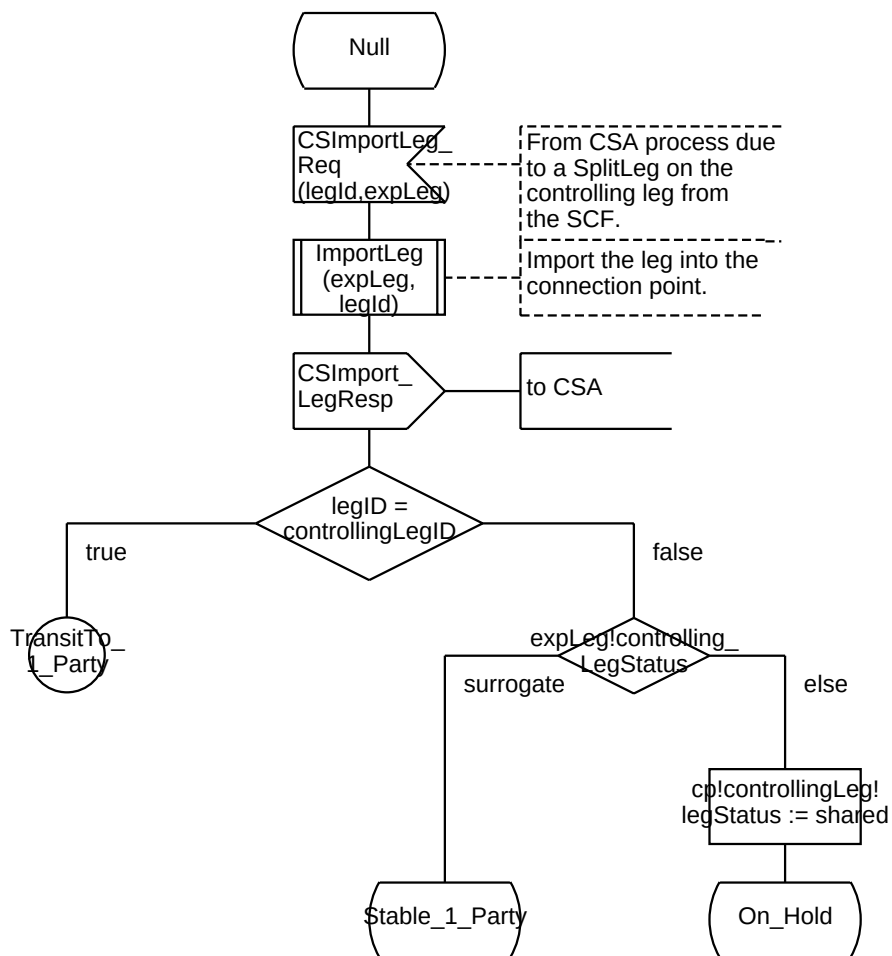
/* Null -> 1-Party
Null -> On-Hold

```

```

Note: Processing of atomic operations used by the CSA. */

```



Process Type CallSegment

12(80)

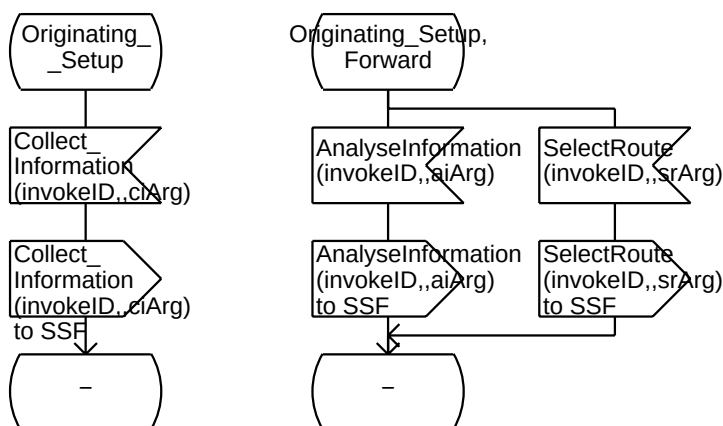
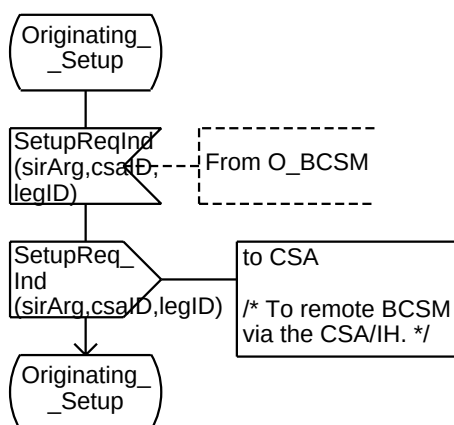
```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

/* TRANSITIONS BETWEEN THE CONNECTION VIEW STATES OF THE CALL SEGMENT. */

/* Note: The states of the call segment are defined in Q.1224. */



Process Type CallSegment

13(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

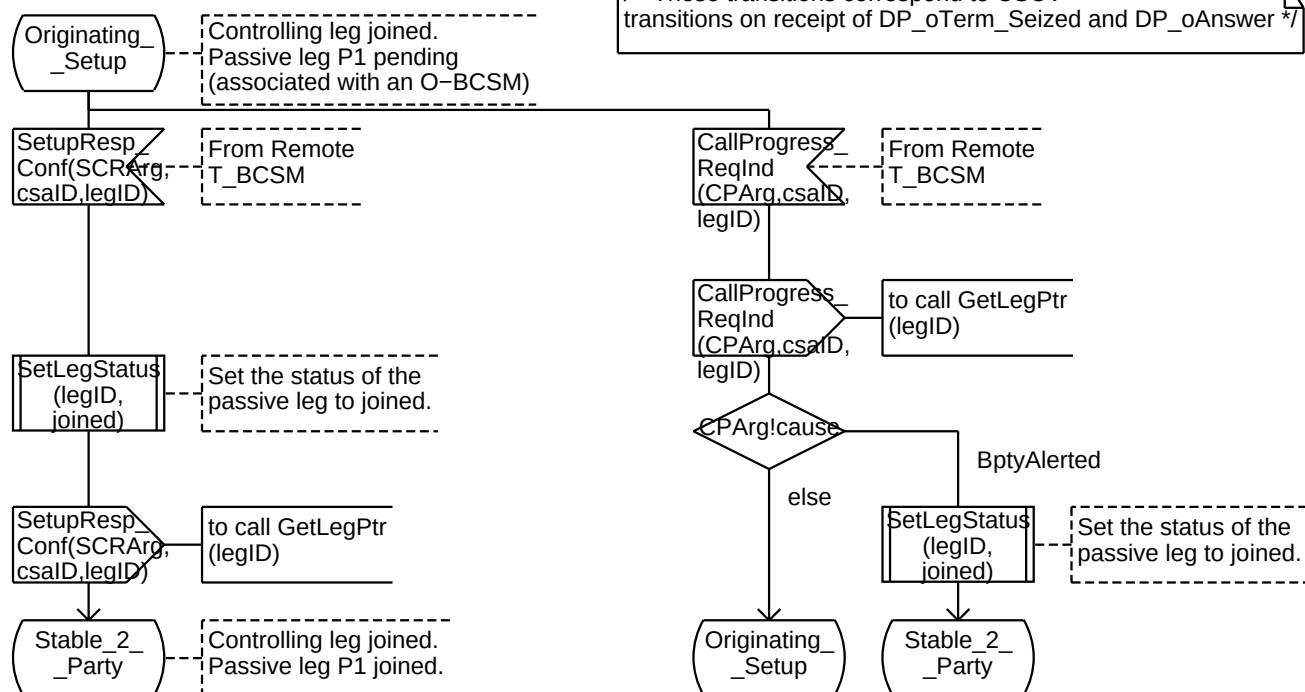
/* Originating Setup -> Stable 2 Party
Terminating Setup -> Stable 2 Party */

```

```

/* These transitions correspond to CSCV
transitions on receipt of DP_oTerm_Seized and DP_oAnswer */

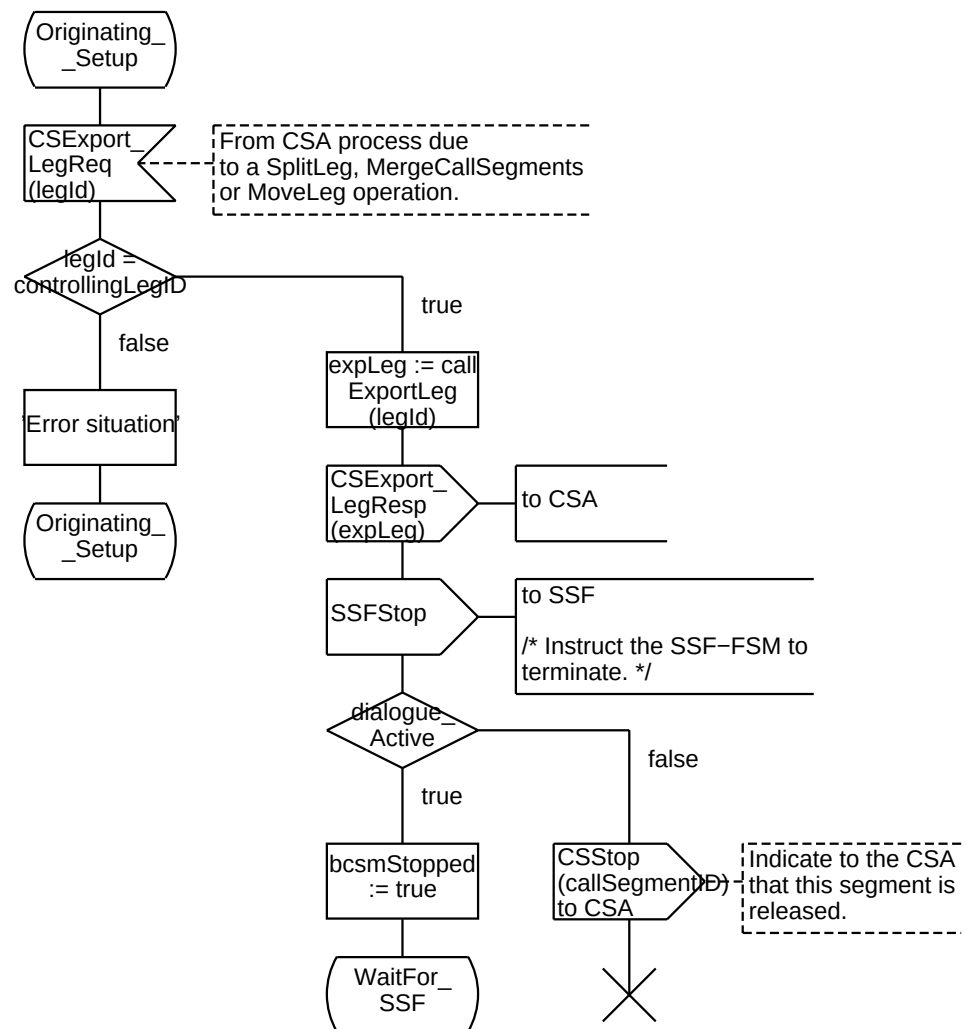
```



Process Type CallSegment

14(80)

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */



Process Type CallSegment

15(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

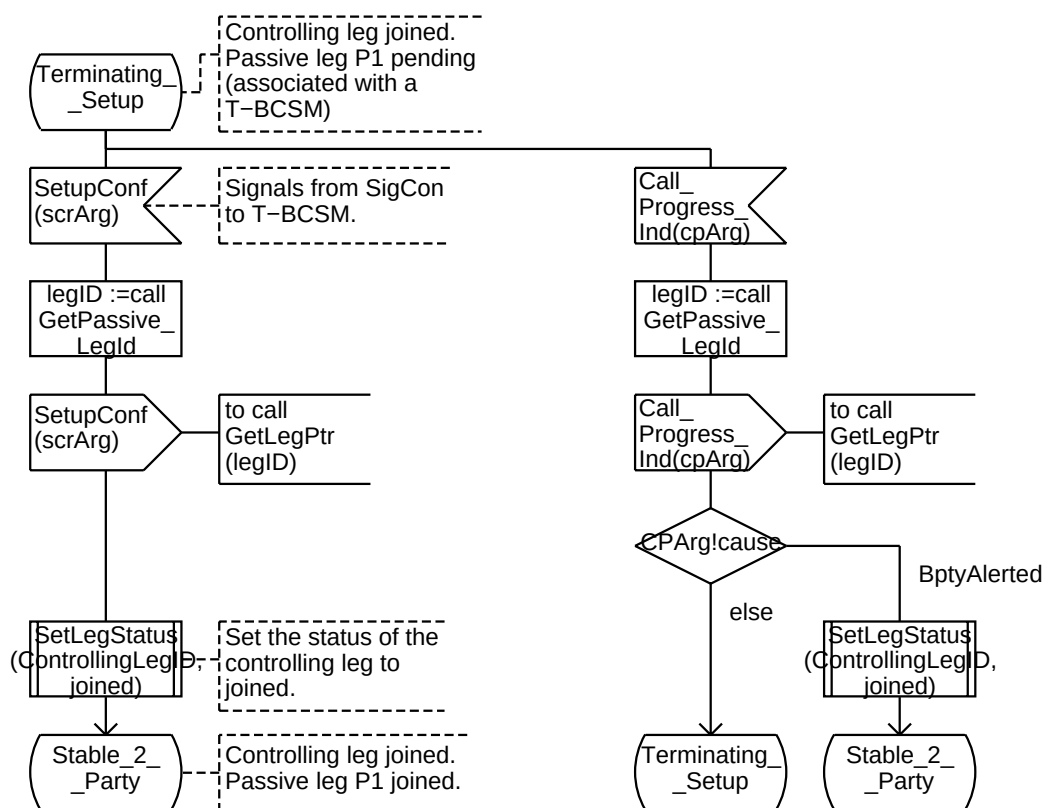
/* Originating Setup -> Stable 2 Party
Terminating Setup -> Stable 2 Party */

```

```

/* These transitions correspond to CSCV
transitions on receipt of DP_oTerm_Seized and DP_oAnswer */

```



Process Type CallSegment

16(80)

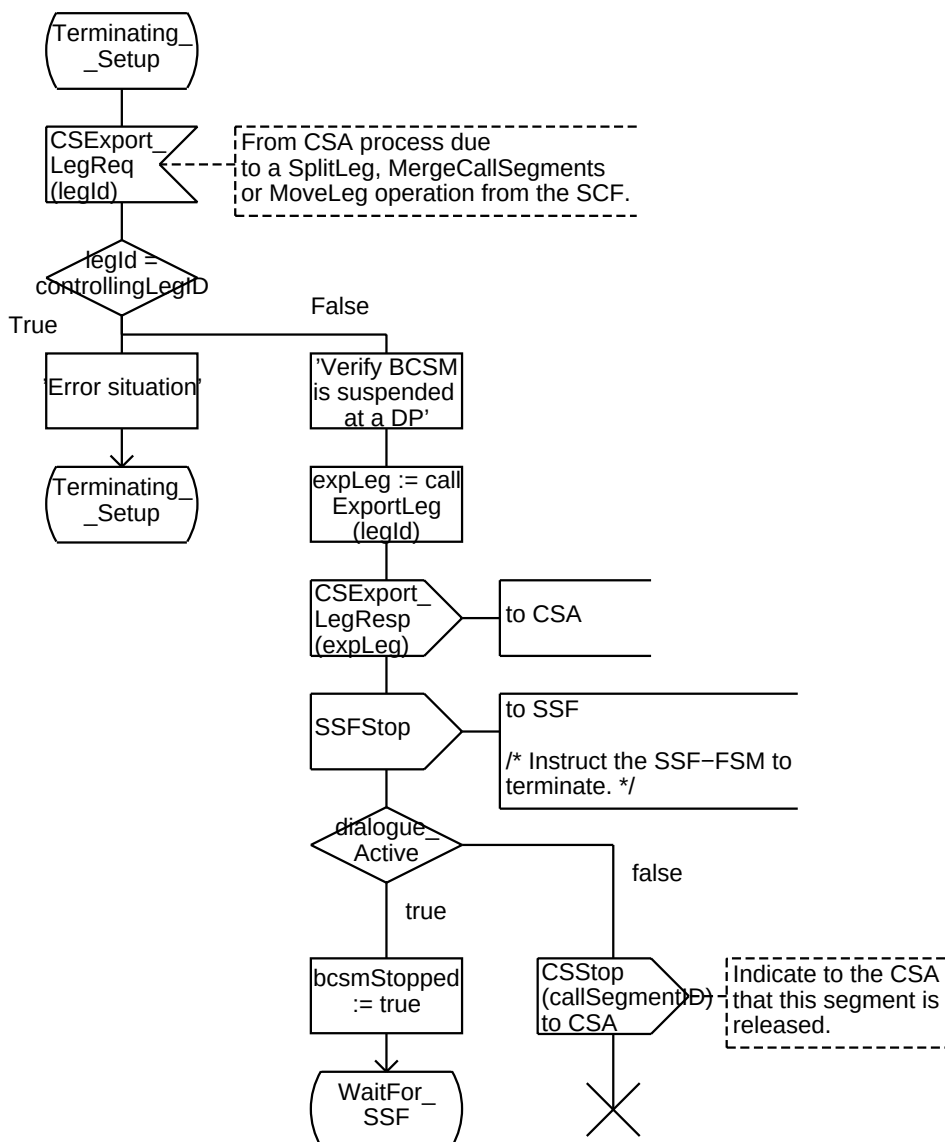
```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

/* Terminating Setup -> Null */

/* This transition was added in the last week of work on Q.1238-2, in order to preserve CS2 call waiting/call hold services - MergeCallSegments MoveLeg and MoveCallSegments now permitted in Terminating_Setup CSCV state */



Process Type CallSegment

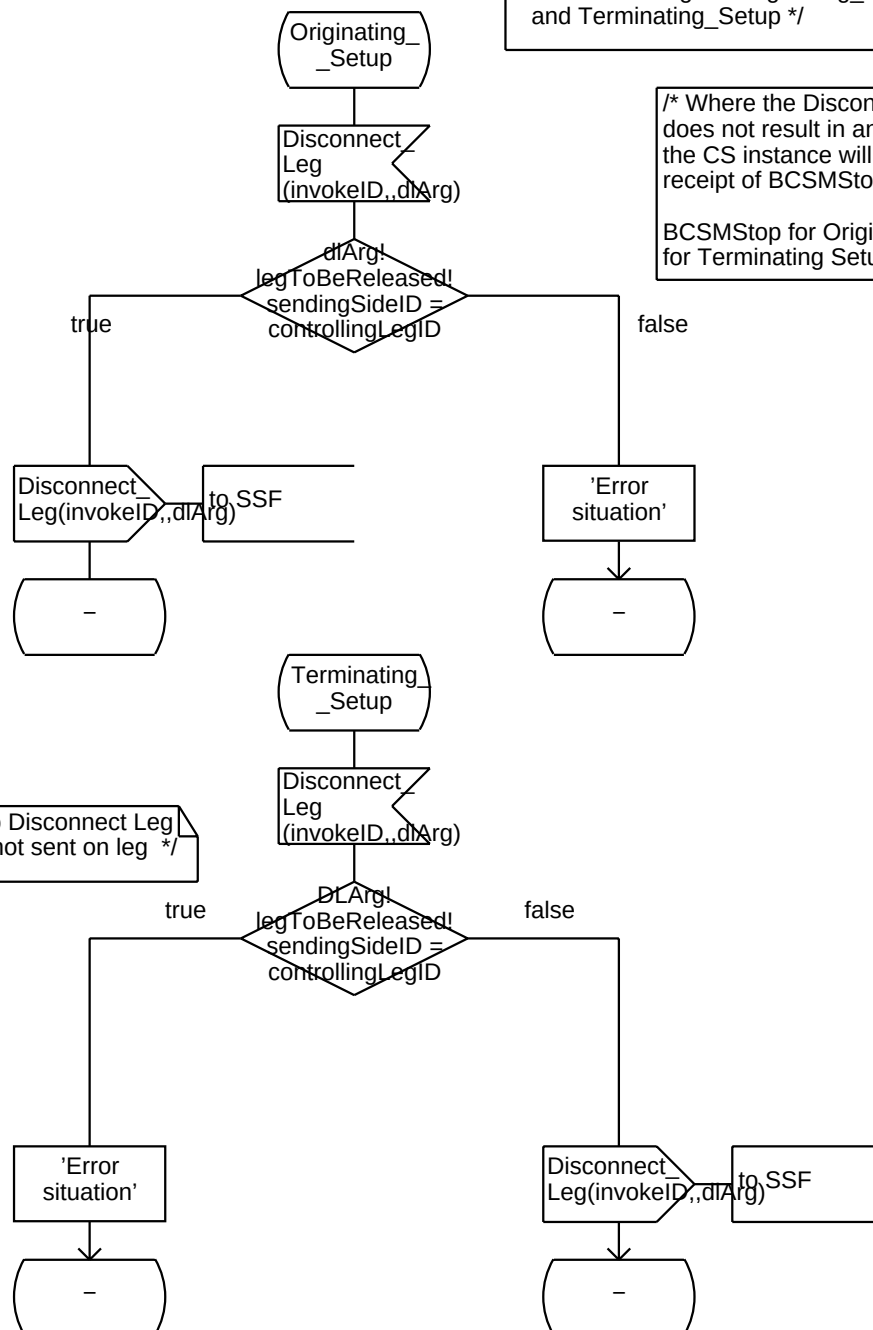
17(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* DisconnectLeg for Originating_Setup
 and Terminating_Setup */

/* Where the DisconnectLeg operation
 does not result in an error situation,
 the CS instance will be deleted on
 receipt of BCSMStop.

BCSMStop for Originating Setup and
 for Terminating Setup is on page 17 */



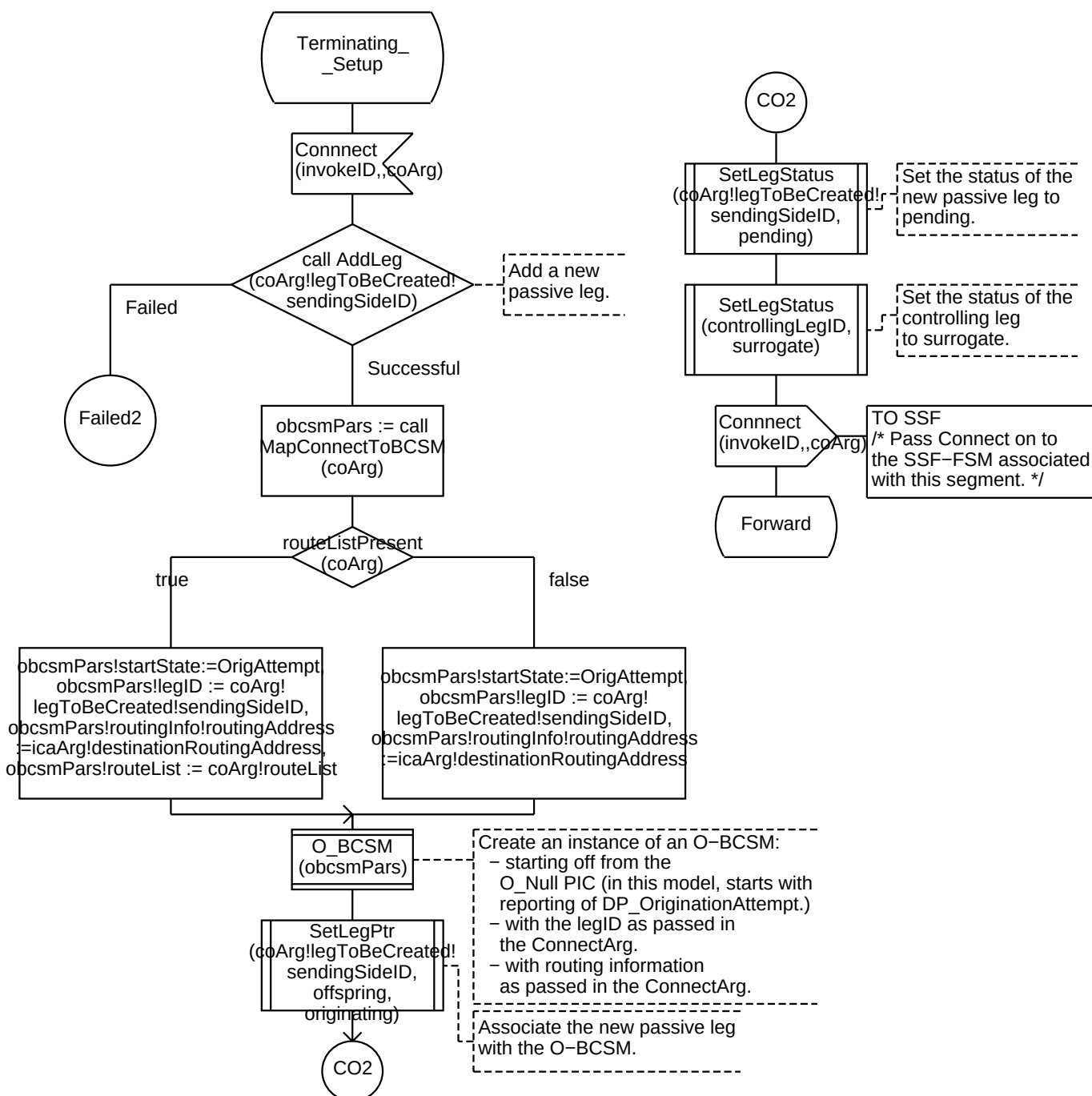
/* Error if attempt to Disconnect Leg
 when SetupReq not sent on leg */

Process Type CallSegment

18(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* Terminating-Setup -> Forward */



Process Type CallSegment

19(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/**** TRANSITIONS BETWEEN THE CONNECTION STATES OF THE CALL SEGMENT. ****/

```

```

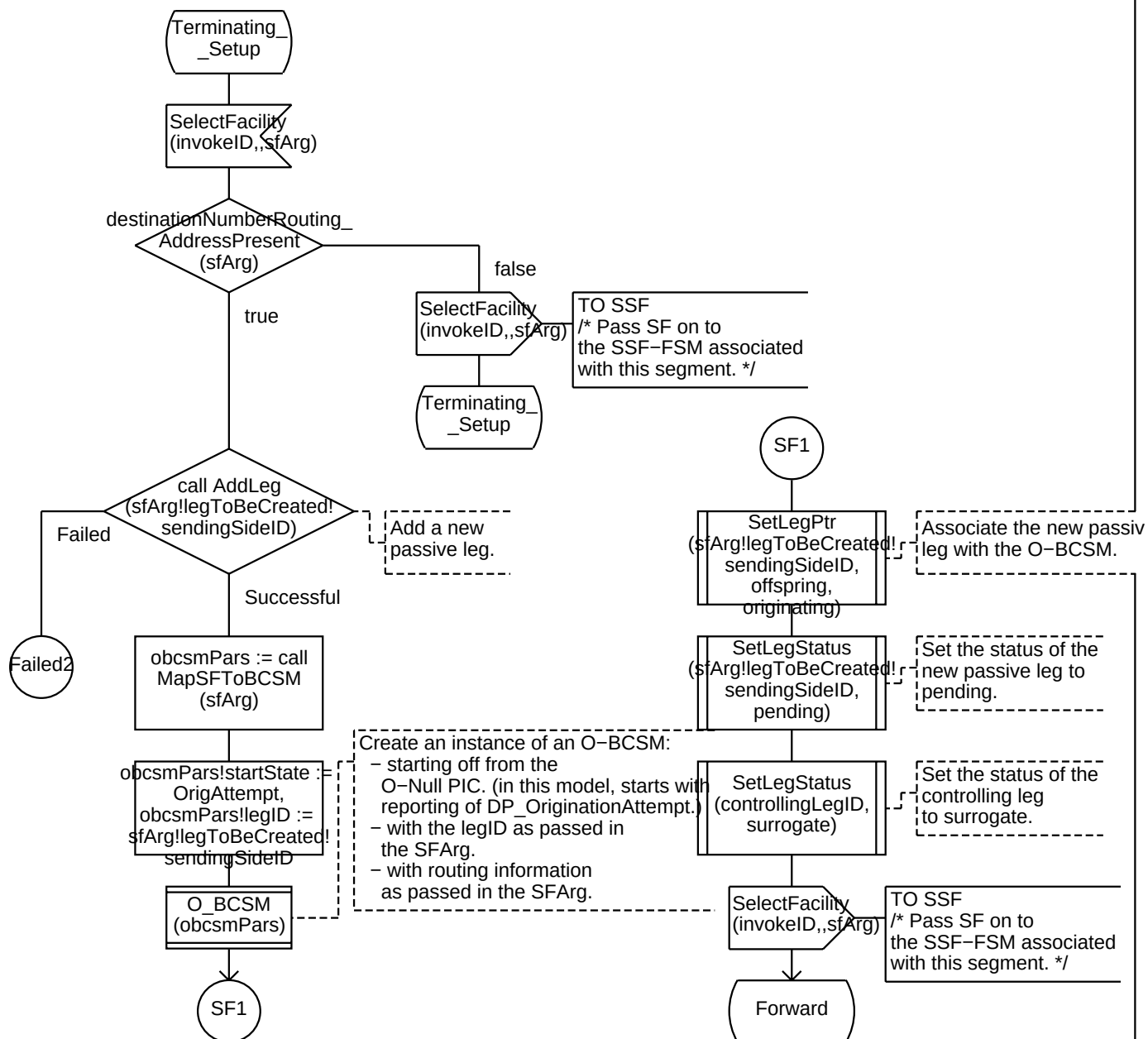
/* Note: The states of the call segment are defined in Q.1238. */

```

```

/* Terminating-Setup -> Forward */

```



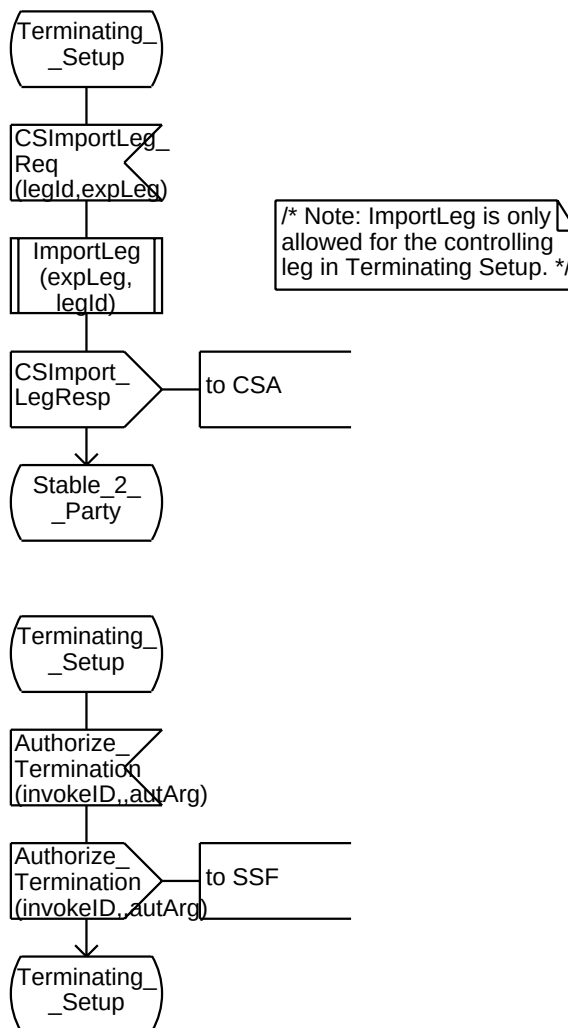
Process Type CallSegment

20(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */
    
```

/* Terminating Setup -> Stable-2-Party
Note: Processing of atomic operations used by the CSA. */



Process Type CallSegment

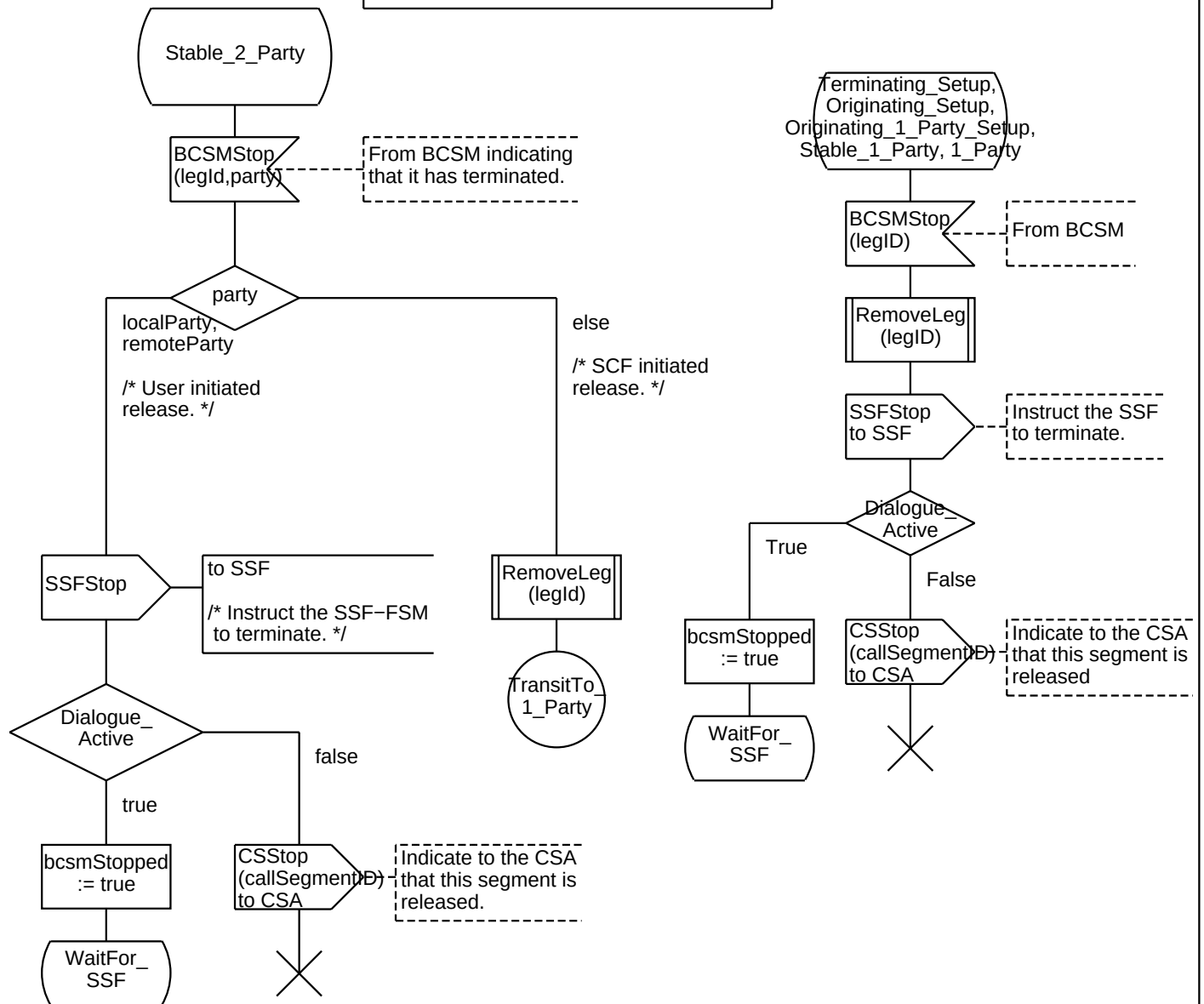
21(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

**** HANDLING OF CALL RELEASE ****



Process Type CallSegment

22(80)

```

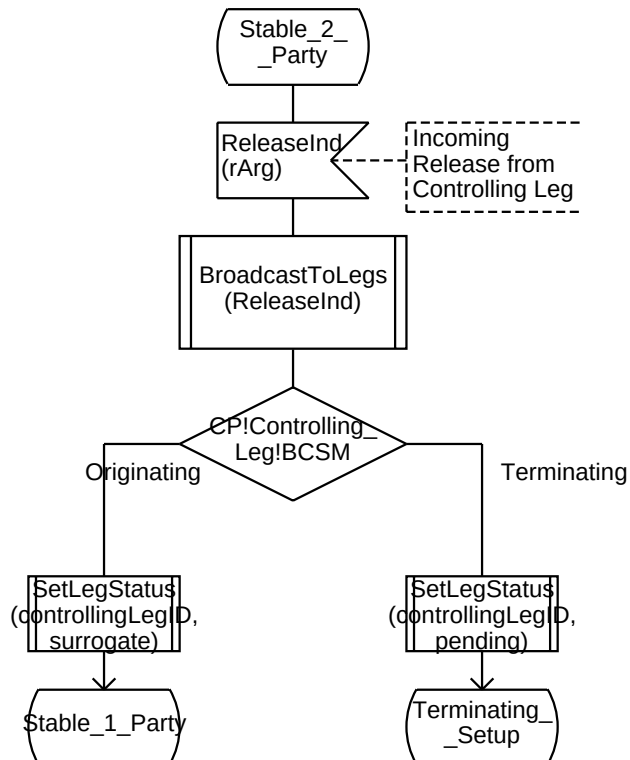
;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/* RELEASE FROM CONTROLLING LEG
   Stable-2-Party -> Stable-1-Party
*/

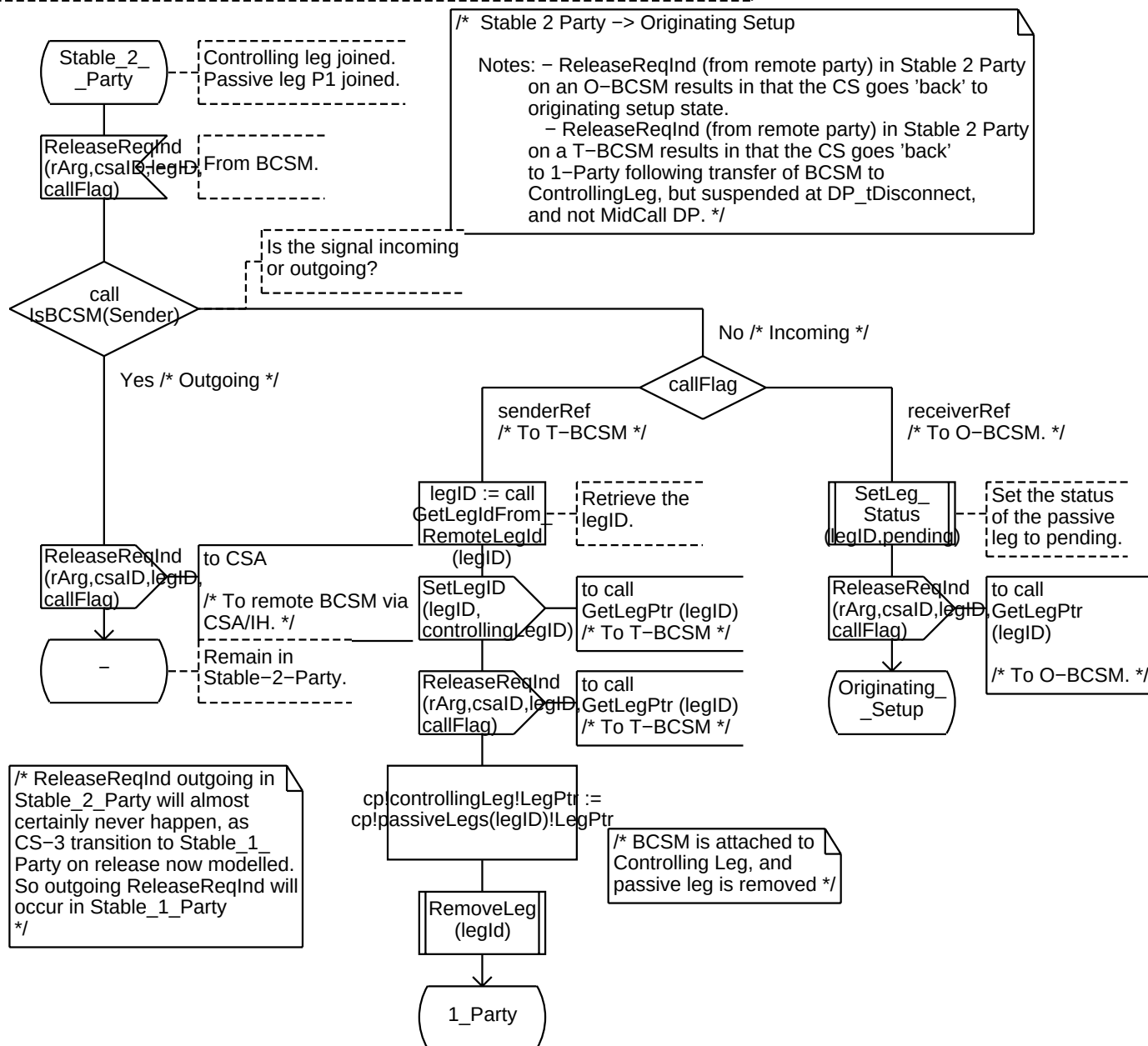
```



Process Type CallSegment

23(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */



Process Type CallSegment

24(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

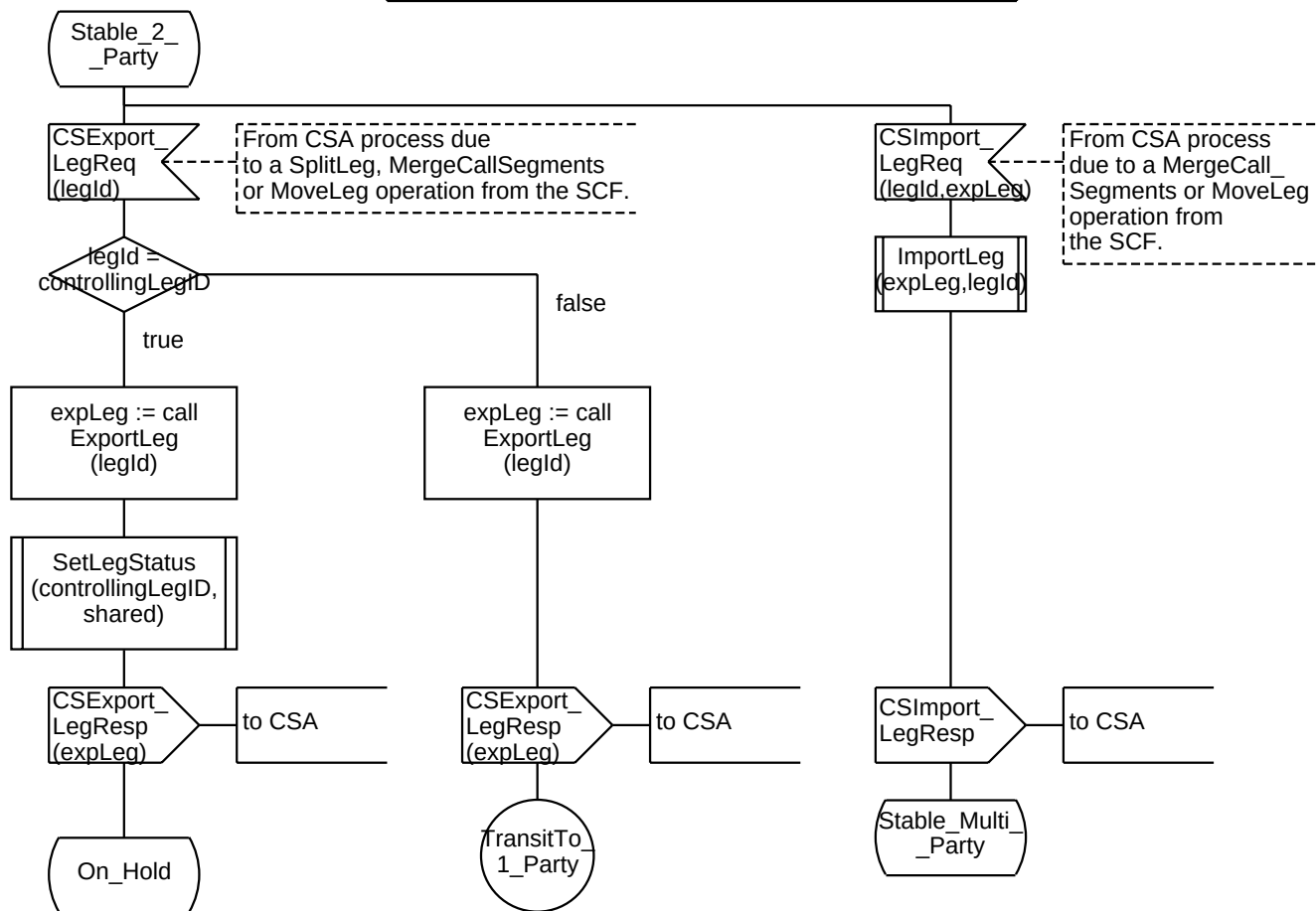
/* Stable-2-Party -> On-Hold
   Stable-2-Party -> 1-Party
   Stable-2-Party -> Stable-Multi-Party

```

```

Note: Processing of atomic operations used by the CSA. */

```



Process Type CallSegment

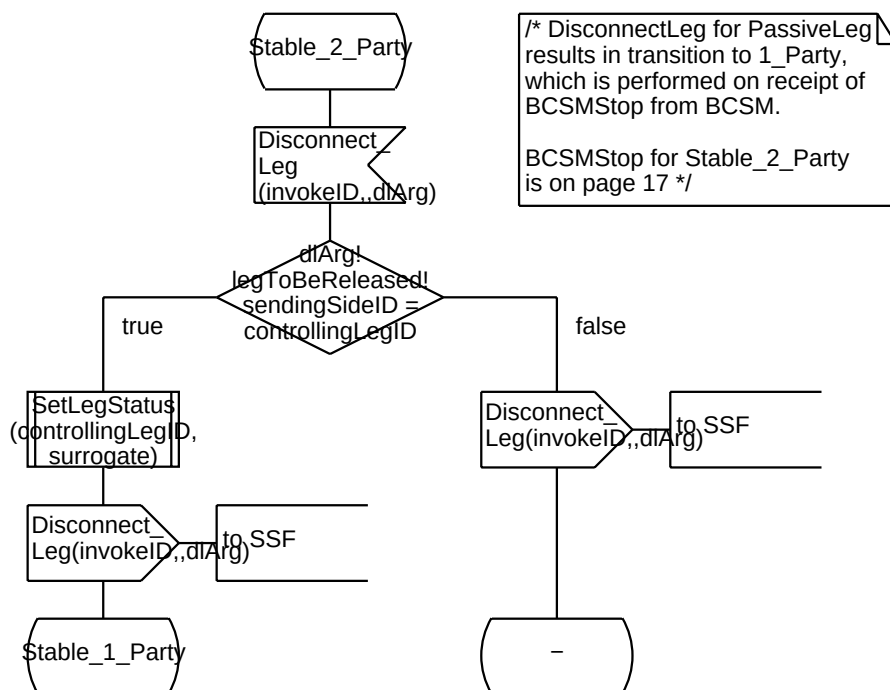
25(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```
/* Stable-2-Party -> Stable-1-Party */
```



/* Notes:

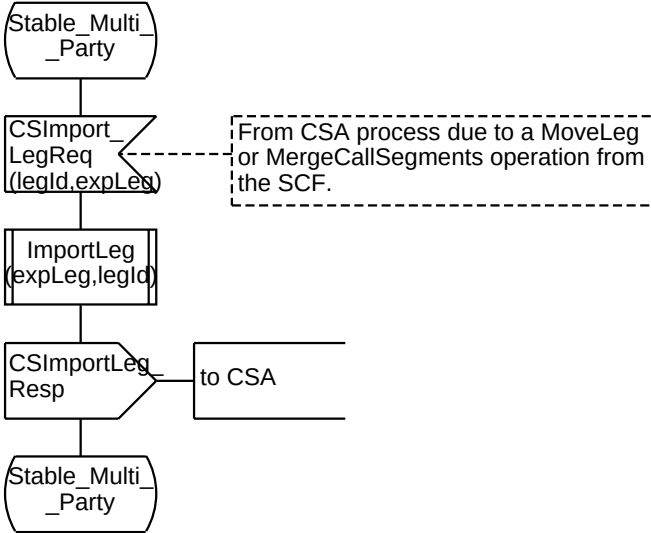
- DisconnectLeg is not defined for the following states:
Originating-1-Party-Setup.
- DisconnectLeg of the controlling leg in any multi party state will result in that all BCSMs will generate a ReleaseReq towards the signalling control. The signalling control should filter these ReleaseReqs and pass on only one Release message to the signalling. */

Process Type CallSegment

26(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* Stable-Multi-Party -> Stable-Multi-Party
Note: Processing of atomic operations used by the CSA. */



Process Type CallSegment

27(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

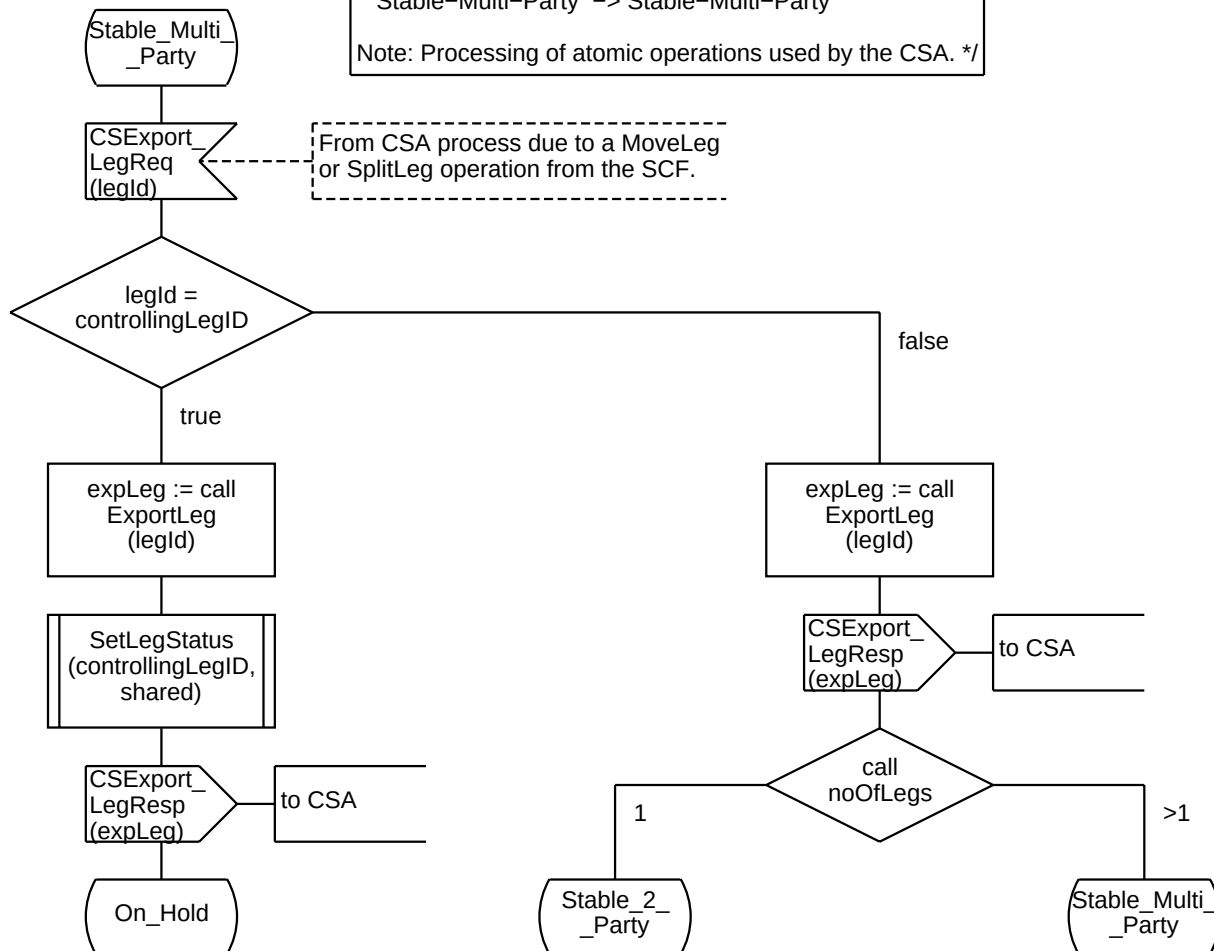
```

```

/* Stable-Multi-Party -> On-Hold
   Stable-Multi-Party -> Stable-2-Party
   Stable-Multi-Party -> Stable-Multi-Party

```

Note: Processing of atomic operations used by the CSA. */



Process Type CallSegment

28(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/* Stable-Multi-Party -> Transfer
On-Hold -> Transfer */

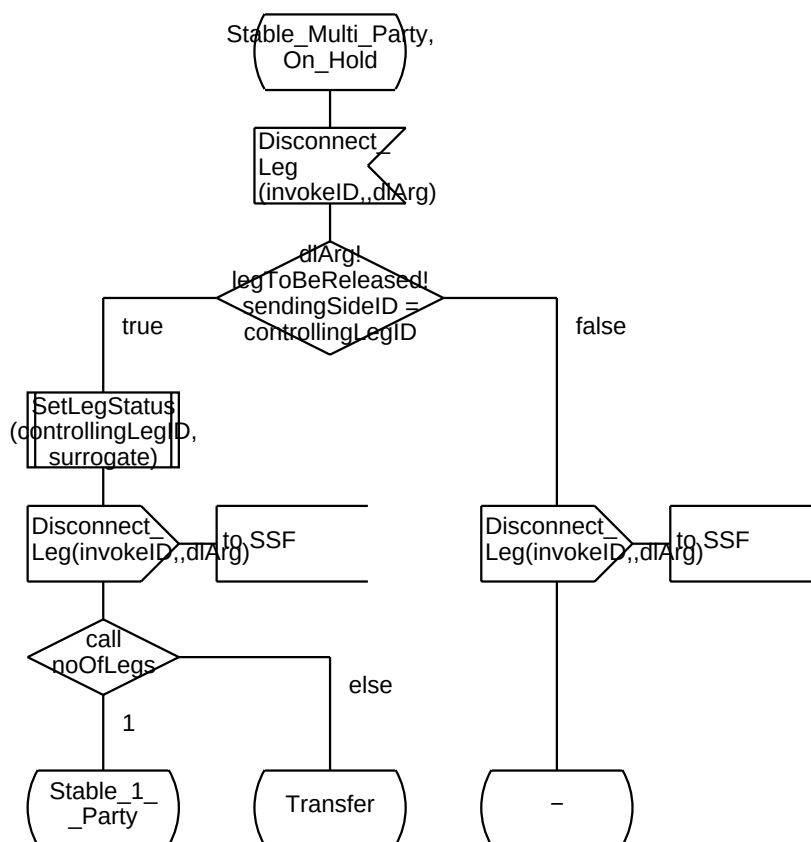
```

/* For the DisconnectLeg(p) operation, it is the BCSMStop event that removes the passive leg from the call segment. The Disconnect Leg operation is passed on to the SSF to allow the SSF to terminate the leg (BCSM) in a controlled way.

The transitions occurring on the BCSM stop event (and specified at other pages) are:

Stable-2-Party -> 1-Party

Stable-Multi-Party -> Stable-2-Party */



Process Type CallSegment

29(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

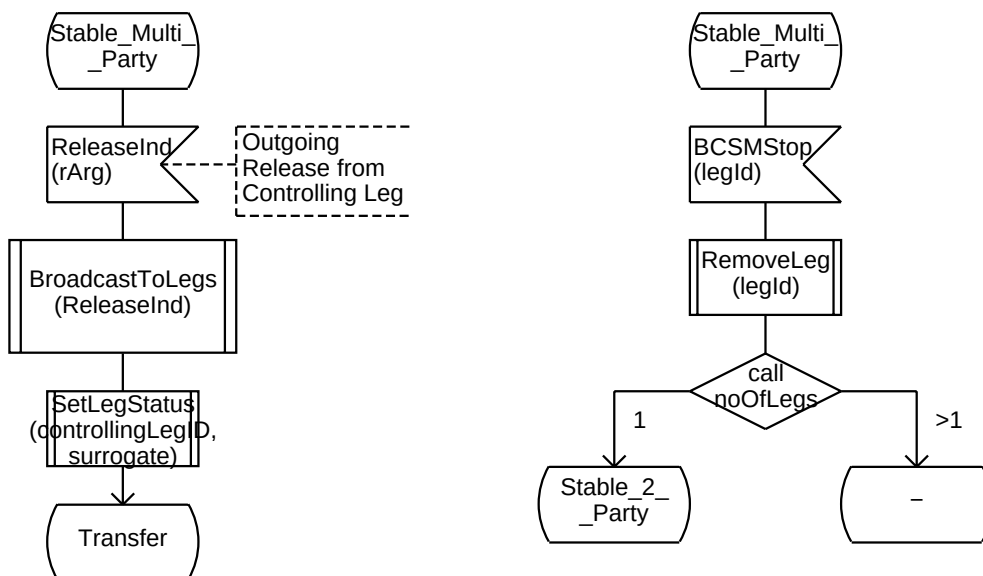
```

```

/* Stable-Multi-Party -> Stable-2-Party
Stable-Multi-Party -> Stable-Multi-Party
Stable-Multi-Party -> Transfer

Note: Processing of atomic operations used by the CSA. */

```

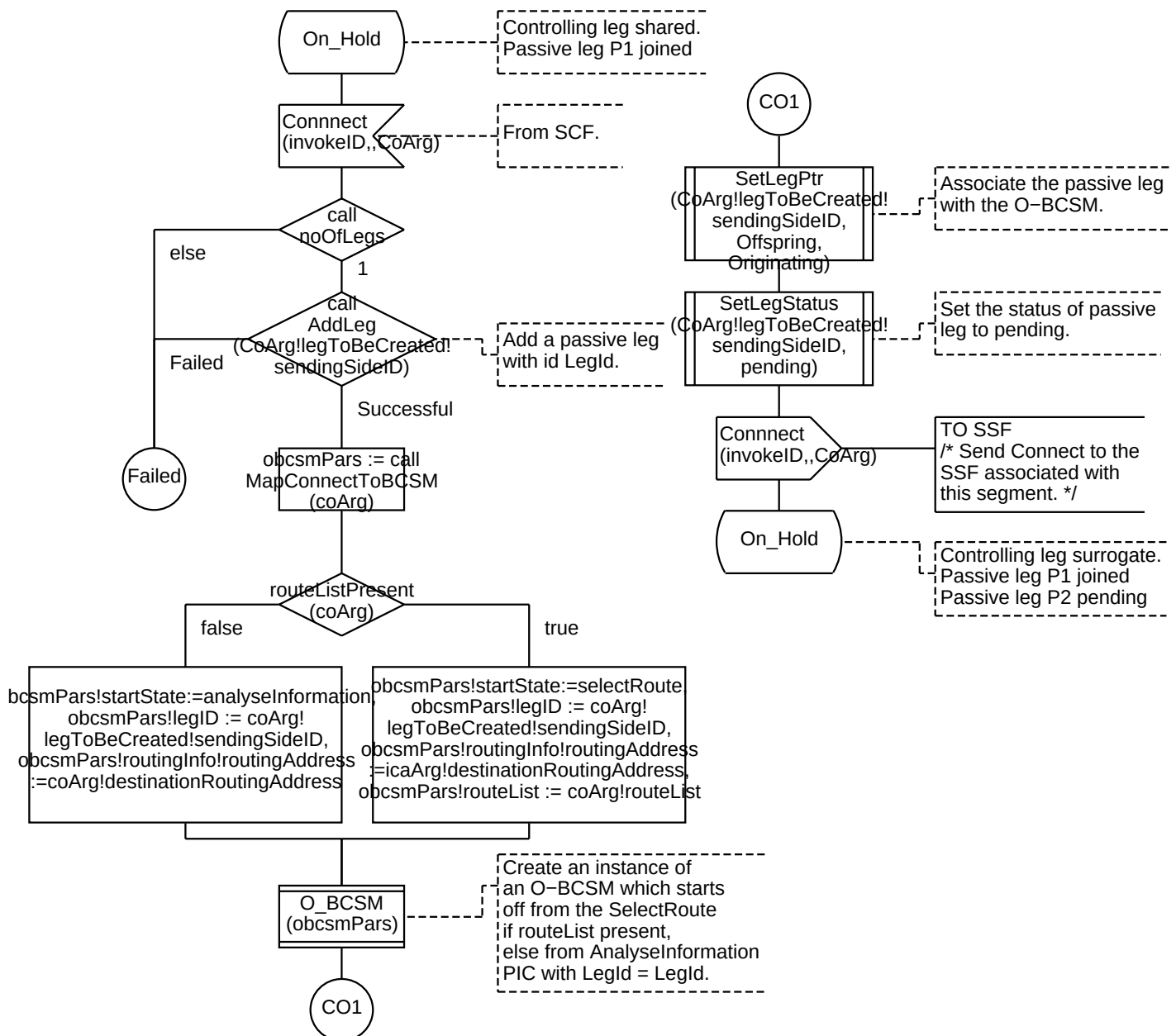


Process Type CallSegment

30(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* On_Hold -> On_Hold */



Process Type CallSegment

31(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

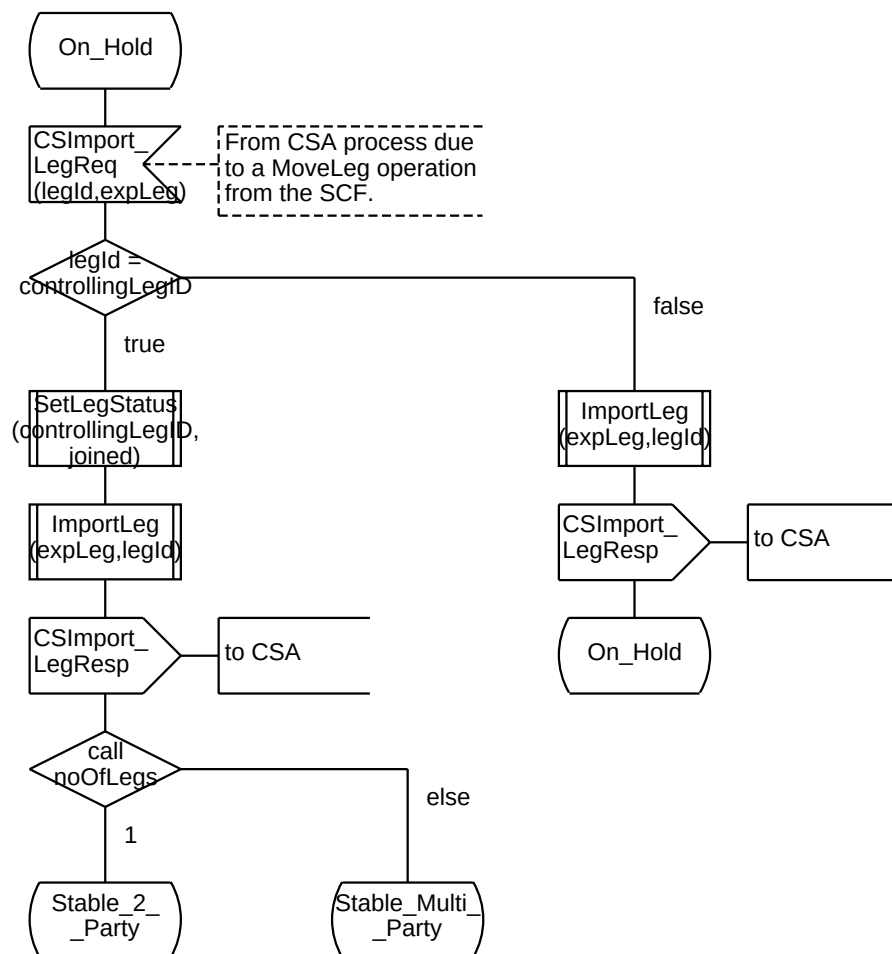
```

```

/* On-Hold -> Stable-2-Party
On-Hold -> Stable-Multi-Party
On-Hold -> On-Hold

Note: Processing of atomic operations used by the CSA. */

```



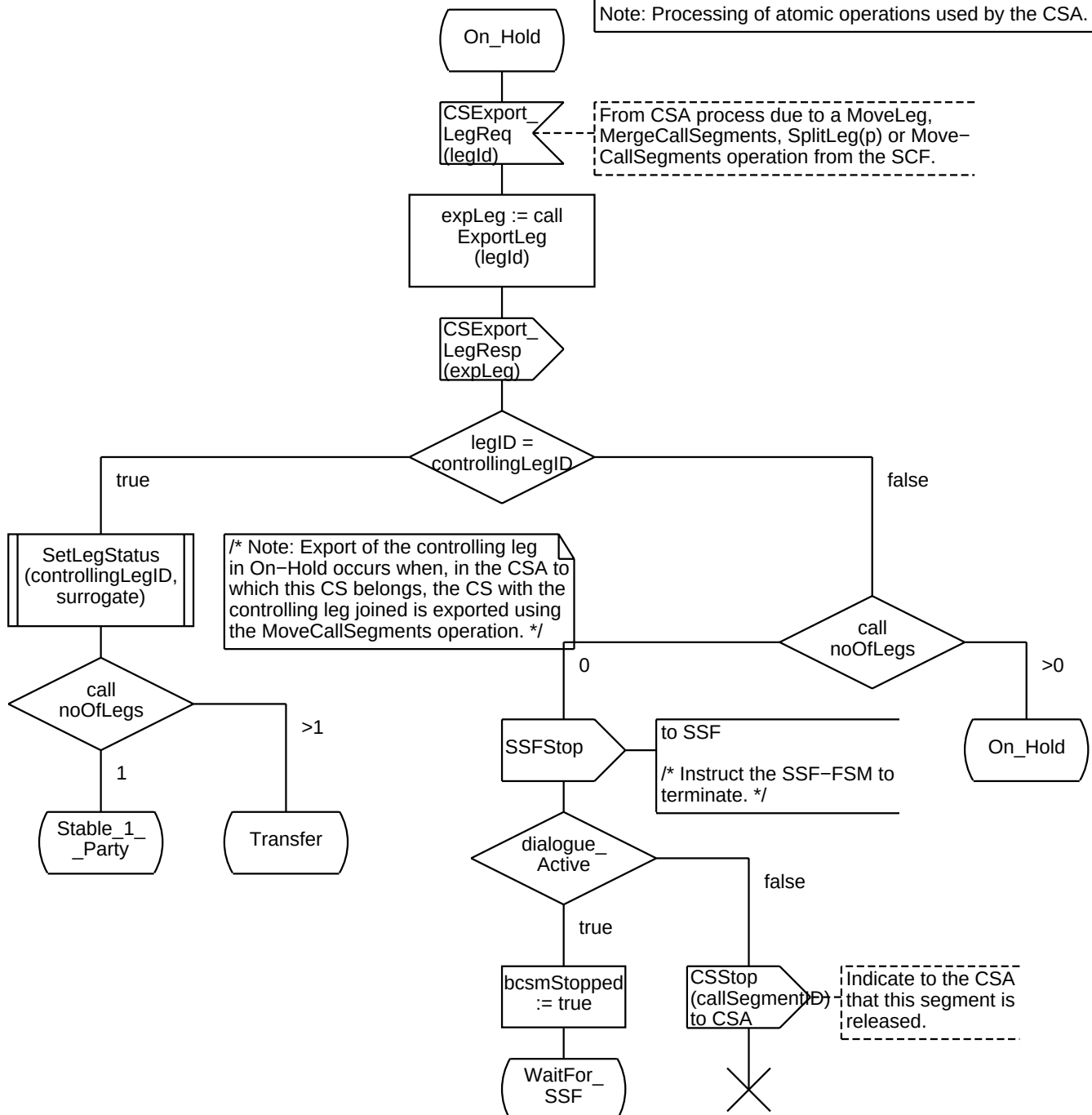
Process Type CallSegment

32(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* On-Hold -> Terminate call segment
 On-Hold -> On-Hold
 On-Hold -> Transfer

Note: Processing of atomic operations used by the CSA. */



Process Type CallSegment

33(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

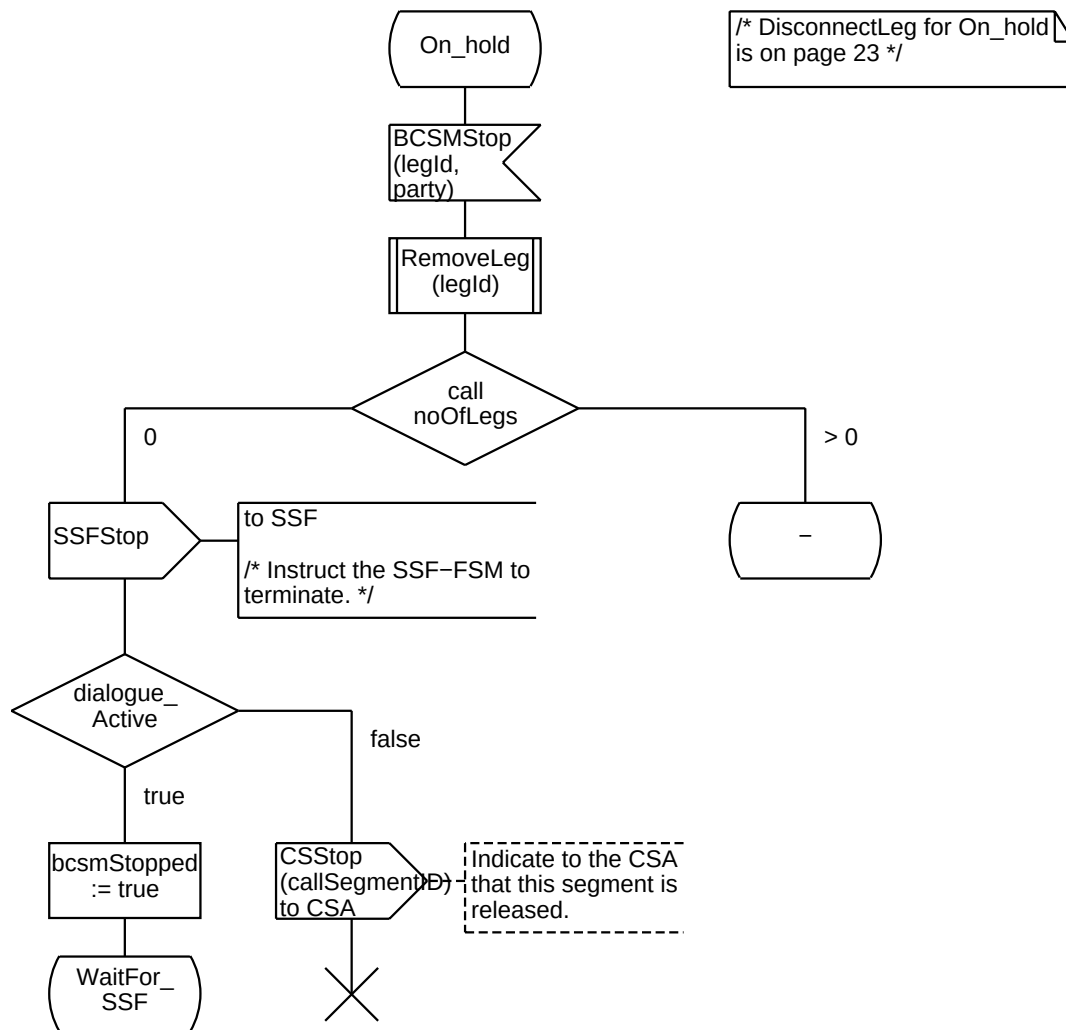
/* On-Hold -> Terminate call segment
On-Hold -> On-Hold

```

```

Note: Processing of atomic operations used by the CSA. */

```



Process Type	CallSegment

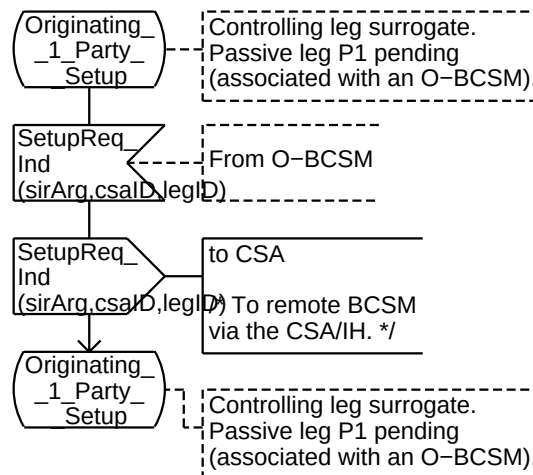
```

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* State state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

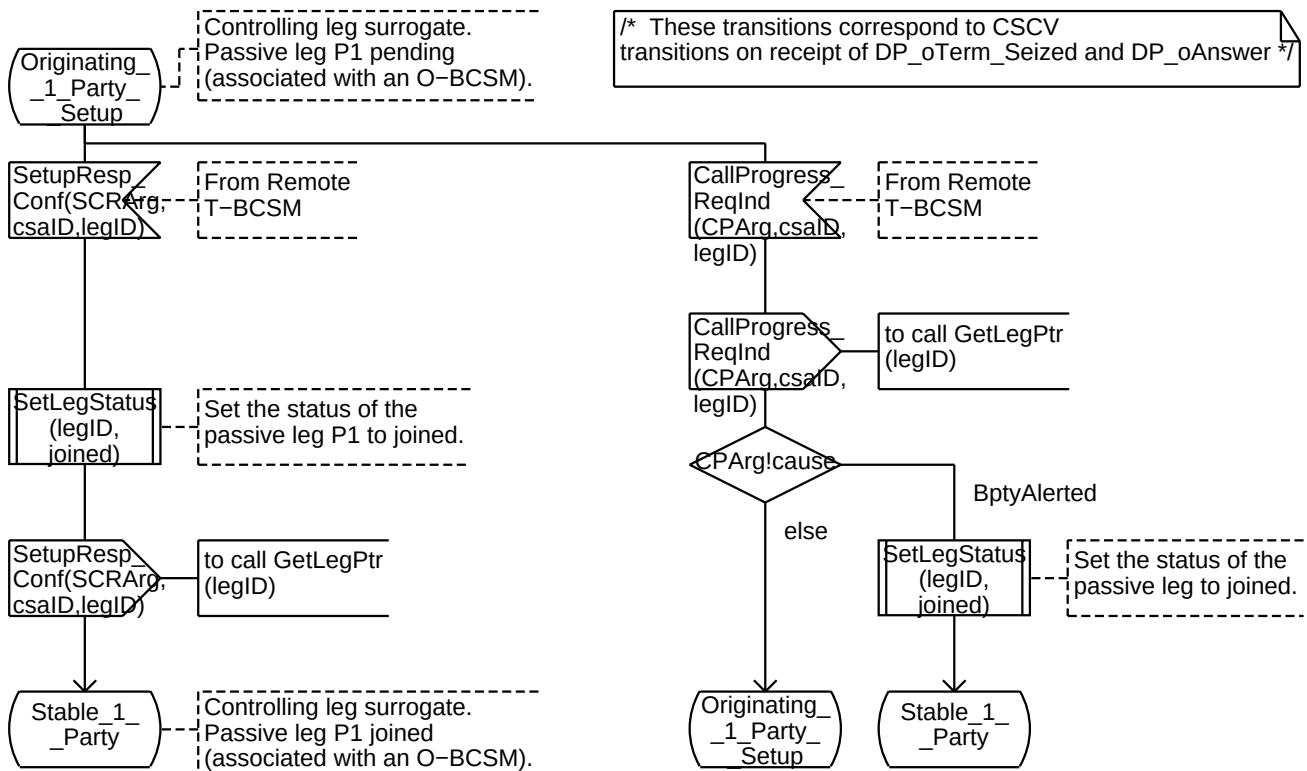
```

```
/* Originating-1-Party-Setup -> Stable 1 Party */
```

```
/* BCSMStop for Originating_
1_Party_Setup is on page 17 */
```



```
/* These transitions correspond to CSCV
transitions on receipt of DP_oTerm_Seized and DP_oAnswer */
```

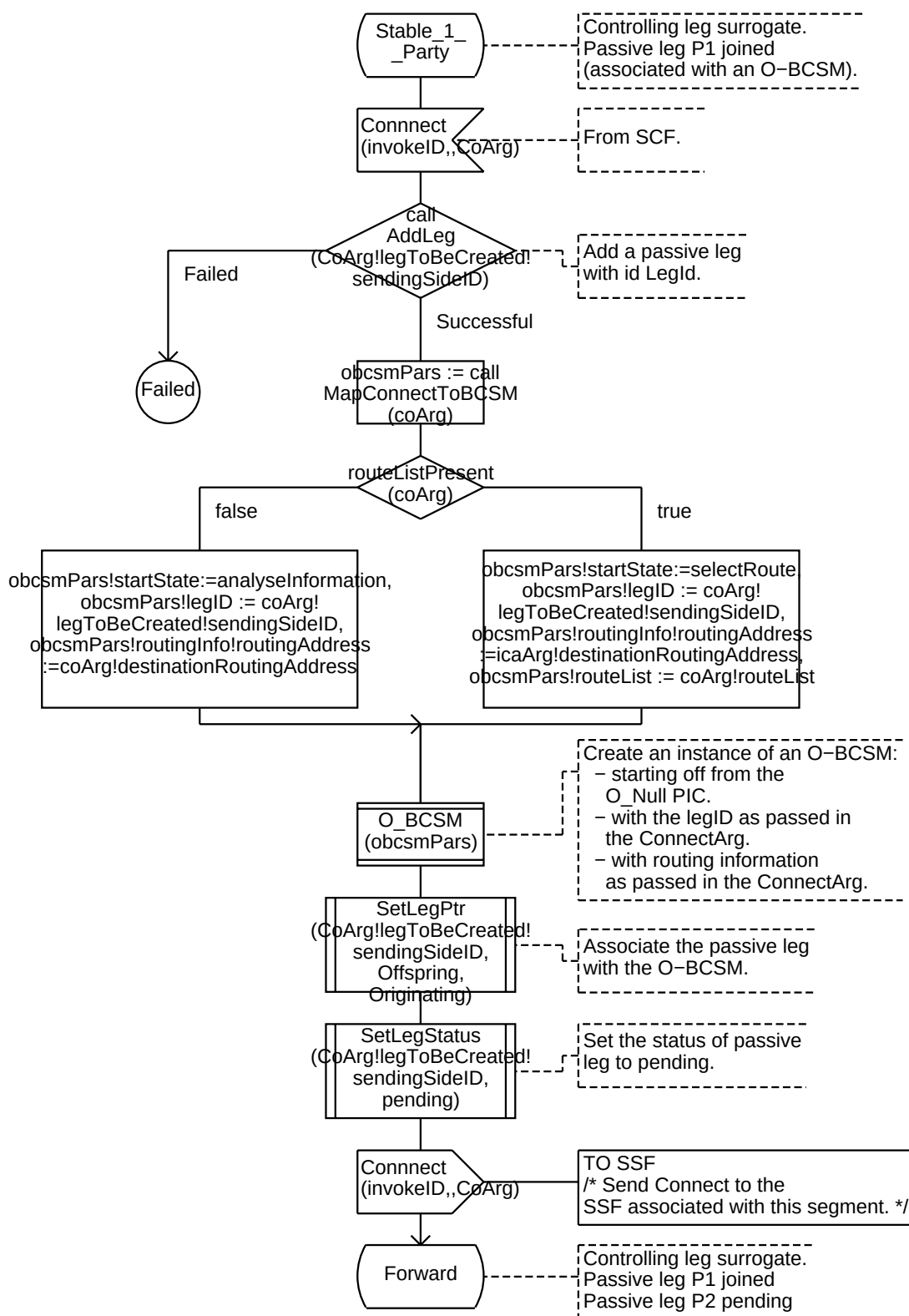


Process Type CallSegment

35(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* Stable 1 Party -> Forward */



Process Type CallSegment

36(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/* Stable-1-Party -> Stable-2-Party
Stable-1-Party -> Transfer

```

```

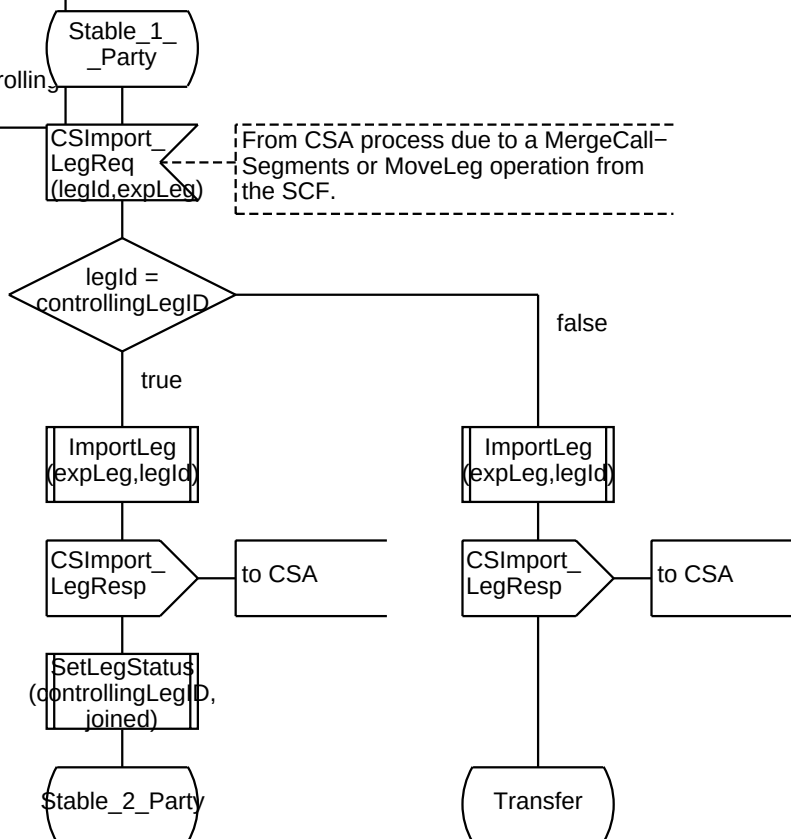
Note: Processing of atomic operations used by the CSA. */

```

```

/* Note: Import of the controlling leg
in Stable-1-Party occurs when, in the CSA to
which this CS belongs, first a CS with the
controlling leg joined is imported using
the MoveCallSegments operation, then the
MoveLeg operation is used to move the controlling
leg in to this CS. */

```



Process Type CallSegment

37(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

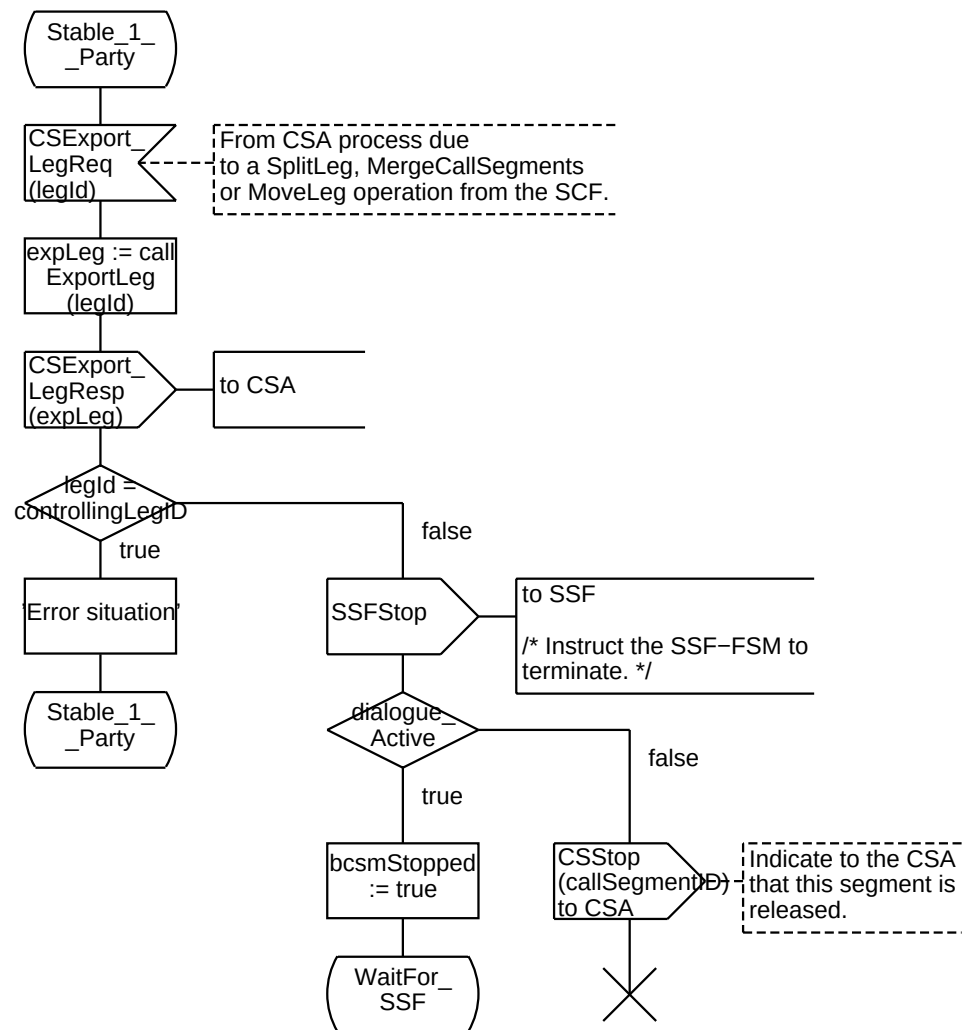
/* Stable-1-Party -> Stable-1-Party
Stable-1-Party -> Null (via WaitForSSF)

```

```

Note: Processing of atomic operations used by the CSA. */

```



Process Type CallSegment

38(80)

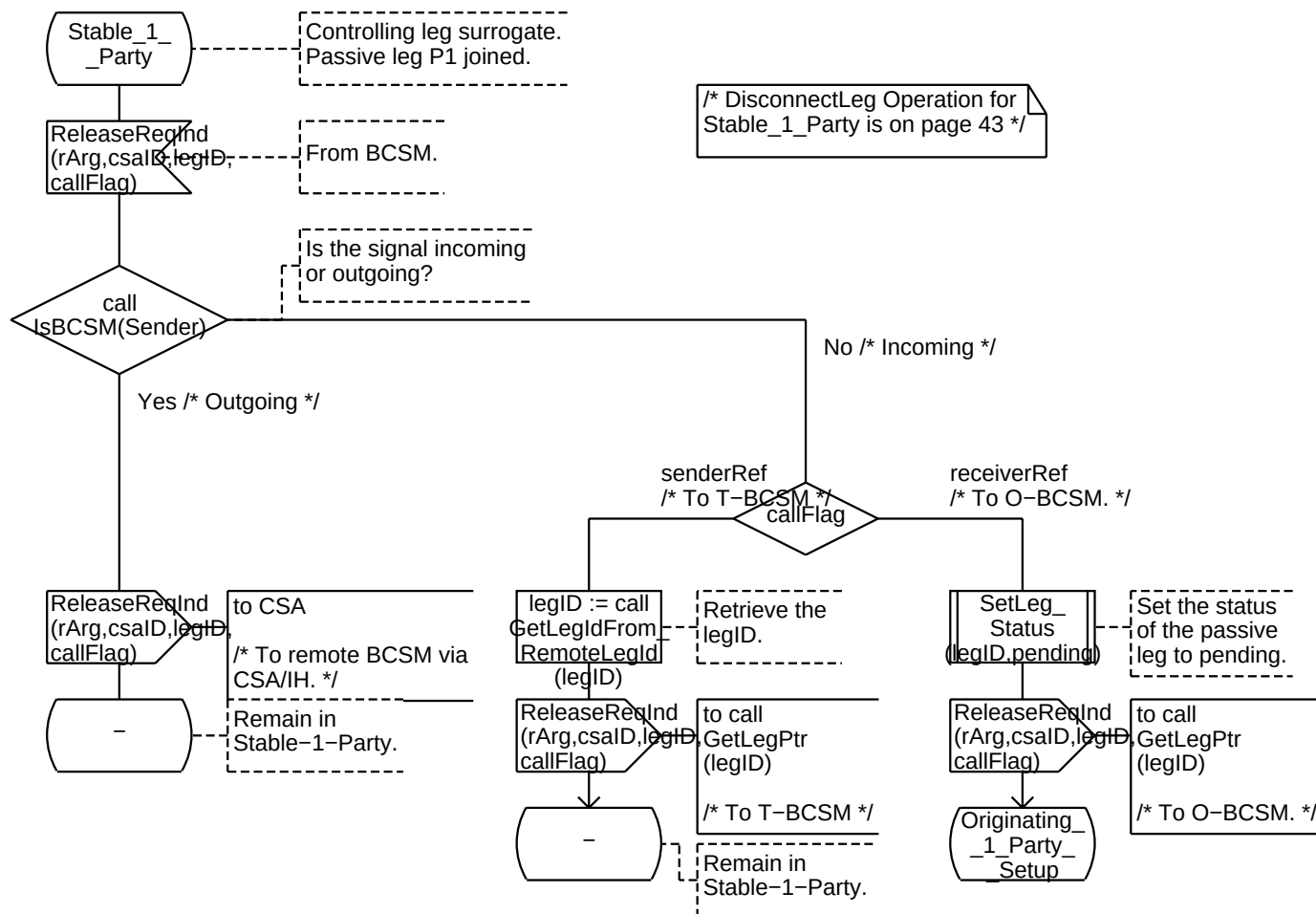
FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* Stable 2 Party -> Originating Setup

Notes: - ReleaseReqInd (from remote party) in stable 2 party on an O/T-BCSM results in that the CS goes 'back' to originating/terminating setup state.

*/

/* DisconnectLeg Operation for Stable_1_Party is on page 43 */



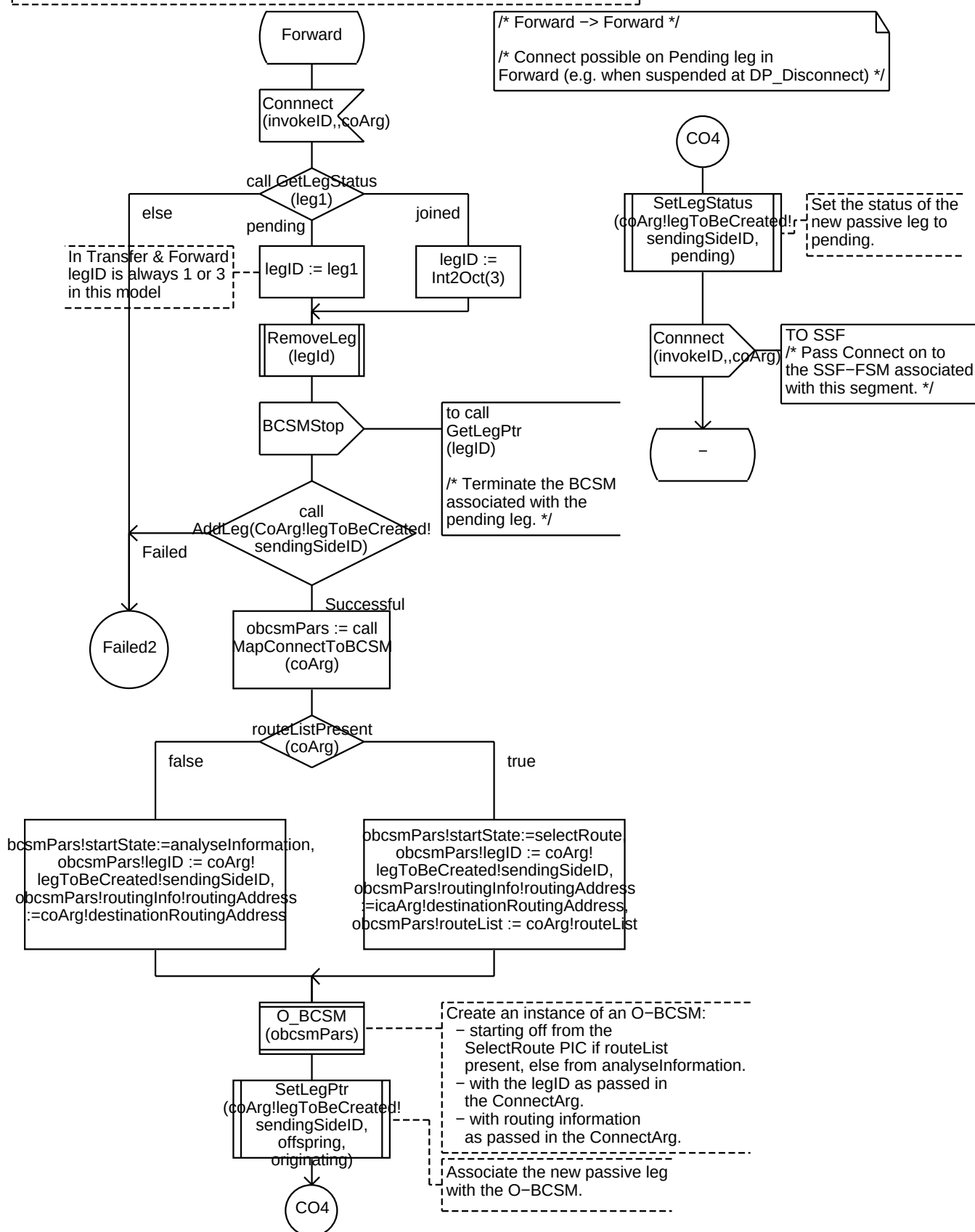
Annex A: CallSegment

Process Type CallSegment

39(80)

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* Forward -> Forward */
 /* Connect possible on Pending leg in Forward (e.g. when suspended at DP_Disconnect) */



Process Type CallSegment

40(80)

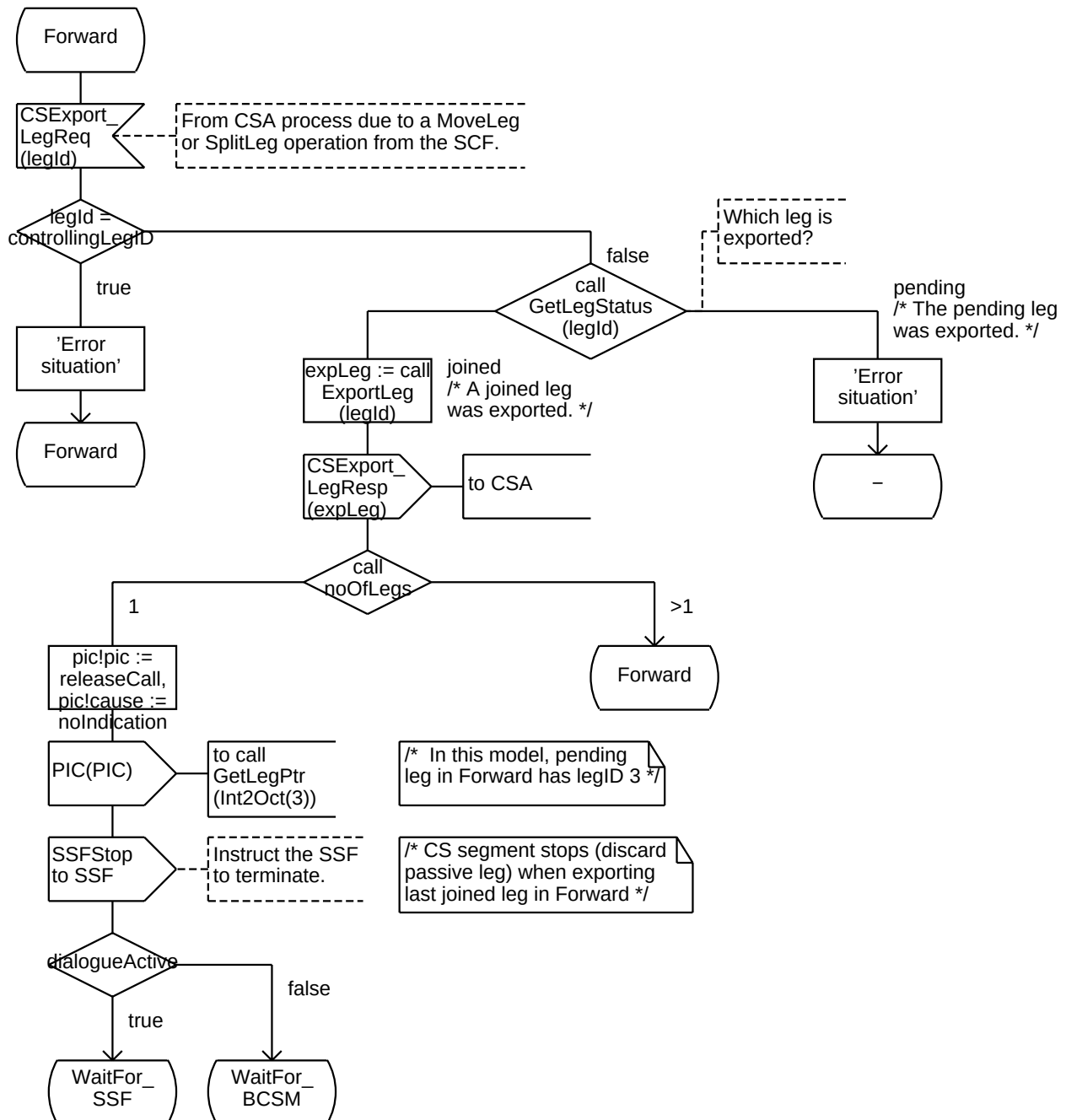
```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

/* Forward -> Forward

Note: Processing of atomic operations used by the CSA. */



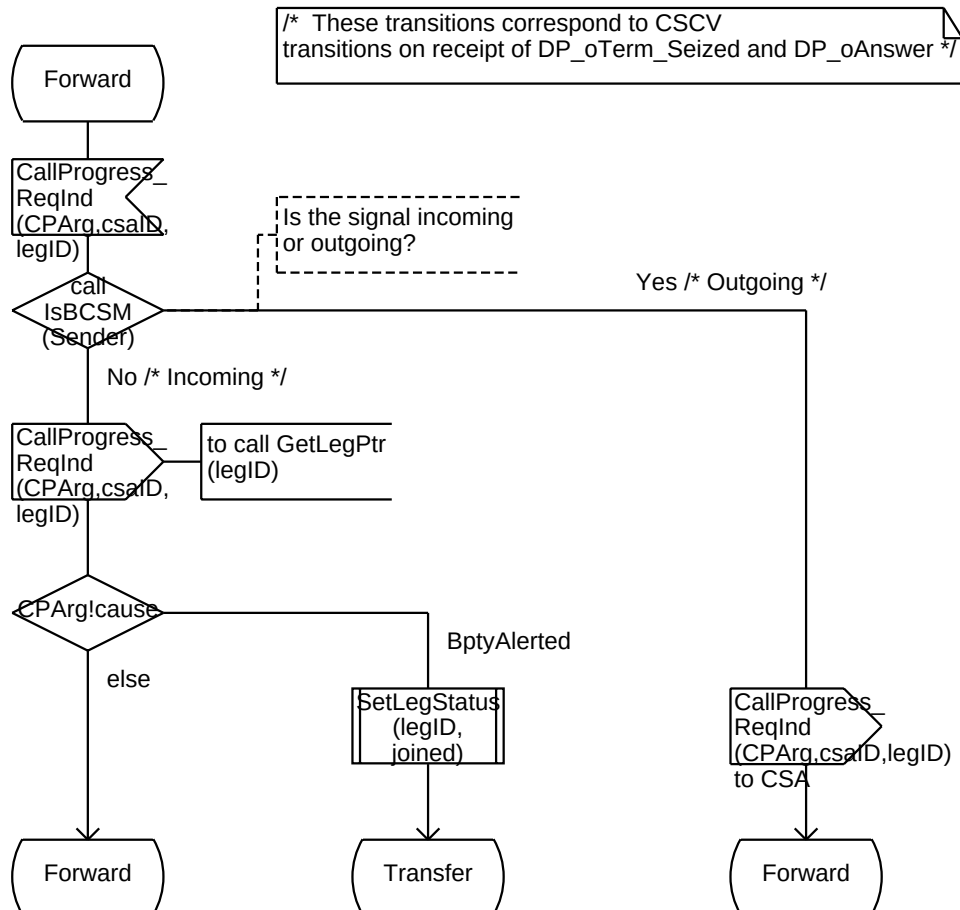
Process Type CallSegment

41(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```



Process Type CallSegment

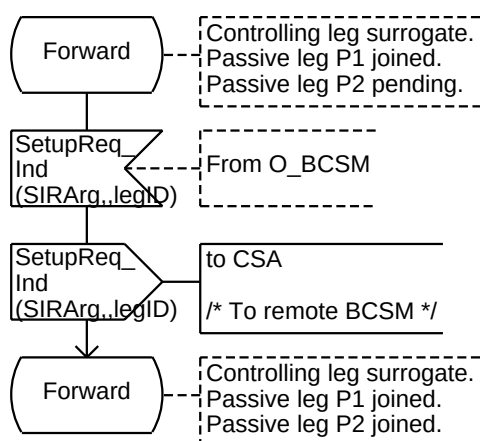
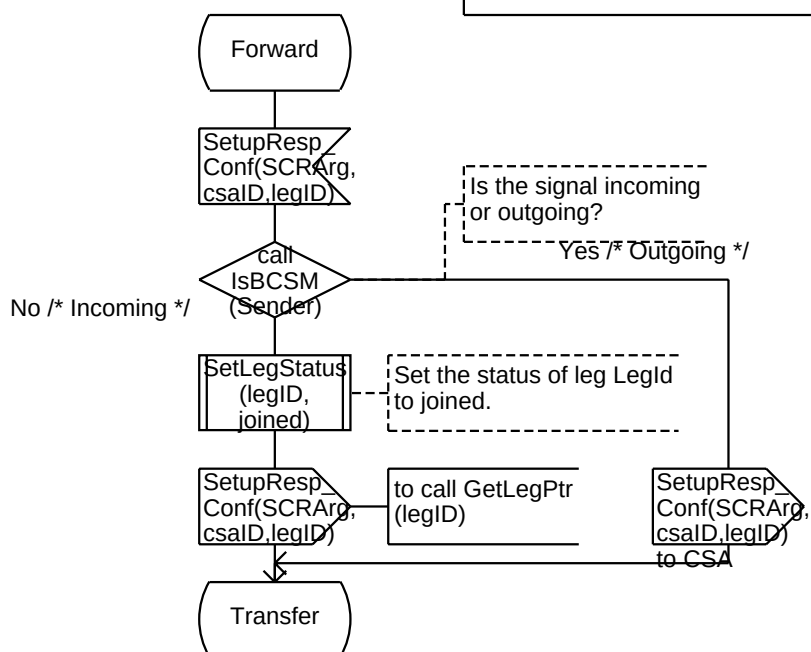
```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```
/* Forward -> Transfer */
```

```
/* These transitions correspond to CSCV
transitions on receipt of DP_oTerm_Seized and DP_oAnswer */
```



Process Type CallSegment

43(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

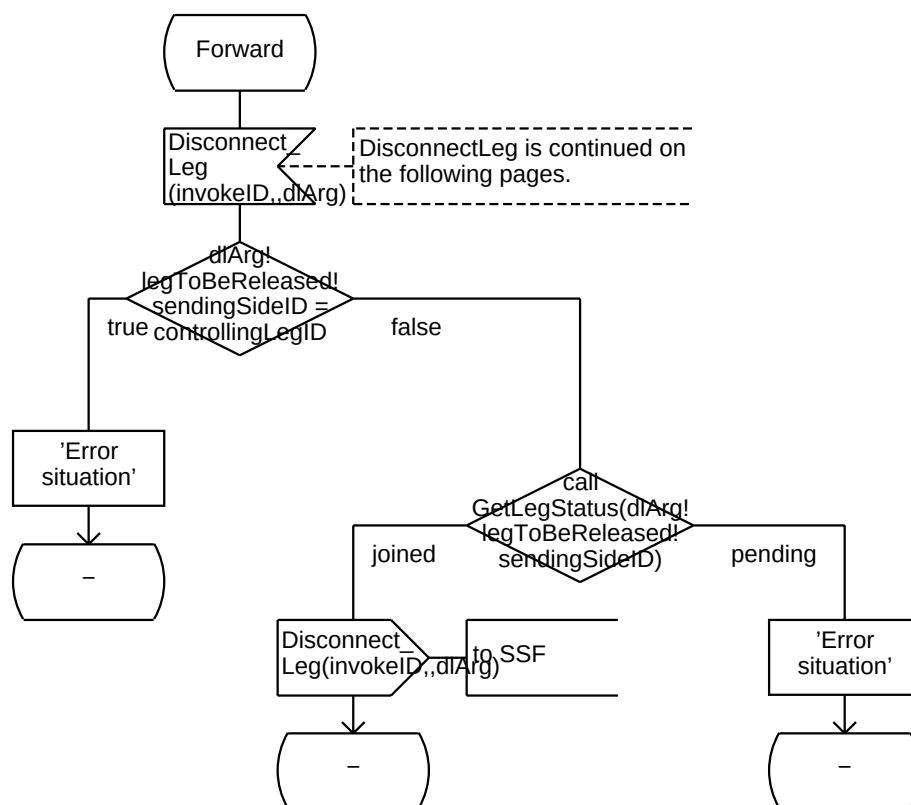
```

/* For the DisconnectLeg(p) operation, it is the BCSMStop event that removes the passive leg from the call segment. The Disconnect Leg operation is passed on to the SSF to allow the SSF to terminate the leg (BCSM) in a controlled way.

The transitions occurring on the BCSM stop event (and specified at other pages) are:

Forward -> Forward
 Forward -> Originating-1-Party-Setup
 Forward -> Stable-1-Party */

/* Forward -> Forward */



Process Type CallSegment

44(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

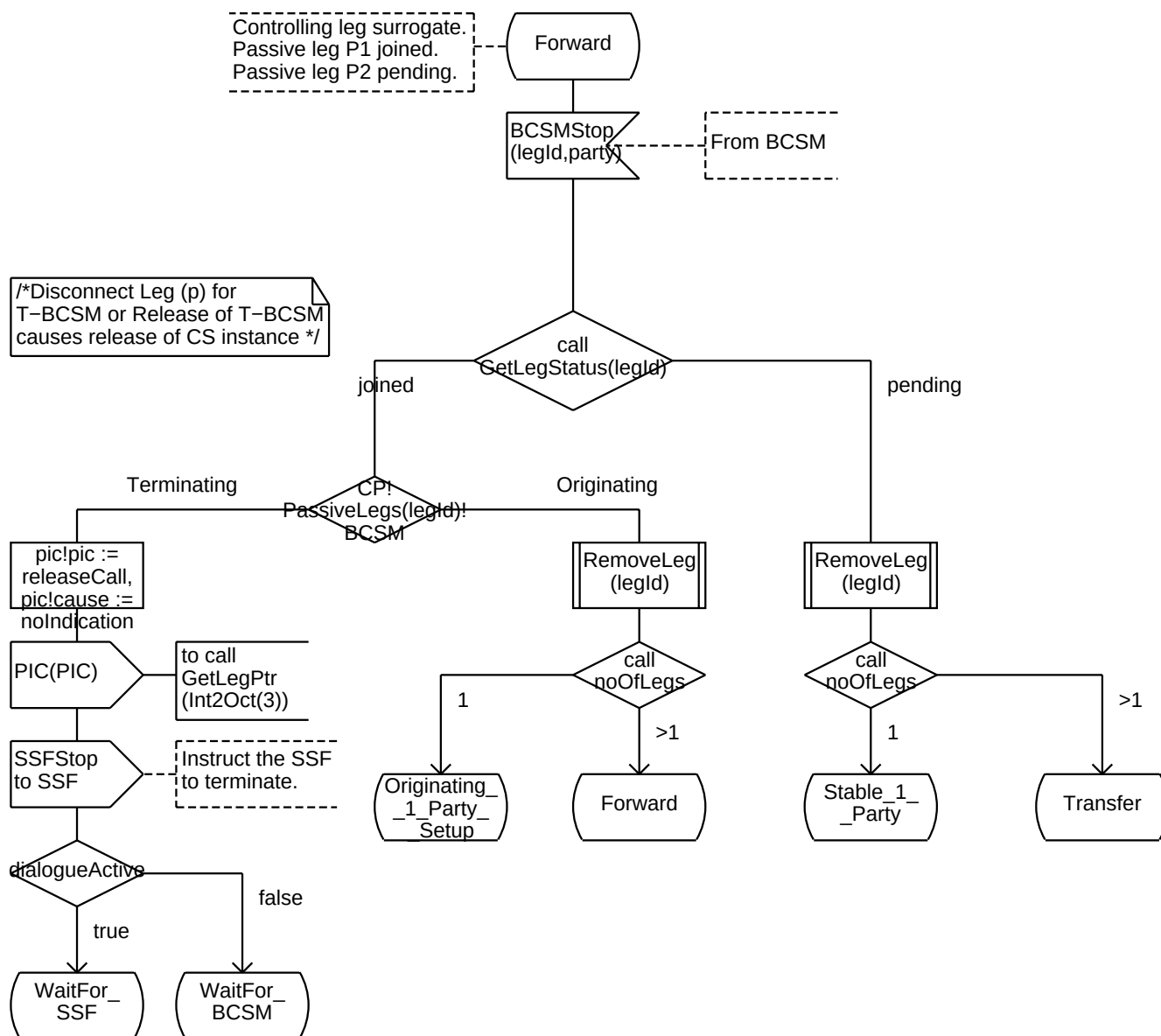
```

/* Forward -> Originating 1-Party-Setup
Forward -> Forward
Forward -> Stable-1-Party
Forward -> Transfer

```

Note: Processing of atomic operations used by the CSA. */

/* Note: The Forward state is composed of one passive pending leg and one or more passive joined legs. It has to be determined if it was the pending leg or one of the joined legs that released. */



Process Type CallSegment

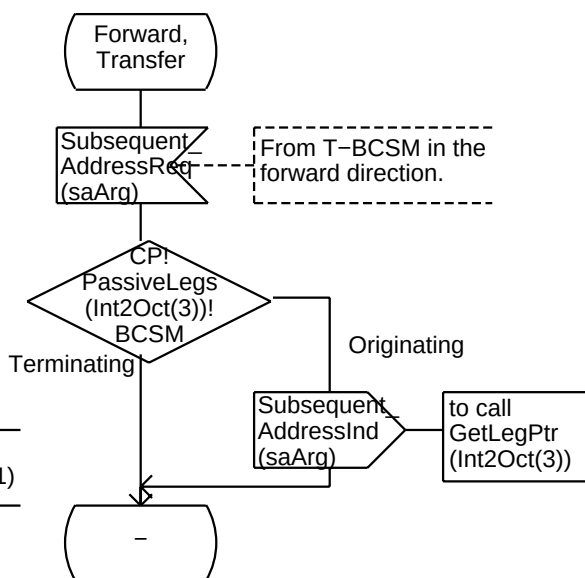
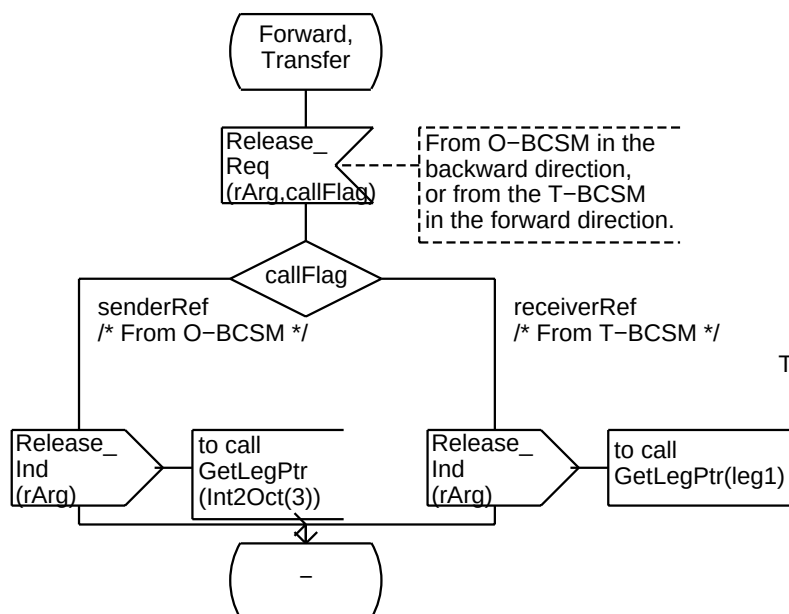
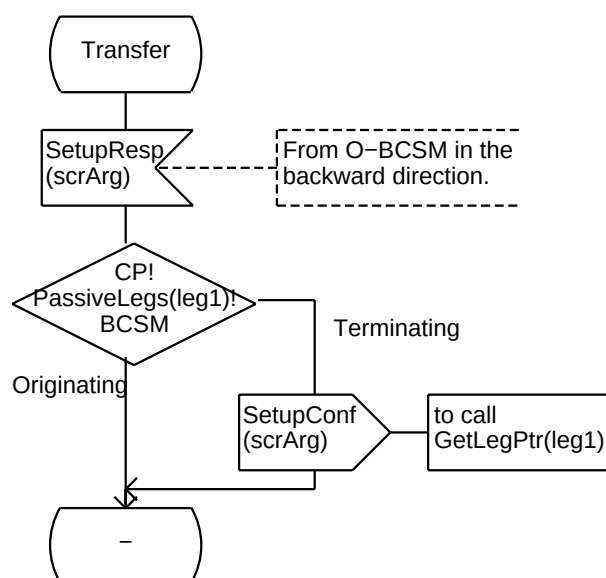
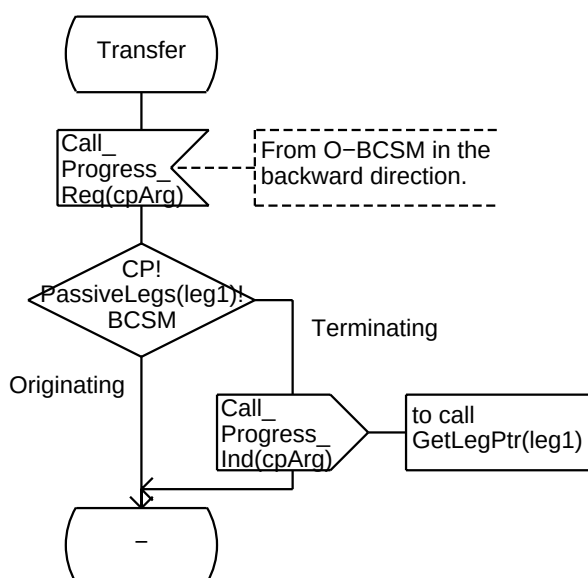
45(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

In the Transfer state, all backward and forward signalling messages to/from the BCSMs must be relayed since the controlling leg is surrogate.

In the Forward state, only ReleaseReq and Subsequent-AddressReq needs to be relayed. */

/* Relaying of messages from O-BCSM to T-BCSM is different than to O-BCSM, as 2 O-BCSMs only exist in case of SCF initiated call. */

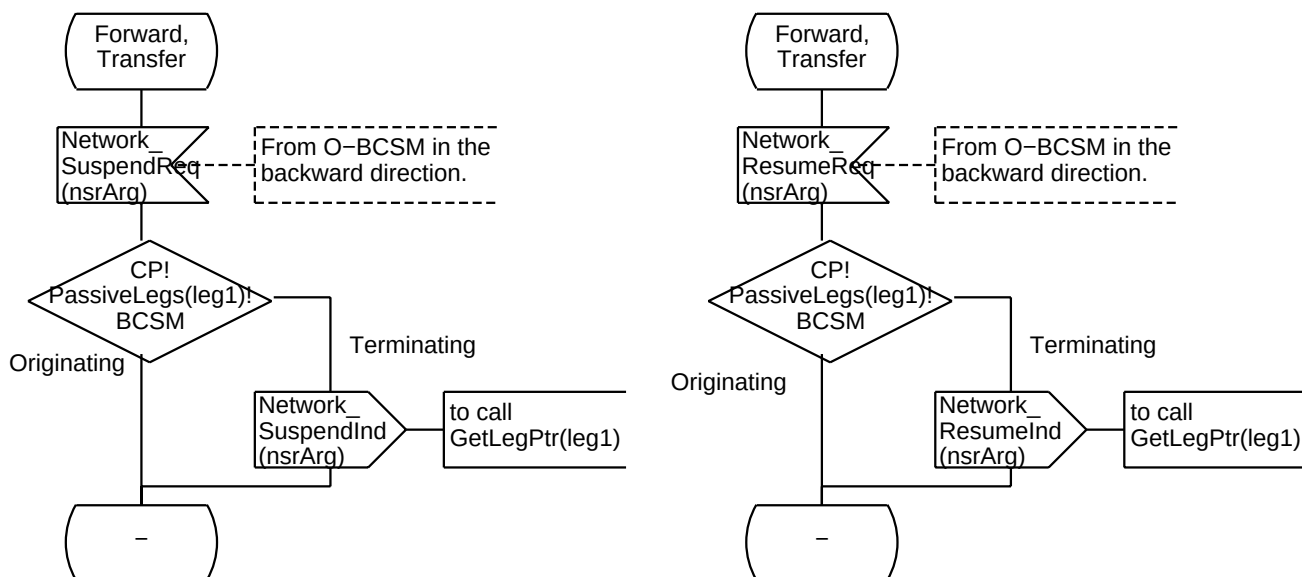


Process Type CallSegment

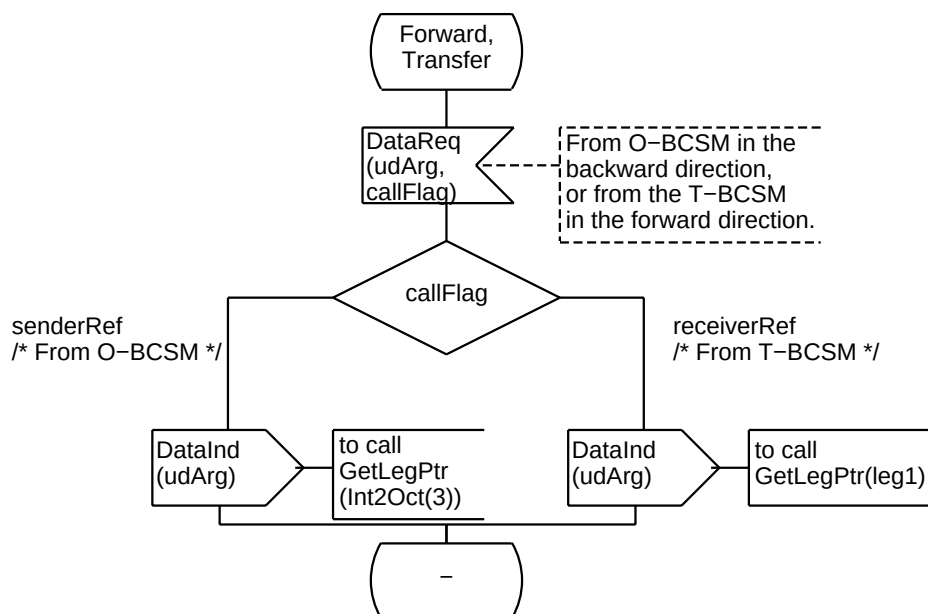
46(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* In the Transfer and Forward states, the NetworkSuspendReq, NetworkResumeReq and DataReq to/from the BCSMs must be relayed since the controlling leg is surrogate. */



/* Relaying of messages from O_BCSM to T_BCSM is different than from O_BCSM to O_BCSM, as 2 O_BCSMs only exist in case of SCF initiated call. */



Process Type CallSegment

47(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/* Transfer -> Stable-Multi-Party
Transfer -> On Hold

```

```

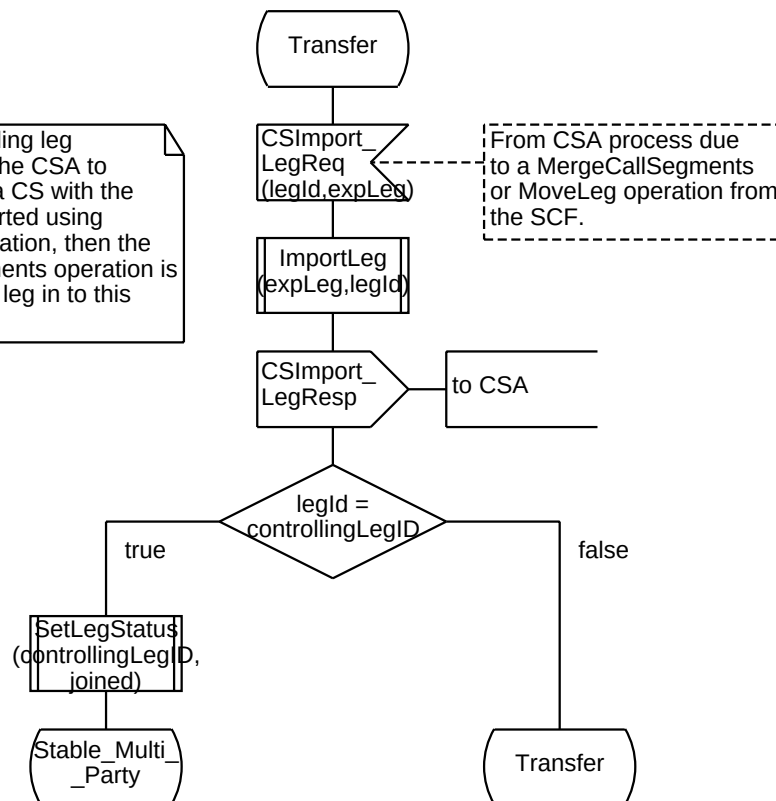
Note: Processing of atomic operations used by the CSA. */

```

```

/* Note: Import of the controlling leg
in Transfer occurs when, in the CSA to
which this CS belongs, first a CS with the
controlling leg joined is imported using
the MoveCallSegments operation, then the
MoveLeg or MergeCallSegments operation is
used to move the controlling leg in to this
CS. */

```



Process Type CallSegment

48(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

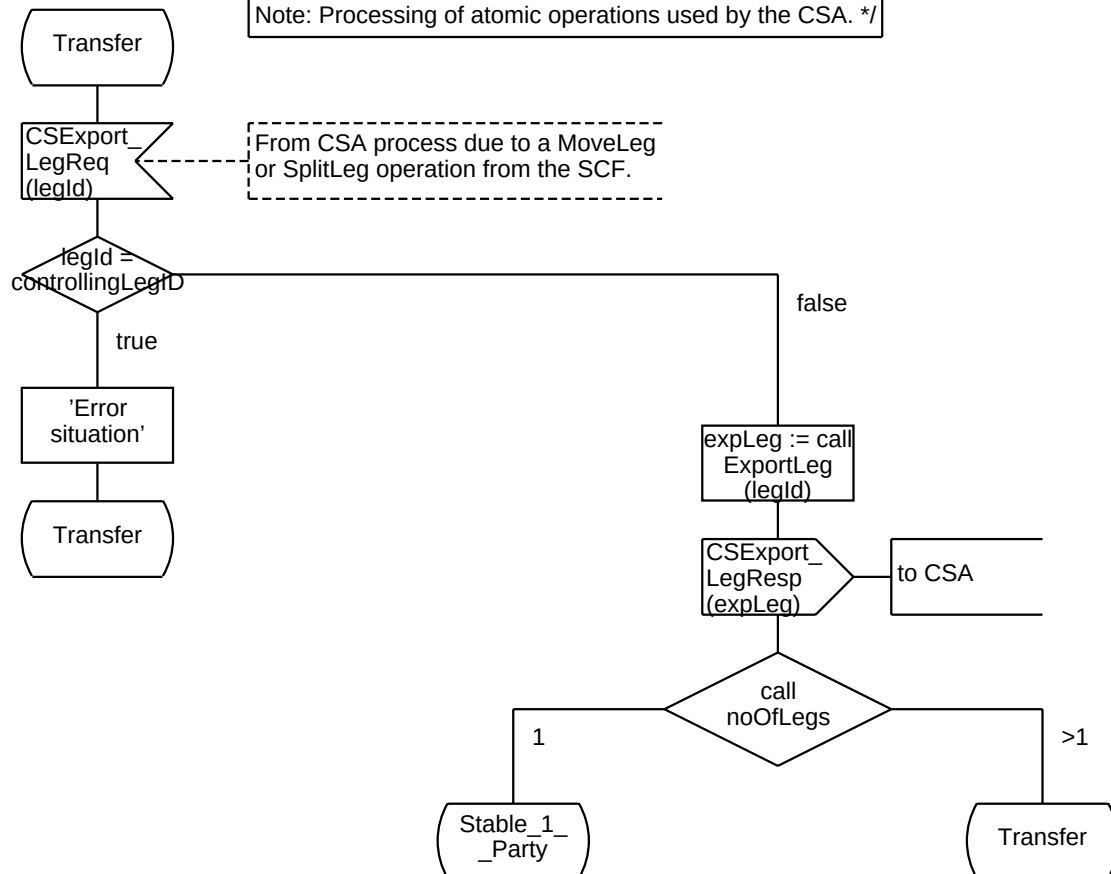
```

```

/* Transfer -> Transfer
Transfer -> Stable-1-Party
Transfer -> Transfer

```

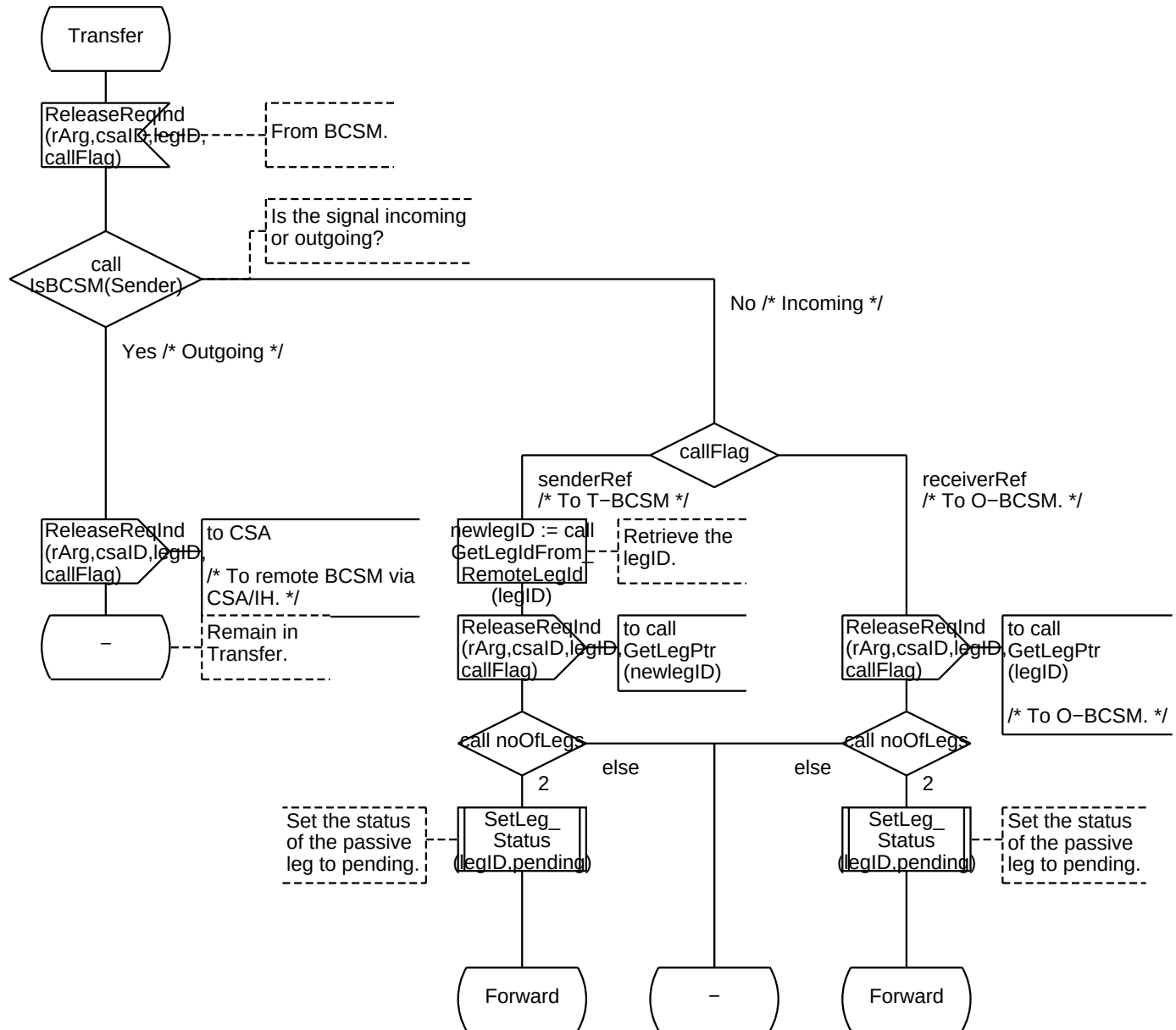
Note: Processing of atomic operations used by the CSA. */



Process Type CallSegment

49(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */



Process Type CallSegment

50(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

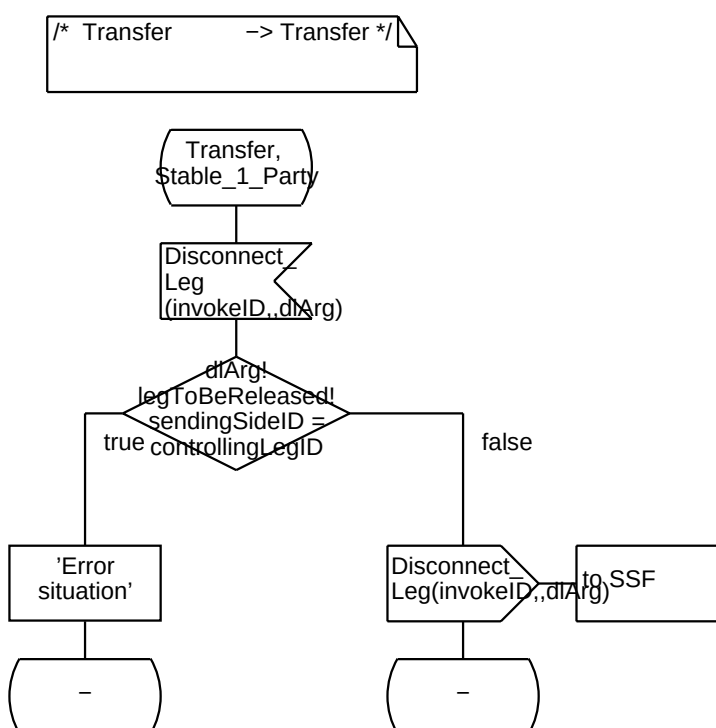
```

/* For the DisconnectLeg(p) operation, it is the BCSMStop event that removes the passive leg from the call segment. The Disconnect Leg operation is passed on to the SSF to allow the SSF to terminate the leg (BCSM) in a controlled way.

The transitions occurring on the BCSM stop event (and specified at other pages) are:

Stable-1-Party -> Null

Transfer -> Stable-1-Party */



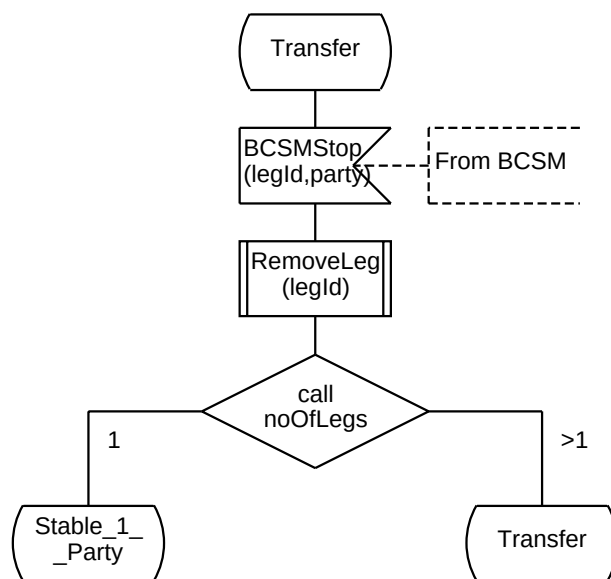
Process Type CallSegment

51(80)

```
FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */
```

```
/* Transfer -> Stable-1-Party
Transfer -> Transfer
```

```
Note: Processing of atomic operations used by the CSA. */
```



Process Type CallSegment

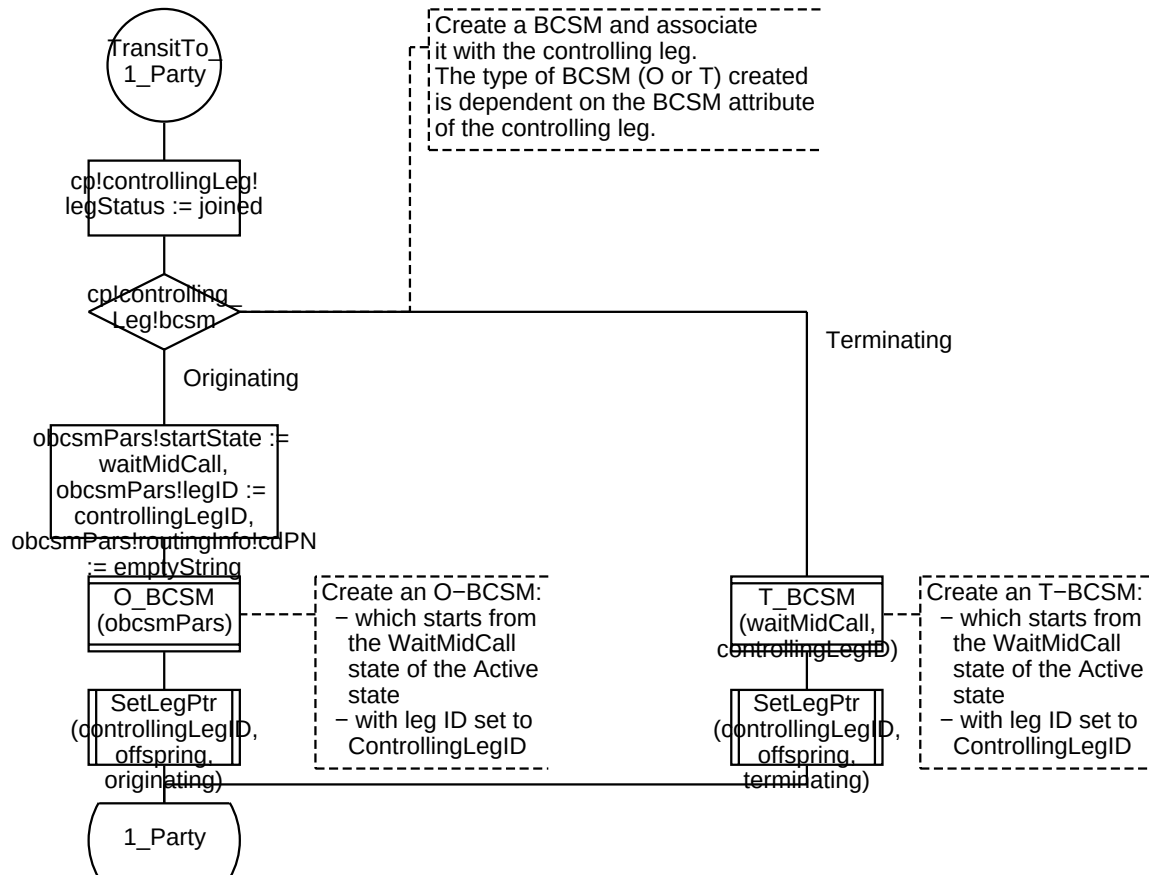
52(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

/* The transition to the 1-Party state involves creating a BCSM and associate it with the controlling leg. */

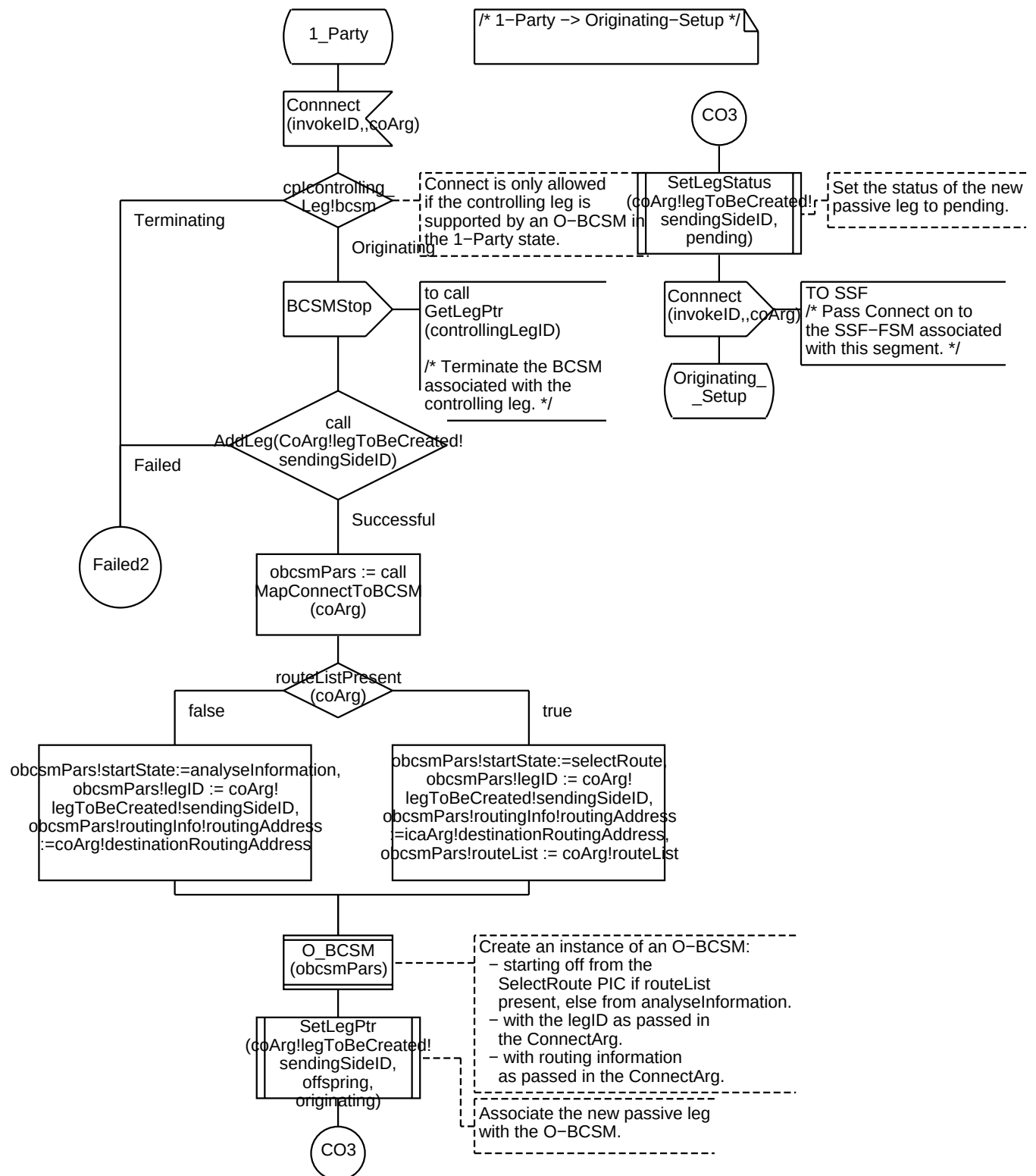


Annex A: CallSegment

Process Type CallSegment

53(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

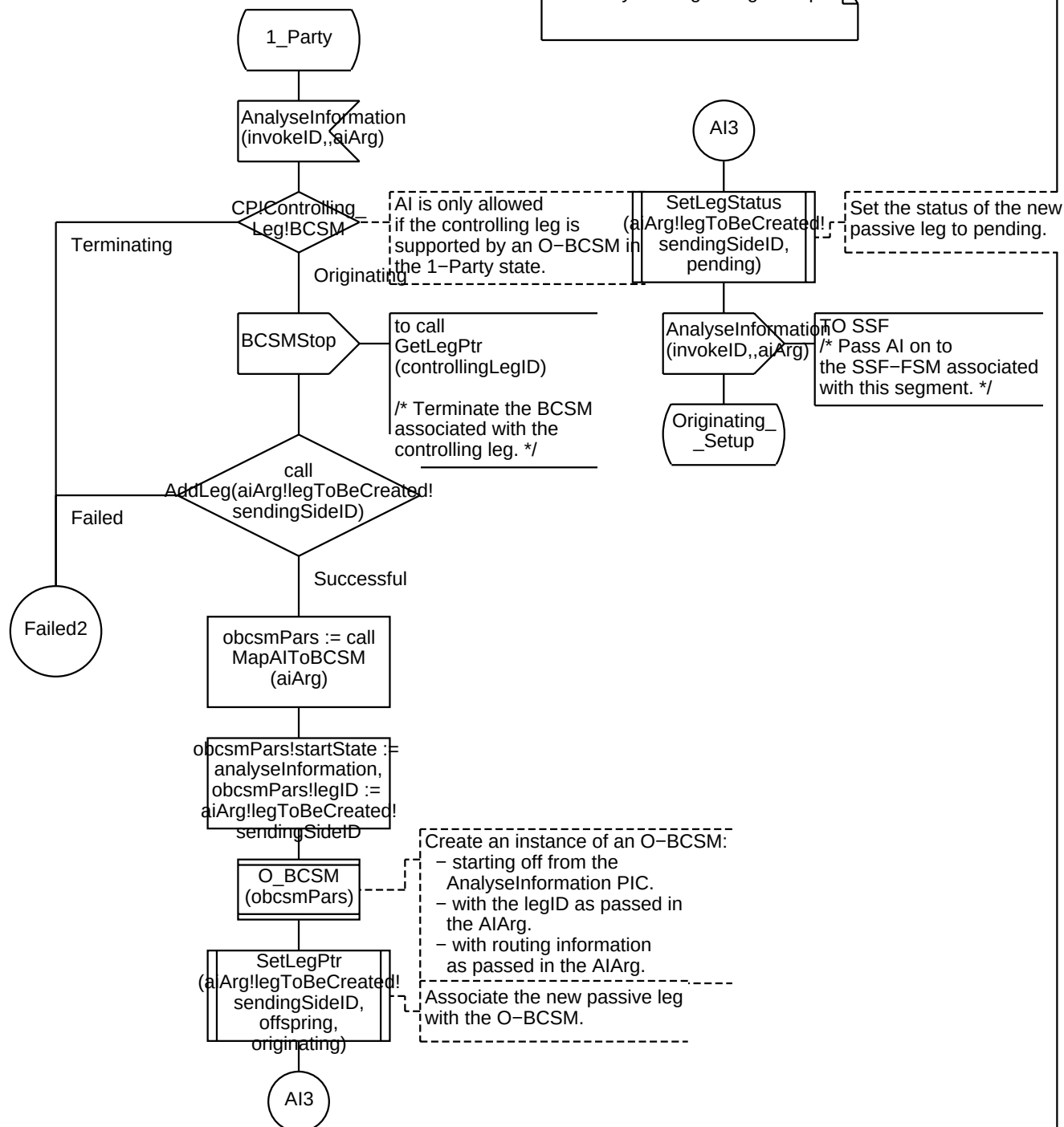


Process Type CallSegment

54(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* 1-Party -> Originating-Setup */

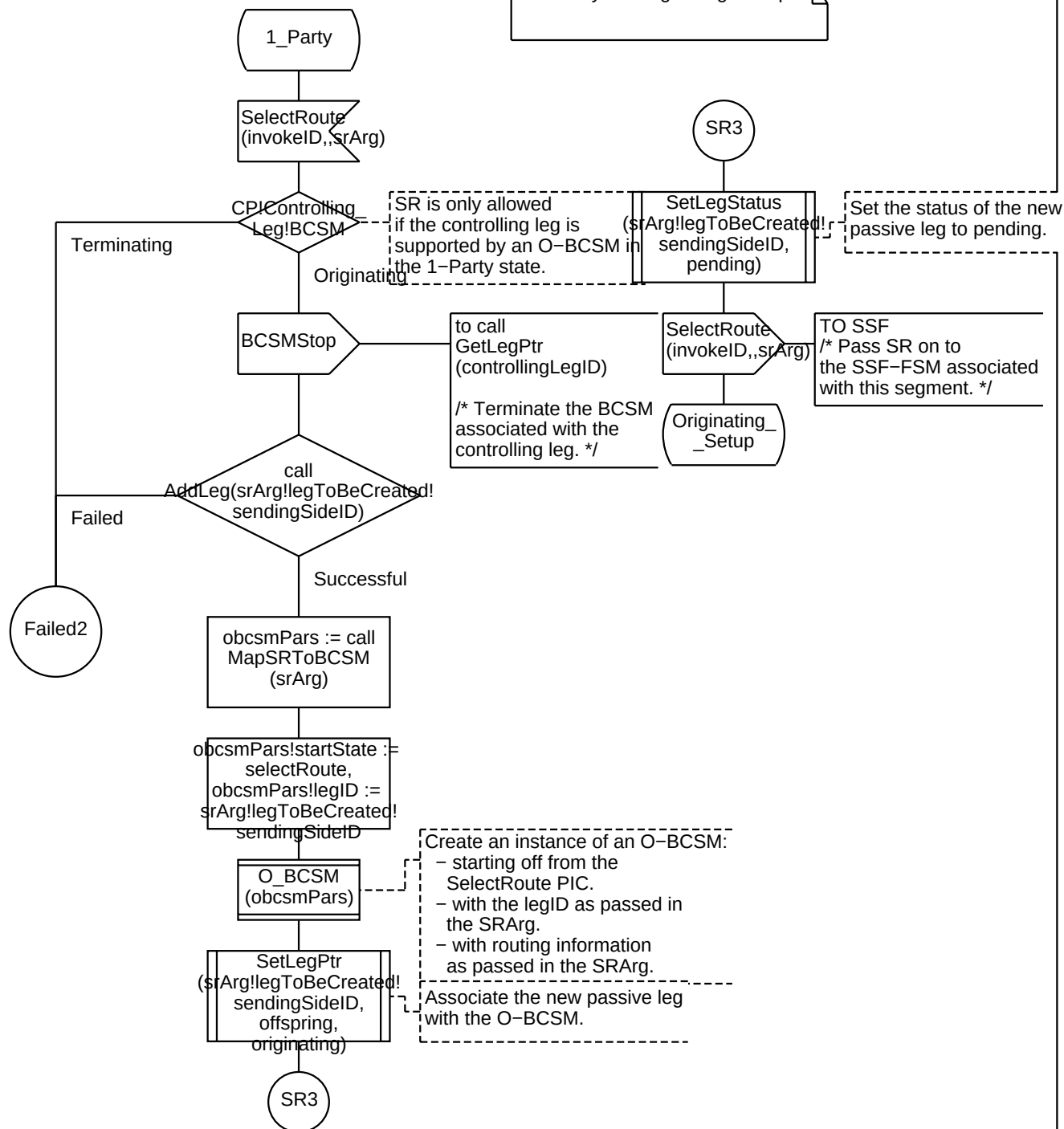


Process Type CallSegment

55(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* 1-Party -> Originating-Setup */



Process Type CallSegment

56(80)

```

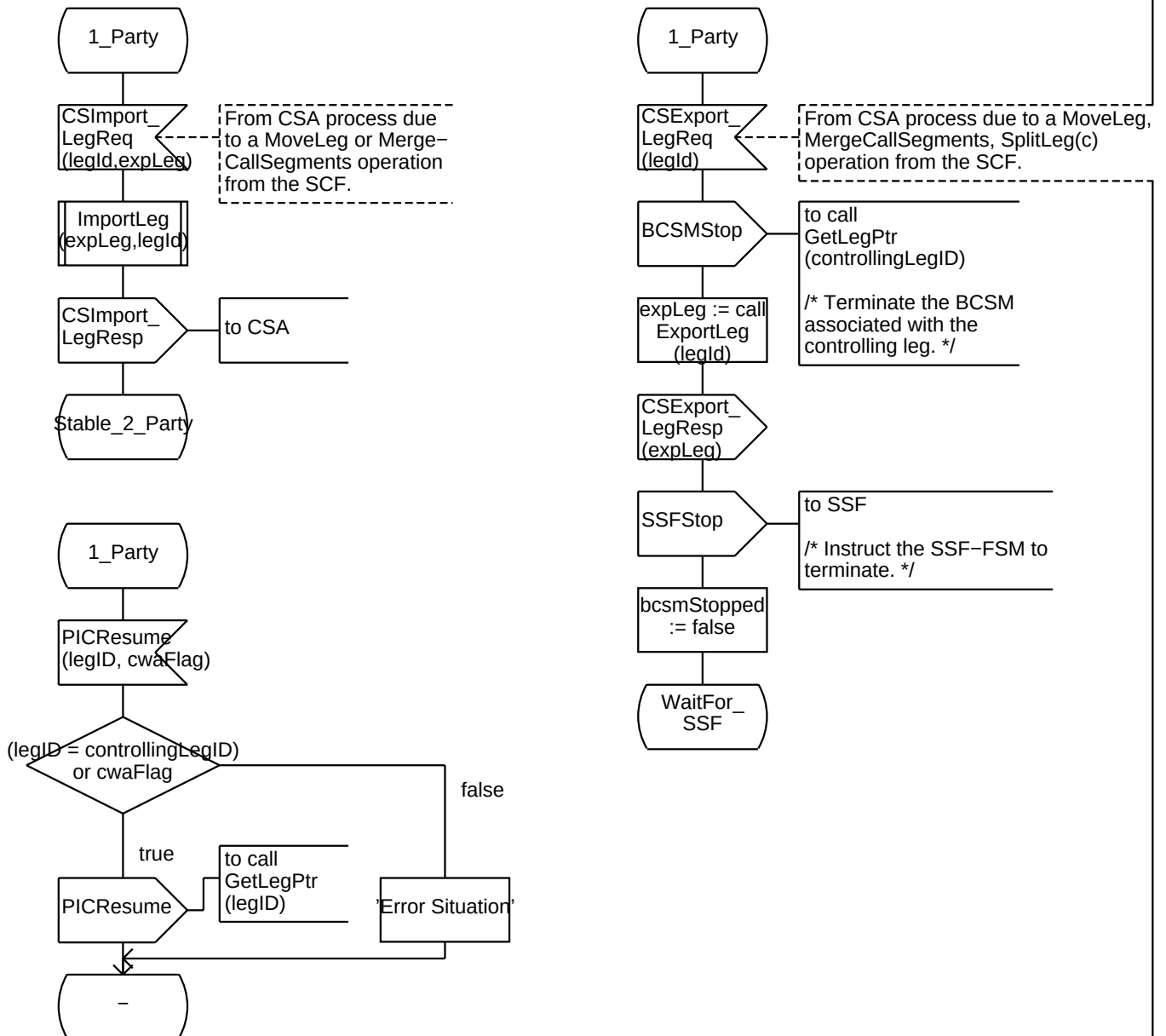
;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/* 1-Party -> Stable-2-Party
1-Party -> Null */

```

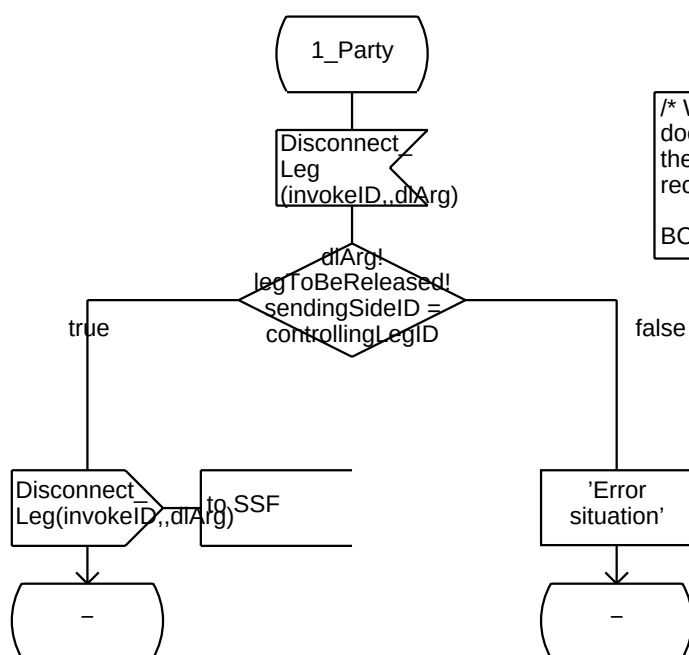


Process Type CallSegment

57(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */
    
```



/* Where the DisconnectLeg operation does not result in an error situation, the CS instance will be deleted on receipt of BCSMStop. BCSMStop for 1_Party is on page 17 */

Process Type CallSegment

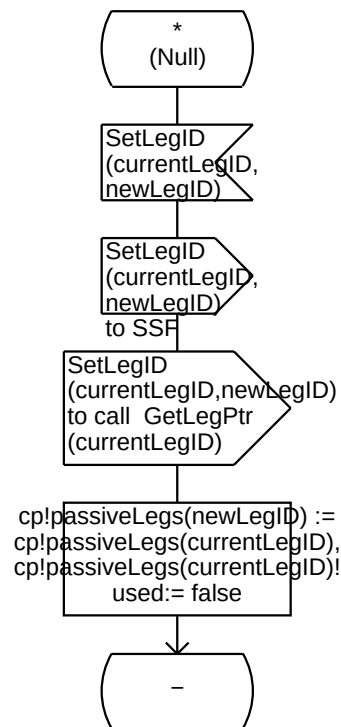
58(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

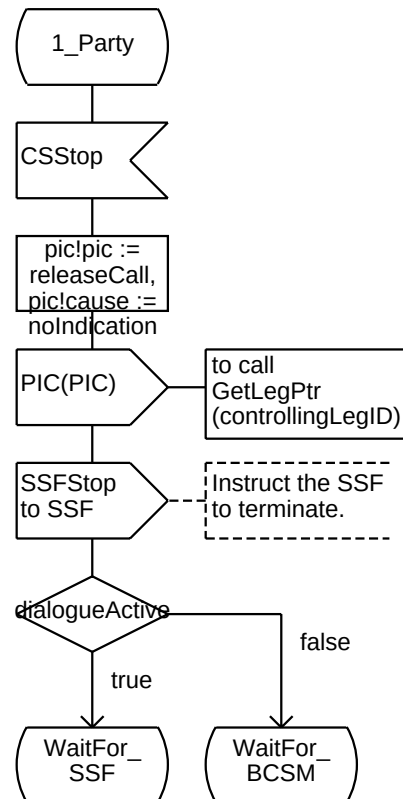
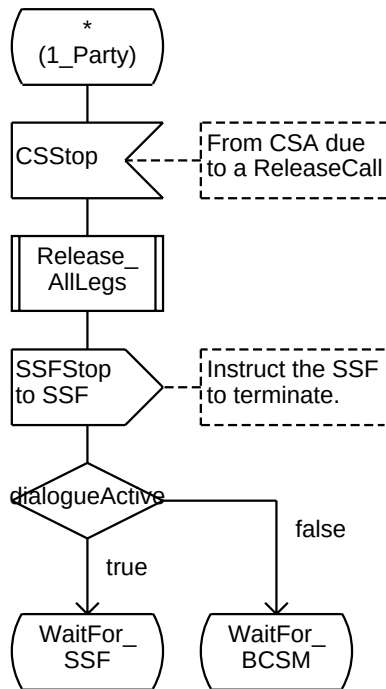
/* When the MoveCallSegments operation is processed by the IH and CSA, the CS may be requested to renumber the legs. In such a case, the CSA sends the signal SetLegID to the CS prior to exporting it into another CSA. */



```

- FPAR /* Parameters assigned by the CSA at creation of the call segment. */
- callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
- controllingLegID LegType, /* Id of the controlling leg. */
- ssfState SSFStateType, /* Start state of the SSF-FSM. */
- scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

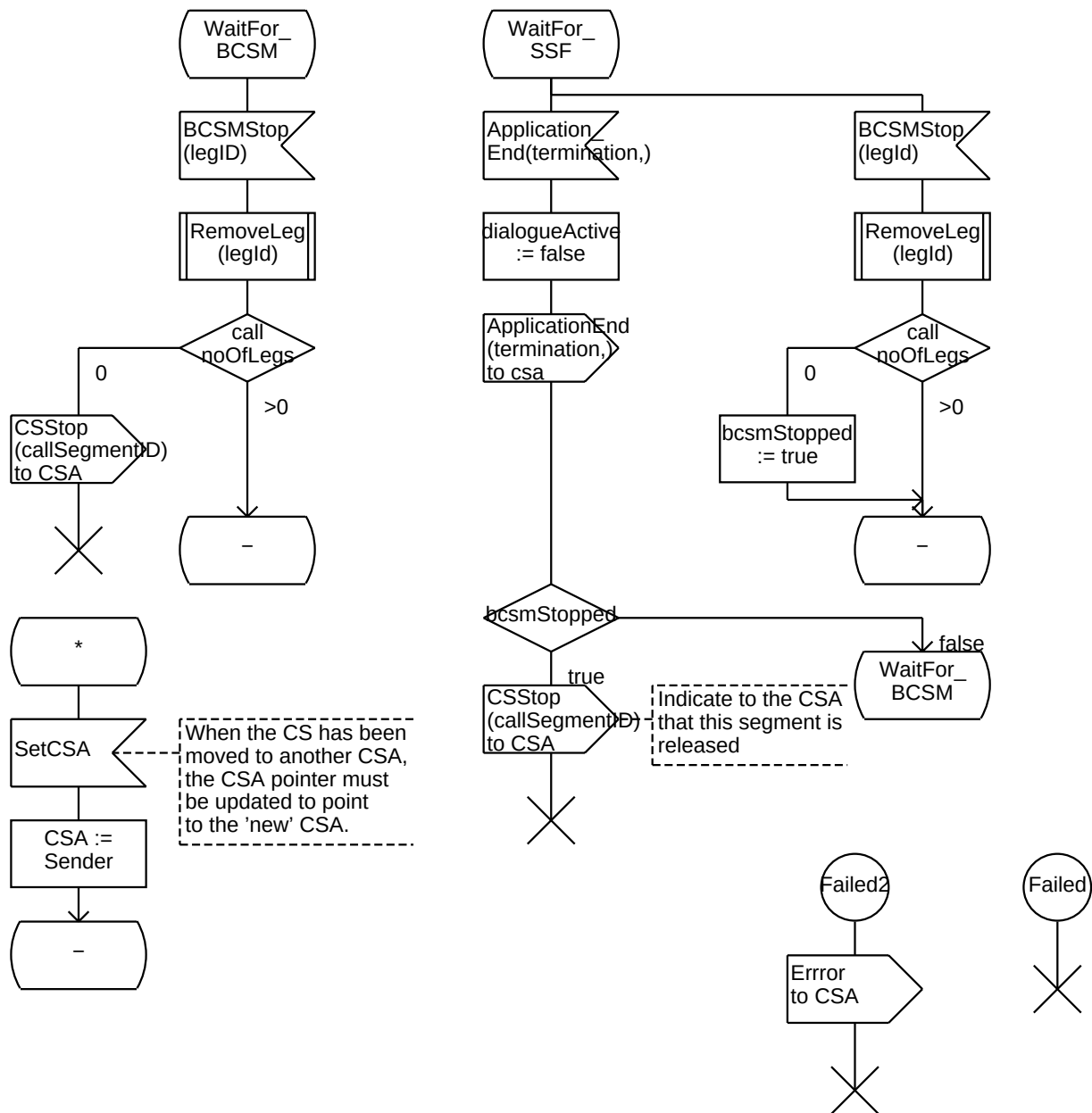
```



Process Type CallSegment

60(80)

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */



Process Type CallSegment

61(80)

```

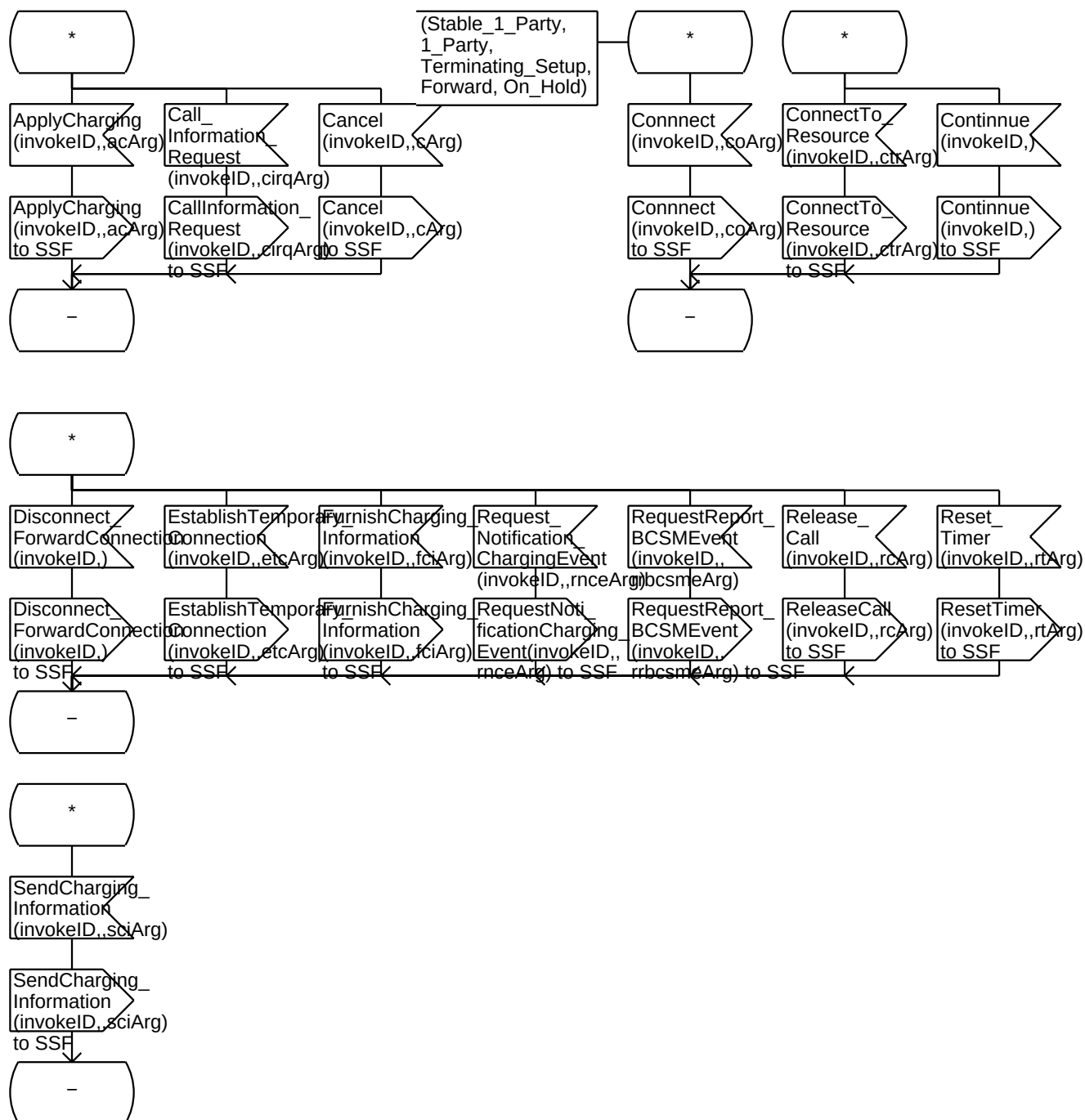
;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/**** ROUTING OF OPERATIONS TO THE SSF-FSM ****/

```

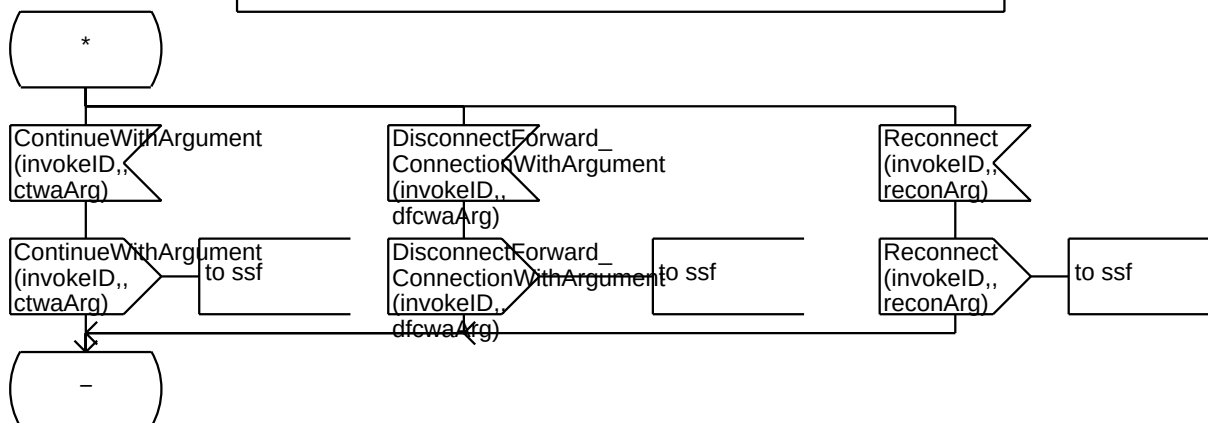


Process Type CallSegment

62(80)

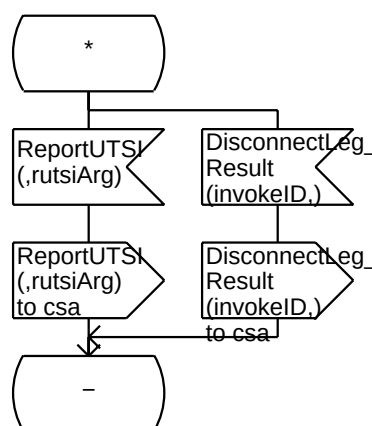
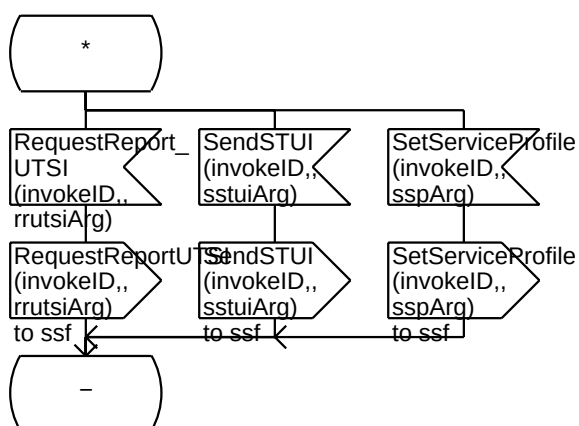
FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

/* ROUTING OF IN OPERATIONS TO/FROM THE SSF-FSM */



/* Note:
 DisconnectLeg does not change the connection state.
 It is the BCSMStop event (indicating that the BCSM
 has stopped) that changes the connection state. */

/* Note:
 SendSTUI on the controlling leg in any multi party state
 will result in that all BCSMs will generate a DataReq towards
 the signalling control. The call segment should filter these
 DataReqs and pass on only one DataReq to the signalling
 control. */

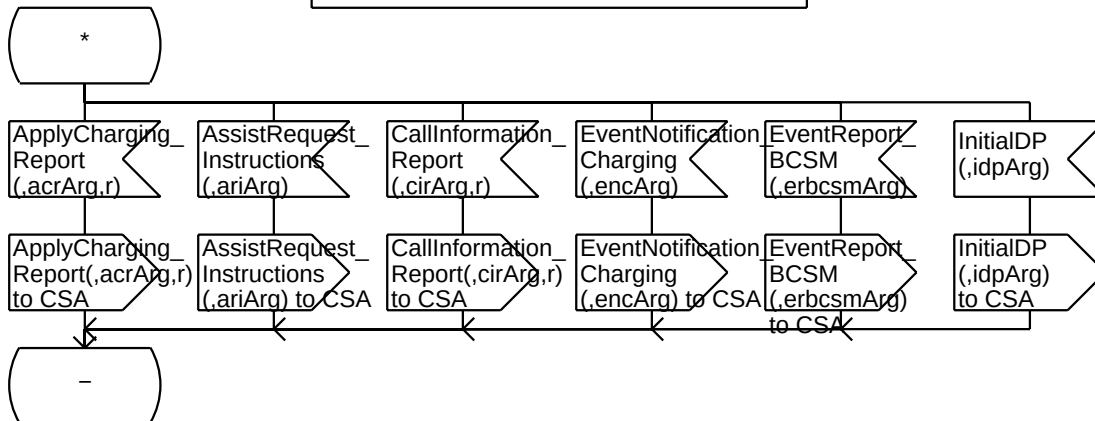


63(80)

```

/* IN operations from the SSF to the SCF. */

```



Process Type CallSegment

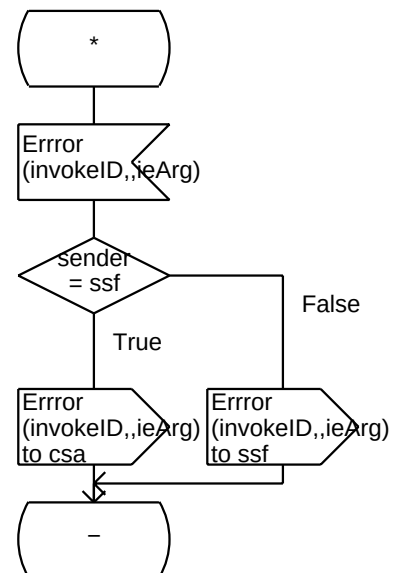
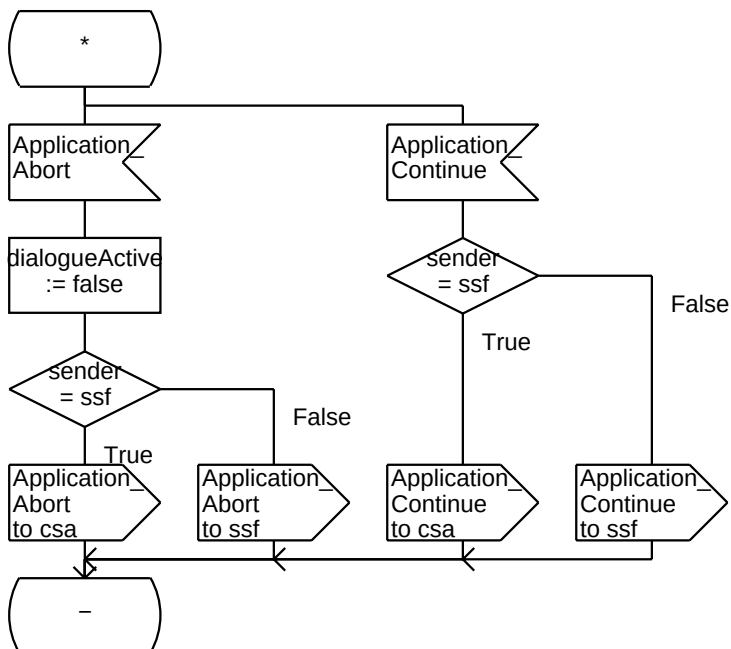
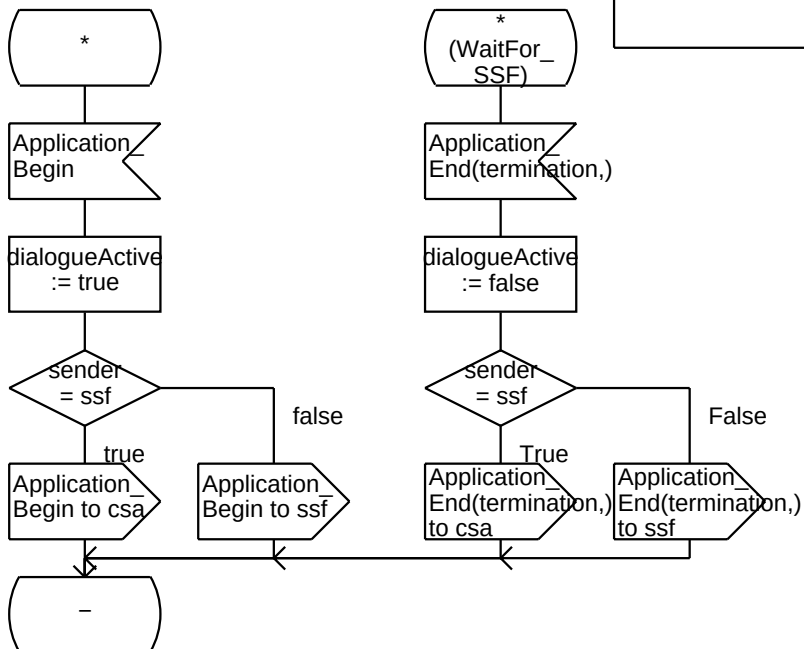
64(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

**** ROUTING OF DIALOGUE AND ERROR PRIMITIVES. ****



Process Type CallSegment

65(80)

```

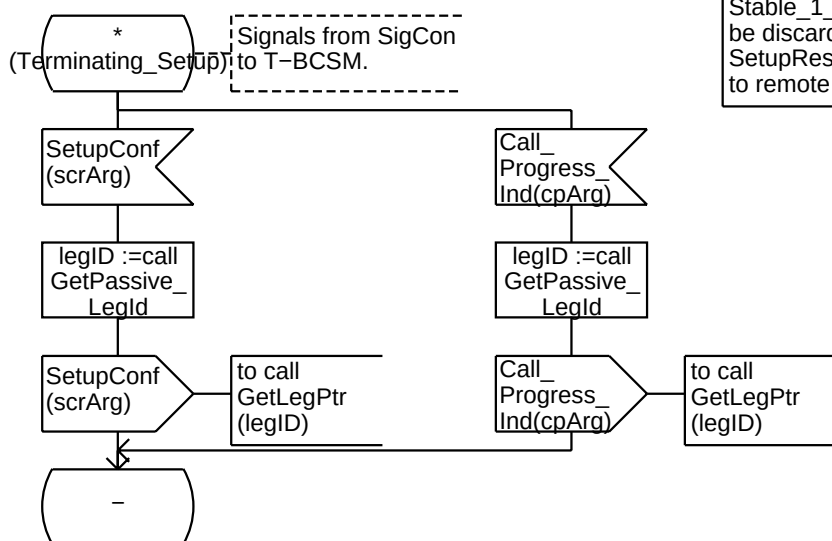
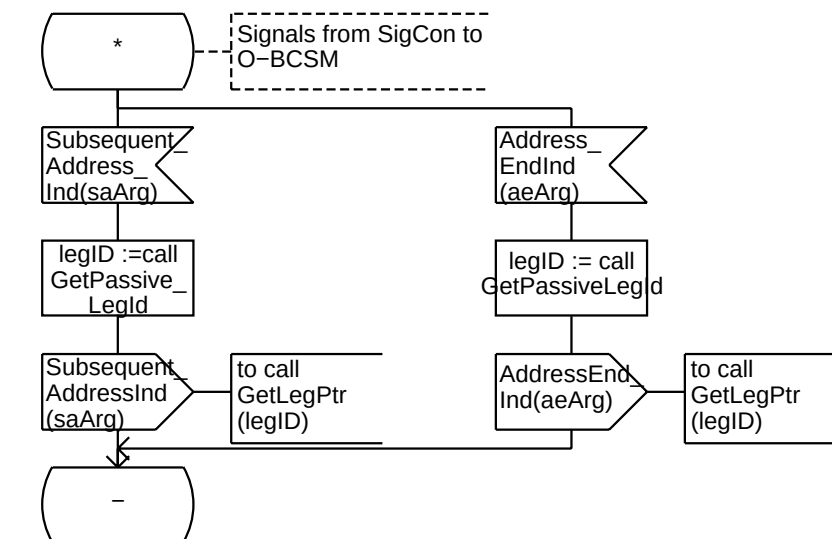
;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/* ROUTING OF SIGNALS TO/FROM SIGCON (VIA THE CSA/IH). */

```



```

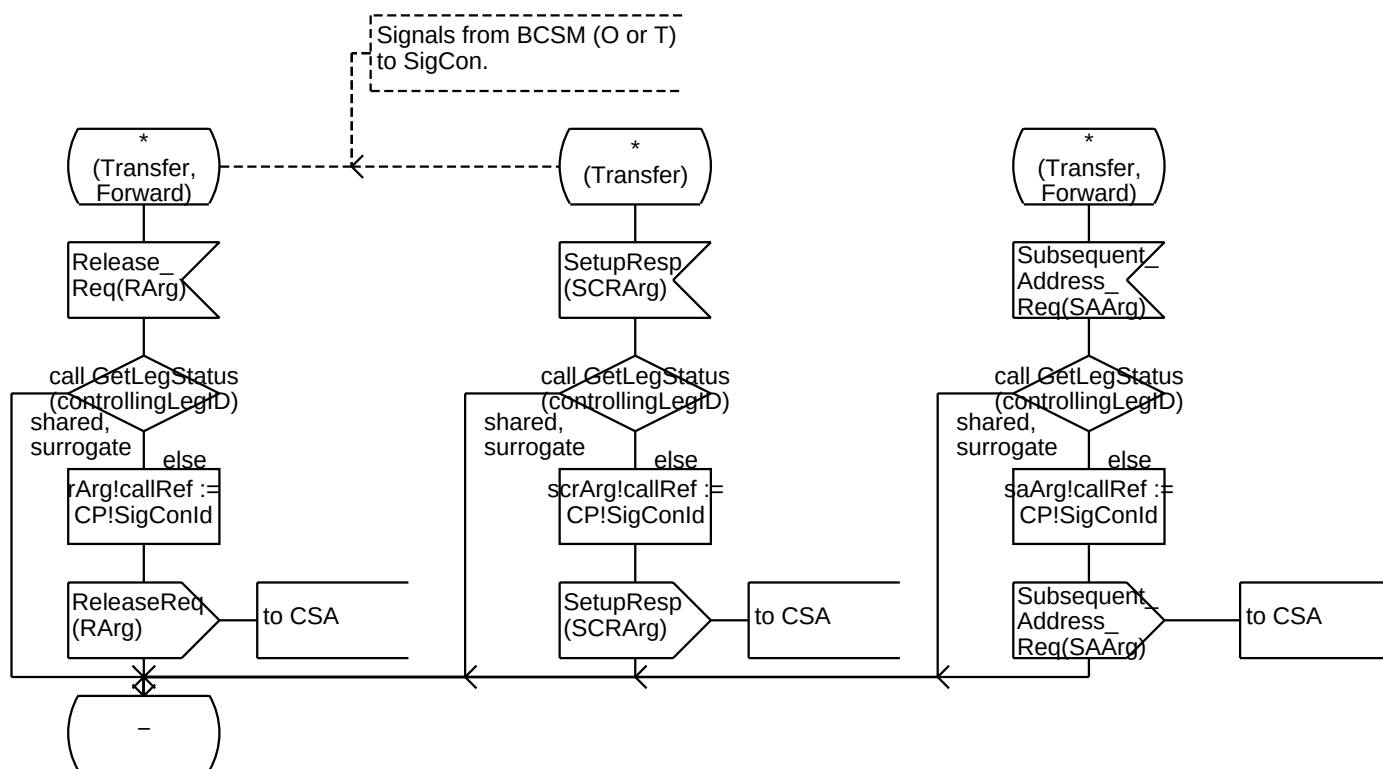
/* Note: SetupConf and CallProgressInd received at T-BCSM when in CS state Stable_1_Party are glare cases. They should be discarded and not forwarded as SetupRespConf and CallProgressReqInd to remote half-call O-BCSM */

```

Process Type CallSegment

66(80)

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */



Process Type CallSegment

67(80)

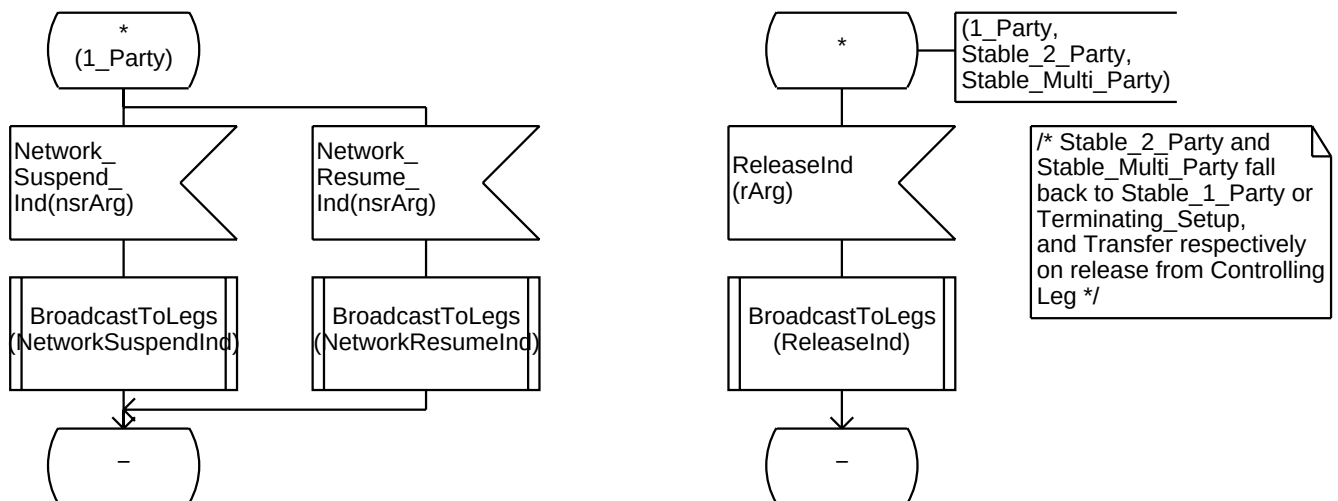
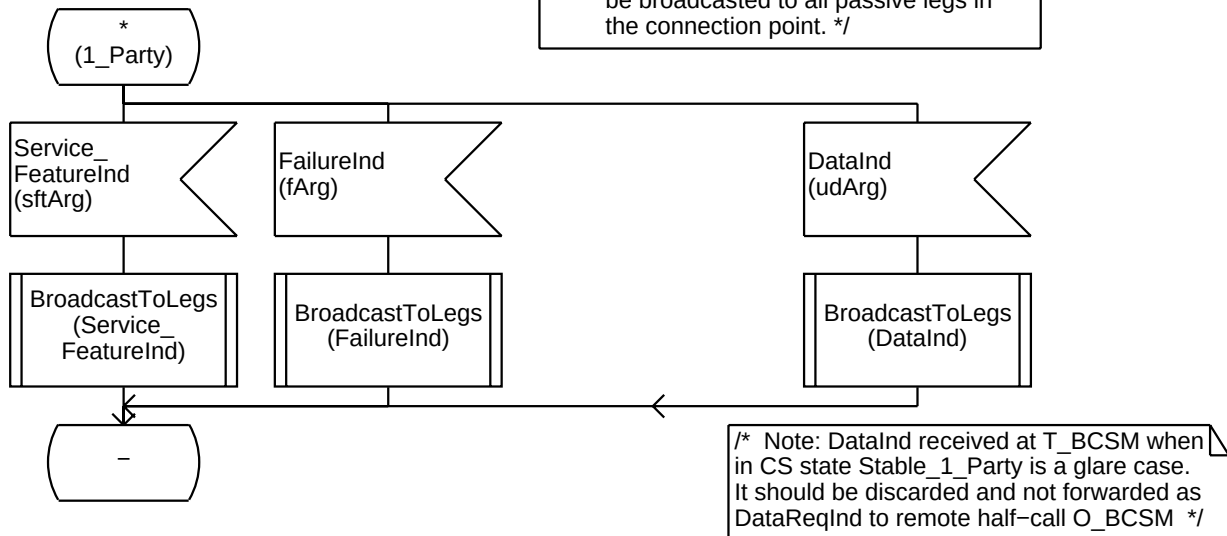
```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

/**** ROUTING OF SIGNALLING CONTROL PRIMITIVES TO/FROM SIGCON. ****/

/* Note: The signalling control primitives from SigCon -> BCSMs must be broadcasted to all passive legs in the connection point. */



Process Type CallSegment

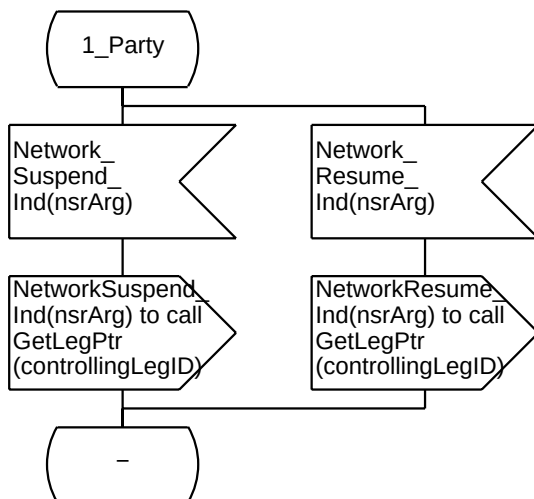
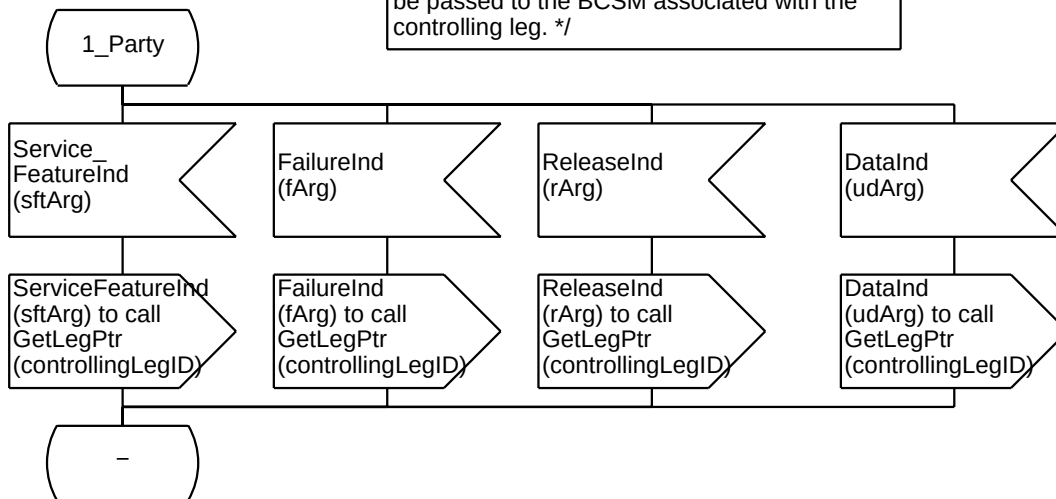
68(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

/* Note: In the 1-Party state, the signalling control primitives from SigCon -> BCSMs must be passed to the BCSM associated with the controlling leg. */



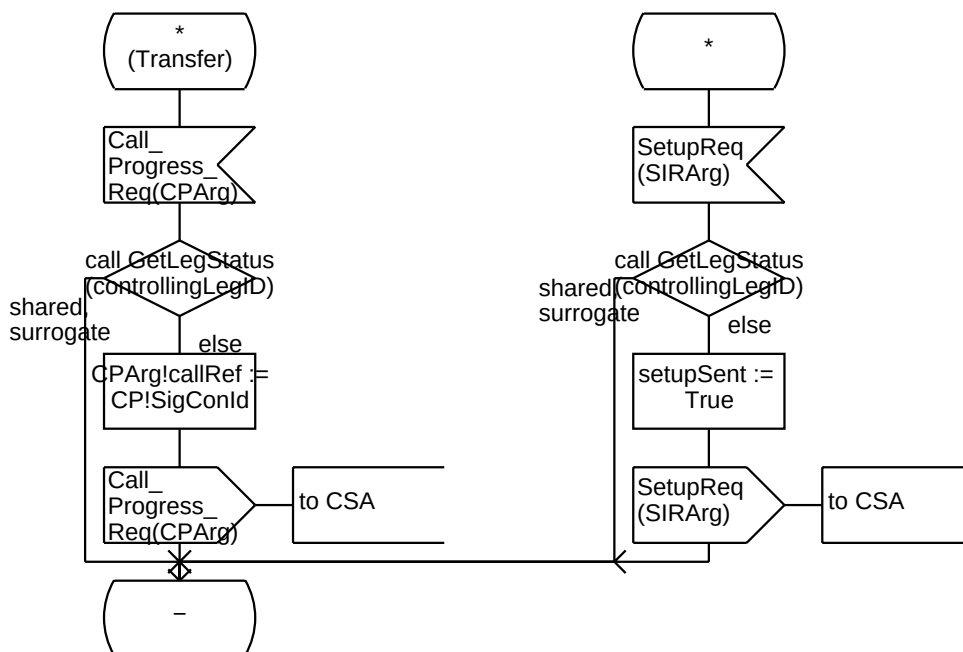
Process Type CallSegment

69(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```



Process Type CallSegment

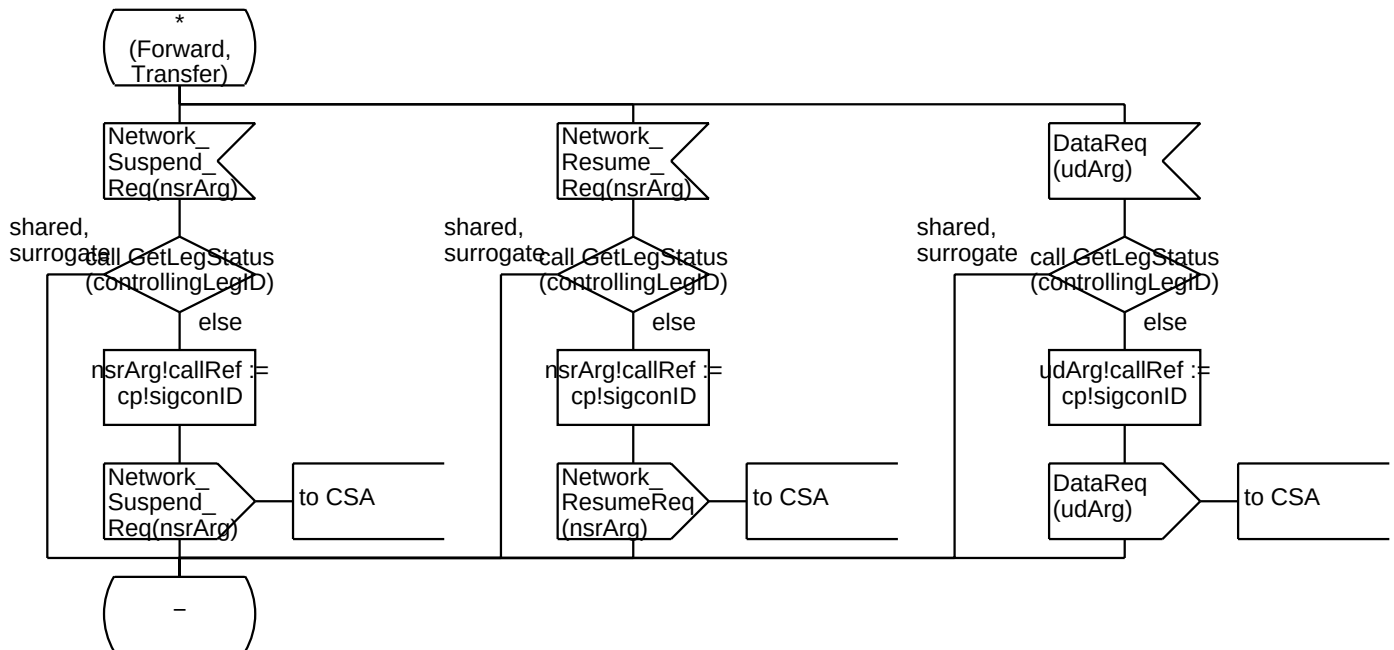
70(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

/* Note: Signalling control primitives from BCSMs -> SigCon are routed via the CSA. */



Process Type CallSegment

71(80)

```

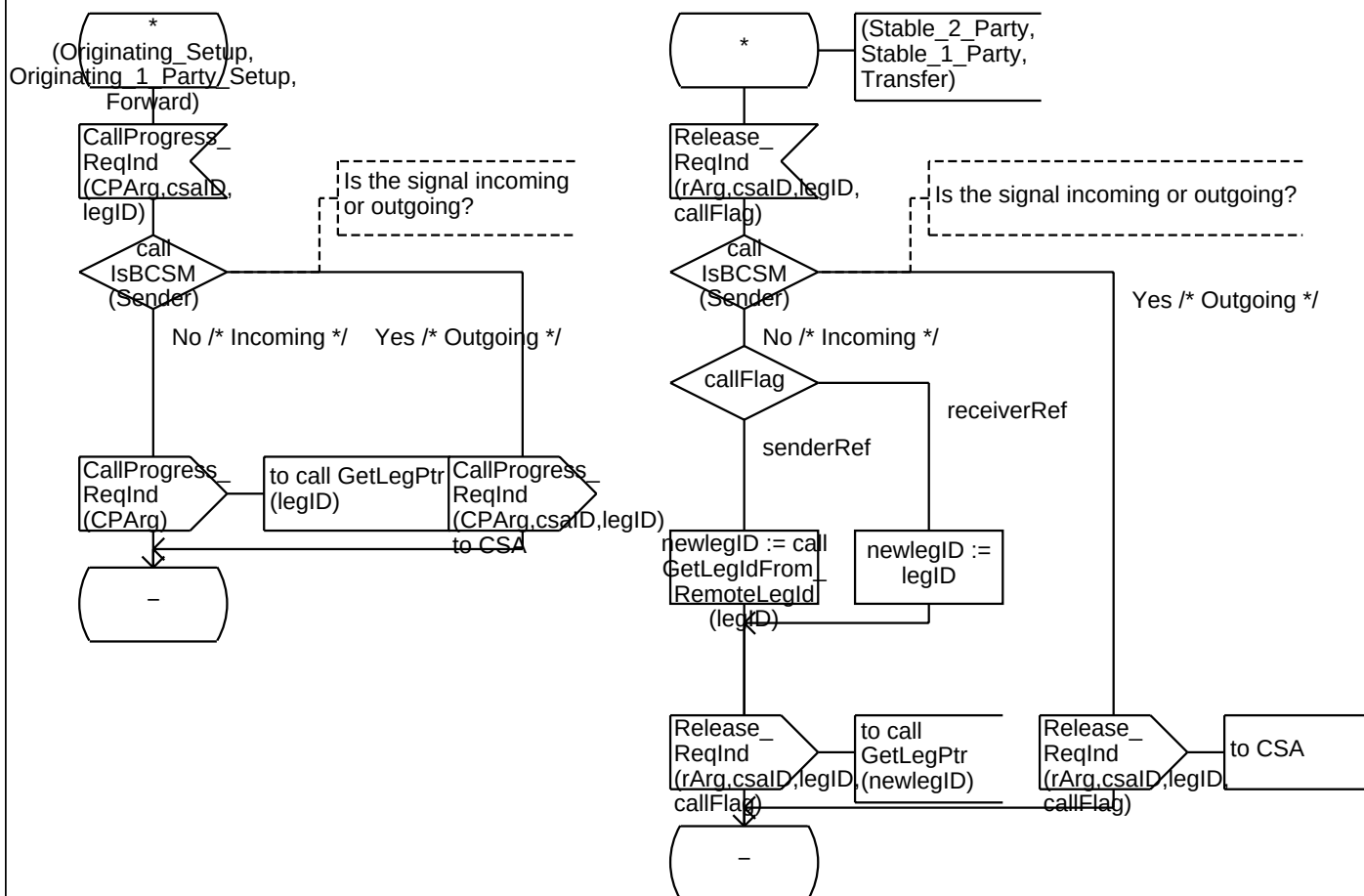
;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

```

/**** ROUTING OF SIGNALS BETWEEN BCSMs ****/

```



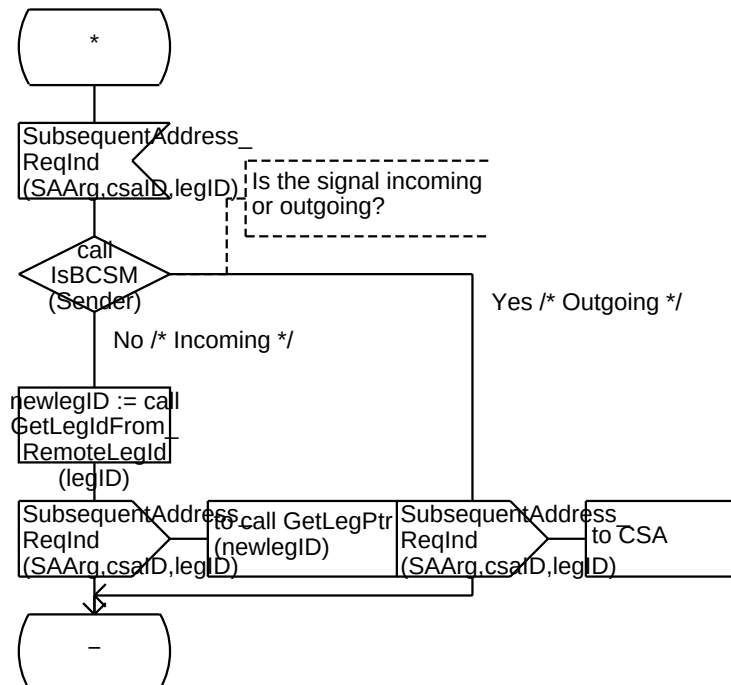
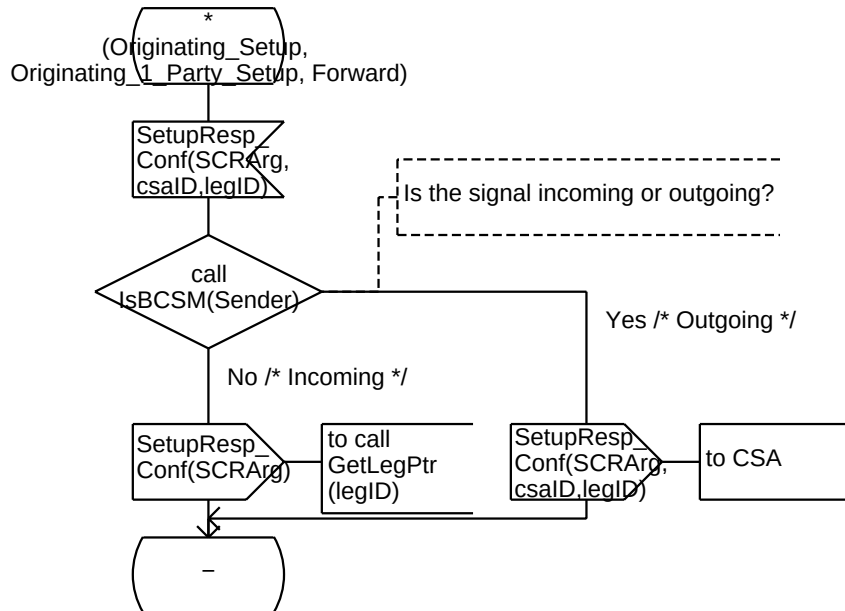
Process Type CallSegment

72(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

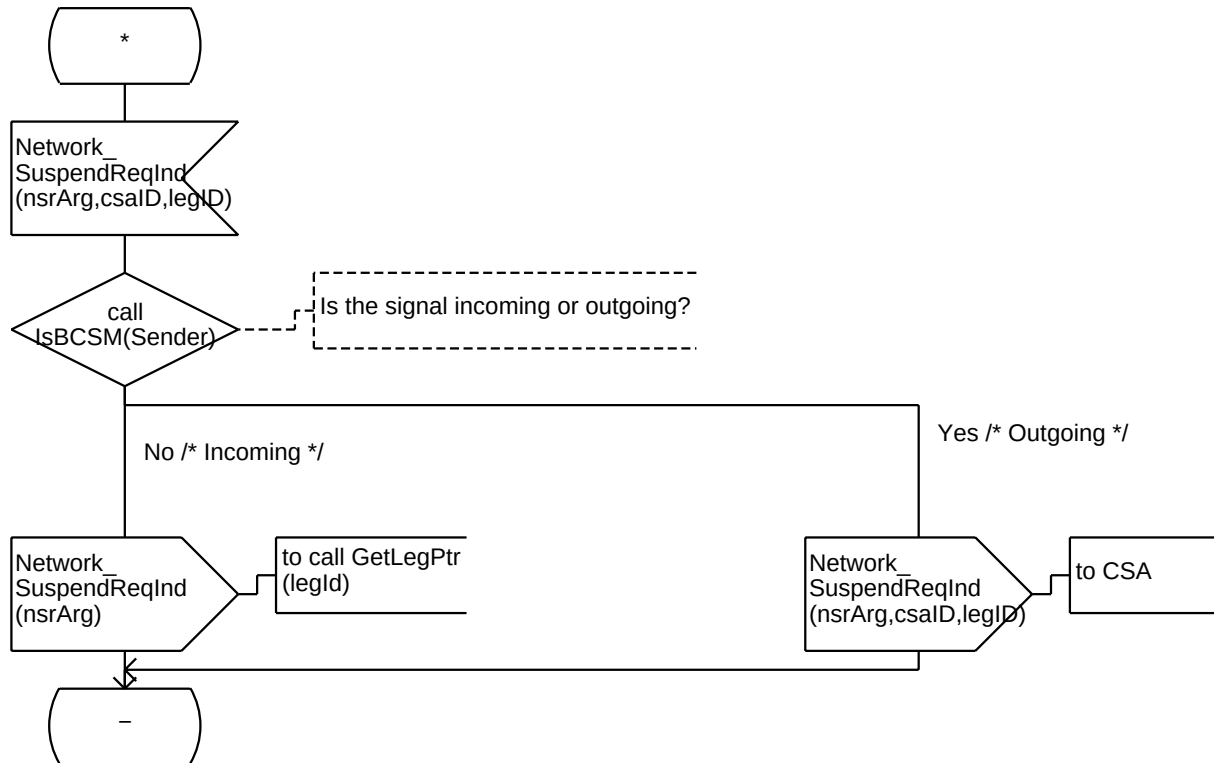


73(80)

```

    FPAR /* Parameters assigned by the CSA at creation of the call segment. */
    callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
    controllingLegID LegType, /* Id of the controlling leg. */
    ssfState SSFStateType, /* State of the SSF-FSM. */
    scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```



Process Type CallSegment

74(80)

;

FPAR

/* Parameters assigned by the CSA at creation of the call segment. */

callSegmentID

CallSegmentID,

/* Id of this call segment as assigned by the CSA. */

controllingLegID

LegType,

/* Id of the controlling leg. */

ssfState

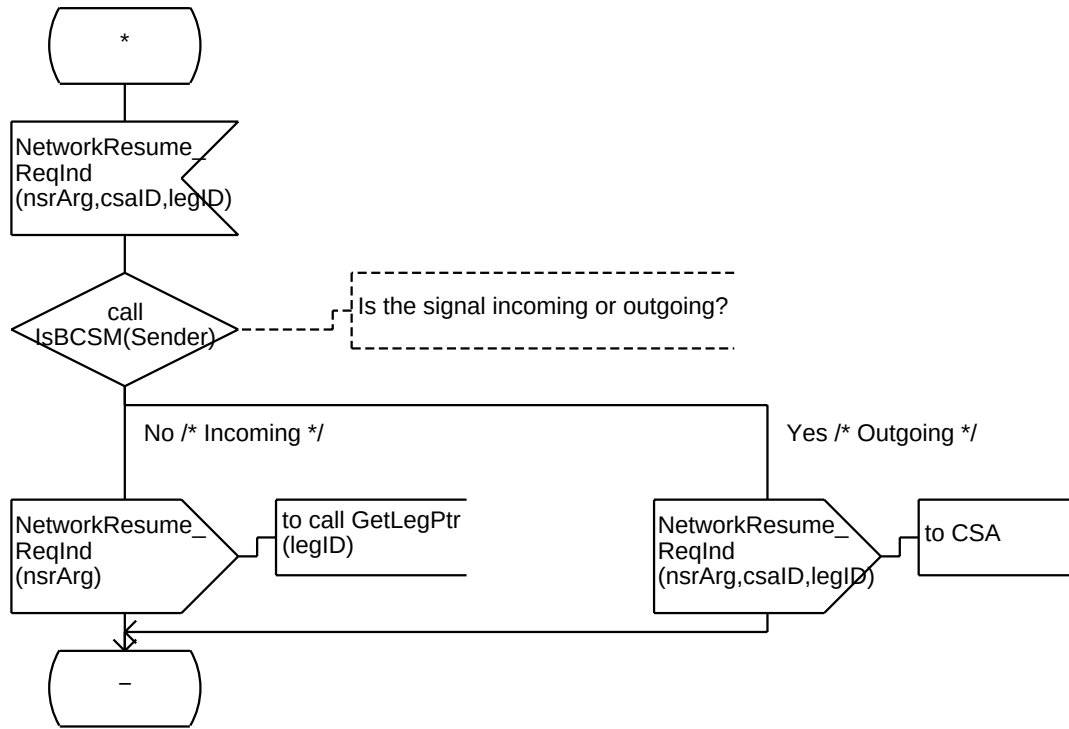
SSFStateType,

/* Start state of the SSF-FSM. */

scfInitiated

Boolean;

/* Is the call initiated by the SCF or not? */



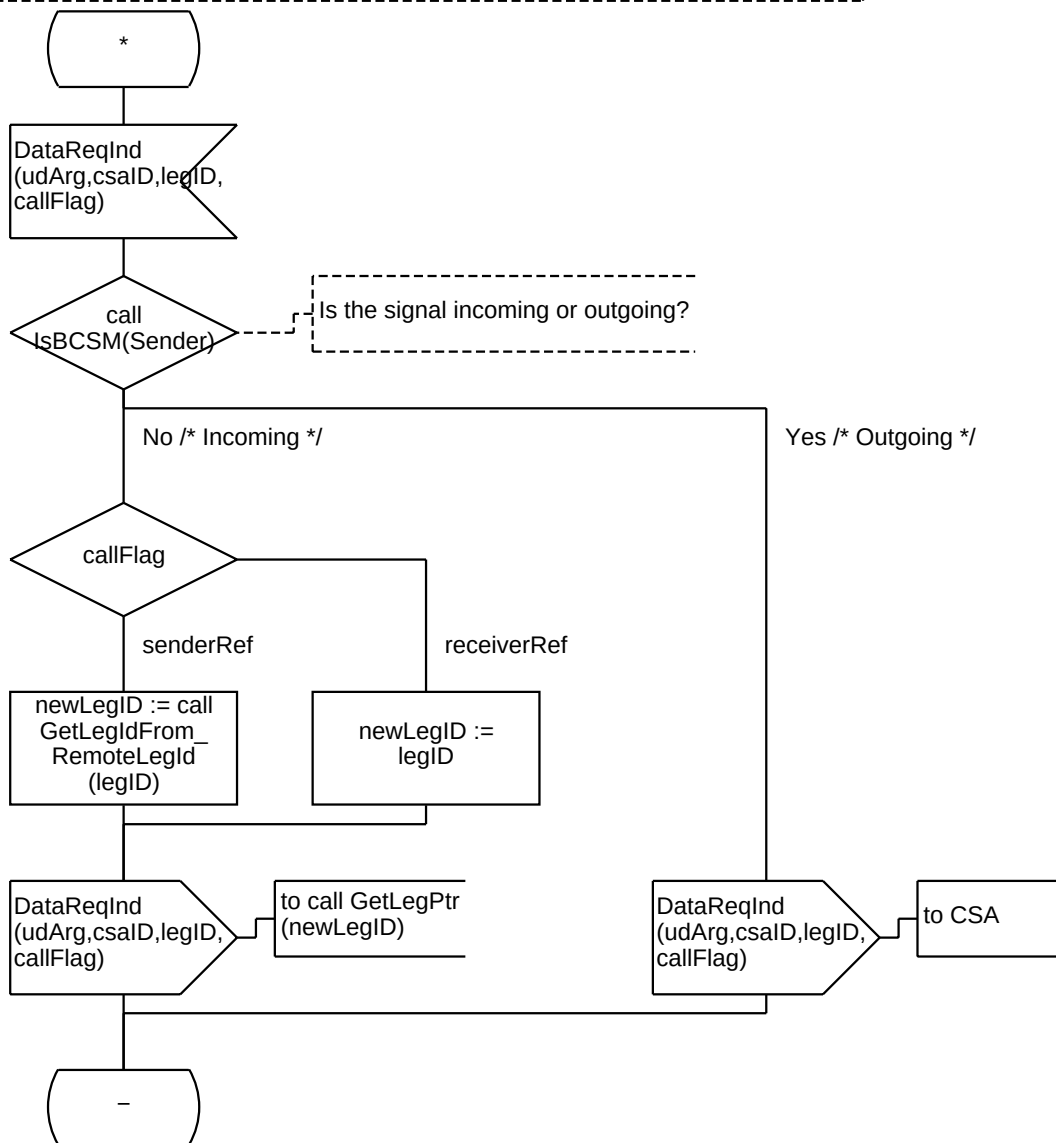
Process Type CallSegment

75(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```



Process Type CallSegment

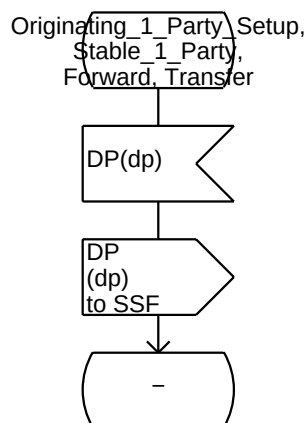
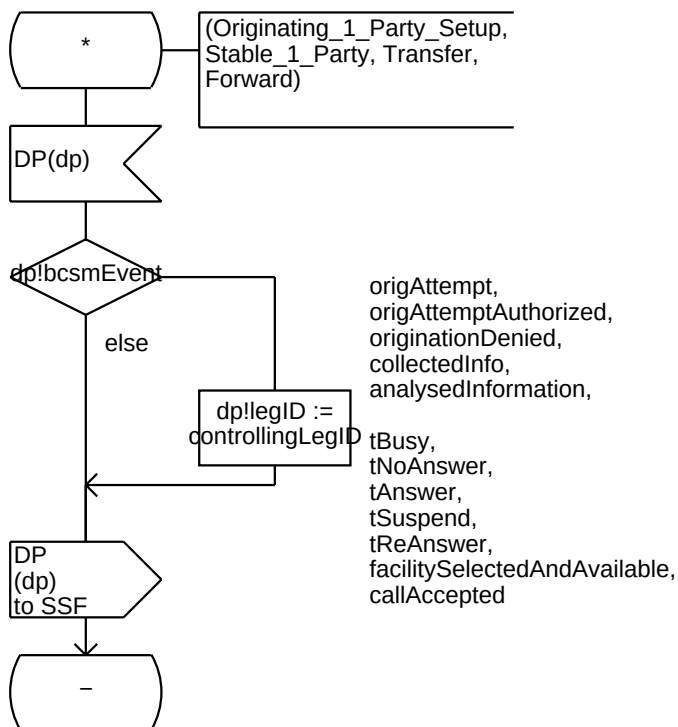
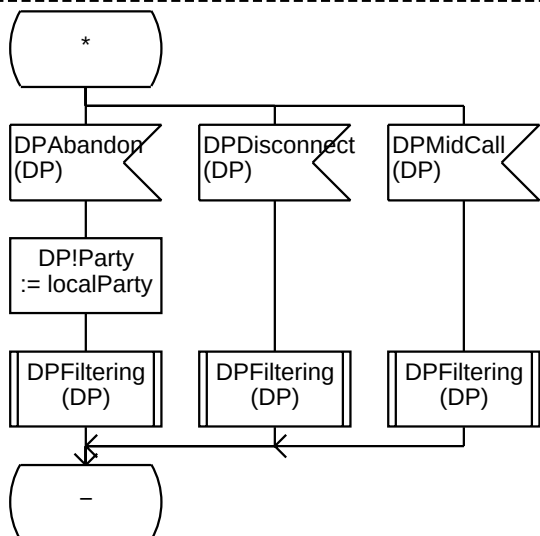
76(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

/* Routing of detection points (DPs) to the SSF. */



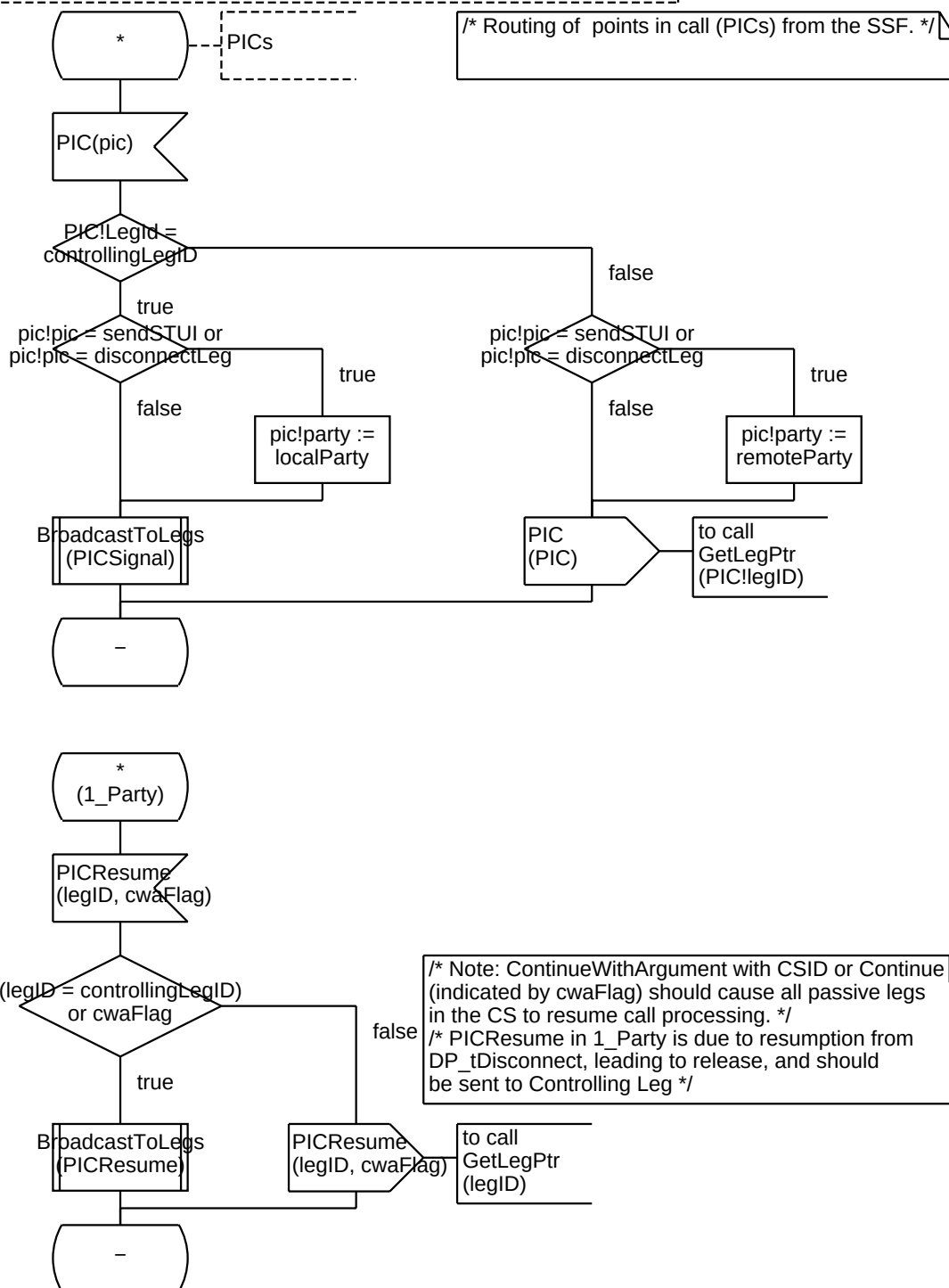
/* Note:
Events occurring when the passive leg is pending (in the case of originating setup), or when the controlling leg is pending (in case of terminating setup) should be reported for the controlling leg.

Note: This does not apply when the controlling leg is surrogate, in Originating_1_Party_Setup, Stable_1_Party, Forward or Transfer. */

Process Type CallSegment

77(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */



Process Type CallSegment

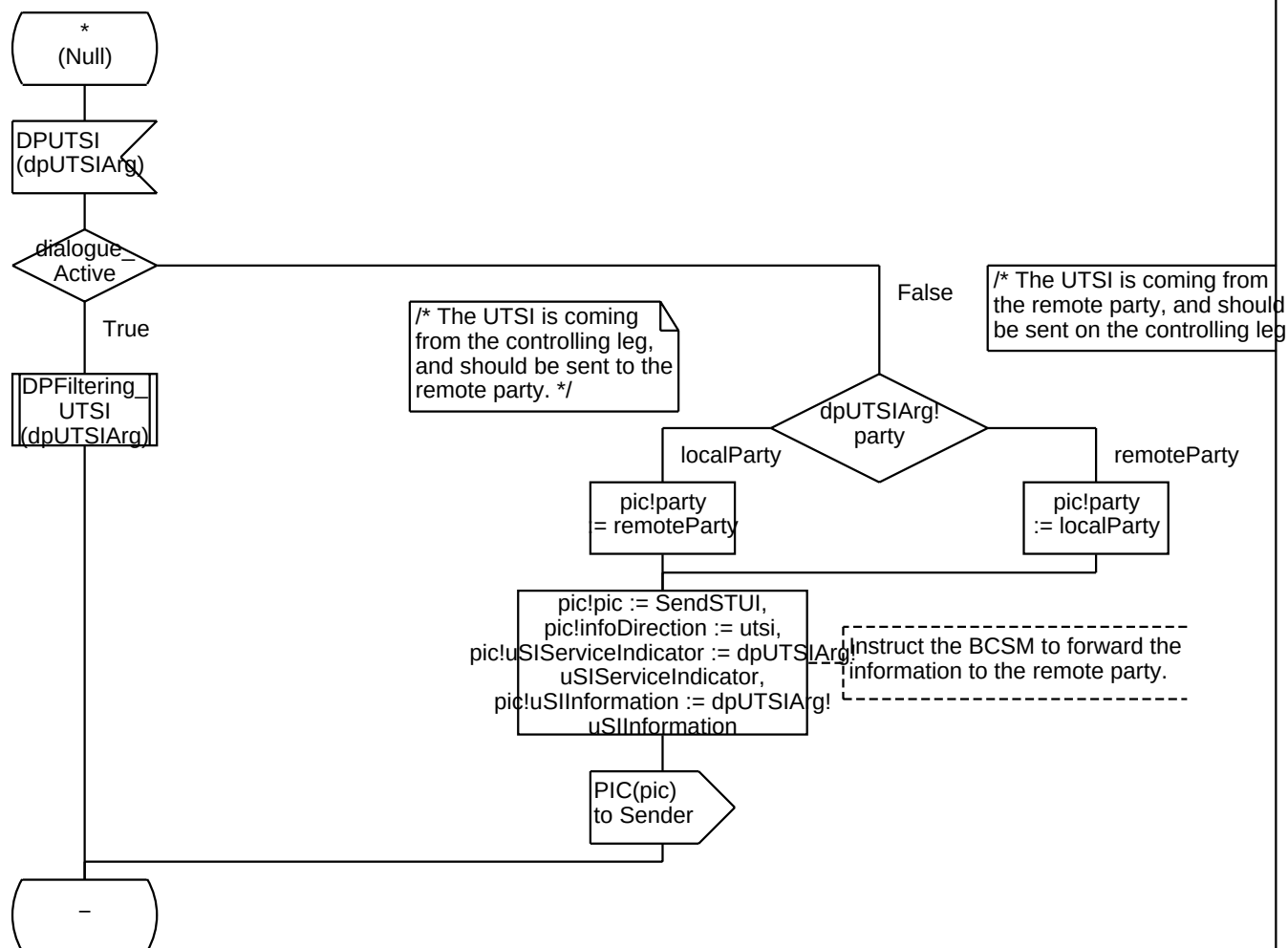
78(80)

```

;FPAR /* Parameters assigned by the CSA at creation of the call segment. */
callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
controllingLegID LegType, /* Id of the controlling leg. */
ssfState SSFStateType, /* Start state of the SSF-FSM. */
scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

```

/* In all states except Null, DPUTSI is only reported to the SSF-FSM if there is a dialogue active, otherwise the DPUTSI will be transformed into a SendSTUI PIC and the information will be send (by the BCSM) to the remote party. */



Process Type

CallSegment

79(80)

;

F

P

A

R

/*

Parameters

assigned

by

the

CSA

at

creation

of

the

call

segment.

*/

callSegmentID

CallSegmentID,

/*

Id

of

this

call

segment

as

assigned

by

the

CSA.

*/

controllingLegID

LegType,

/*

Id

of

the

controlling

leg.

*/

ssfState

SSFStateType,

/*

Start

state

of

the

SSF-FSM.

*/

scfInitiated

Boolean;

/*

Is

the

call

initiated

by

the

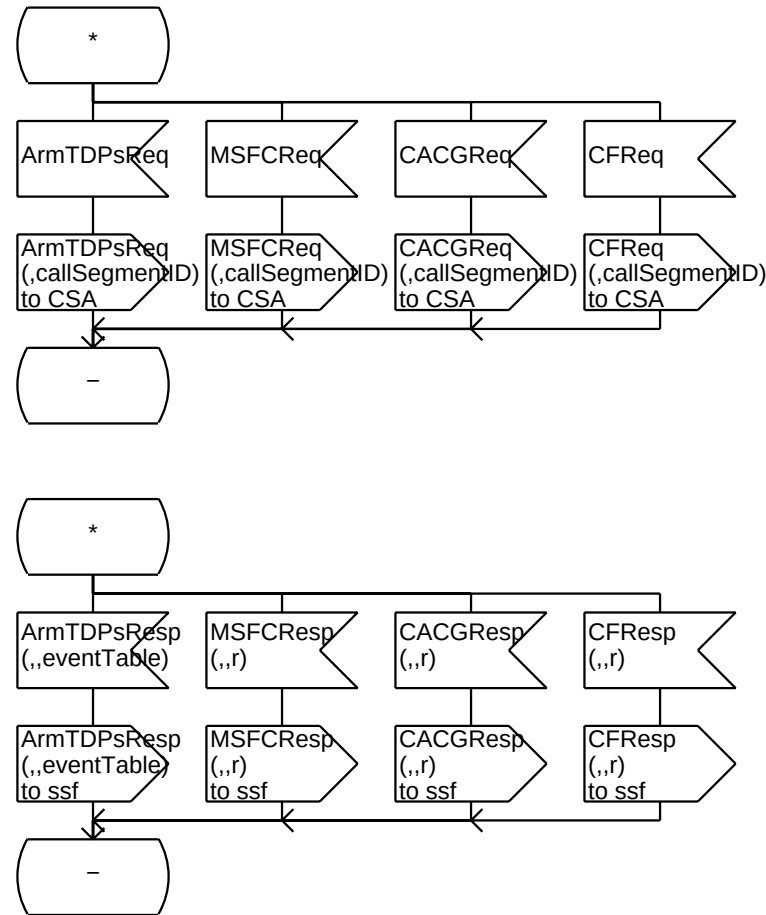
SCF

or

not?

*/

/* ROUTING OF SIGNALS BETWEEN THE SSF-FSM AND THE SSME-FSM */



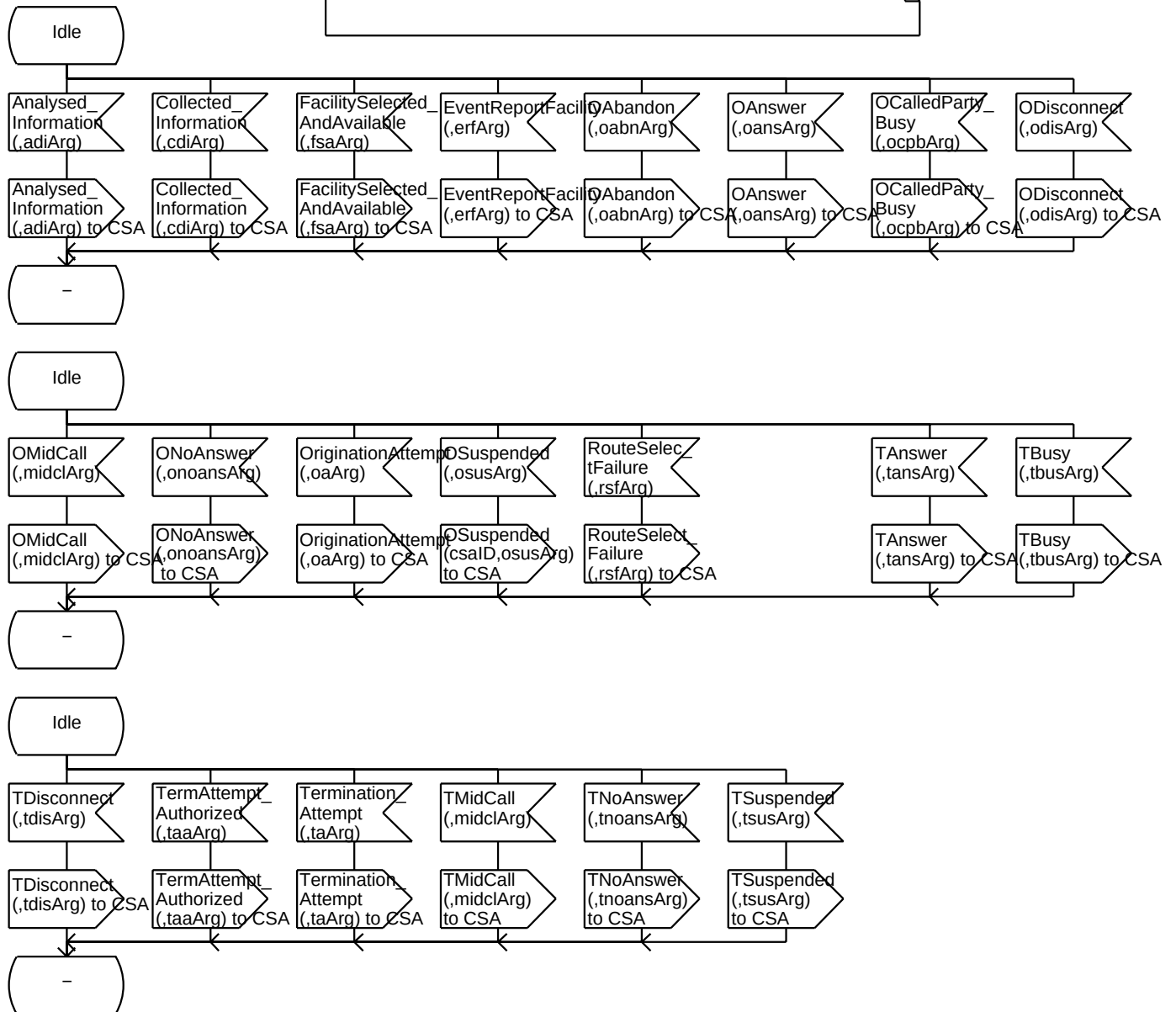
Annex A: CallSegment

Process Type CallSegment

80(80)

FPAR /* Parameters assigned by the CSA at creation of the call segment. */
 callSegmentID CallSegmentID, /* Id of this call segment as assigned by the CSA. */
 controllingLegID LegType, /* Id of the controlling leg. */
 ssfState SSFStateType, /* Start state of the SSF-FSM. */
 scfInitiated Boolean; /* Is the call initiated by the SCF or not? */

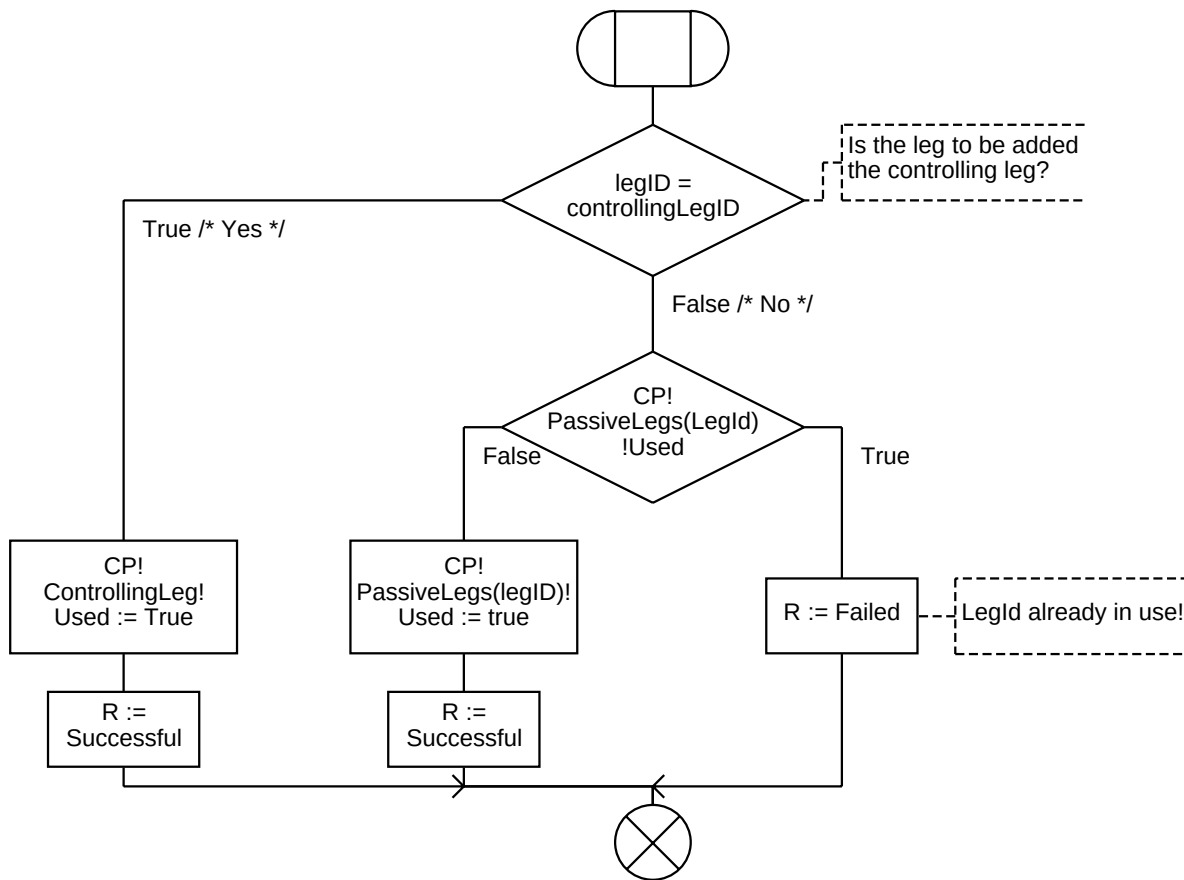
/* ROUTING OF DP SPECIFIC OPERATIONS FROM SSF TO SCF. */



Procedure AddLeg

1(1)

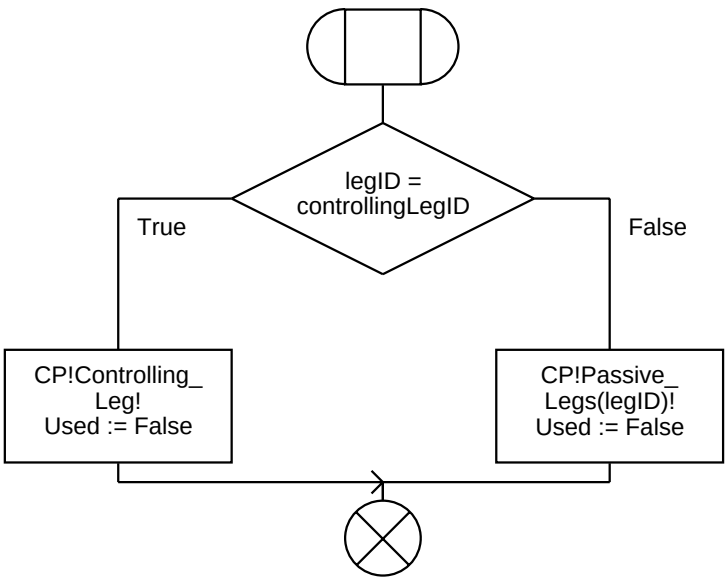
```
;FPAR
IN legID LegType;
RETURNS R ResultType;
```



Procedure <<Process Type CallSegment>> RemoveLeg

1(1)

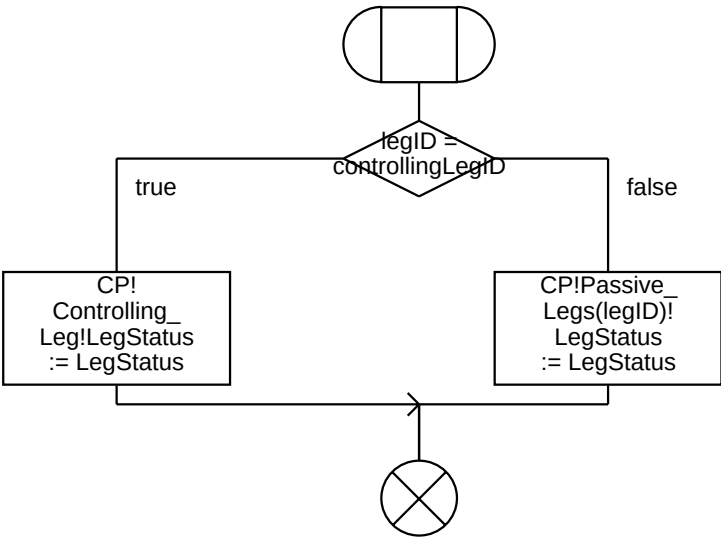
FPAR
IN legID LegType;



Procedure SetLegStatus

1(1)

;FPAR
IN legID LegType,
IN legStatus LegStatusType;

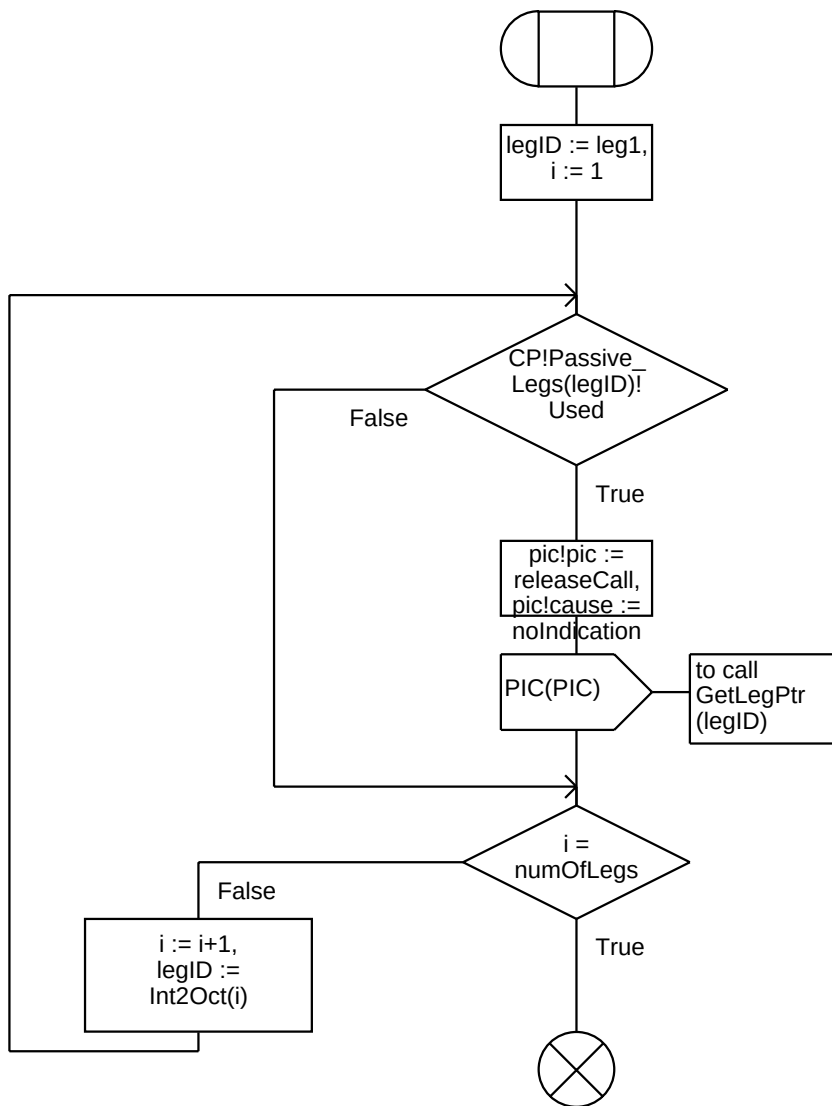


Procedure ReleaseAllLegs

1(1)



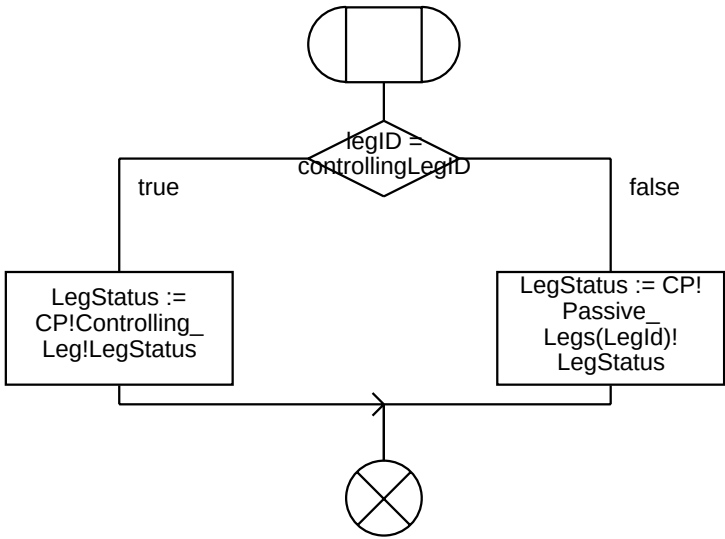
DCL
legID LegType,
i INTEGER;



Procedure GetLegStatus

1(1)

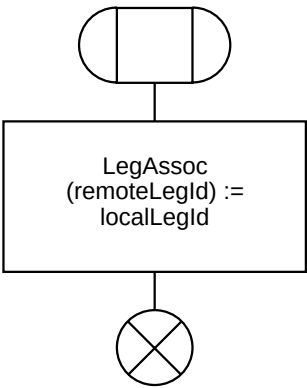
;FPAR
IN legID LegType;
RETURNS LegStatus LegStatusType;



Procedure <<Process Type CallSegment>> SetLegAssoc

1(1)

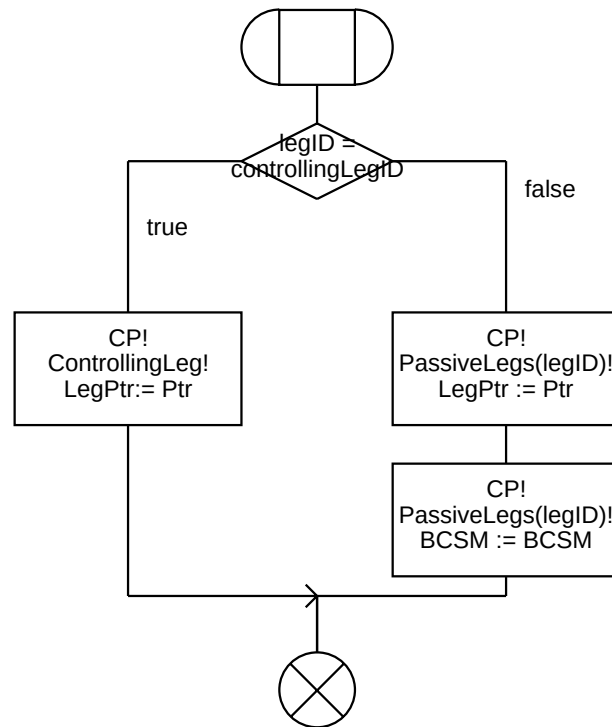
;FPAR
IN remoteLegId LegType;
IN localLegId LegType;



Procedure SetLegPtr

1(1)

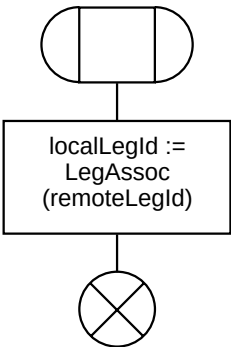
;FPAR
IN legID LegType,
IN Ptr Pid,
IN BCSM BCSMType;



Procedure <<Process Type CallSegment>> GetLegIdfromRemoteLegId

1(1)

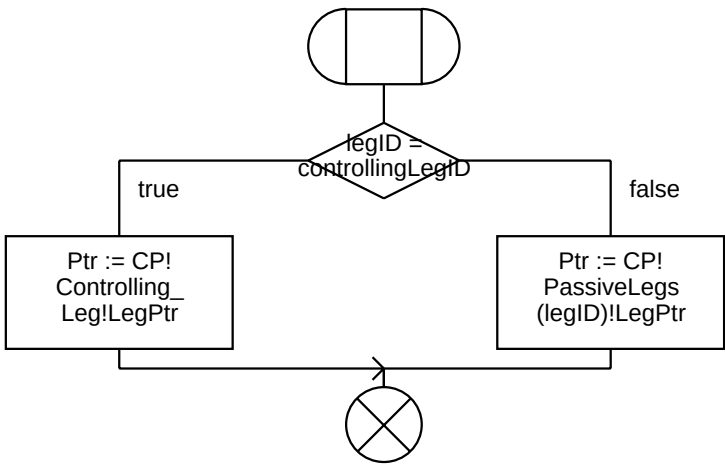
;FPAR
IN remoteLegId LegType;
RETURNS localLegId LegType;



Procedure GetLegPtr

1(1)

;FPAR
IN legID LegType;
RETURNS Ptr PId;

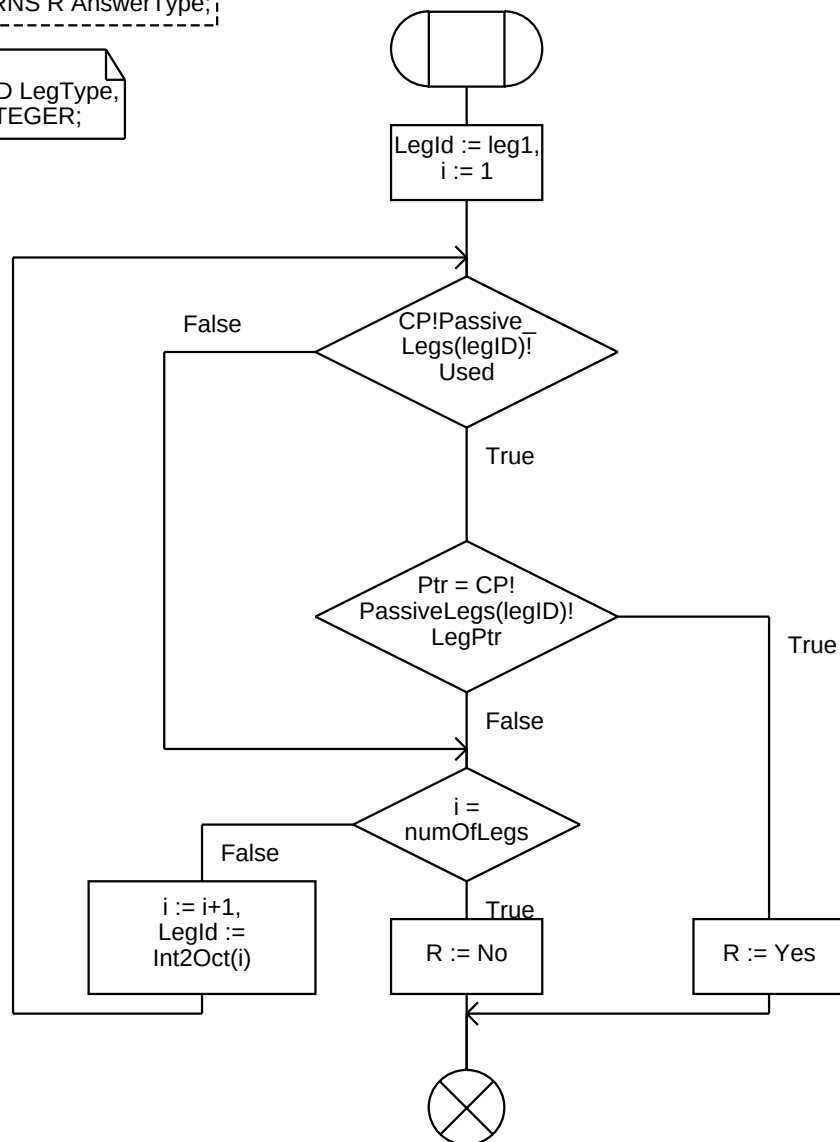


Procedure IsBCSM

1(1)

FPAR
IN Ptr Pid;
RETURNS R AnswerType;

DCL
legID LegType,
i INTEGER;

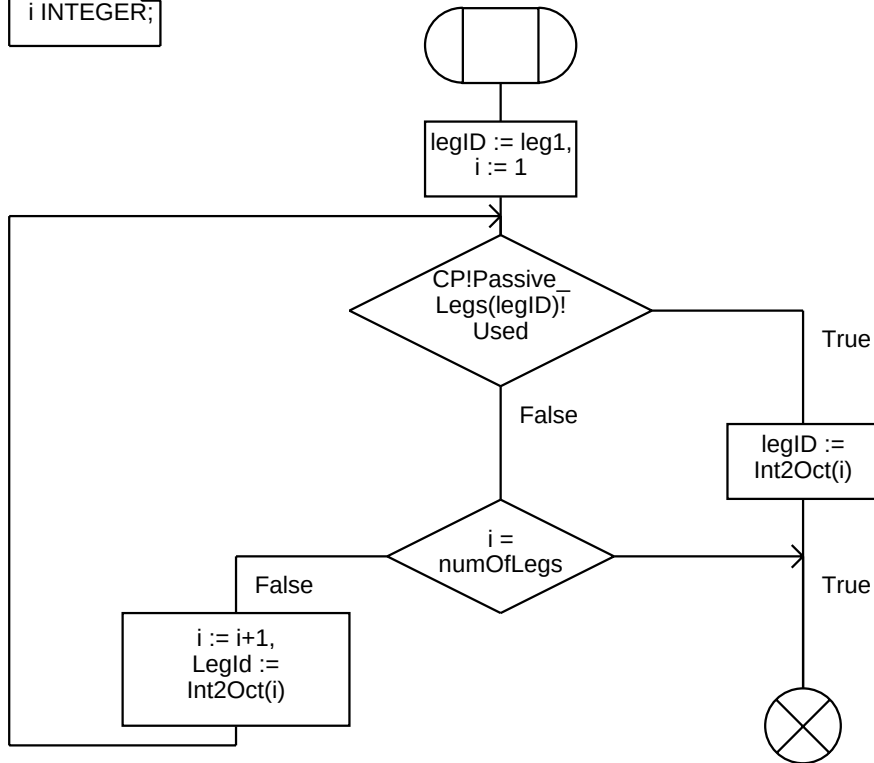


Procedure GetPassiveLegId

1(1)

;RETURNS legID LegType;

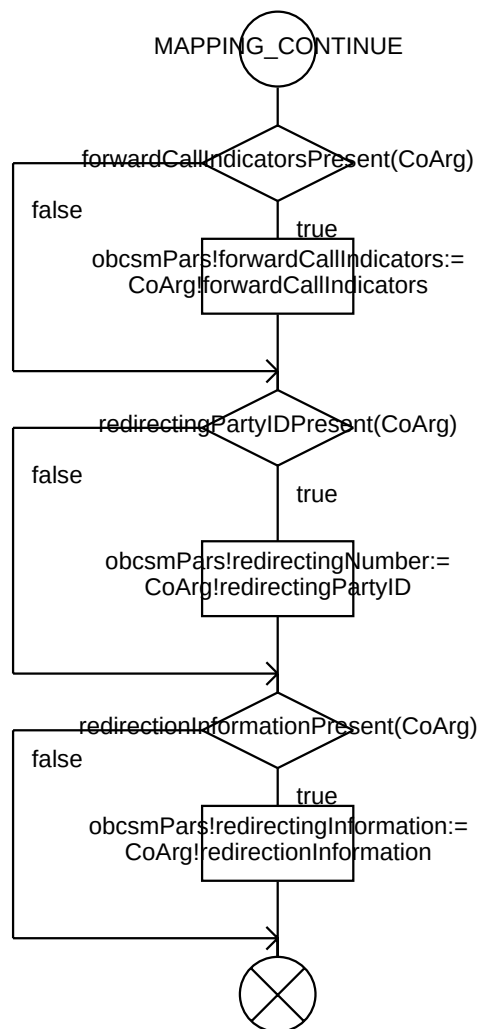
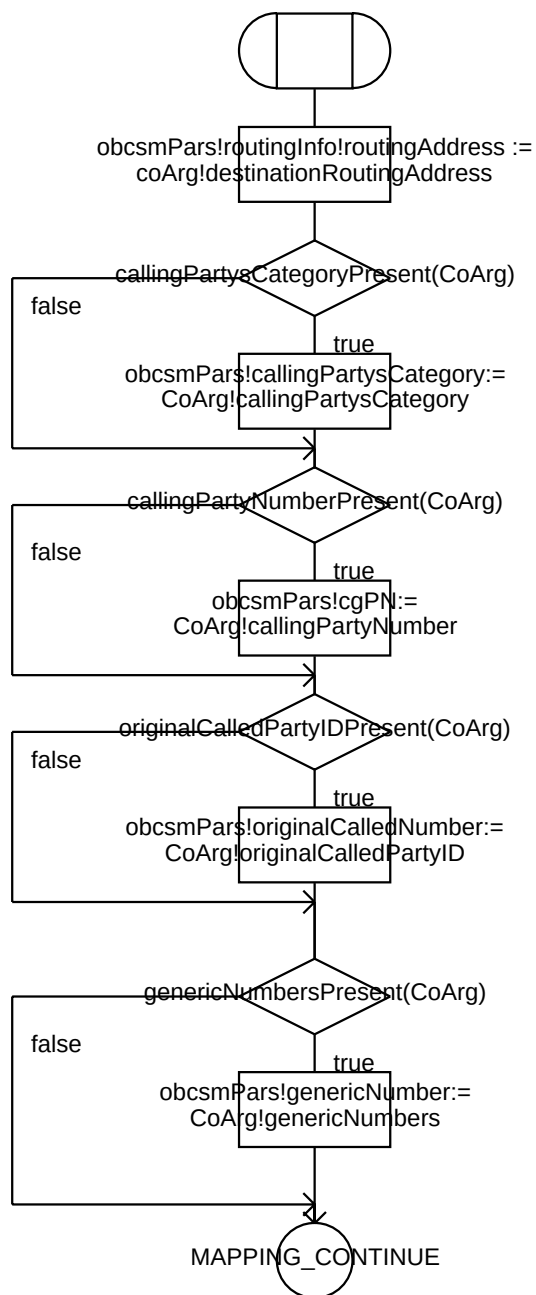
DCL
i INTEGER;



Procedure MapConnectToBCSM

1(1)

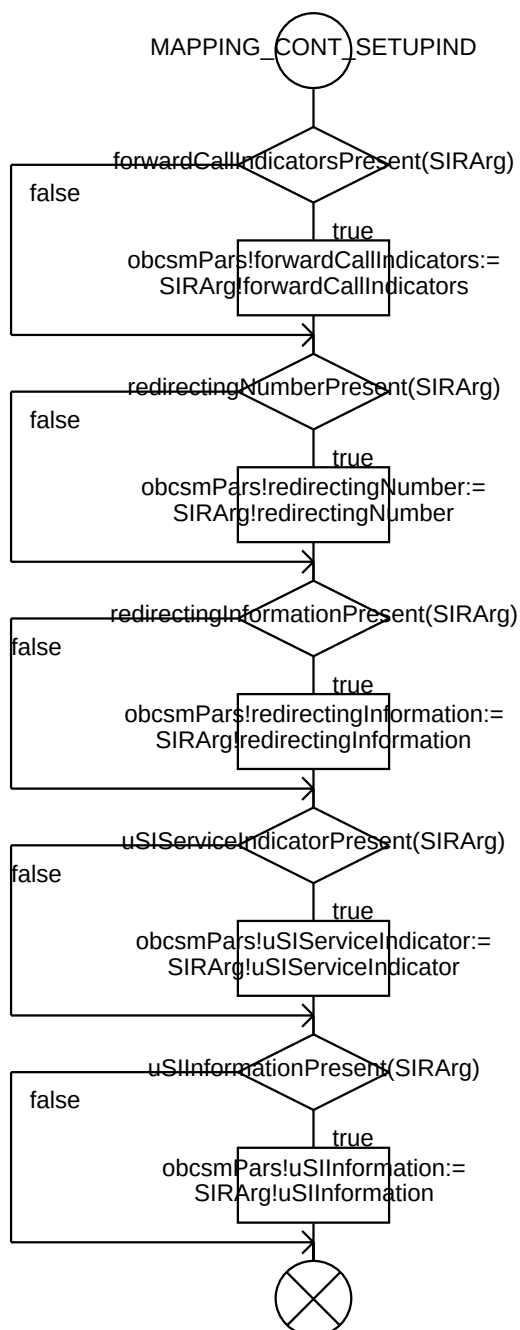
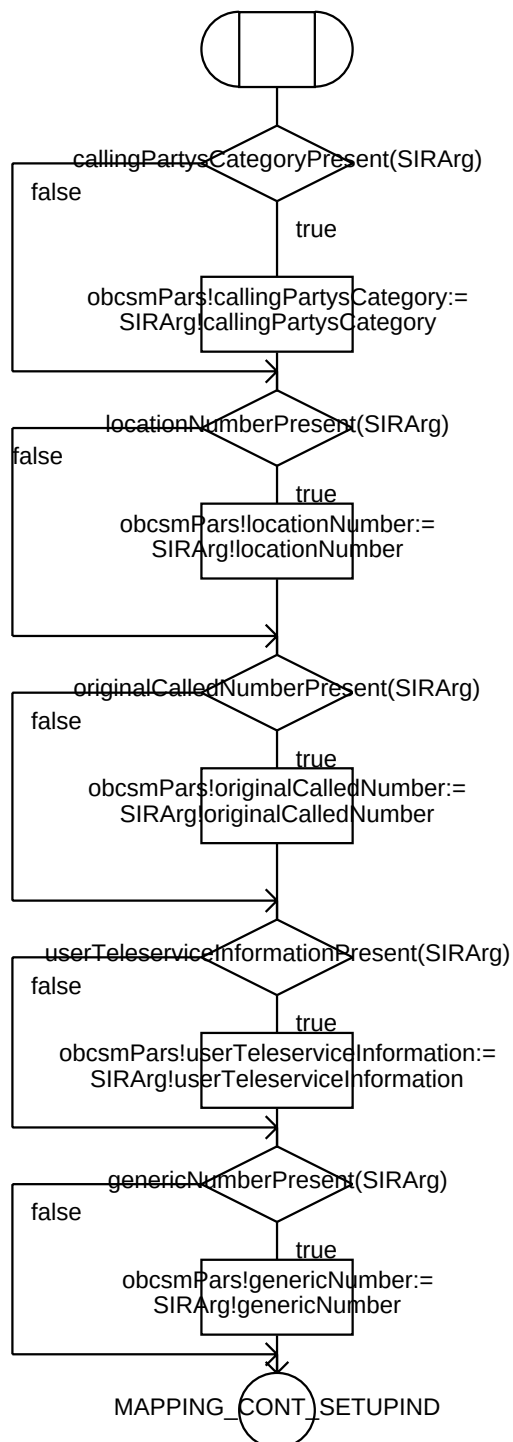
FPAR
IN coArg ConnectArg;
RETURNS obcsmPars OBCSMPars;



Procedure MapSIToBCSM

1(1)

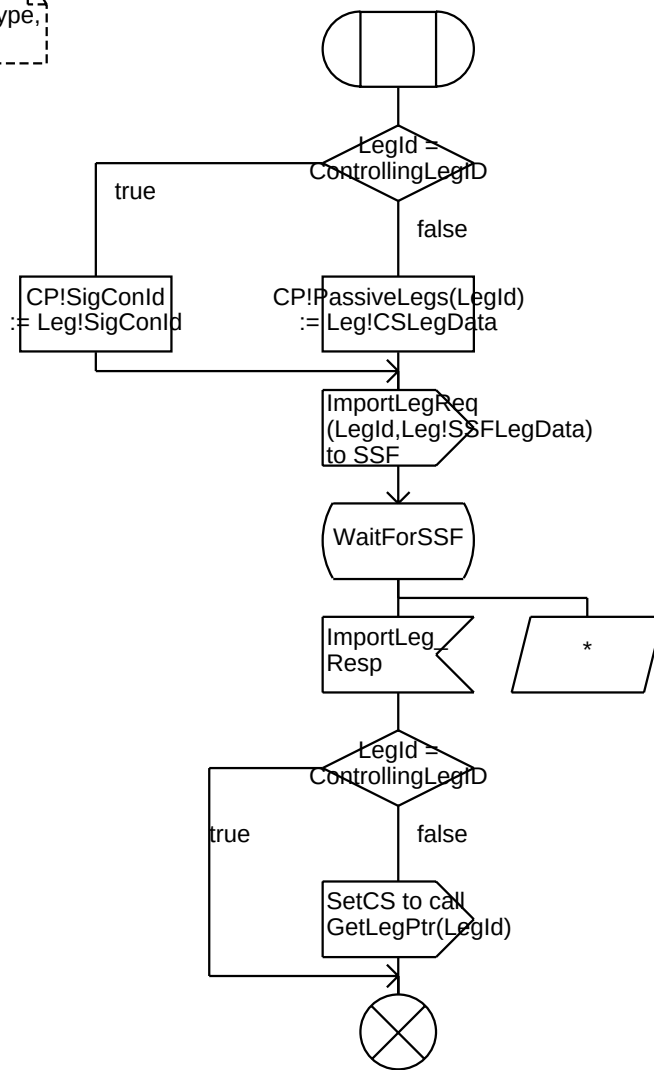
;FPAR
IN sirArg SetupIRType;
RETURNS obcsmPars OBCSMPars;



Procedure <<Process Type CallSegment>> ImportLeg

1(1)

;FPAR
IN Leg ExportLegType,
IN LegId LegType;

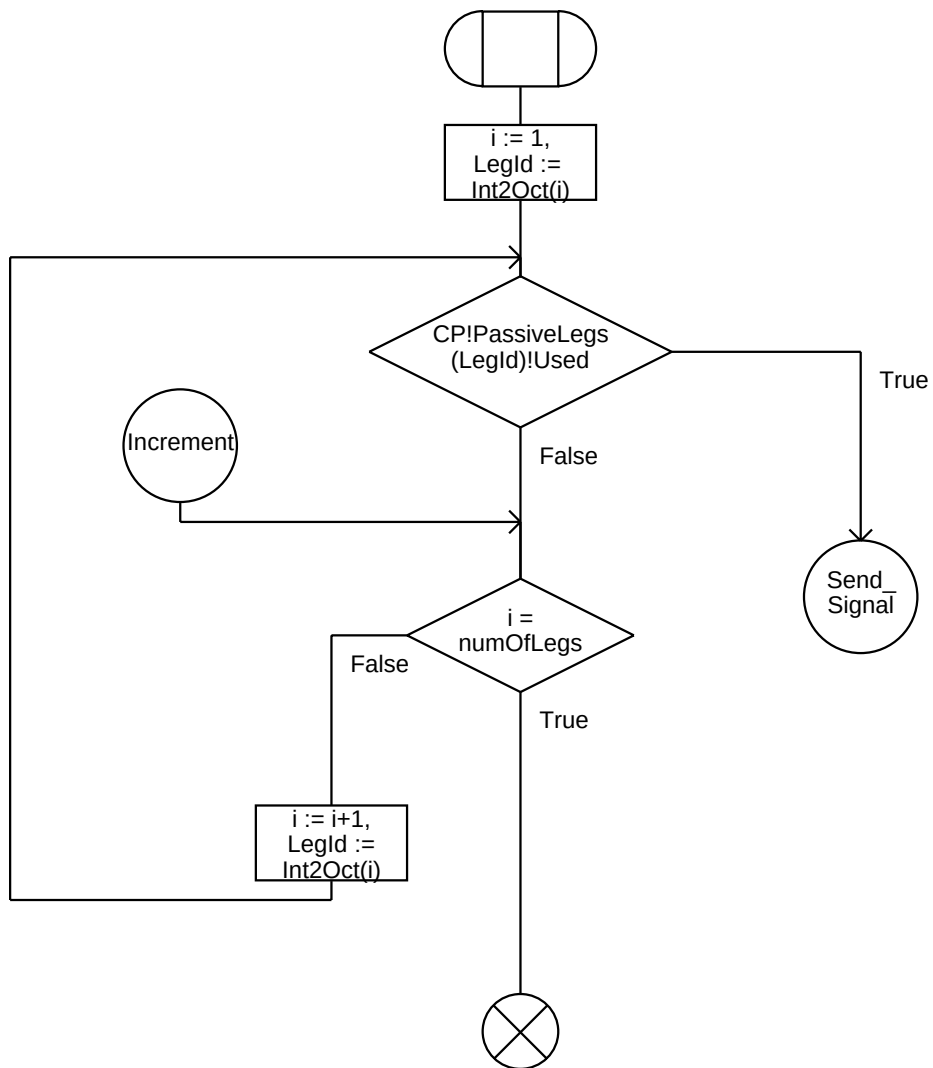


Procedure BroadcastToLegs

1(2)

FPAR
 IN SignalToBroadcast BroadcastSignalNames;

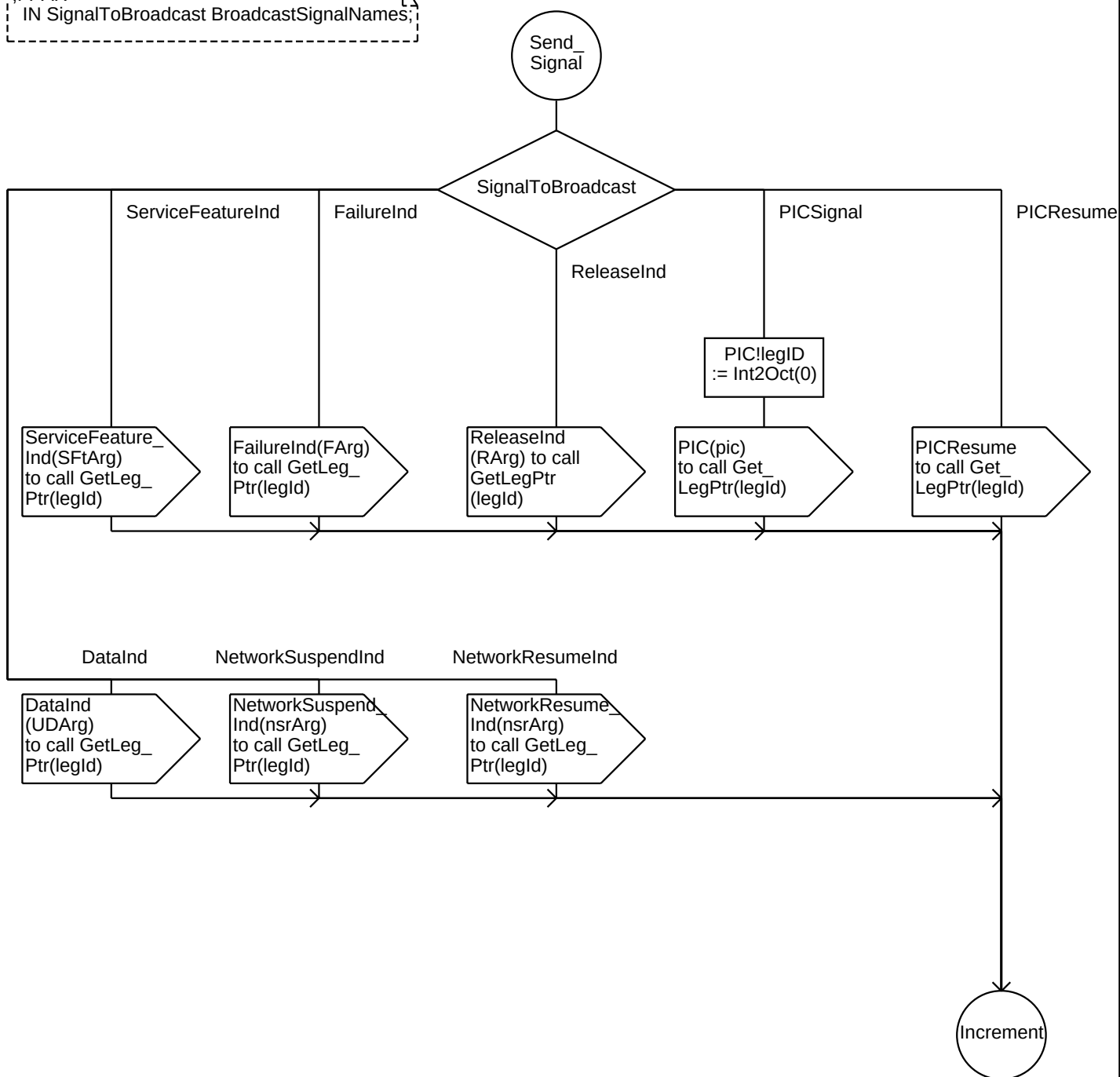
DCL
 LegId LegType,
 i Integer;



Procedure BroadcastToLegs

2(2)

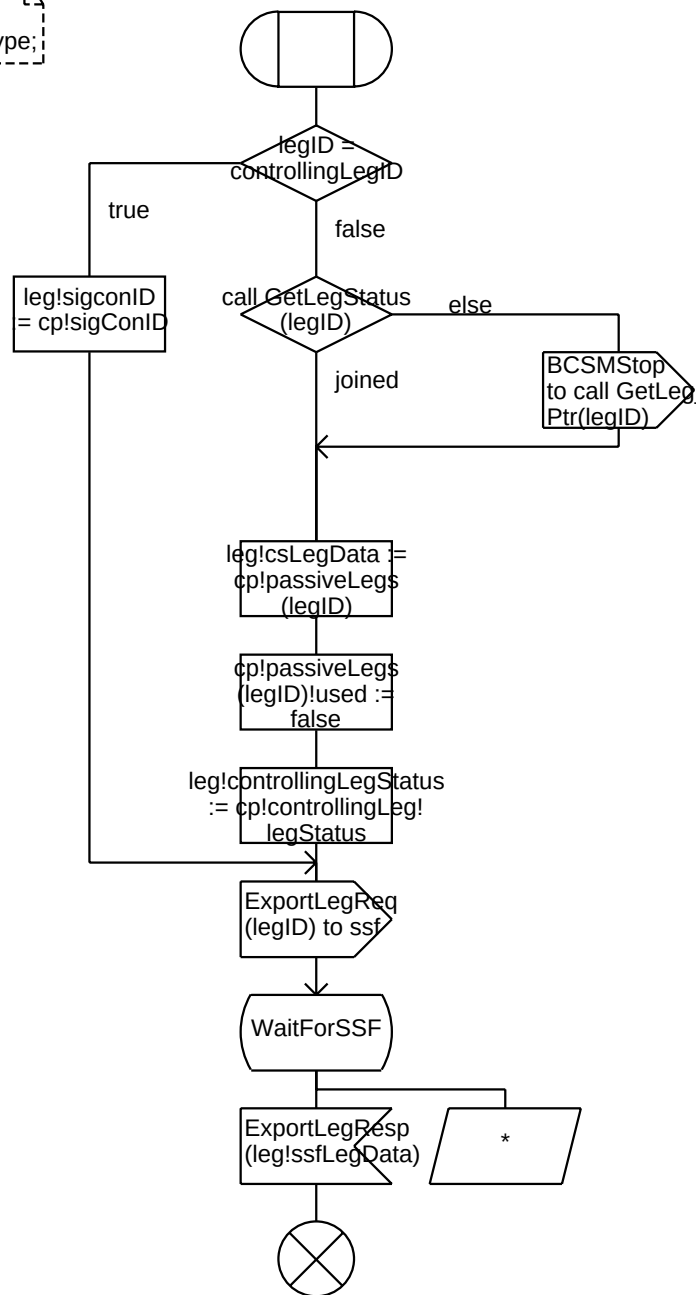
FPAR
IN SignalToBroadcast BroadcastSignalNames;



Procedure <<Process Type CallSegment>> ExportLeg

1(1)

FPAR
LegID LegType;
RETURNS Leg ExportLegType;

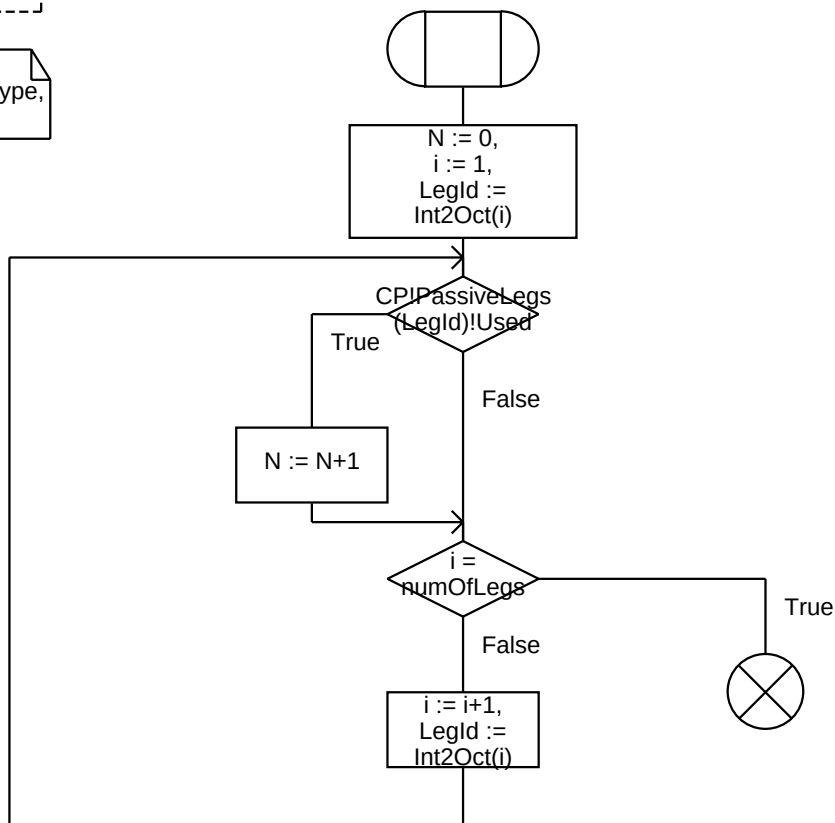


Procedure NoOfLegs

1(1)

RETURNS N Integer;

DCL
LegId LegType,
i Integer;

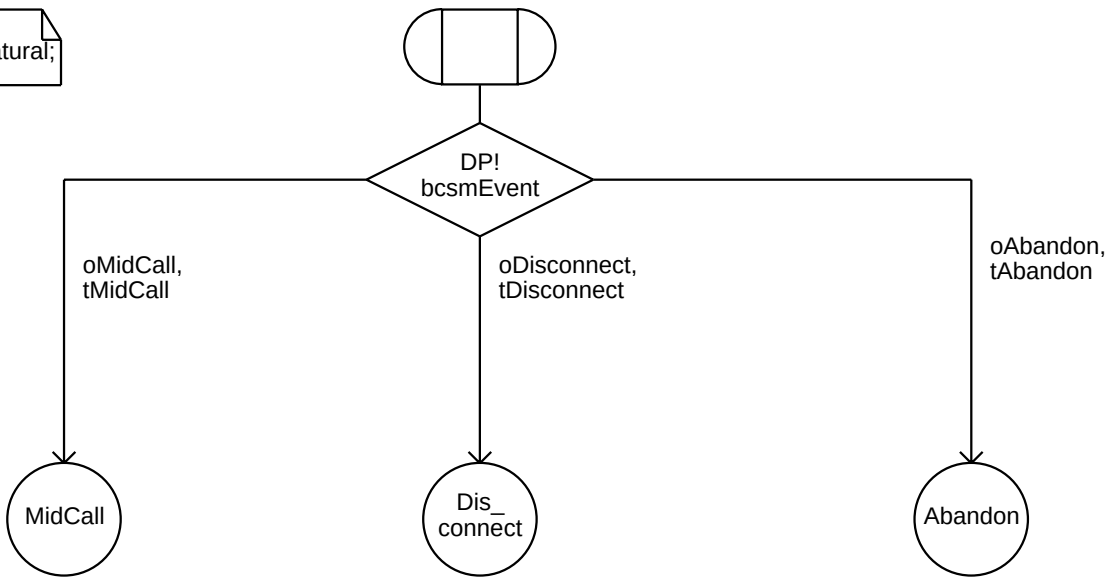


Procedure DPFiltering

1(4)

;FPAR
IN DP DPArg;

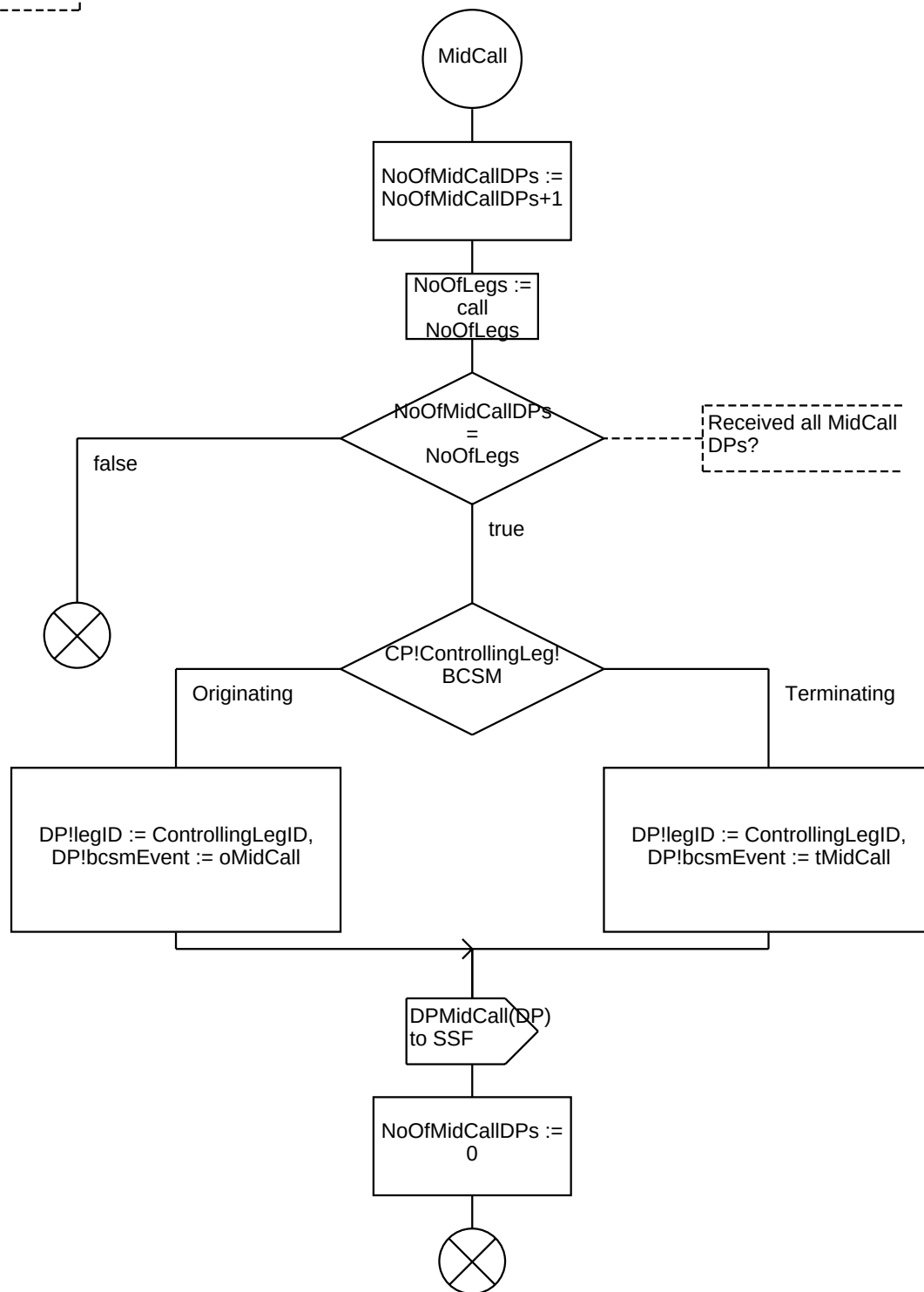
DCL
NoOfLegs Natural;



Procedure DPFiltering

2(4)

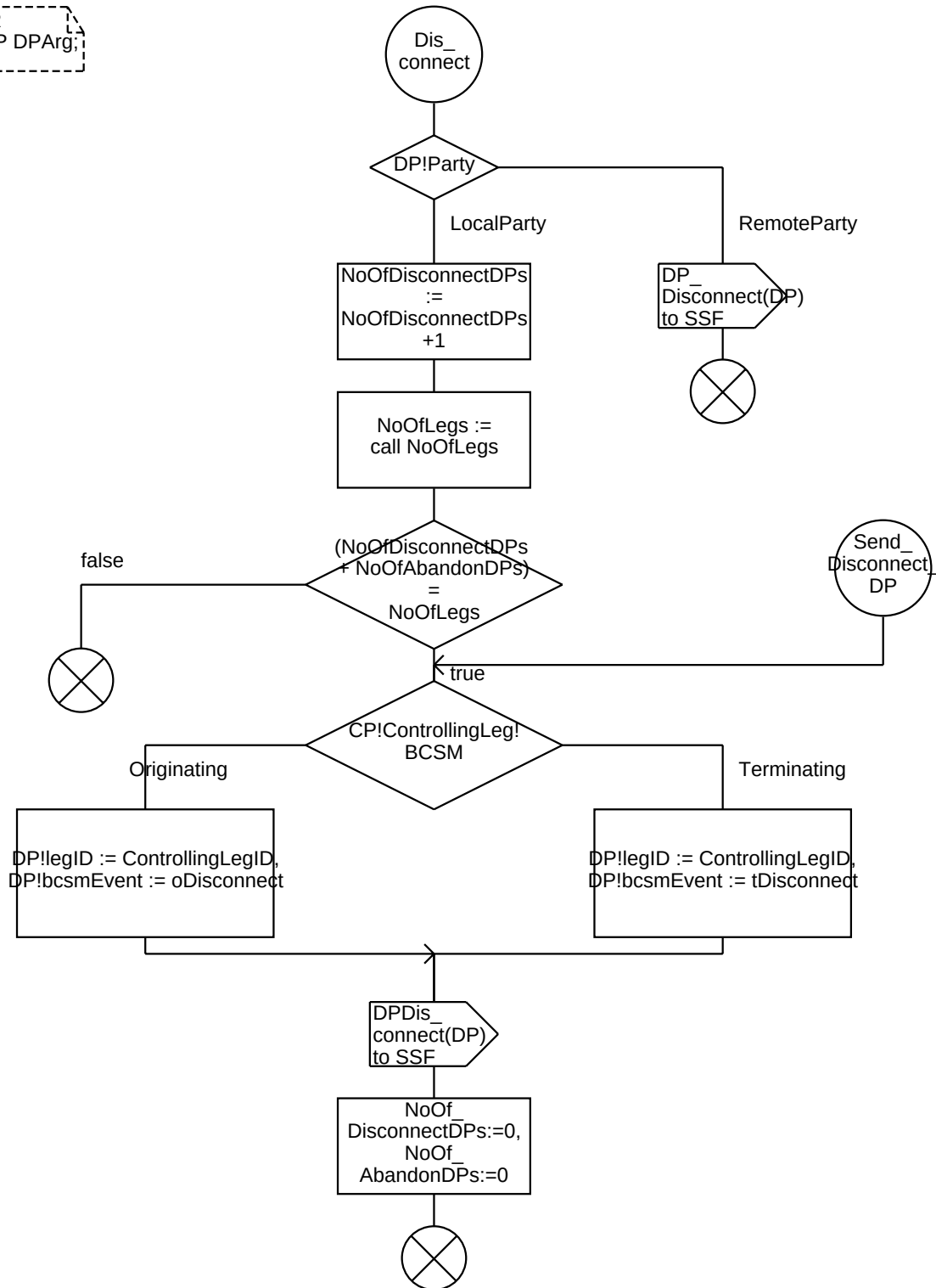
FPAR
IN DP DPArg;



Procedure DPFiltering

3(4)

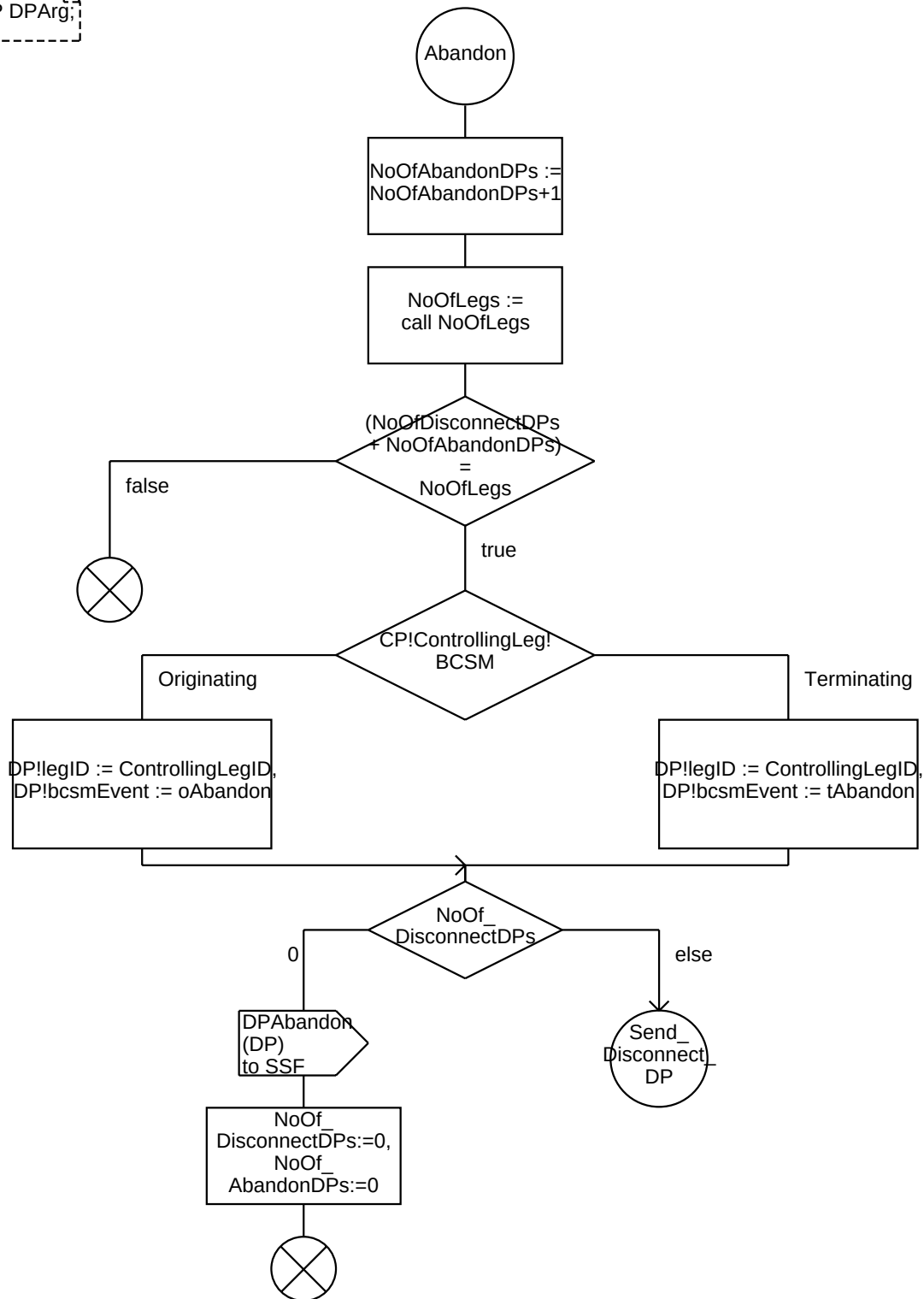
FPAR
IN DP DPArg;



Procedure DPFiltering

4(4)

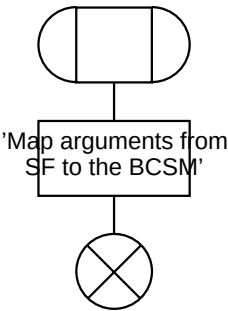
FPAR
IN DP DPArg;



Procedure MapSFTToBCSM

1(1)

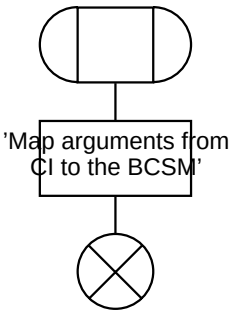
;FPAR
IN sfArg SelectFacilityArg;
RETURNS obcsmPars OBCSMPars;



Procedure MapCIToBCSM

1(1)

;FPAR
IN ciArg CollectInformationArg;
RETURNS obcsmPars OBCSMPars;

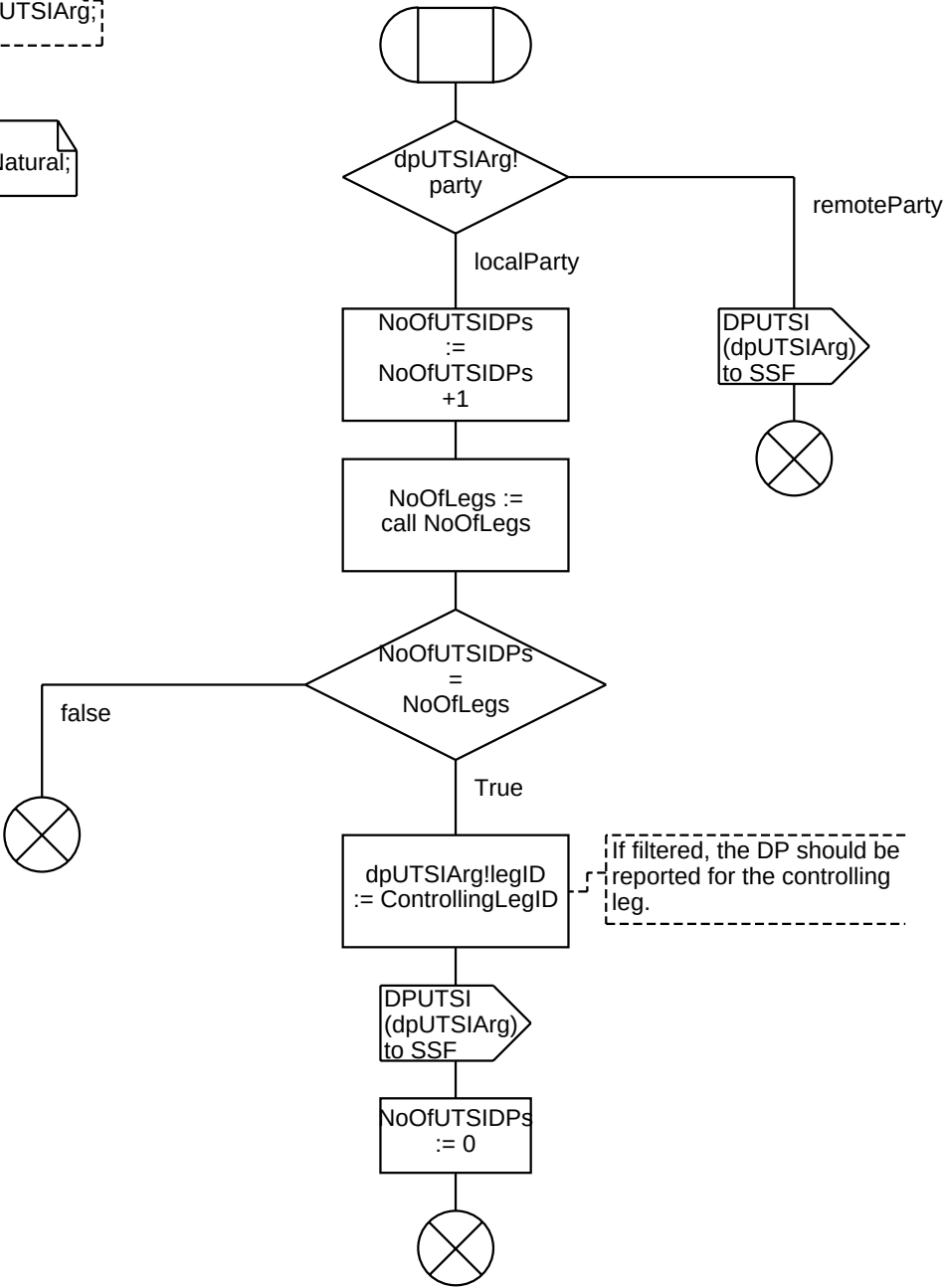


Procedure DPFilteringUTSI

1(1)

FPAR
IN dpUTSIArg dpUTSIArg;

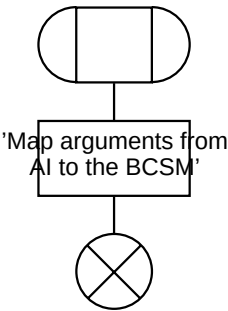
DCL
NoOfLegs Natural;



Procedure MapAIToBCSM

1(1)

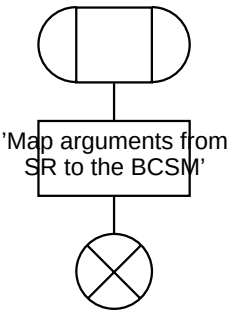
;FPAR
IN aiArg AnalyseInformationArg;
RETURNS obcsmpars OBCSMPars;



Procedure MapSRToBCSM

1(2)

;FPAR
IN srArg SelectRouteArg;
RETURNS obcsmPars OBCSMPars;



Procedure MapSRToBCSM

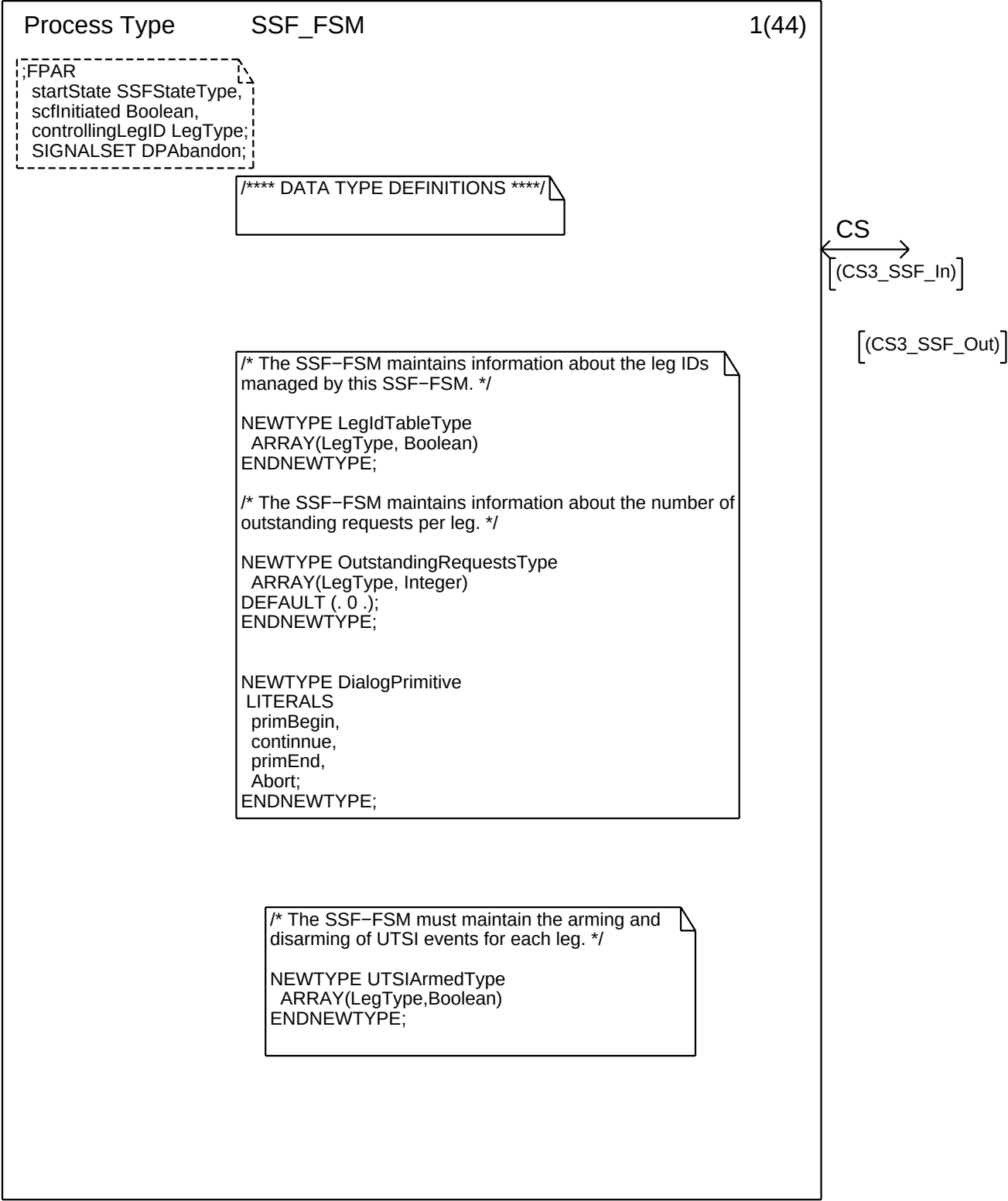
2(2)

;

FPAR

IN srArg SelectRouteArg;

RETURNS obcsmPars OBCSMPars;



Process Type SSF_FSM

2(44)

```

;FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

```

```

/**** VARIABLE DECLARATIONS ****/

```

DCL

```

/* IN operation arguments. */
aiArg AnalyseInformationArg,
adiArg AnalysedInformationArg,
acArg ApplyChargingArg,
acrArg ApplyChargingReportArg,
ariArg AssistRequestInstructionsArg,
cirArg CallInformationReportArg,
cirqArg CallInformationRequestArg,
cArg CancelArg,
ciArg CollectInformationArg,
cdiArg CollectedInformationArg,
coArg ConnectArg,
ctrArg ConnectToResourceArg,
etcArg EstablishTemporaryConnectionArg,
encArg EventNotificationChargingArg,
erBCSMArg EventReportBCSMArg,
fciArg FurnishChargingInformationArg,
hcinArg HoldCallInNetworkArg,
idpArg InitialDPArg,
icaArg InitiateCallAttemptArg,
oabnArg OAbandonArg,
oansArg OAnswerArg,
ocpbArg OCalledPartyBusyArg,
odisArg ODisconnectArg,
midclArg MidCallArg,
onoansArg ONoAnswerArg,
oaaArg OriginationAttemptAuthorizedArg,
rcArg ReleaseCallArg,
rnceArg RequestNotificationChargingEventArg,
rrBCSMEArg RequestReportBCSMEEventArg,
rtArg ResetTimerArg,
rsfArg RouteSelectFailureArg,
sciArg SendChargingInformationArg,
sfArg SelectFacilityArg,
srArg SelectRouteArg,
tansArg TAnswerArg,
tbusArg TBusyArg,
tdisArg TDisconnectArg,
taaArg TermAttemptAuthorizedArg,
tnoansArg TNoAnswerArg,

paArg PlayAnnouncementArg,
pacuiArg PromptAndCollectUserInformationArg,

ieArg ErrorArg;

```

DCL

```

currentLegID, newLegID LegType,
utsiArmed UTSArmedType,
exportEventRecord EventRecordType,
importEventRecord EventRecordType,
dpUTSIArg DPUTSIArg,
dpFacilityArg DPFacilityArg,
legID LegType;

```

DCL

```

eventTable EventTableType,
applicationActive Boolean := false,
applyChargingReportPending Boolean := false,
callInformationReportPending Boolean := false,
eventNotificationChargingPending Boolean := false,
dp DPArg,
invokeID InvokeID,
termination Boolean,
flagContinue Boolean,
pic PICArg,
cs PId,
lastPrimitive DialogPrimitive,
sfEncountered Boolean := false,
outstandingRequests OutstandingRequestsType,

storedDP DPArg, /* Last received TDP-R or EDP-R. */
dpStored Boolean := false, /* Flag to indicate if a DP is stored. */

/* Signal Parameter Declarations */
legIDTable LegIDTableType,
serviceKey ServiceKey;

```

DCL

```

/* IN operation arguments. */
autArg AuthorizeTerminationArg,
ctwaArg ContinueWithArgumentArg,
dfcwaArg DisconnectForwardConnectionWithArgumentArg,
dlArg DisconnectLegArg,
erArg EntityReleasedArg,
erfArg EventReportFacilityArg,
fsaArg FacilitySelectedAndAvailableArg,
oaArg OriginationAttemptArg,
osusArg OSuspendedArg,
reconArg ReconnectArg,
rutsiArg ReportUTSIArg,
rrutsiArg RequestReportUTSIArg,
rrfeArg RequestReportFacilityEventArg,
sendfArg SendFacilityInformationArg,
sstuiArg SendSTUIArg,
sspArg SetServiceProfileArg,
taArg TerminationAttemptArg,
tsusArg TSuspendedArg,

parmArg PromptAndReceiveMessageArg,
scArg ScriptCloseArg,
siArg ScriptInformationArg,
scrArg ScriptRunArg;

```

Process Type SSF_FSM

3(44)

;FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

/**** TIMER DECLARATIONS *****/

```
/*
The timer Tssf is used to prevent excessive
call suspension time and to guard the
association between the SSF and the SCF.
*/

TIMER    Tssf;

/*
The timer Tssf can have four different values

Tssf_Duration1 - Sending an InitialDP
Tssf_Duration2 - Entering state Waiting_for_Instruction
Tssf_Duration3 - Receiving an HoldCallInNetwork
Tssf_Duration4 - Entering Waiting_for_End_of_User_Interaction or
                  Waiting_for_End_of_Temporary_Connection
*/

/* The Tssf durations have been initialised to nominal values (1hr) */
DCL
Tssf_Duration1    DURATION := 3600000,
Tssf_Duration2    DURATION := 3600000,
Tssf_Duration3    DURATION := 3600000,
Tssf_Duration4    DURATION := 3600000,

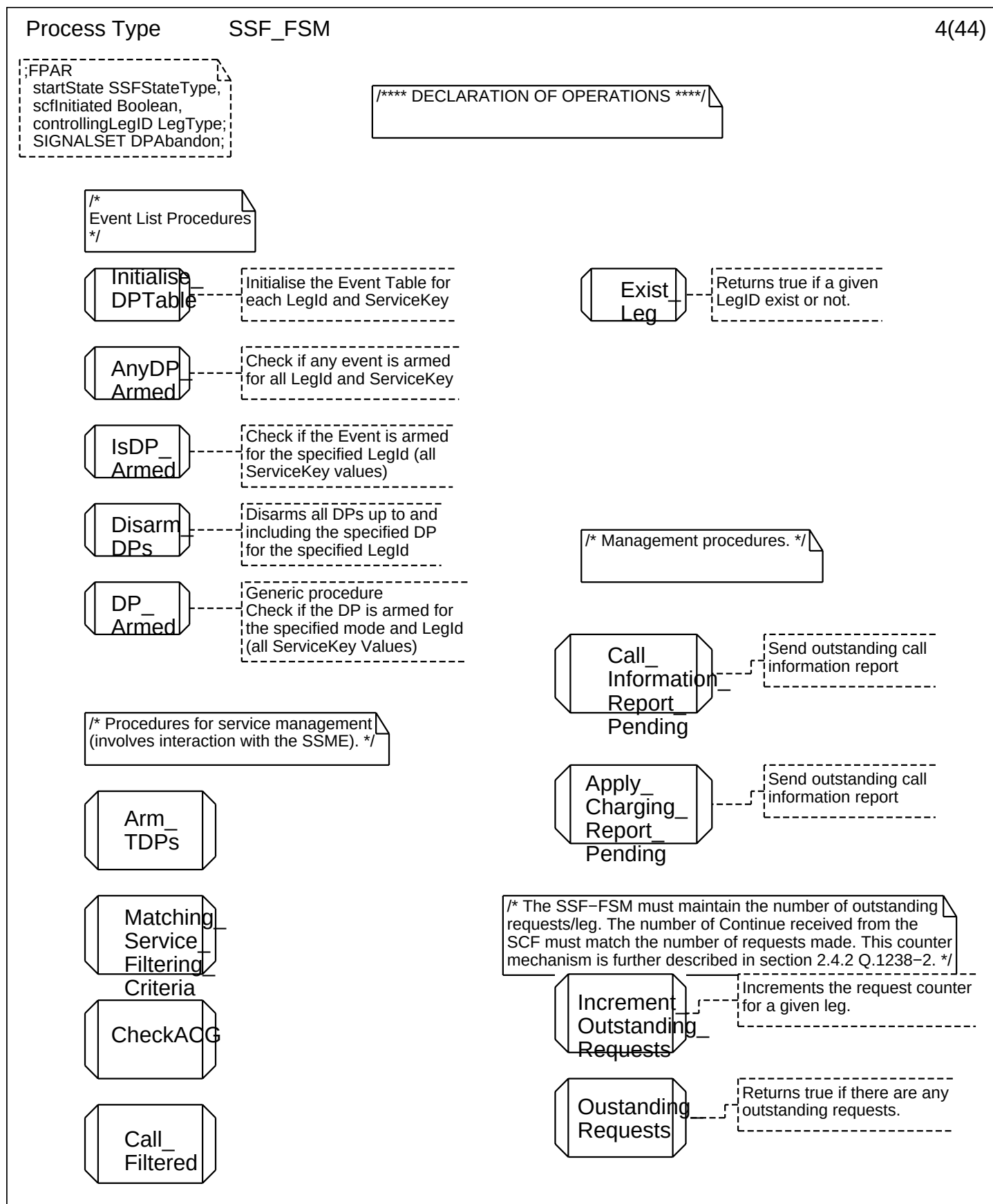
Tssf_Duration DURATION;

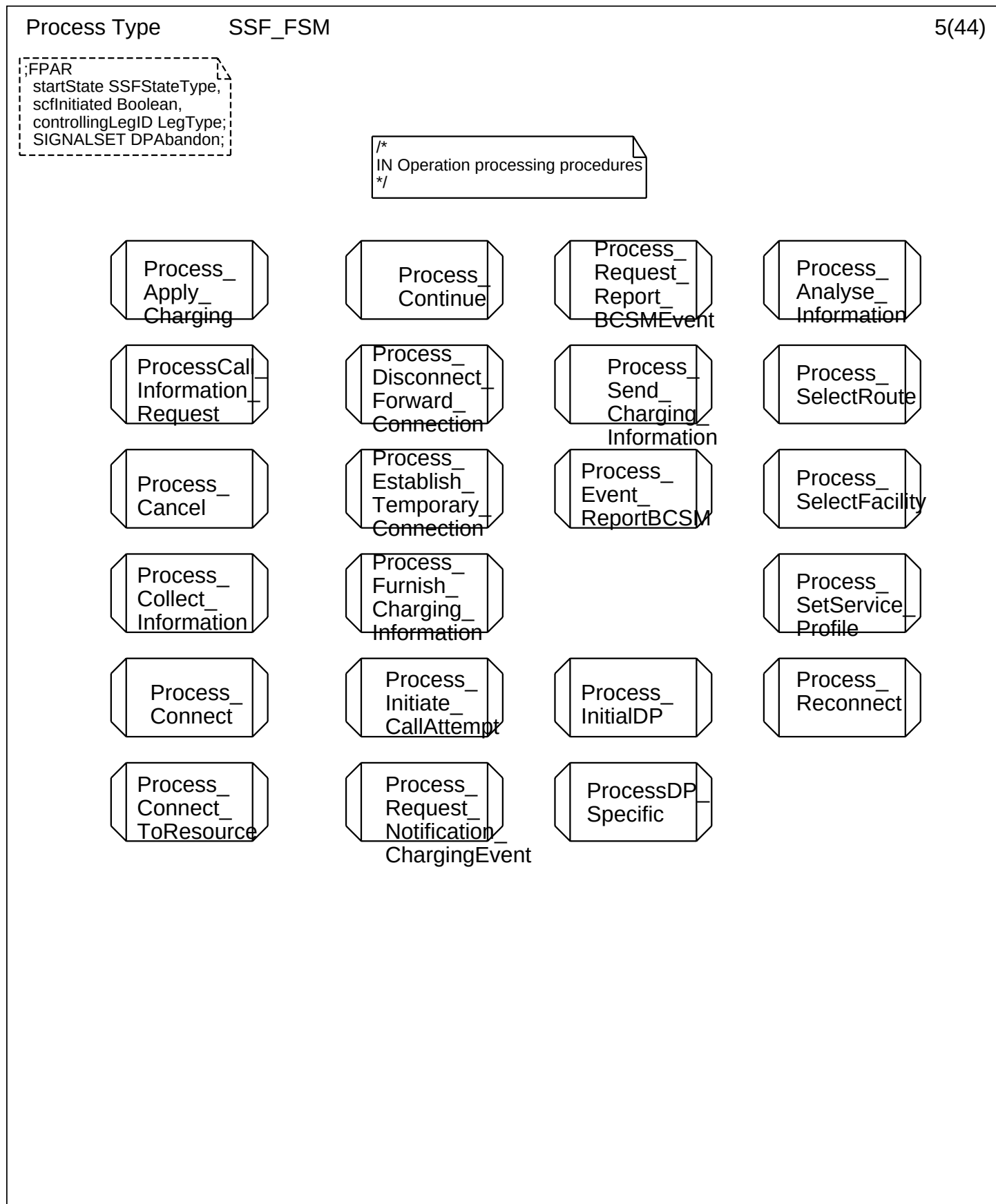
/* The following flags are used to control the operation */
/* of the timer Tssf in Waiting_for_Instruction state */

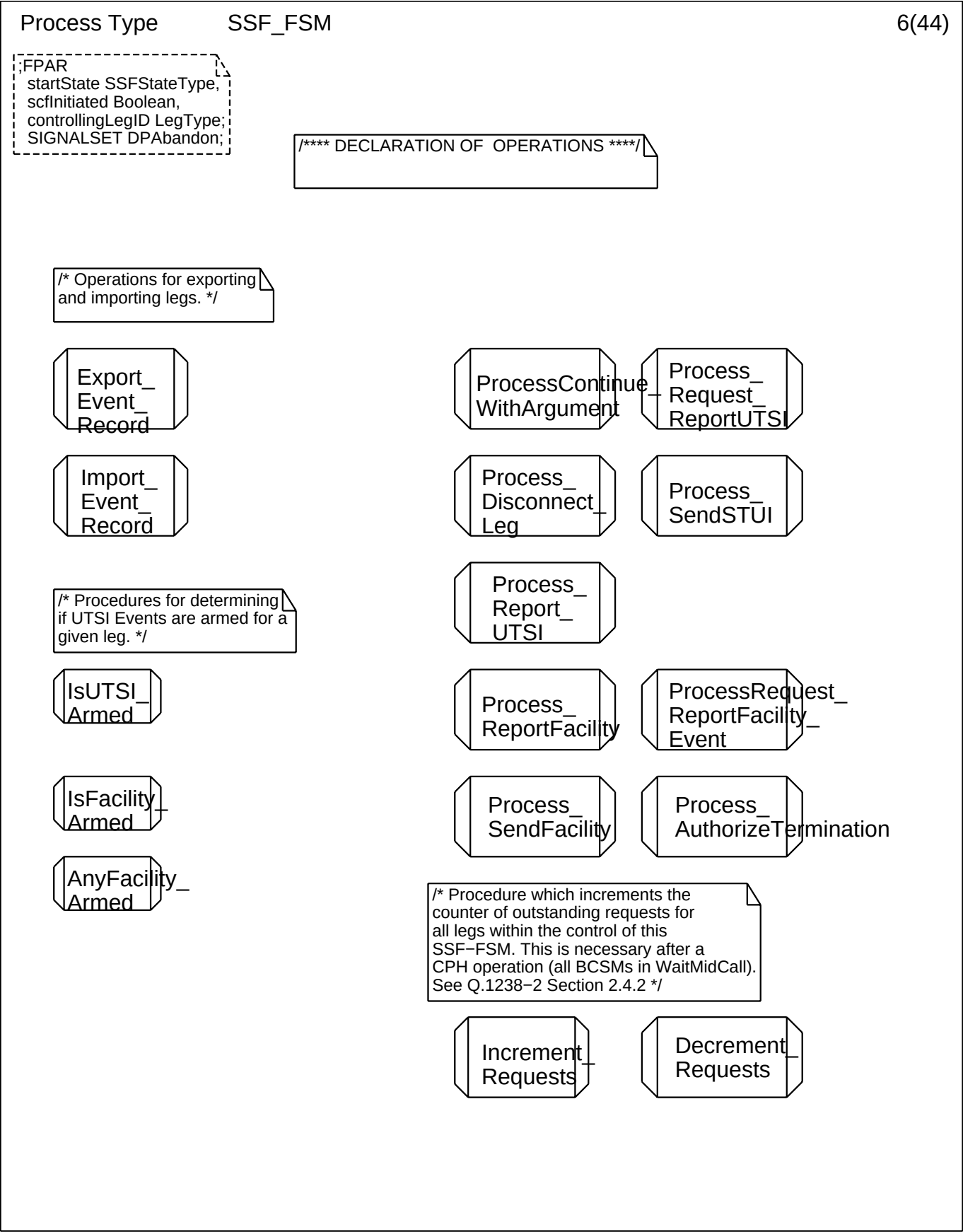
DCL
InitialTssfActive BOOLEAN := FALSE,
InitialTssfReset  BOOLEAN := FALSE;
```

/* Declaration Required only for Simulation */

```
DCL
DP_Specific  BOOLEAN := FALSE;
```





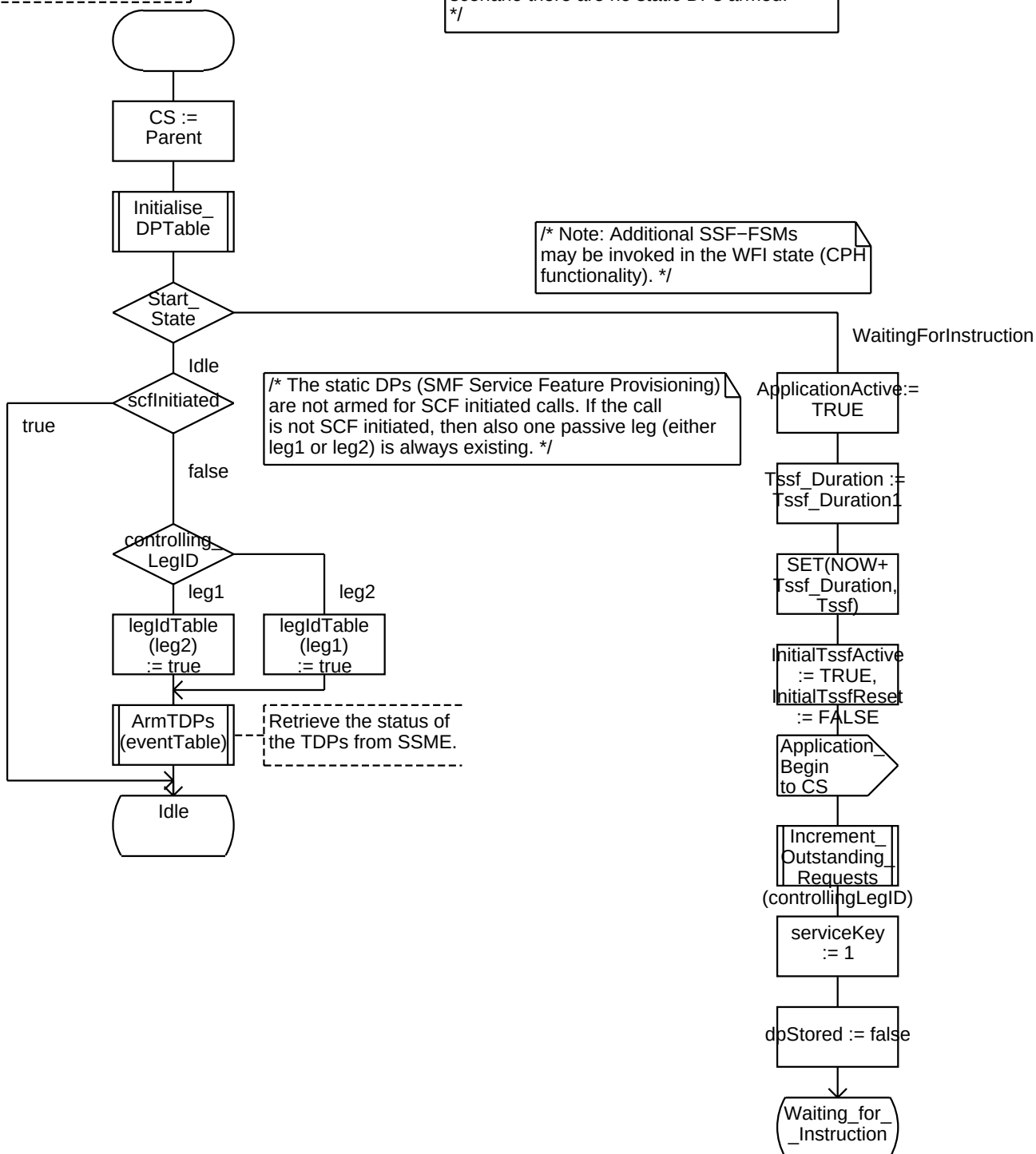
Process Type SSF_FSM

7(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

/*
The SSF is invoked in Waiting For Instruction
state for SCF Initiated Call Attempts. For this
scenario there are no static DPs armed.
*/

/* Note: Additional SSF-FSMs
may be invoked in the WFI state (CPH
functionality). */



Process Type

SSF_FSM

8(44)

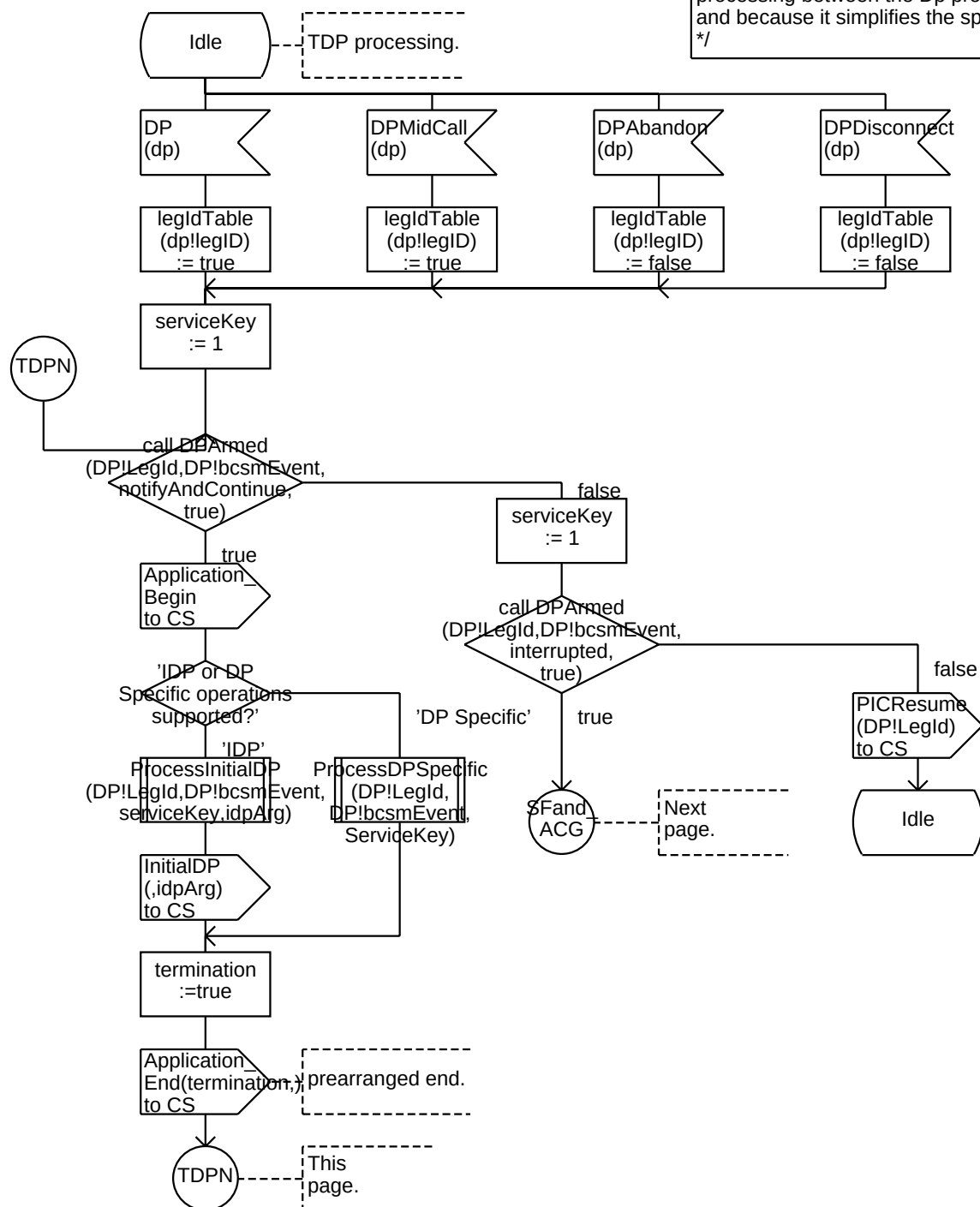
```

;FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

```

/***** DP PROCESSING *****/

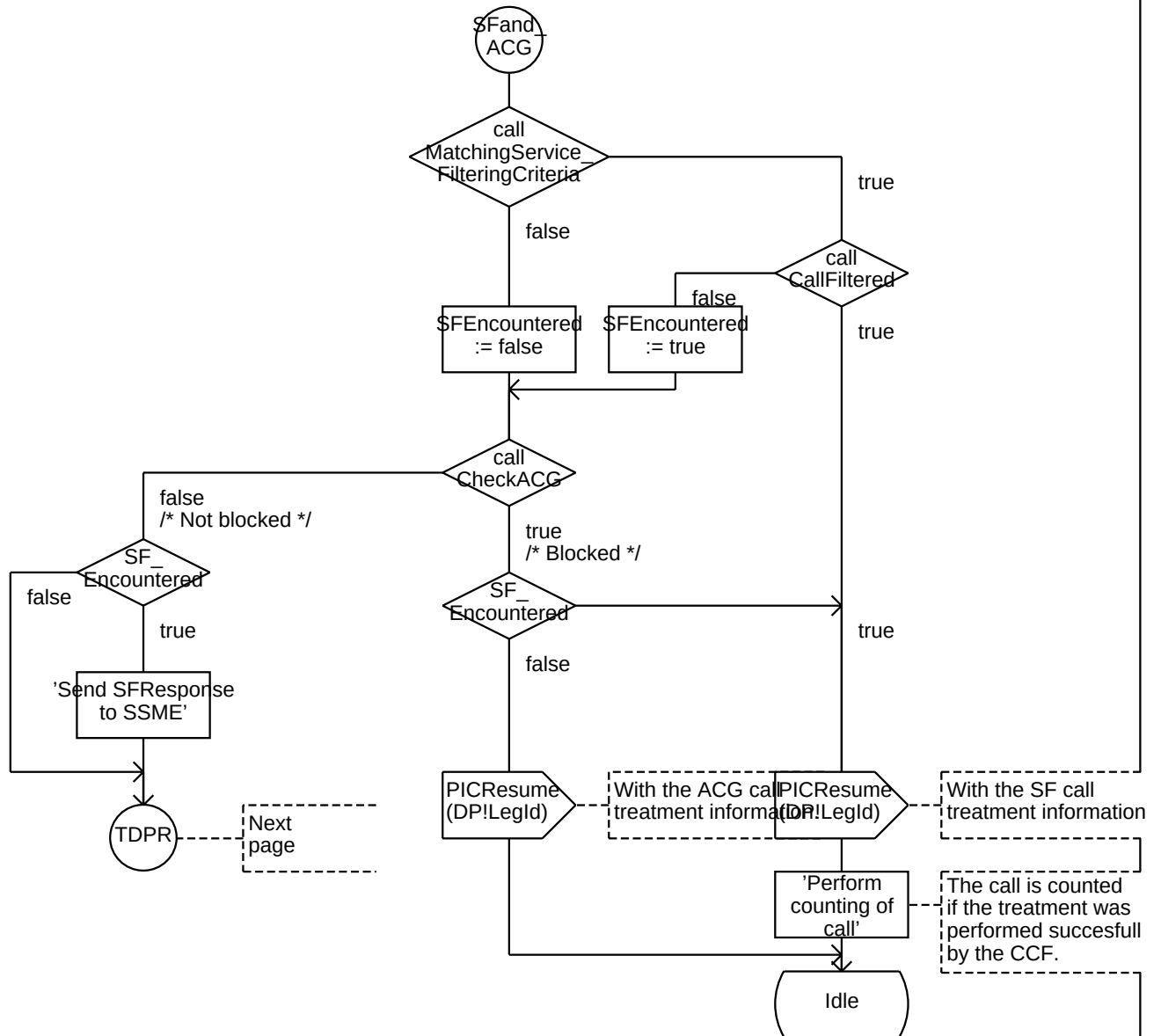
*
The detection point processing is part of the SSF because there is no concept of concurrent processing between the Dp processing and the SS and because it simplifies the specification.
*/



```

;FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

```

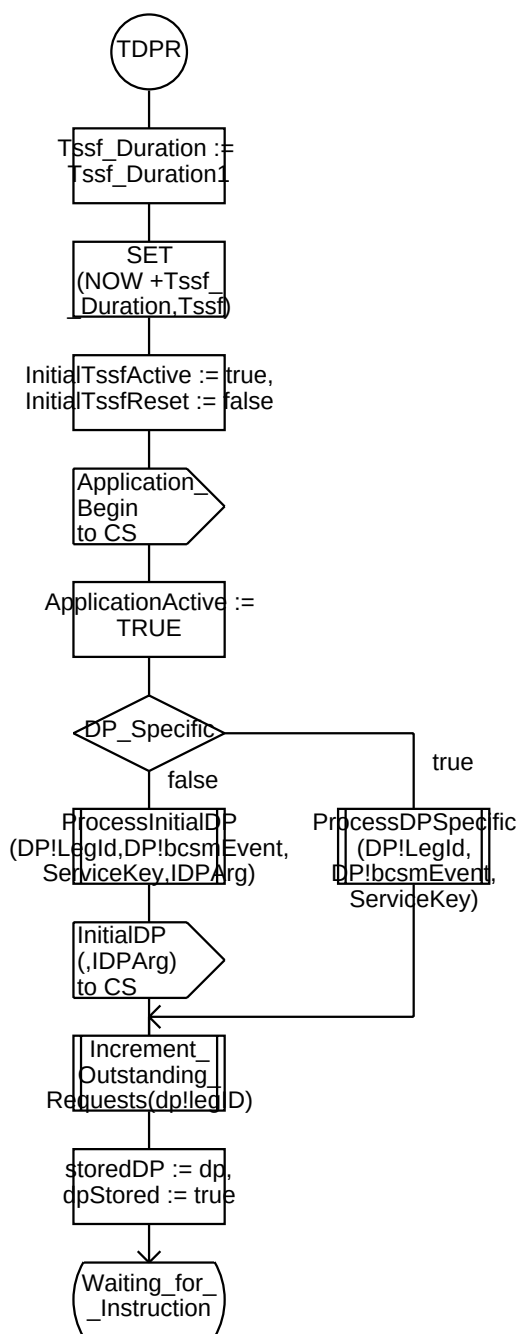


Process Type

SSF_FSM

10(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

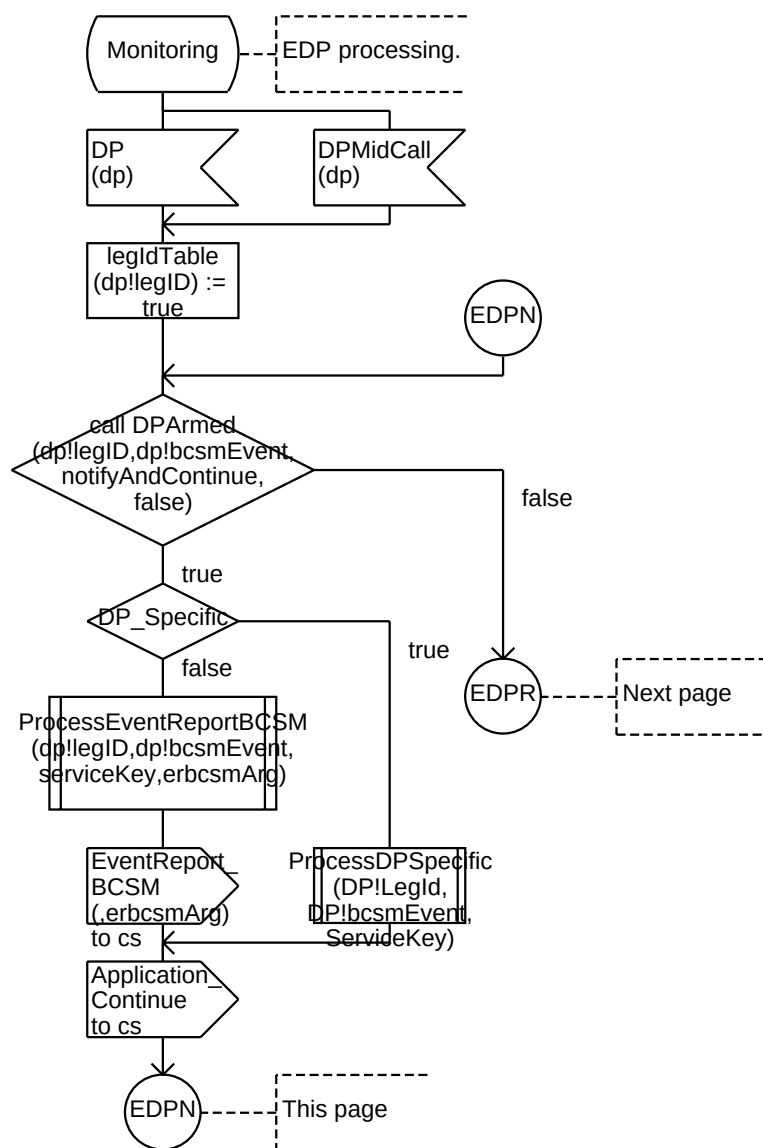


Process Type

SSF_FSM

11(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

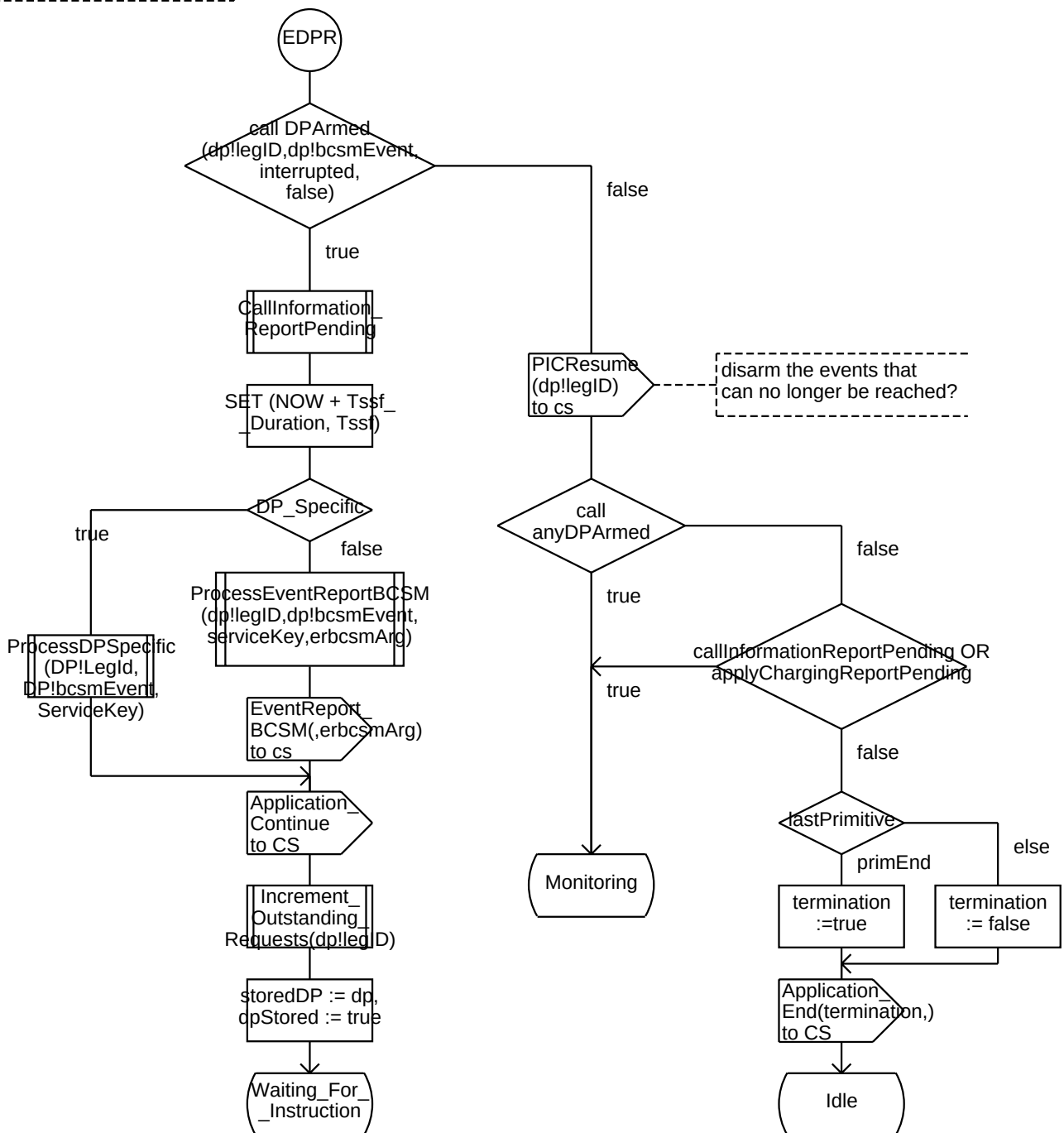


Process Type

SSF_FSM

12(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

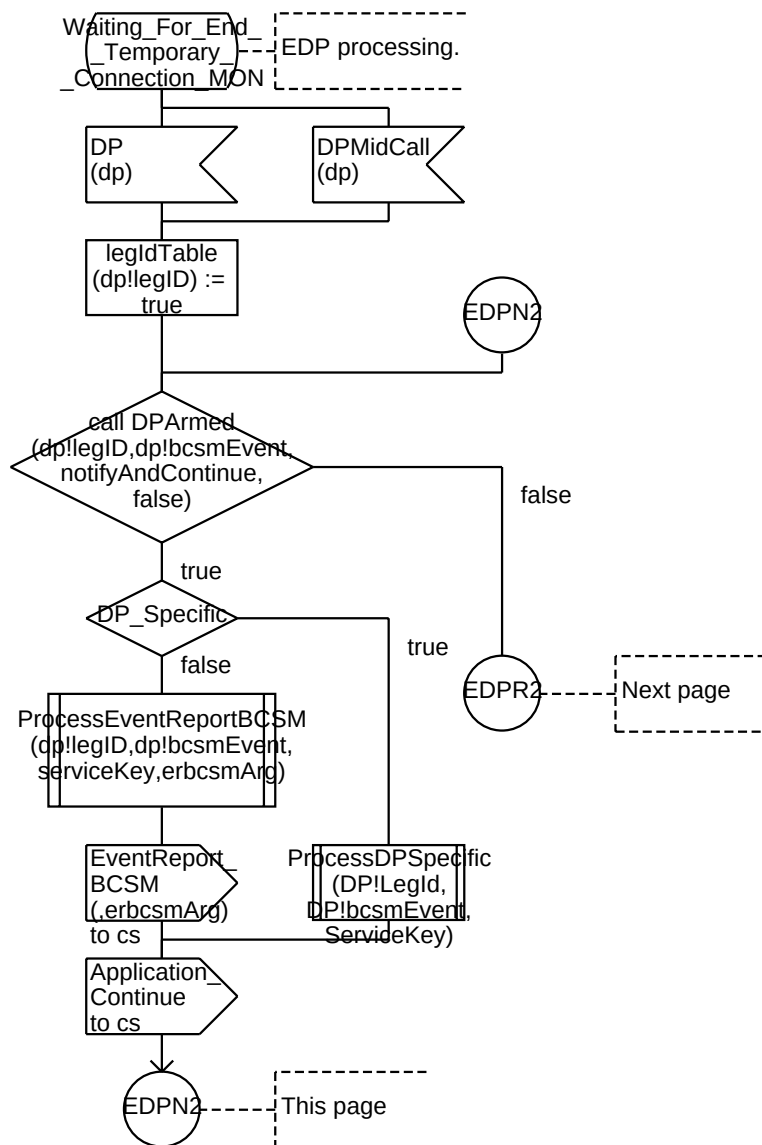


Process Type

SSF_FSM

13(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

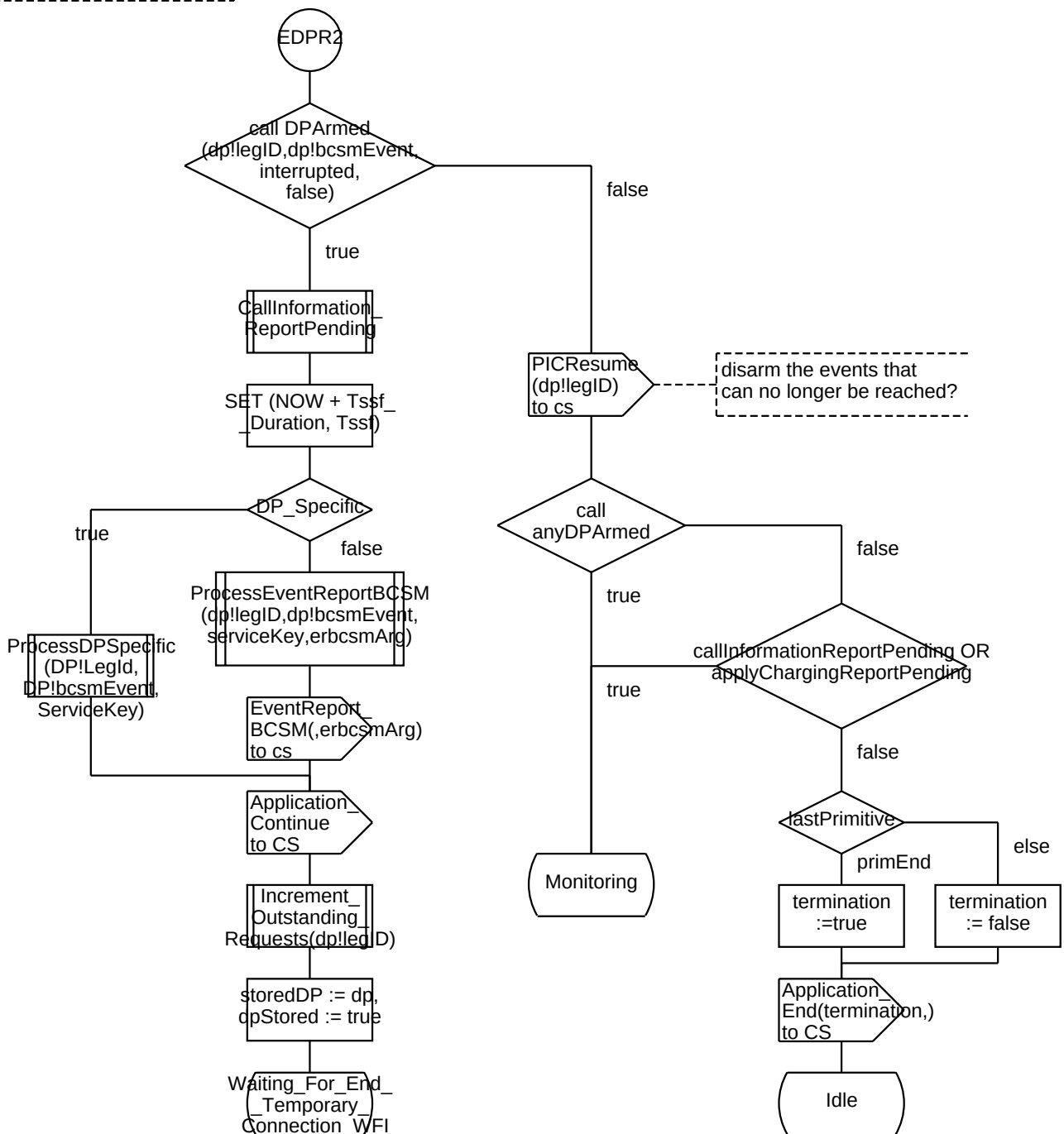


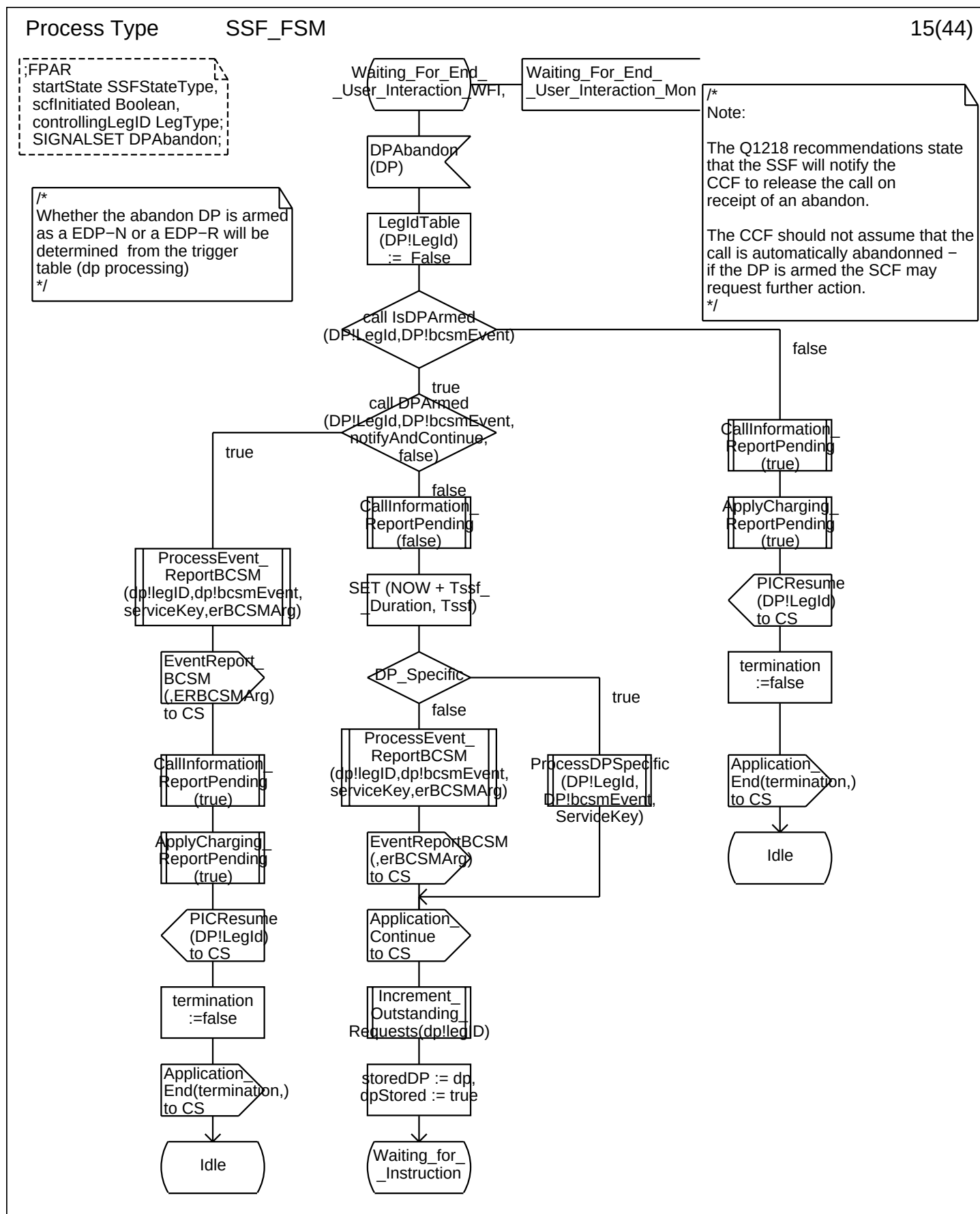
Process Type

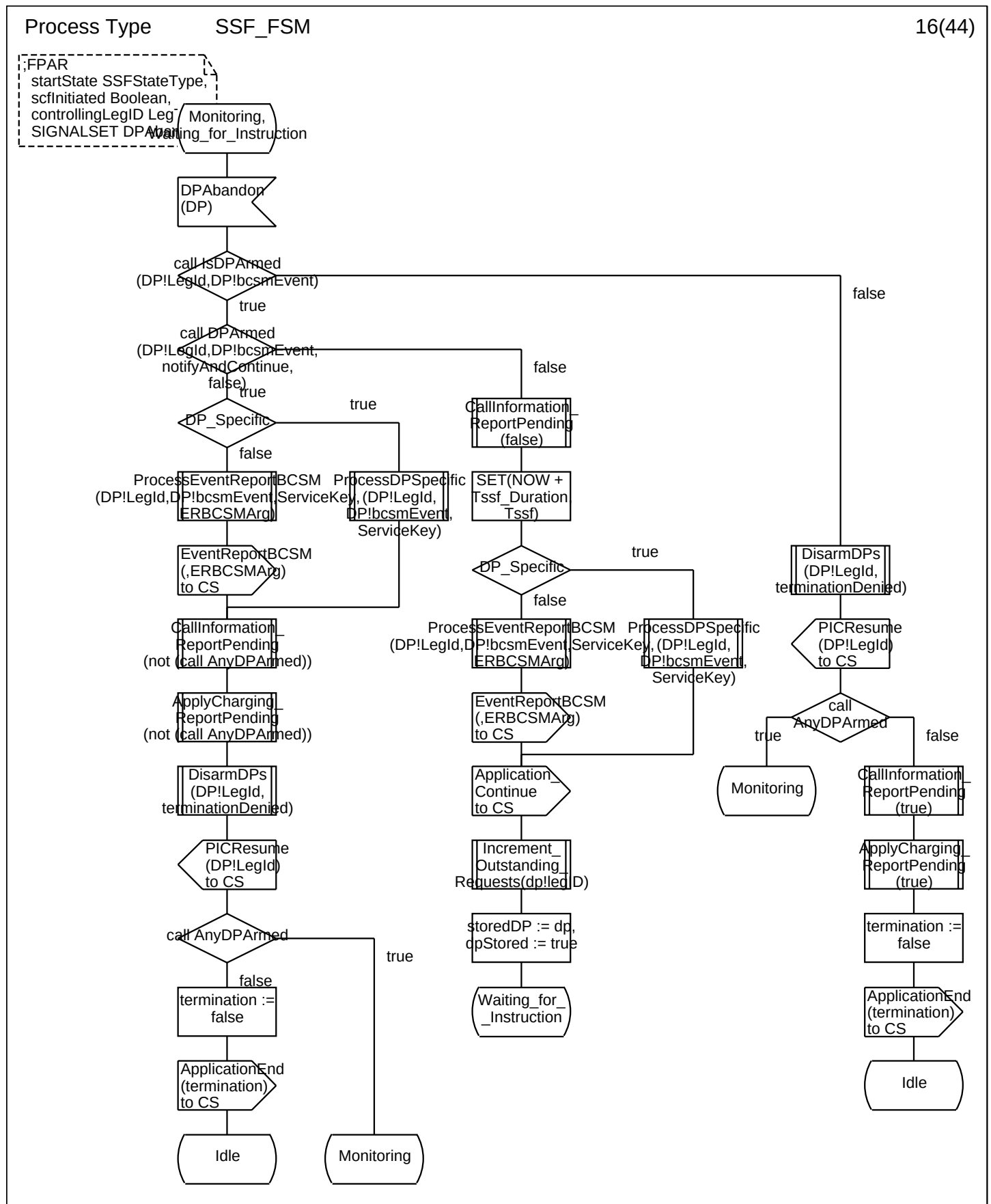
SSF_FSM

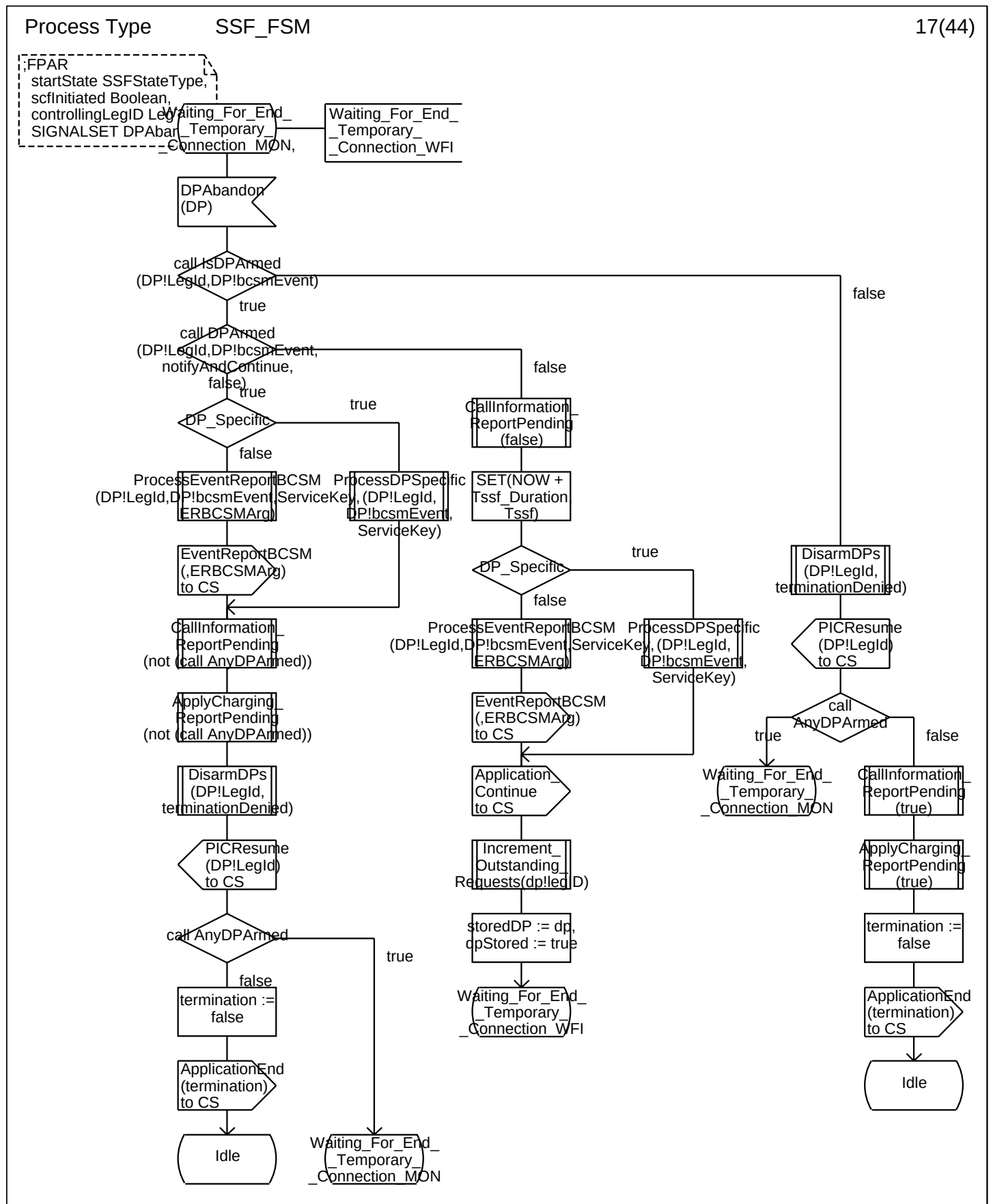
14(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;









Process Type

SSF_FSM

18(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

/*
Whether the disconnect DP is armed
as a EDP-N or a EDP-R will be
determined from the trigger
table (dp processing)
*/

Waiting_For_End_Interaction_WF
_User_Interaction_Mon

DPDisconnect
(DP)

'LegIdTable
(DP!LegId)
:= False'

call IsDPArmed
(DP!LegId,DP!bcsmEvent)

call DPArmed
(DP!LegId,DP!bcsmEvent,
notifyAndContinue,
false)

true

true

DP_Specific

false

ProcessEvent
ReportBCSM
(DP!LegId,DP!bcsmEvent,
ServiceKey,ERBCSMArg)

ProcessDPSpecific
(DP!LegId,
DP!bcsmEvent,
ServiceKey)

CallInformation
ReportPending
(true)

ApplyCharging
Report_Pending
(true)

PICResume
(DP!LegId)
to CS

termination
:=false

Application
End(termination,)
to CS

Idle

CallInformation
ReportPending
(false)

SET (NOW + Tssf_
Duration, Tssf)

DP_Specific

true

false

ProcessEvent_
ReportBCSM
(DP!LegId,DP!bcsmEvent,
ServiceKey,ERBCSMArg)

EventReport
BCSM
(ERBCSMArg)
to CS

ProcessDPSpecific
(DP!LegId,
DP!bcsmEvent,
ServiceKey)

Application
Continue
to CS

Increment
Outstanding
Requests(dp!legID)

storedDP := dp,
dpStored := true

Waiting_for_
_Instruction

/*
Note:

The Q1218 recommendations state
that the SSF will notify the
CCF to release the call on
receipt of a disconnect.

The CCF should not assume that the
call is automatically disconnected -
if the DP is armed the CCF may
request further action.
*/

CallInformation
ReportPending
(true)

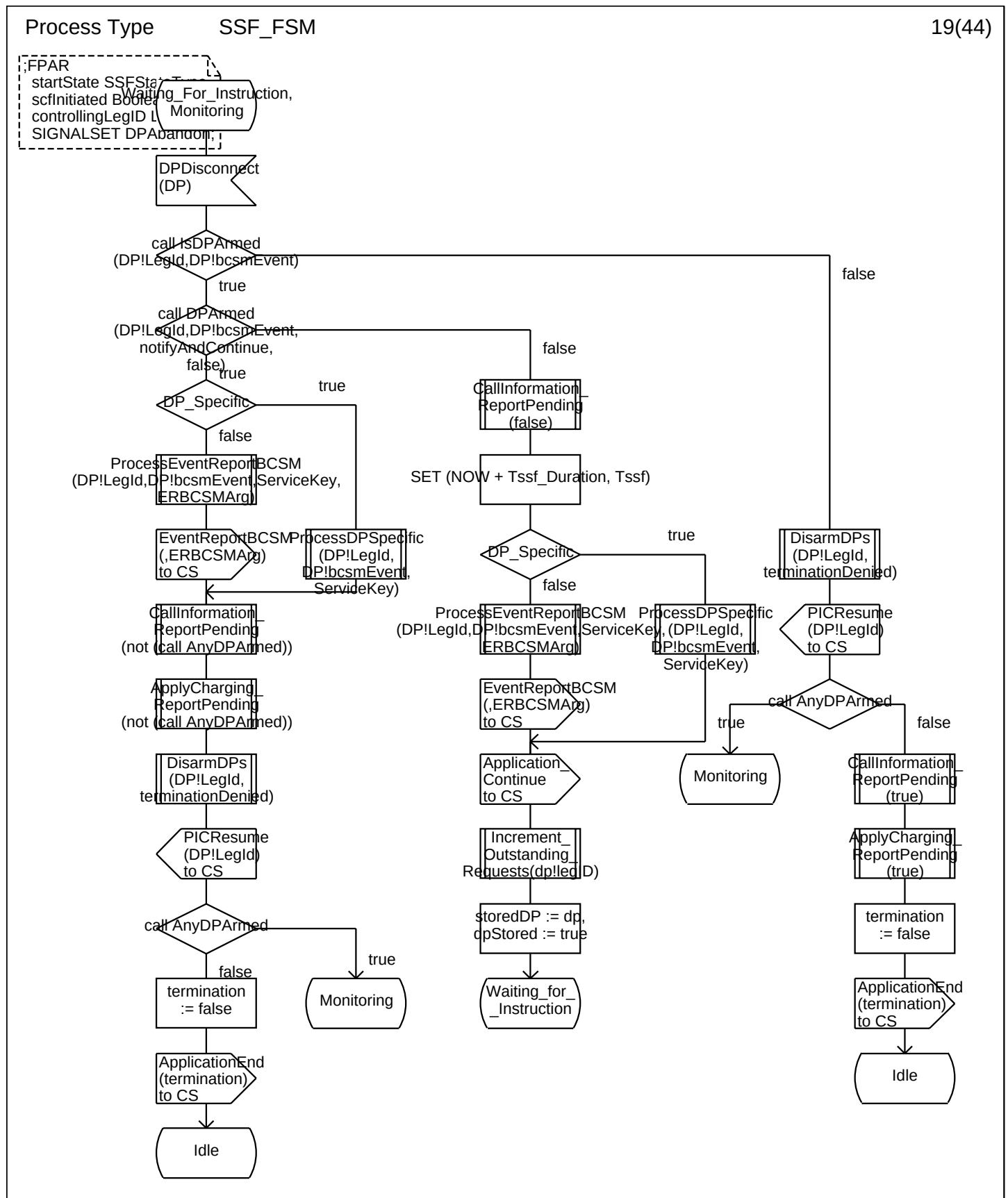
ApplyCharging
ReportPending
(true)

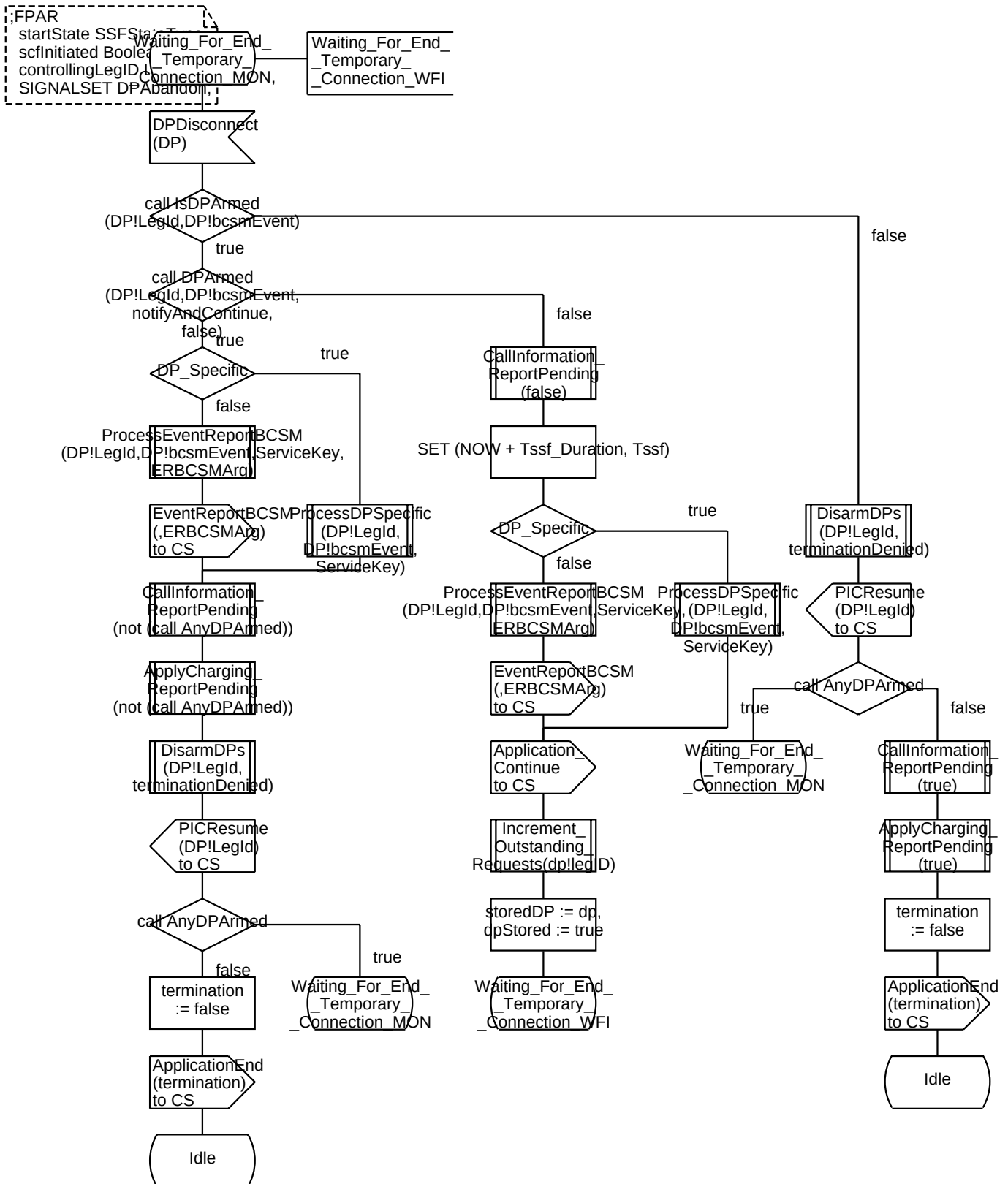
PICResume
(DP!LegId)
to CS

termination
:=false

Application
End(termination,)
to CS

Idle





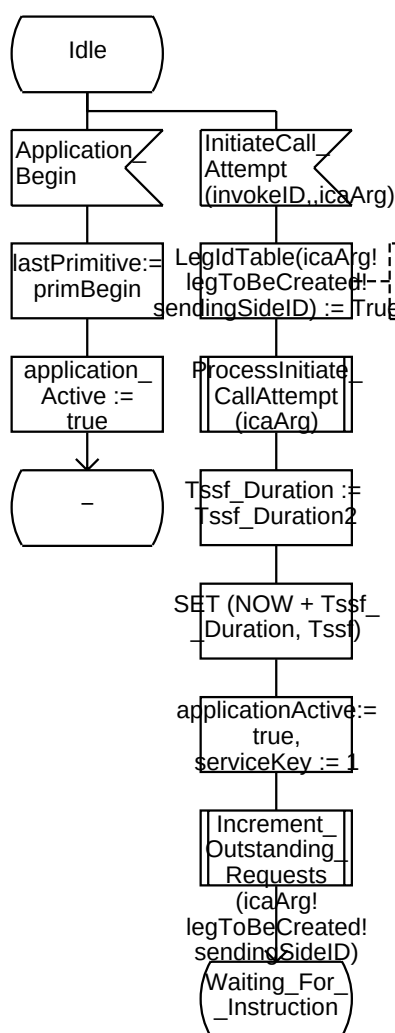
Process Type

SSF_FSM

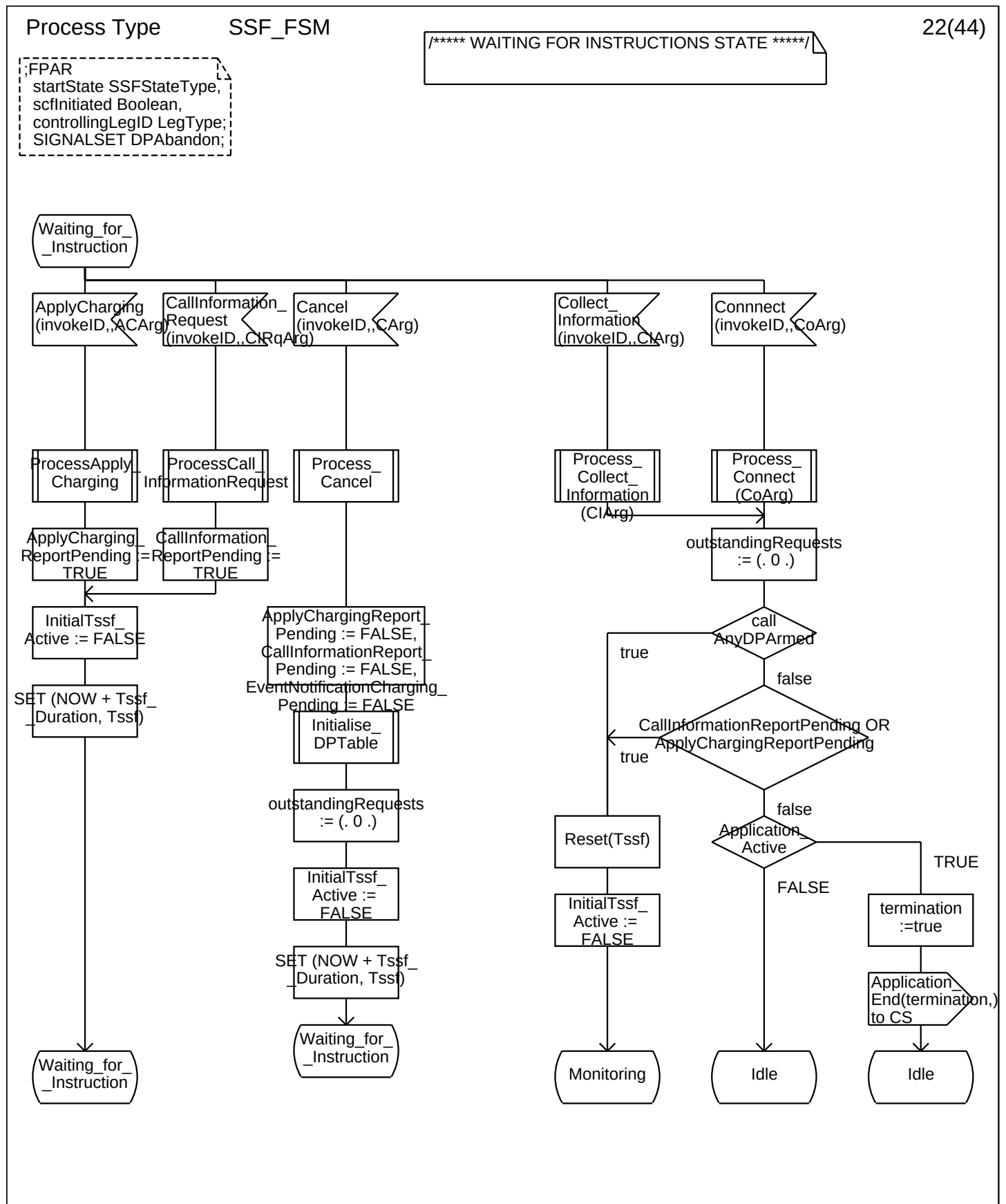
21(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

/* ***** IDLE STATE ***** */



/*
Note:
An SSF instance only exists for the duration of a Trigger Detection Point and subsequent Event Detection Points and Call Information Requests.
When the FSM returns to the Idle state from the WaitingForInstruction or Monitoring state it is implying that the SSF terminates. However, because the model has a static configuration the SSF is not terminated.
The SSF contains explicit application begin, abort, and end primitives which are used to determine if the SSF application is active. The SSF application can receive or send the primitives as appropriate.
*/

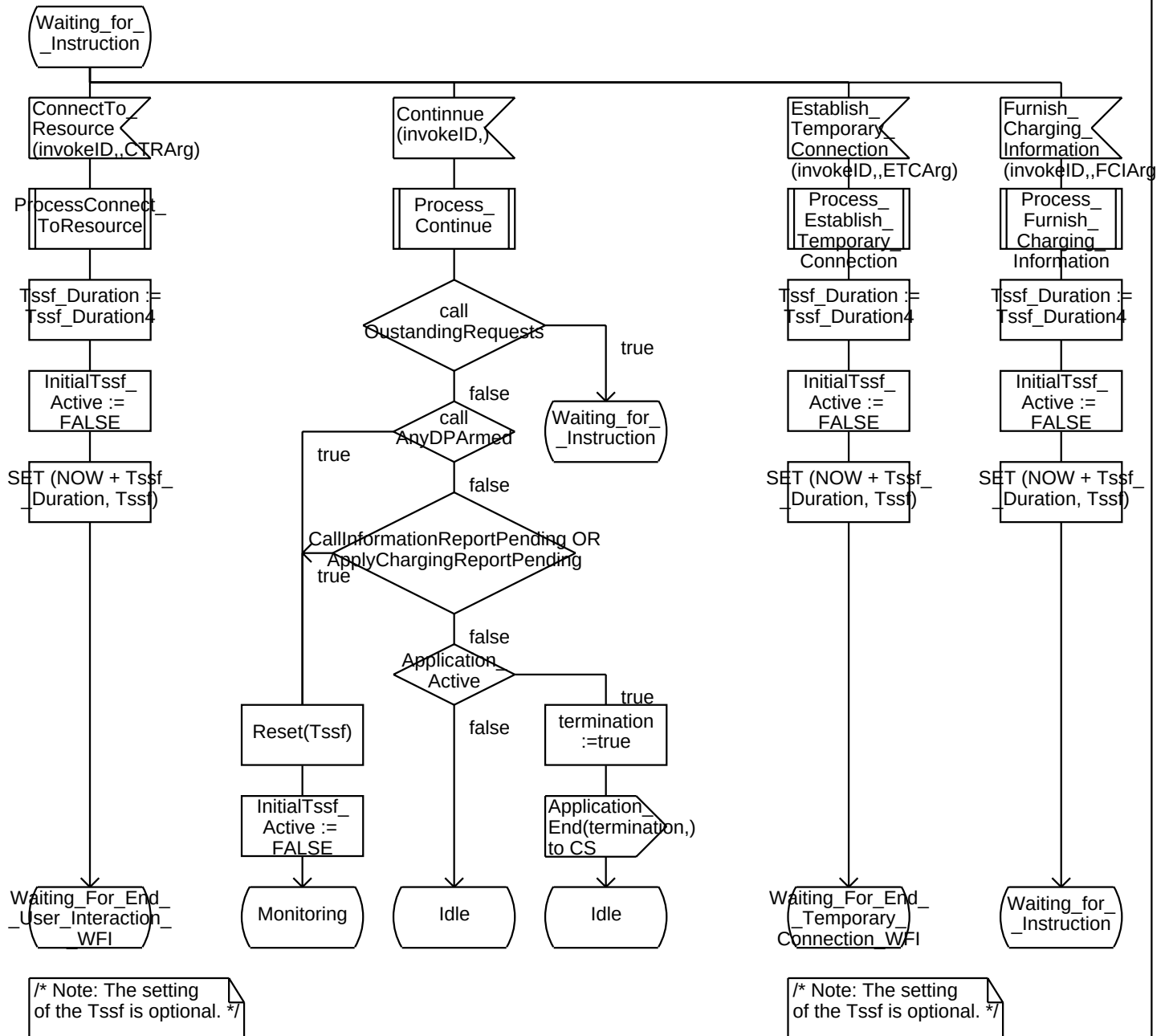


Process Type

SSF_FSM

23(44)

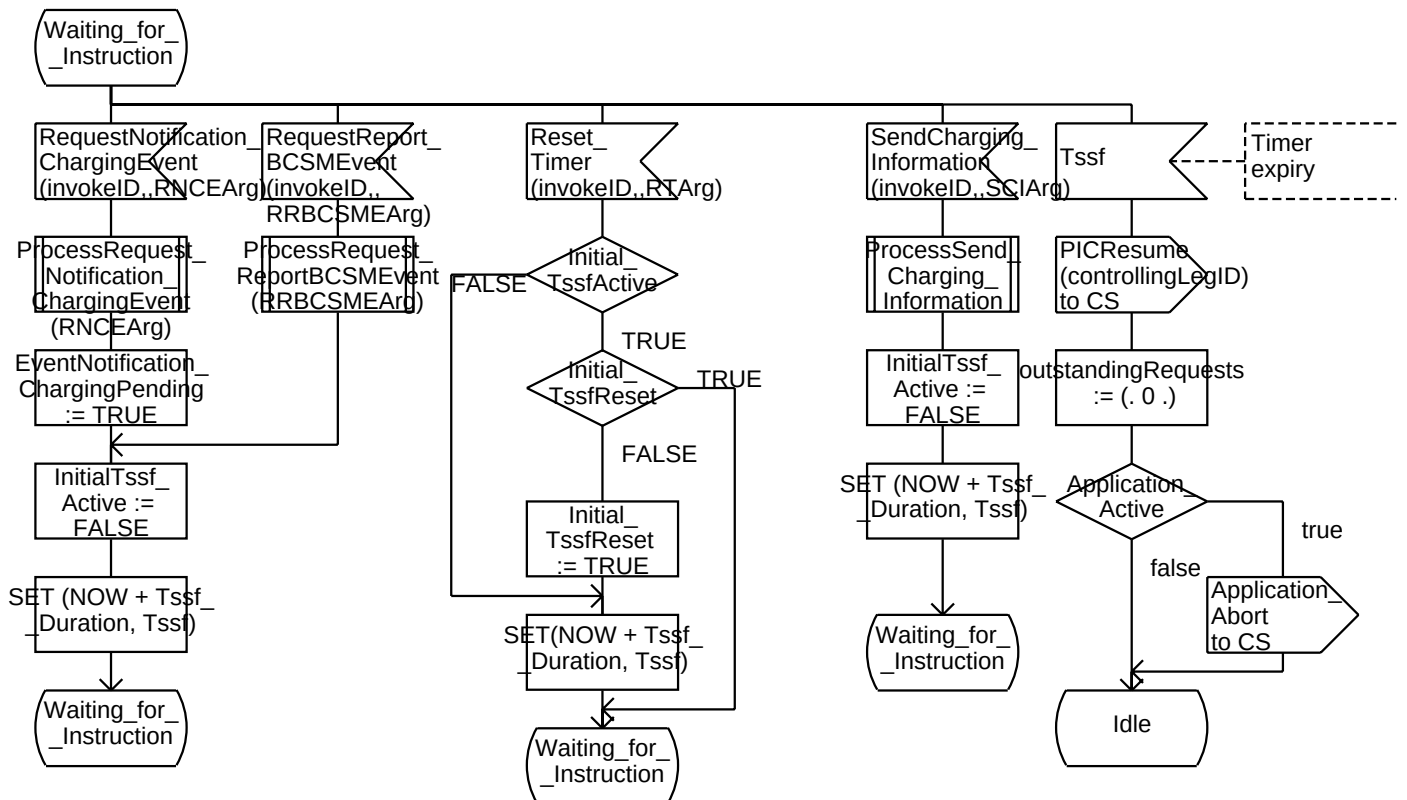
FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;



```

; FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

```

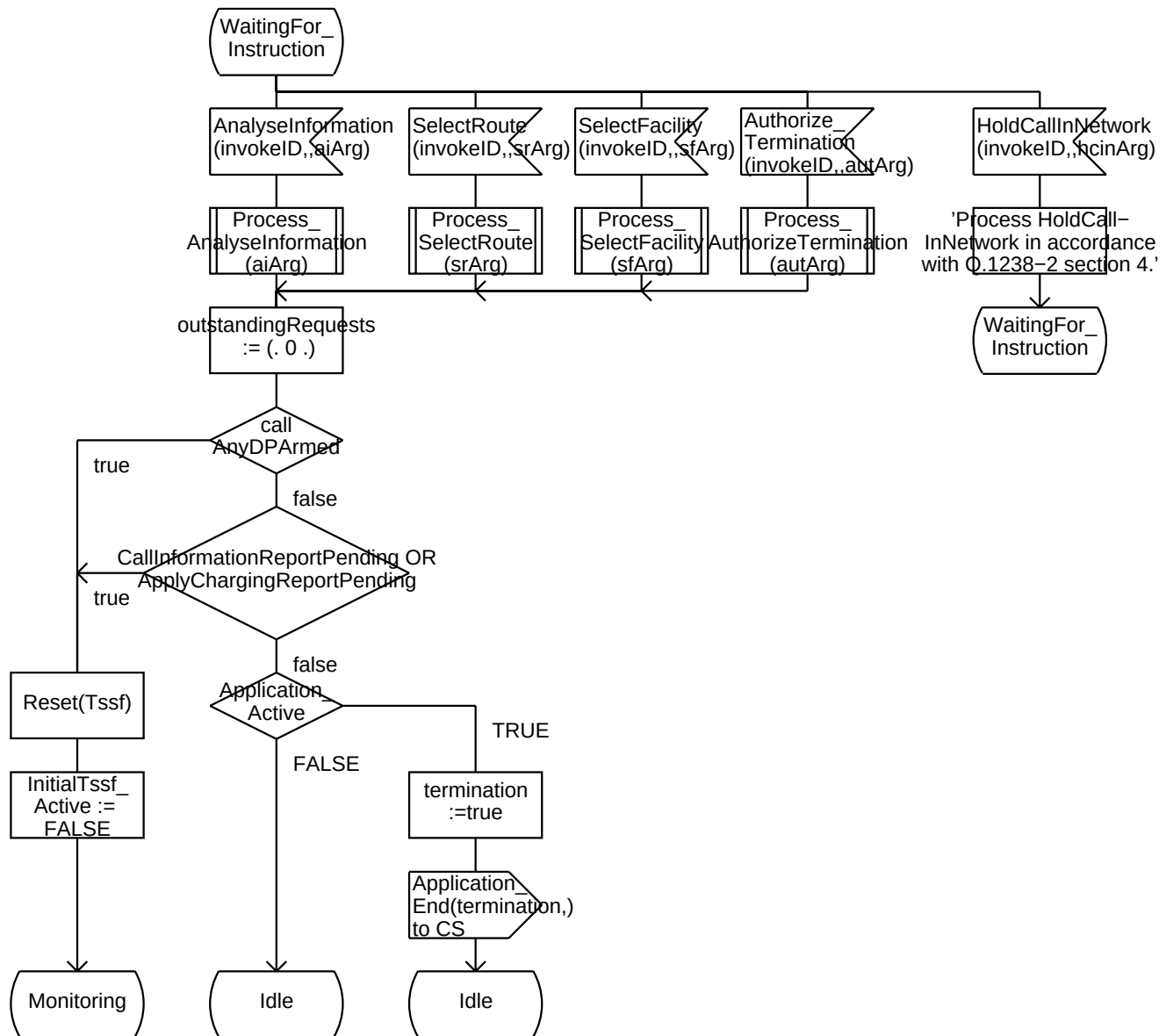


Process Type

SSF_FSM

25(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;



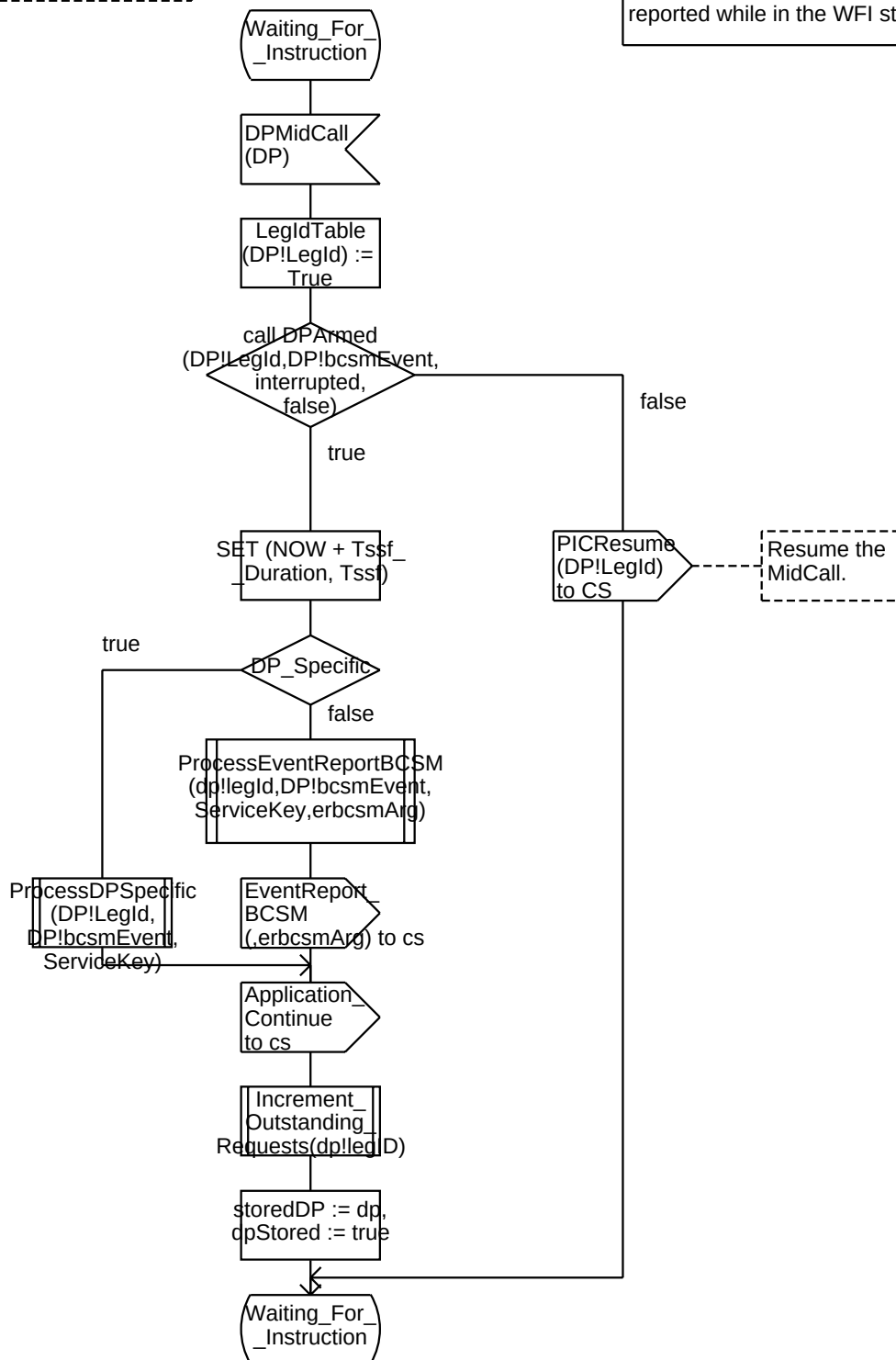
Process Type

SSF_FSM

26(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

/* Note: MidCall EDPs must be reported while in the WFI state. */



;FPAR

startState SSFStateType,

scfInitiated Boolean,

controllingLegID LegType;

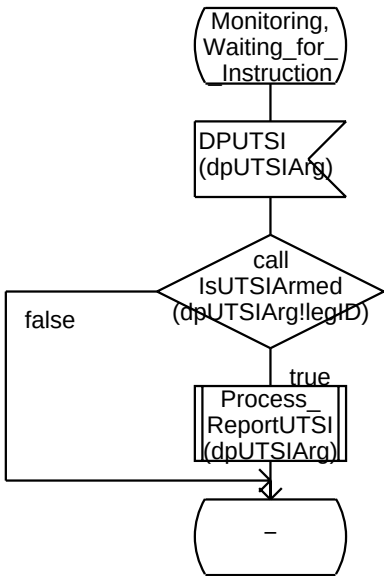
SIGNALSET DPAbandon;

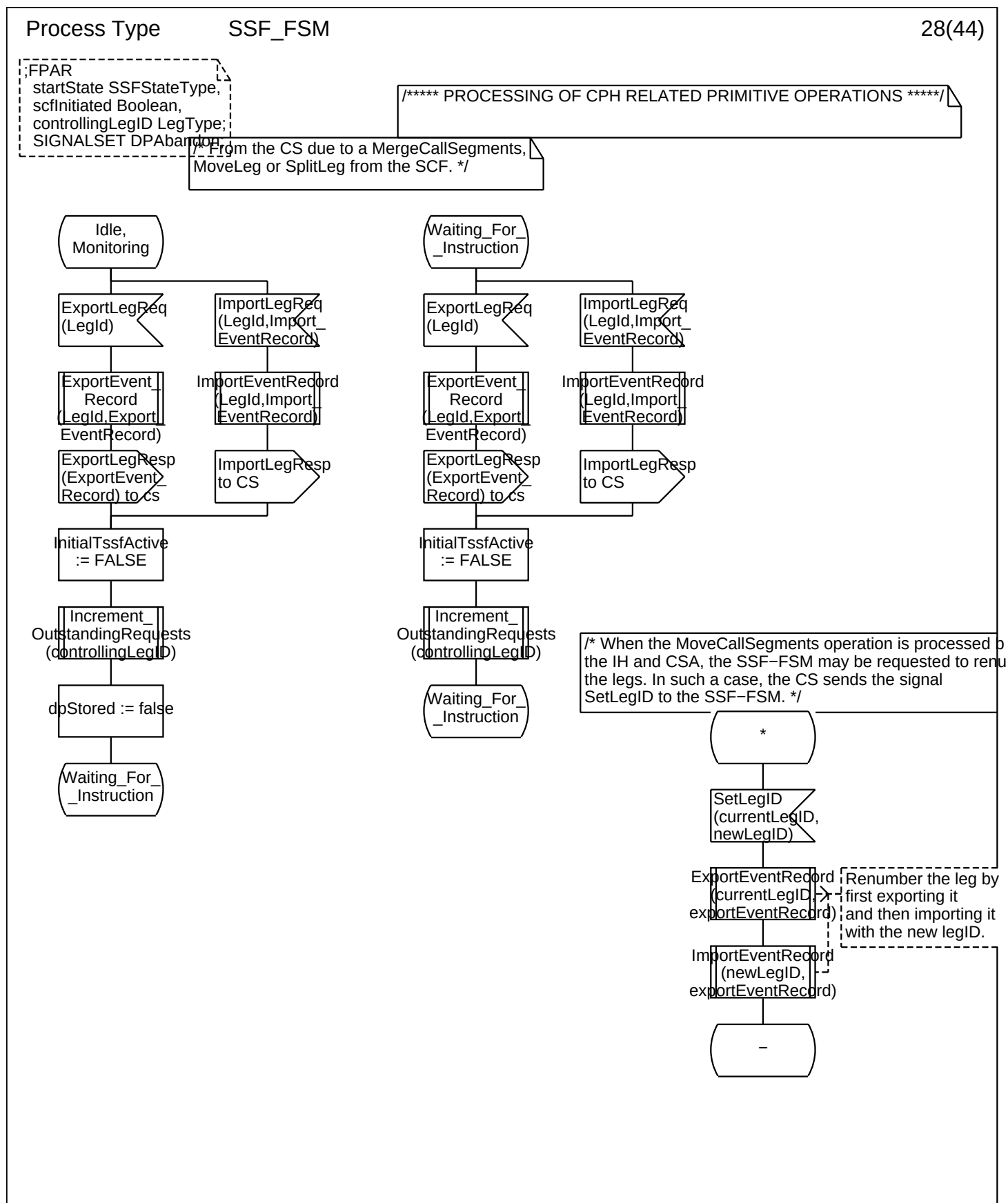
/* Note: DP UTSI used to report

UTSI events. UTSI events can be

reported in the Monitoring and WFI

states. */

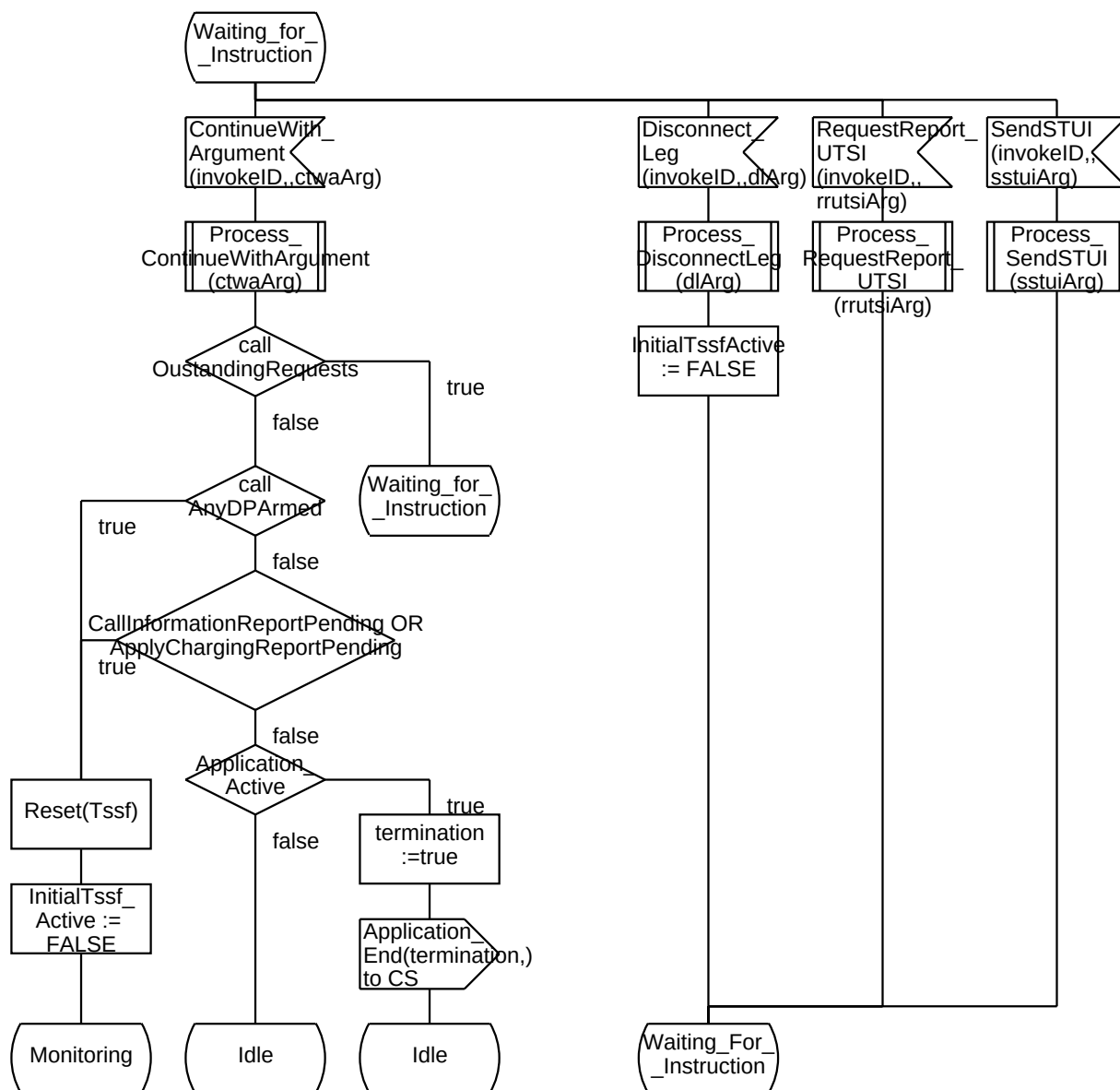




Process Type SSF_FSM

29(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;



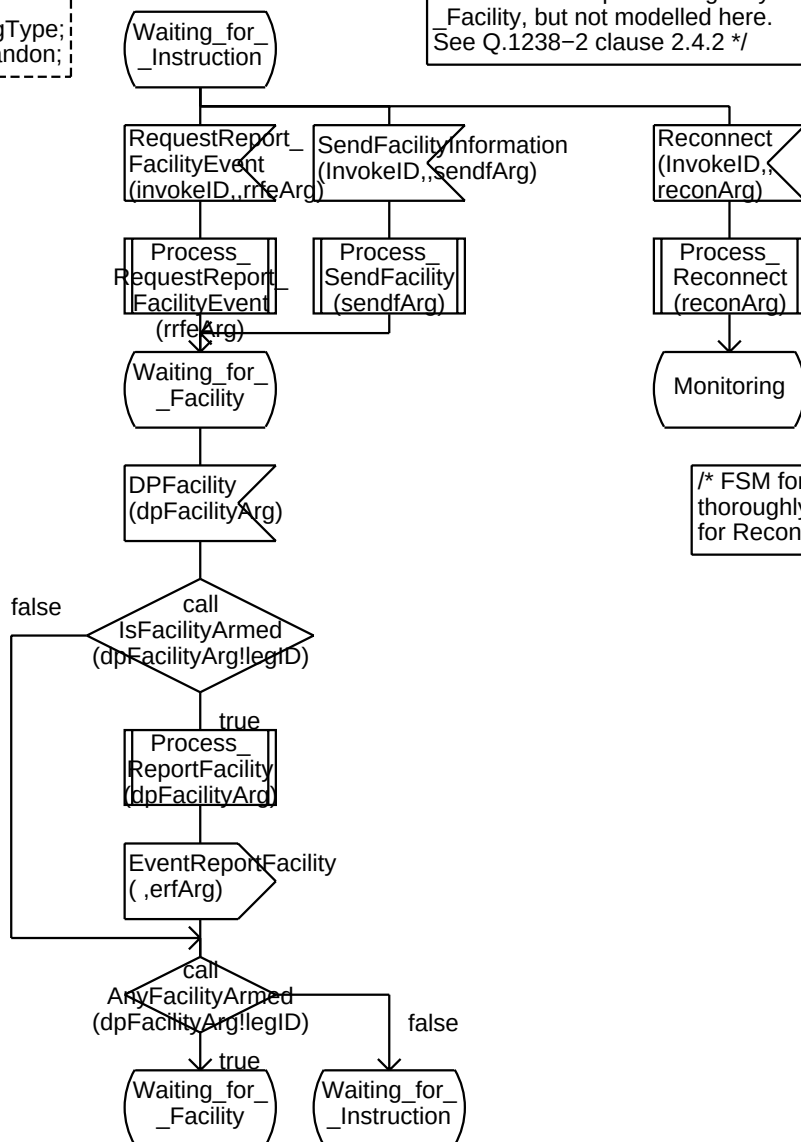
Process Type

SSF_FSM

30(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

/* State Waiting_For_Facility and Facility events not properly modelled as no explanation given as to what constitutes a Facility event. Further operations which do not resume call processing may be received in Waiting_For_Facility, but not modelled here. See Q.1238-2 clause 2.4.2 */



/* FSM for CS states not thoroughly clear in Q.1238-2 for Reconnect operation */

31(44)

```

;FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

```

```

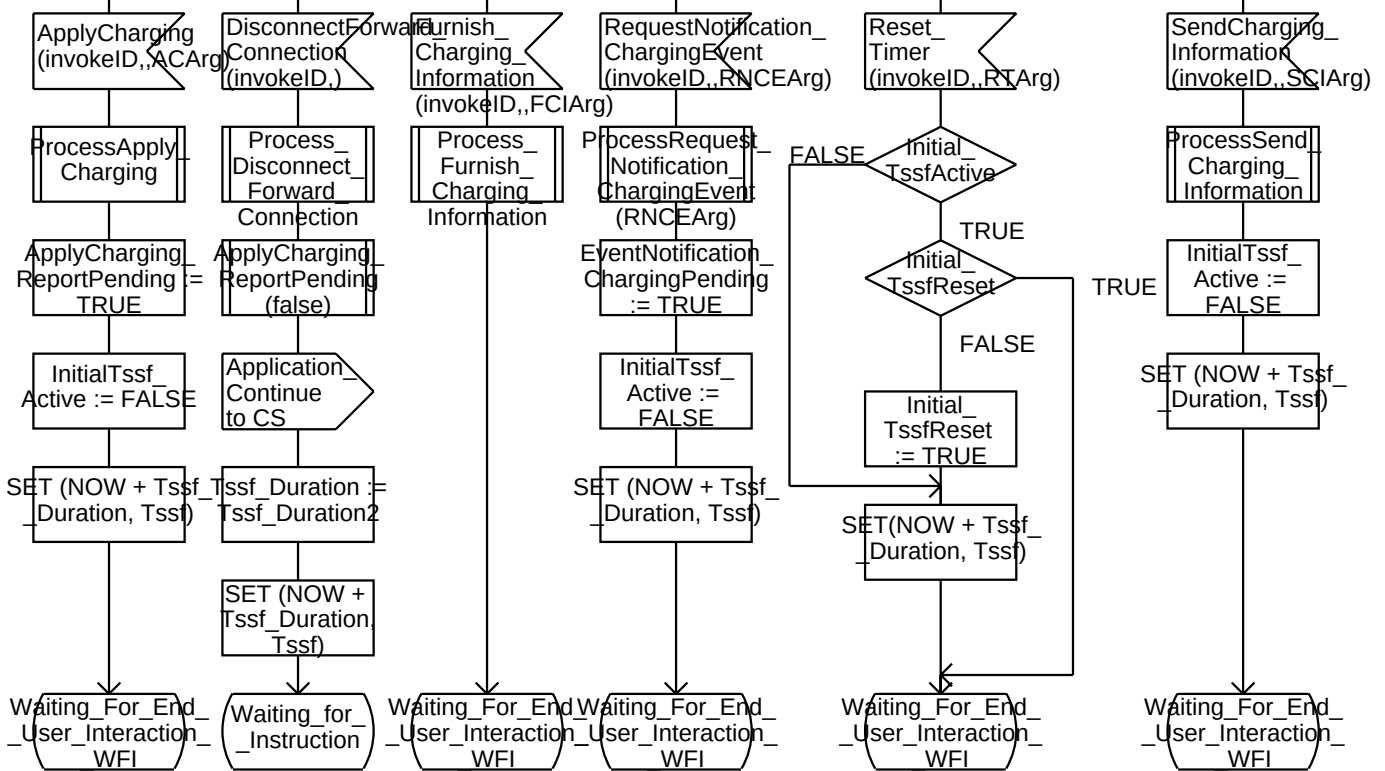
/***** WAITING FOR END OF USER INTERACTION (WFI) STATE *****/

```

```

graph LR
    WFI{{WFI  
Waiting_For_End  
_User_Interaction}} -.- SSF[SSF related operations.]
  
```

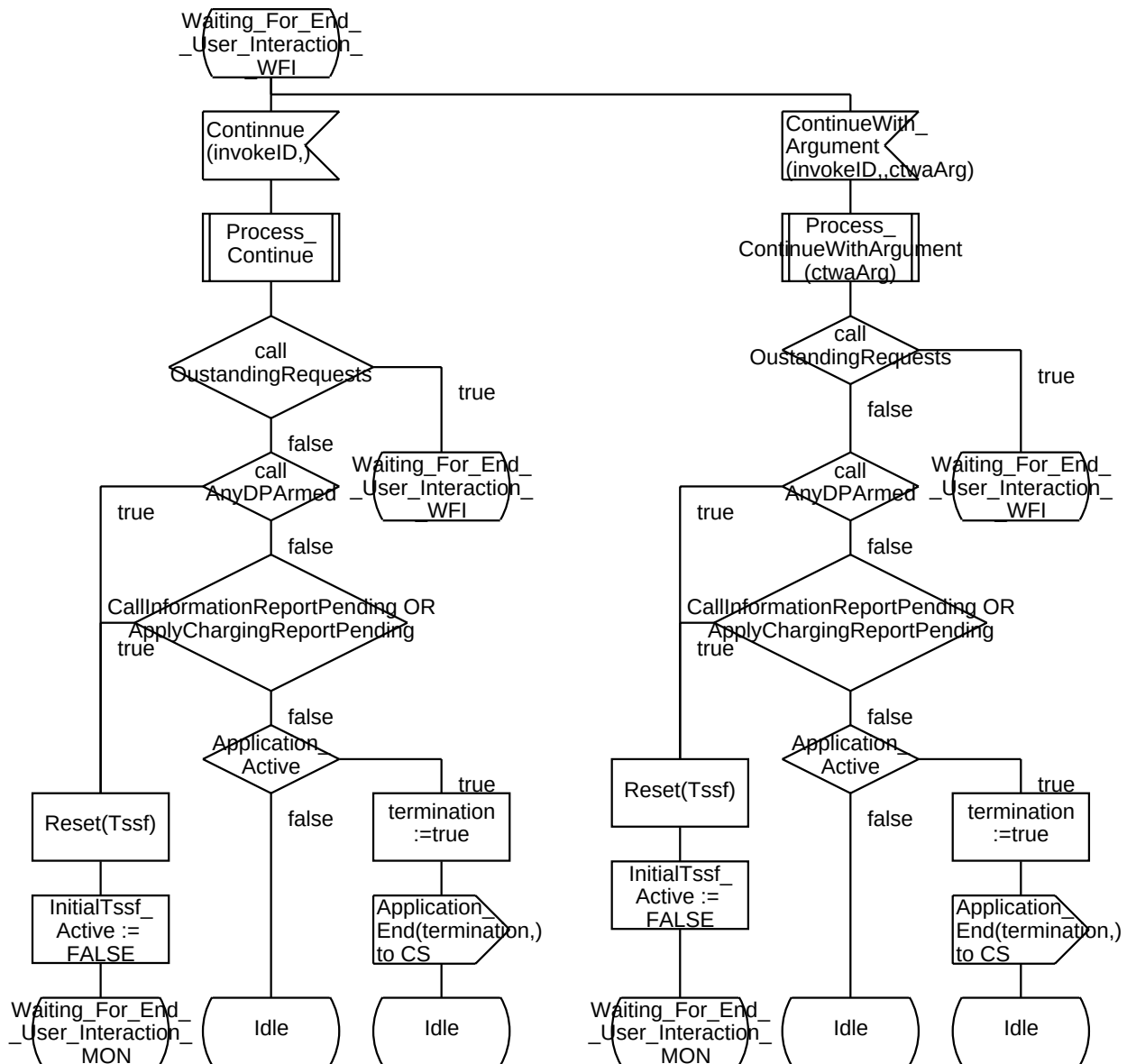
```
/* Note: ResetTimer is optional in this state. */
```



Process Type SSF_FSM

32(44)

FPAR
 startState SSFStateType,
 scfInitiated Boolean,
 controllingLegID LegType;
 SIGNALSET DPAbandon;

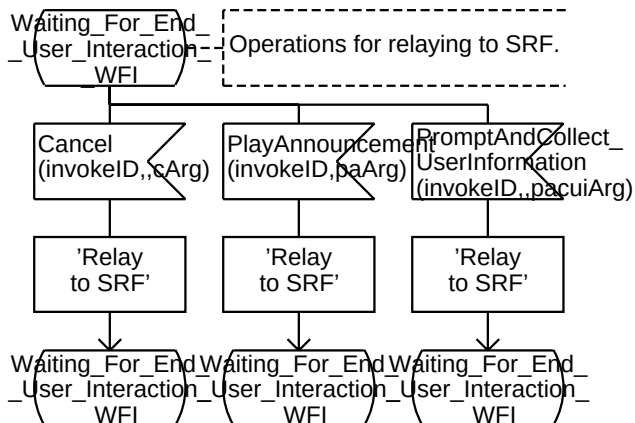


Process Type SSF_FSM

```

;FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

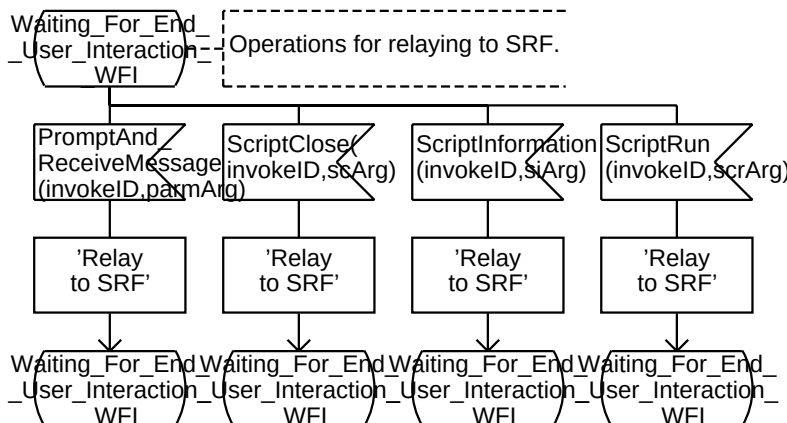
```



/* Note: The following events received from the SRF shall be relayed to the SCF:

- PromptAndCollectUserInformationResult
- SpecializedResourceReport

*/



/* Note: The following events received from the SRF shall be relayed to the SCF:

- PromptAndReceiveMessageResult
- ScriptEvent

*/

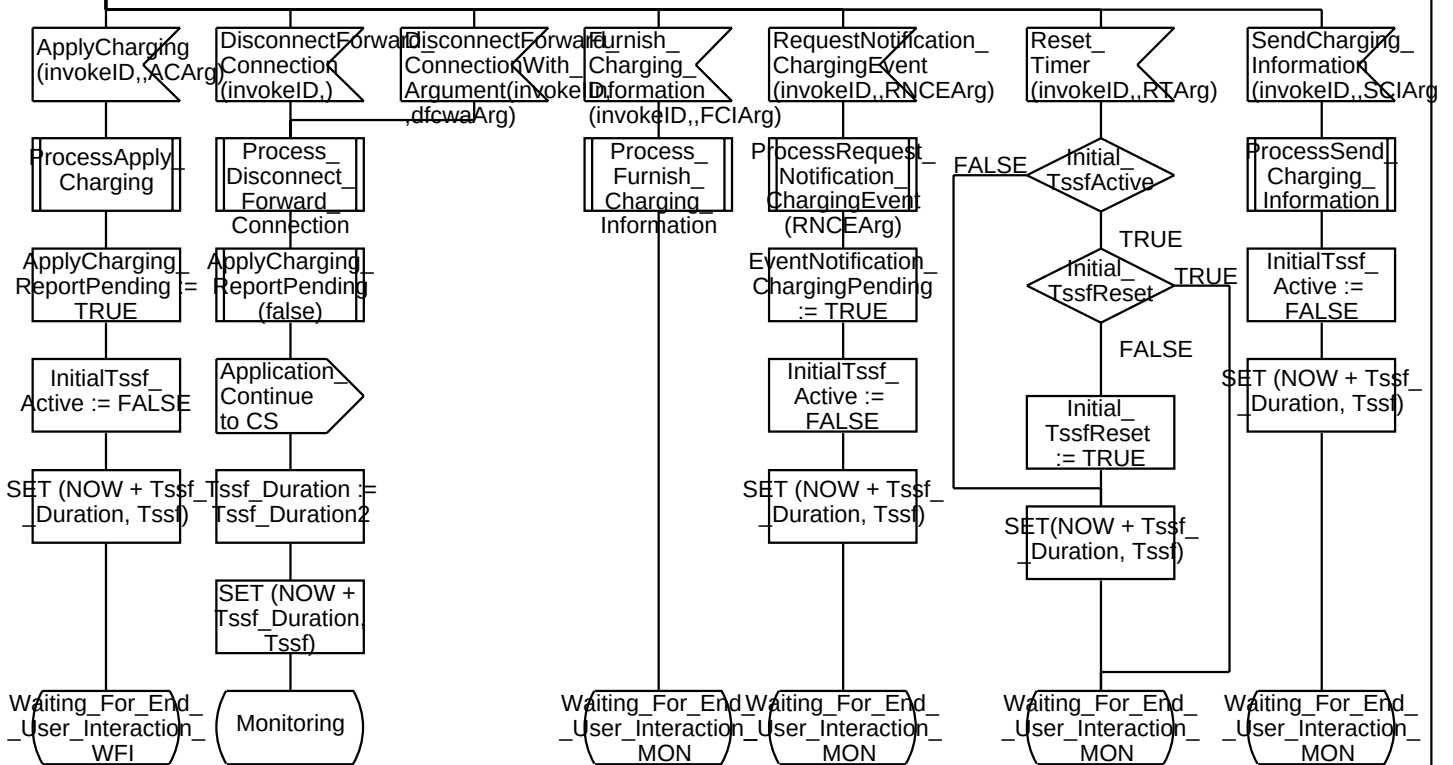
34(44)

```

/***** WAITING FOR END OF USER INTERACTION (MON) STATE *****/

```

```
/* Note: ResetTimer is optional in this state. */
```



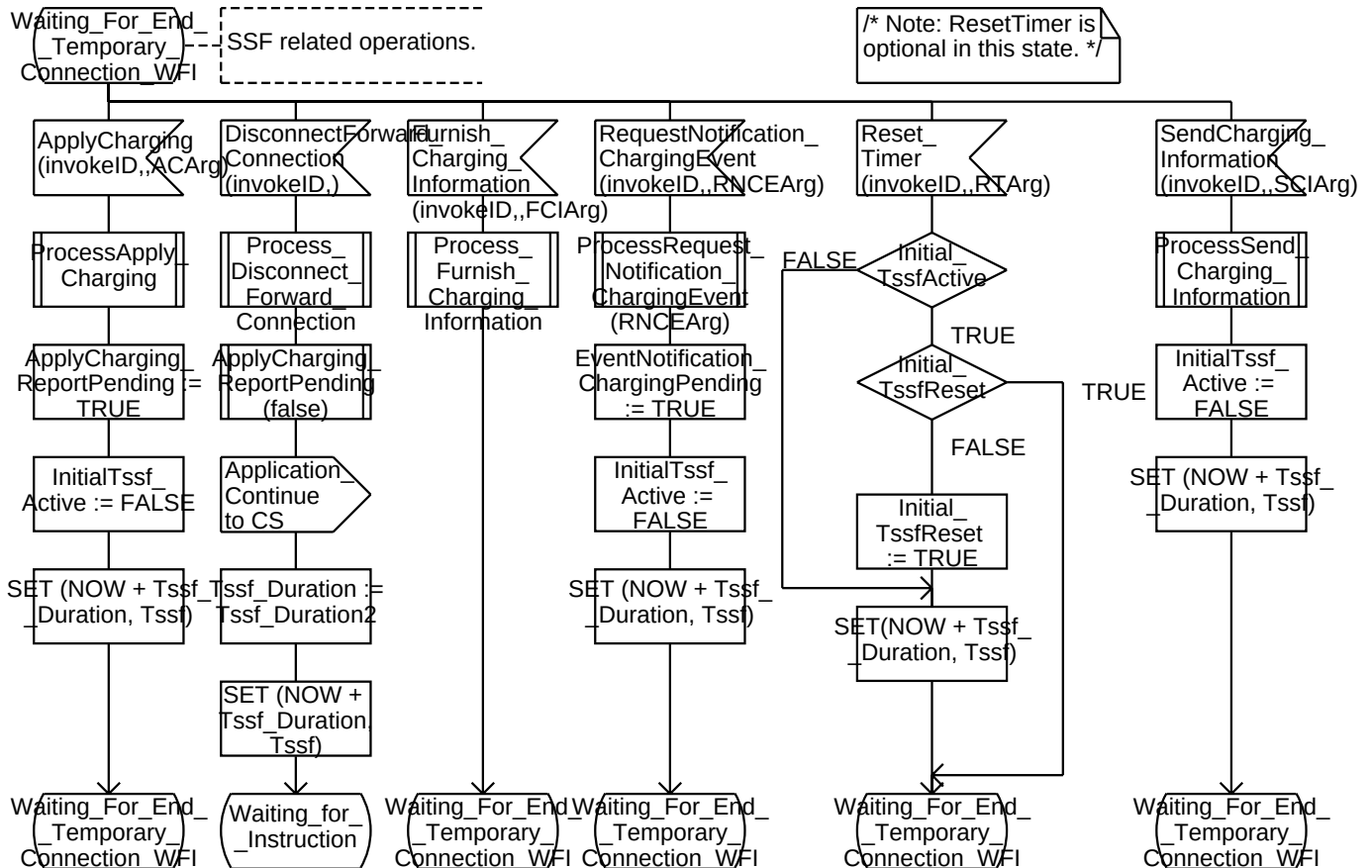
Process Type

SSF_FSM

35(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

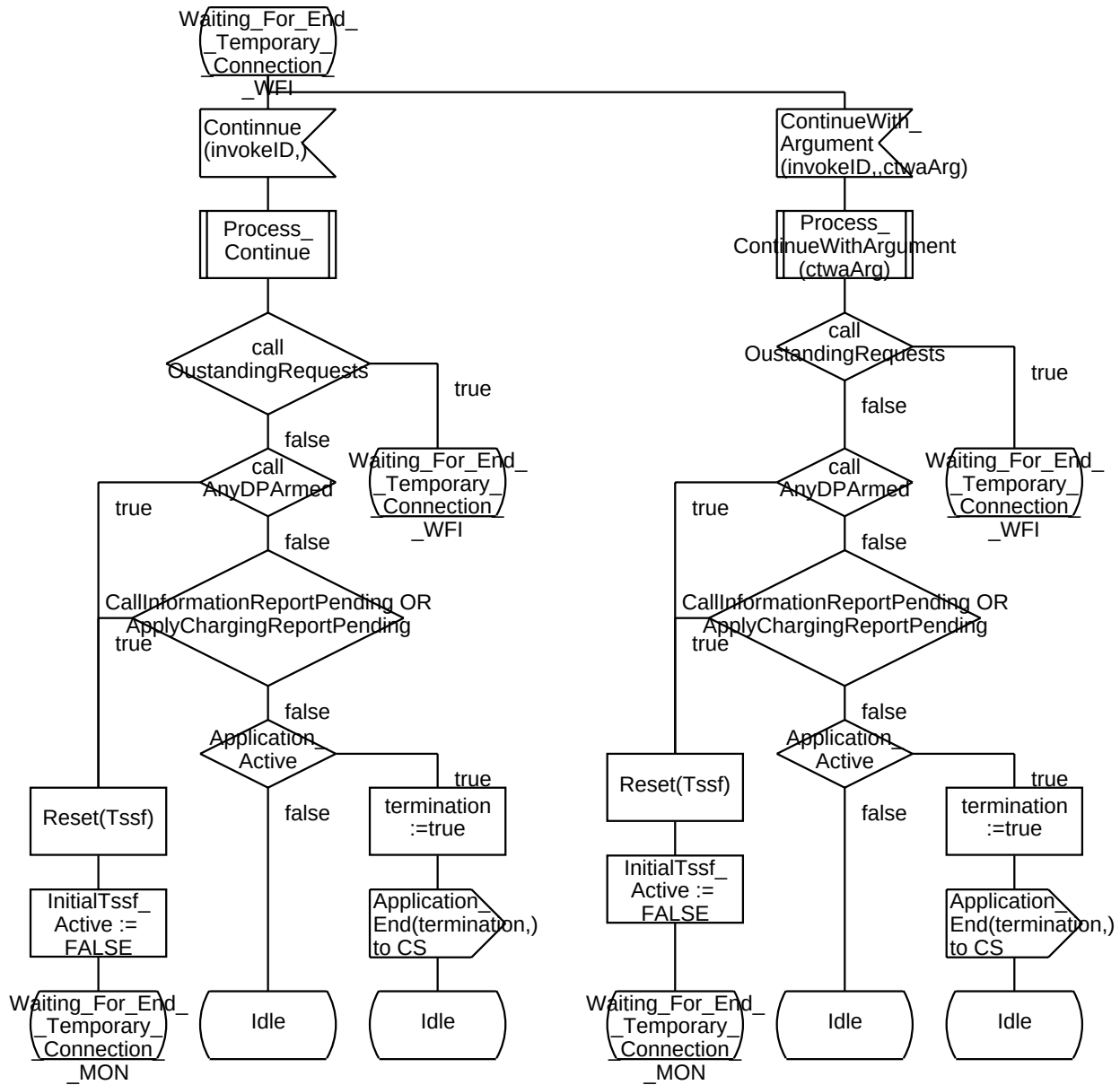
/***** WAITING FOR END OF TEMPORARY CONNECTION (WFI) STATE *****/

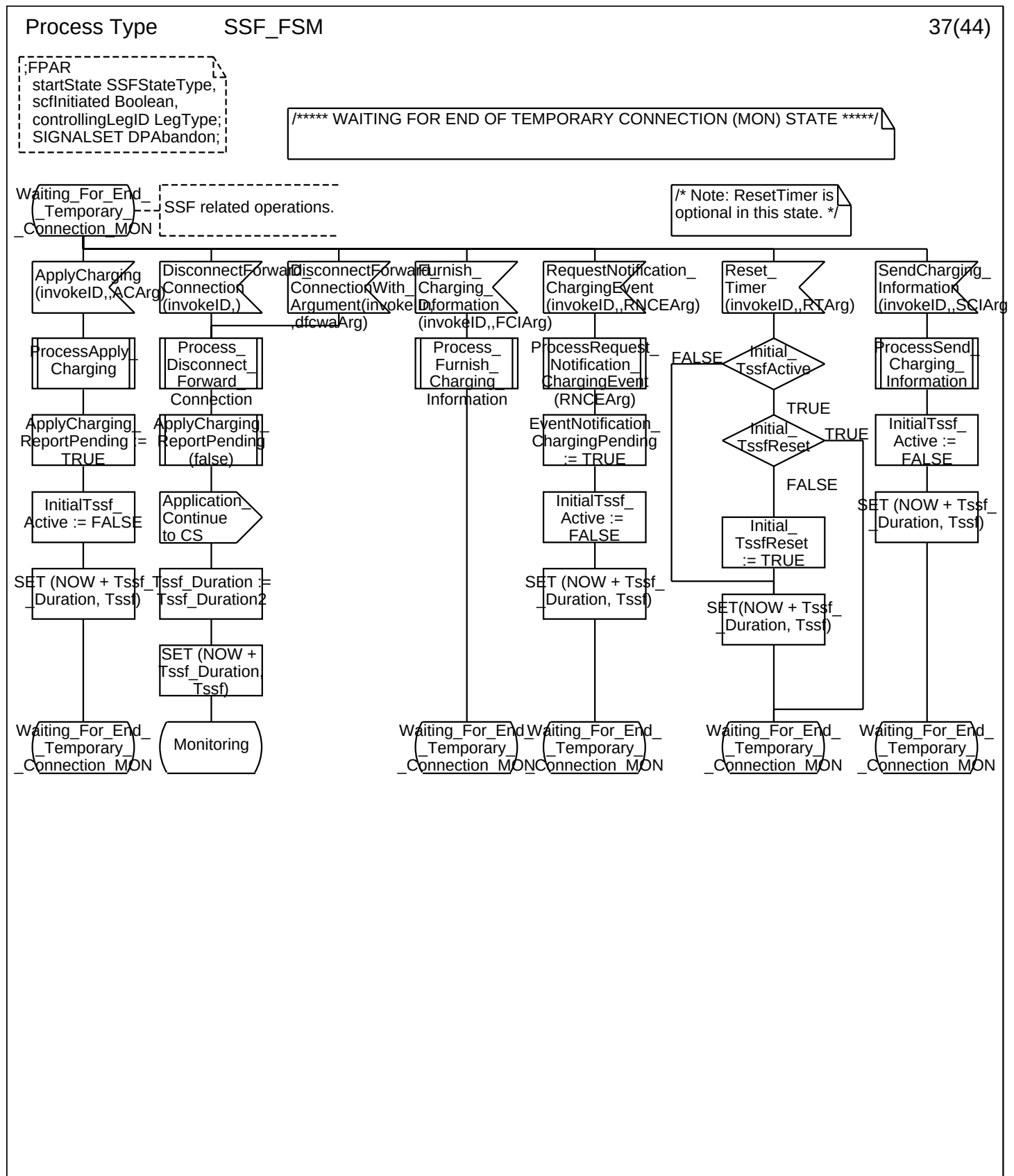


Process Type SSF_FSM

36(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;



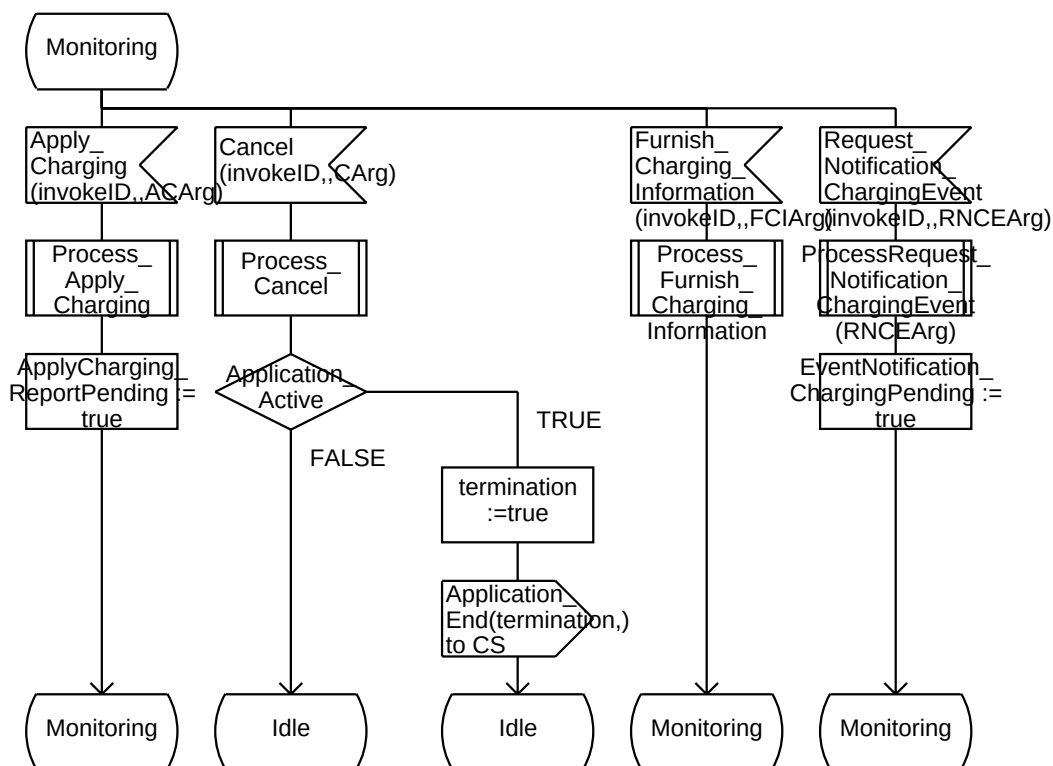


Process Type SSF_FSM

38(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

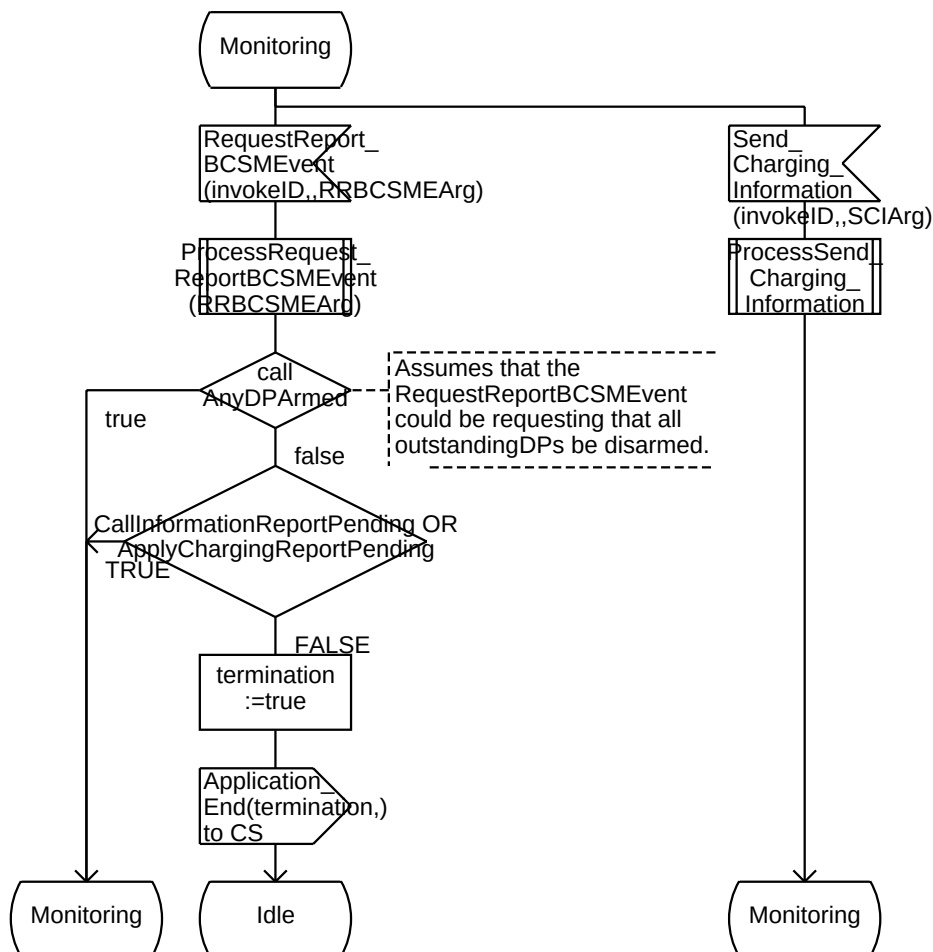
/***** MONITORING STATE *****/

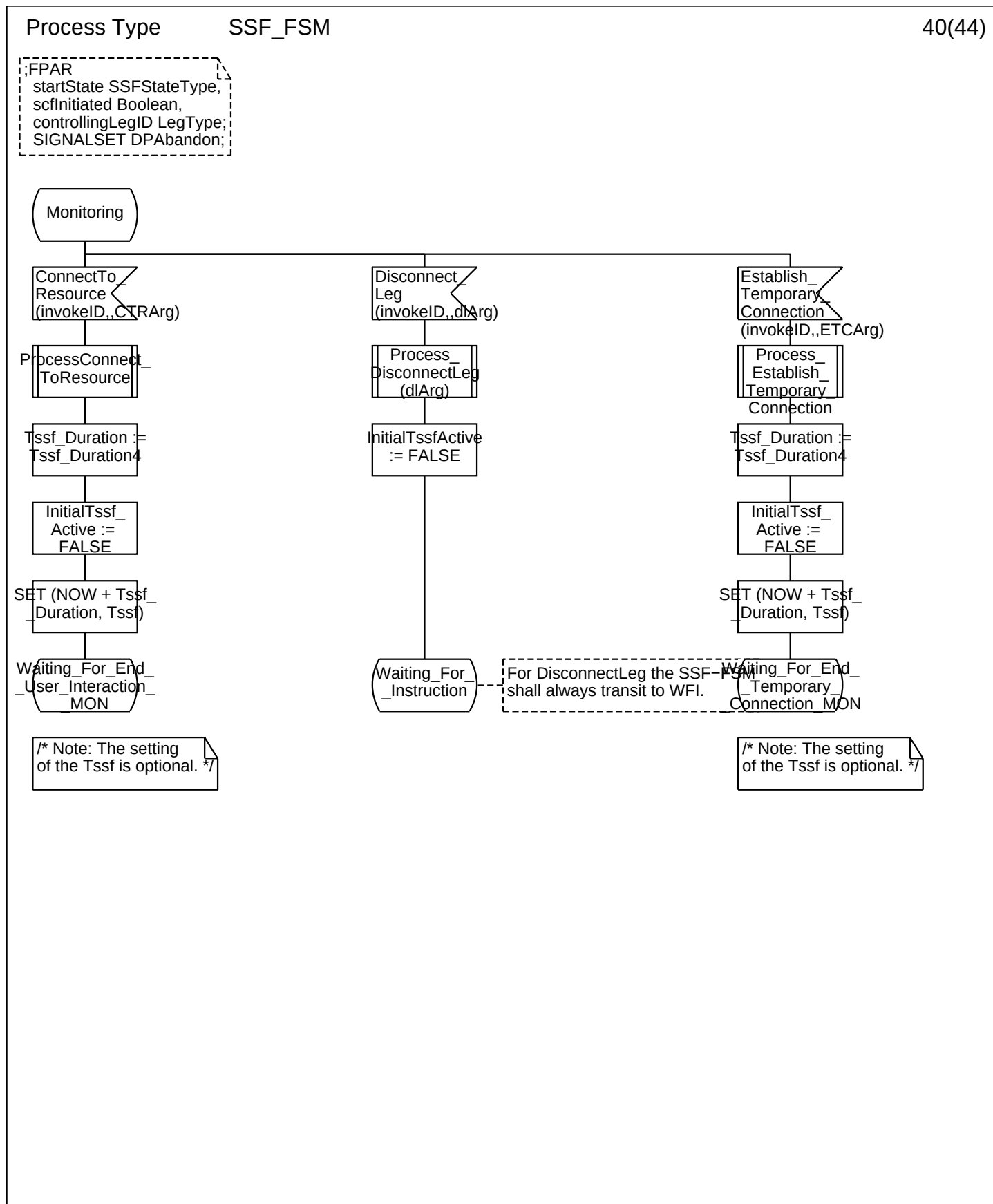


Process Type SSF_FSM

39(44)

FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;





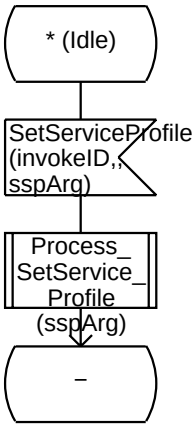
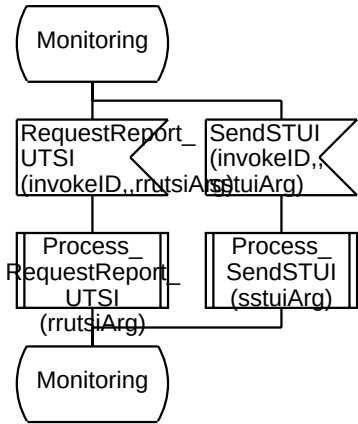
;FPAR

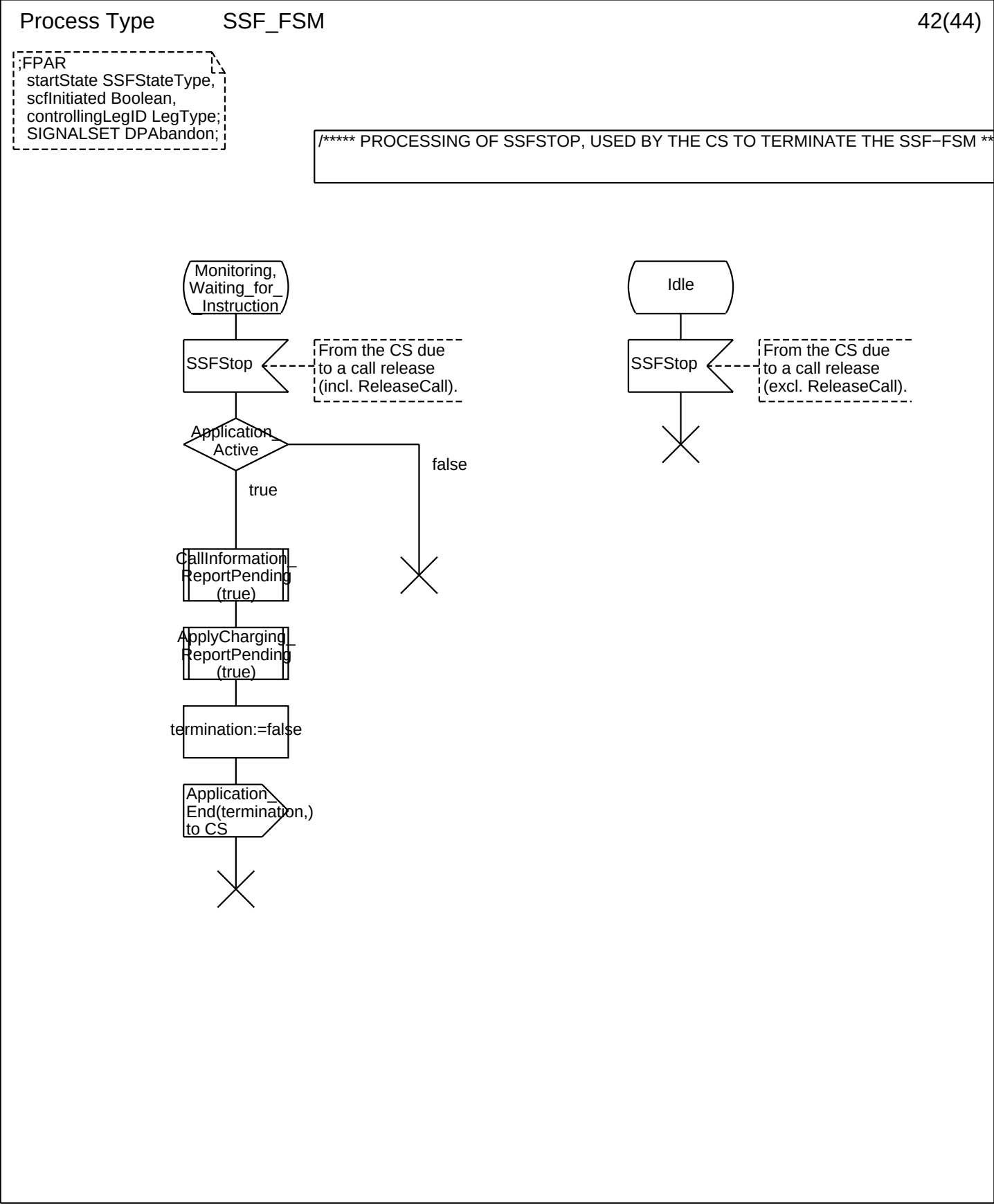
startState SSFStateType,

scfInitiated Boolean,

controllingLegID LegType;

SIGNALSET DPAbandon;





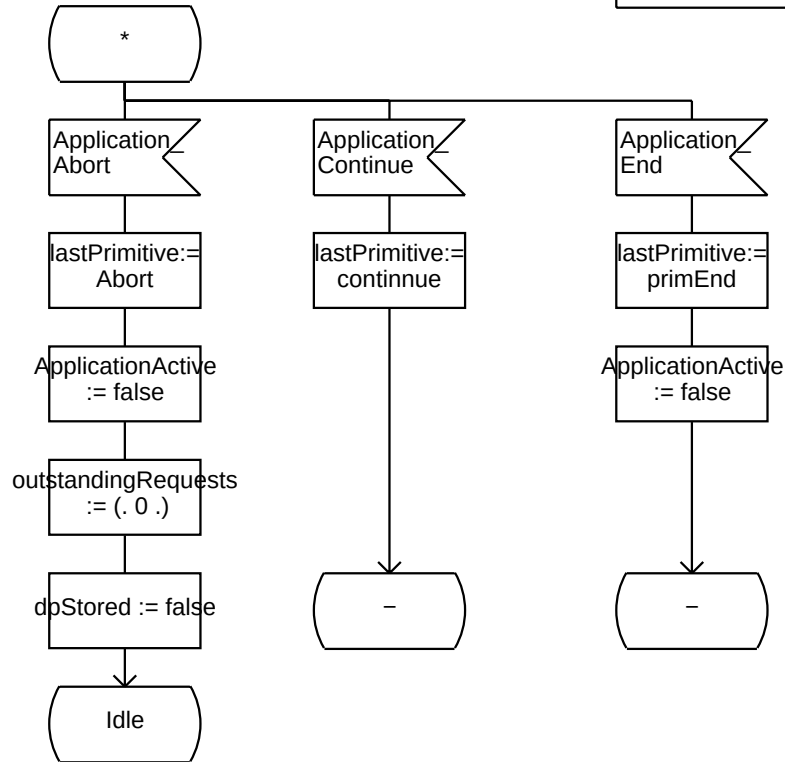
Process Type SSF_FSM

43(44)

FFPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

/* ***** PROCESSING OF APPLICATION PRIMITIVES ***** */

/*
Note that only the relevant application
primitives are defined in the SSF FSM.
*/

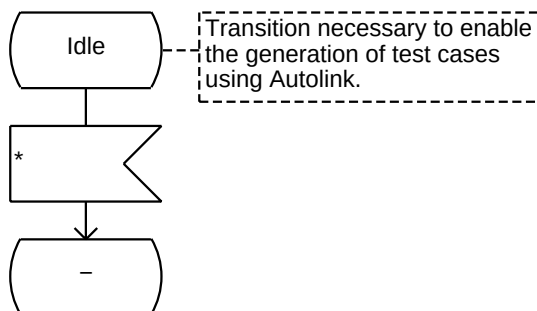
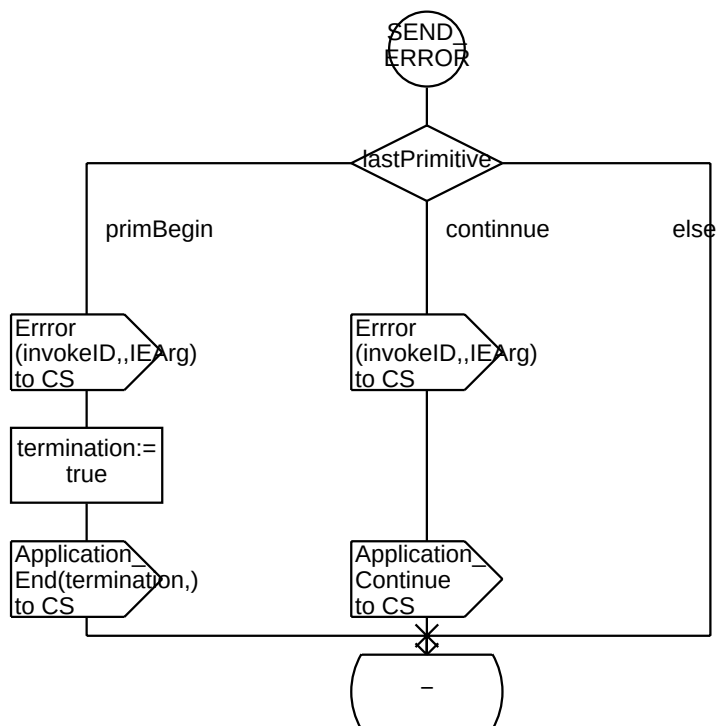


Process Type SSF_FSM

44(44)

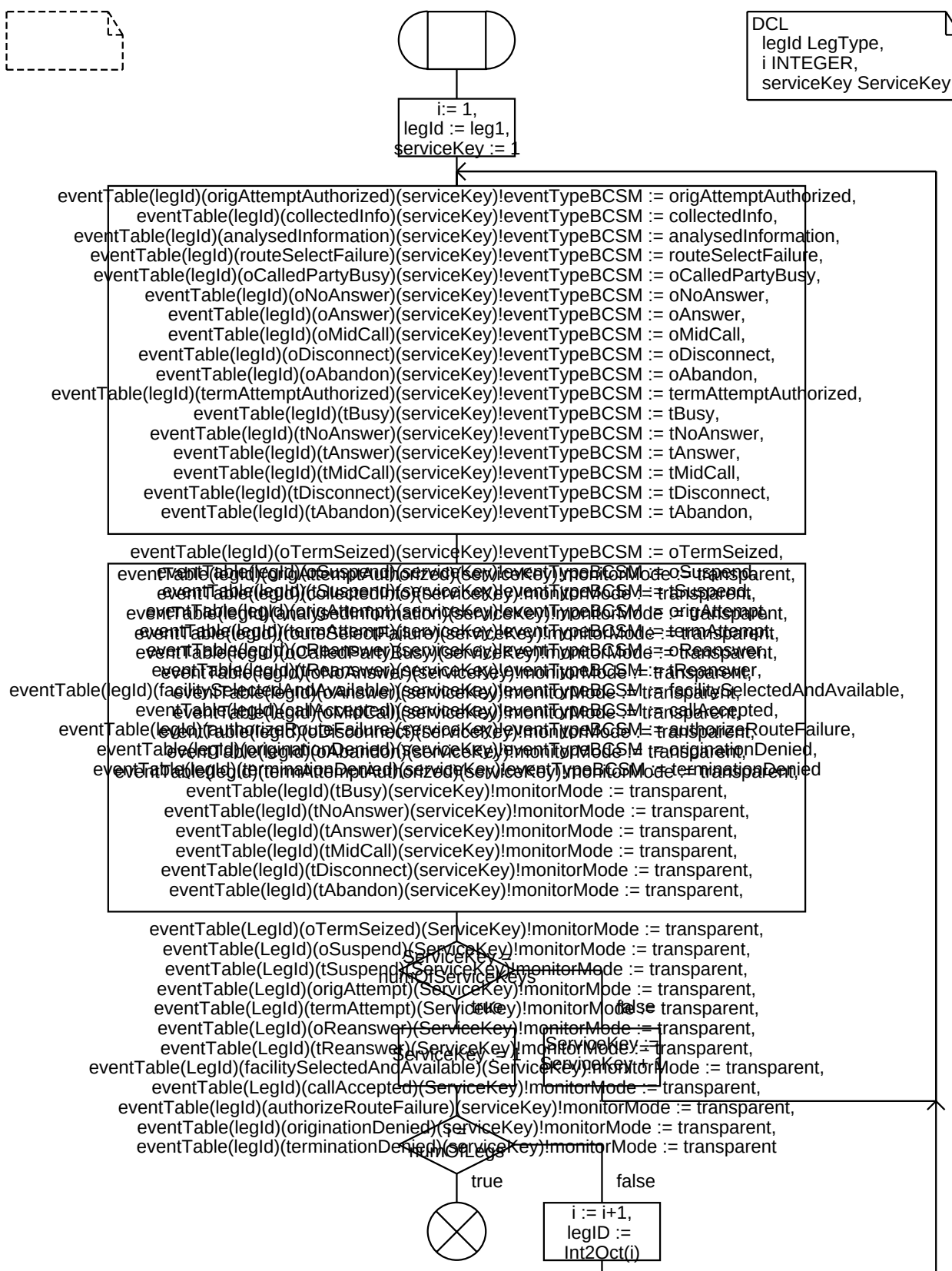
FPAR
startState SSFStateType,
scfInitiated Boolean,
controllingLegID LegType;
SIGNALSET DPAbandon;

/***** SENDING OF ERROR PRIMITIVE *****/



Procedure InitialiseDPTTable

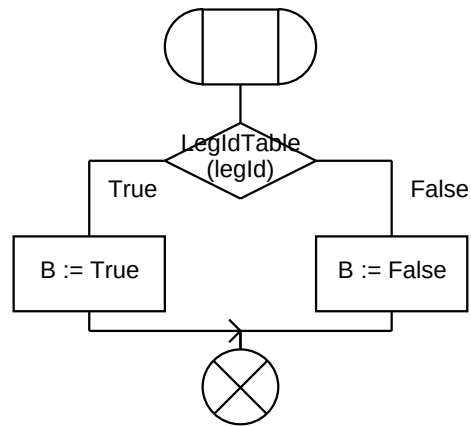
1(1)



Procedure <<Process Type SSF_FSM>> ExistLeg

1(1)

;FPAR
IN legId LegType;
RETURNS B Boolean;

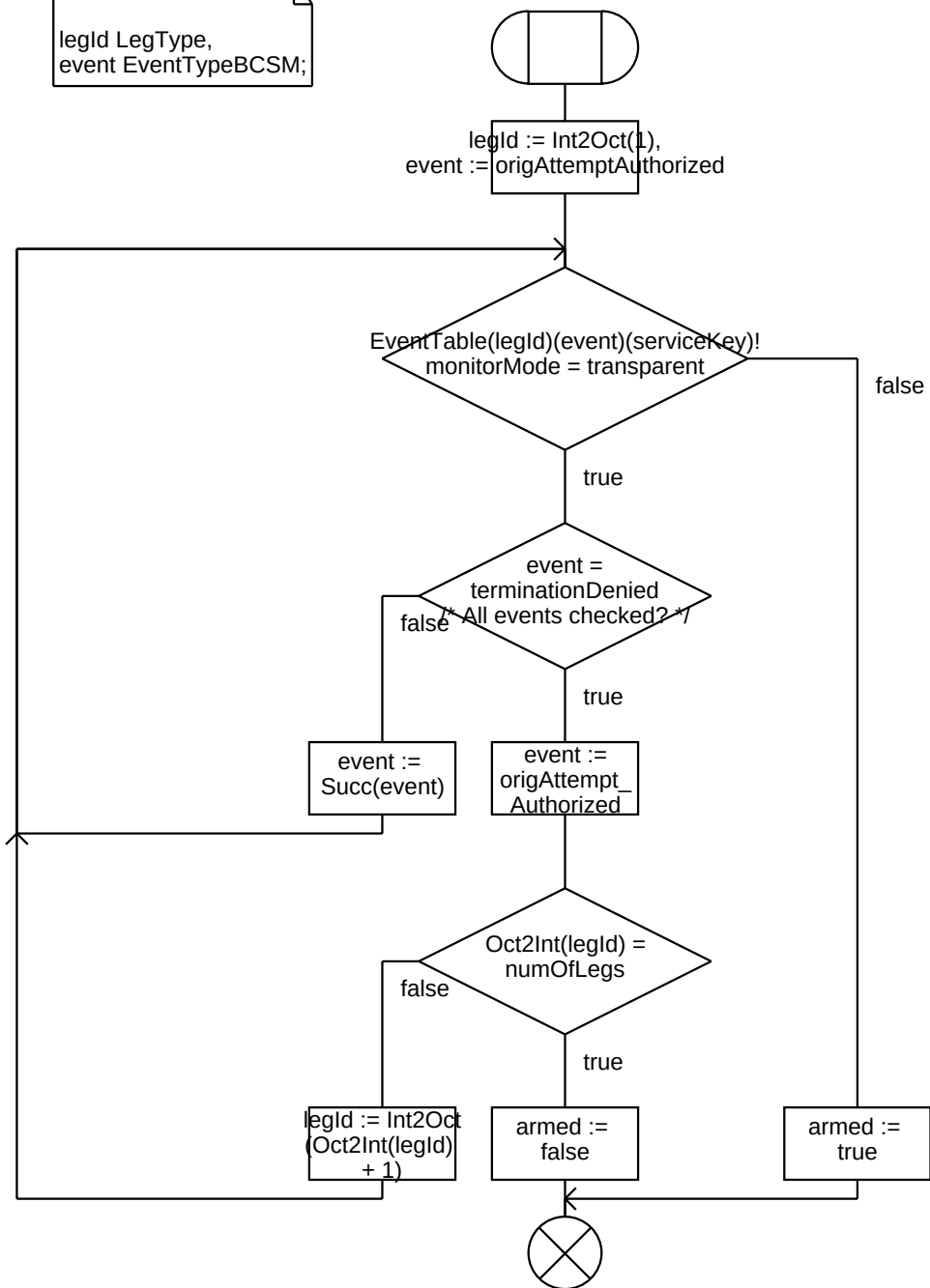


Procedure AnyDPArmed

1(1)

returns
armed Boolean;

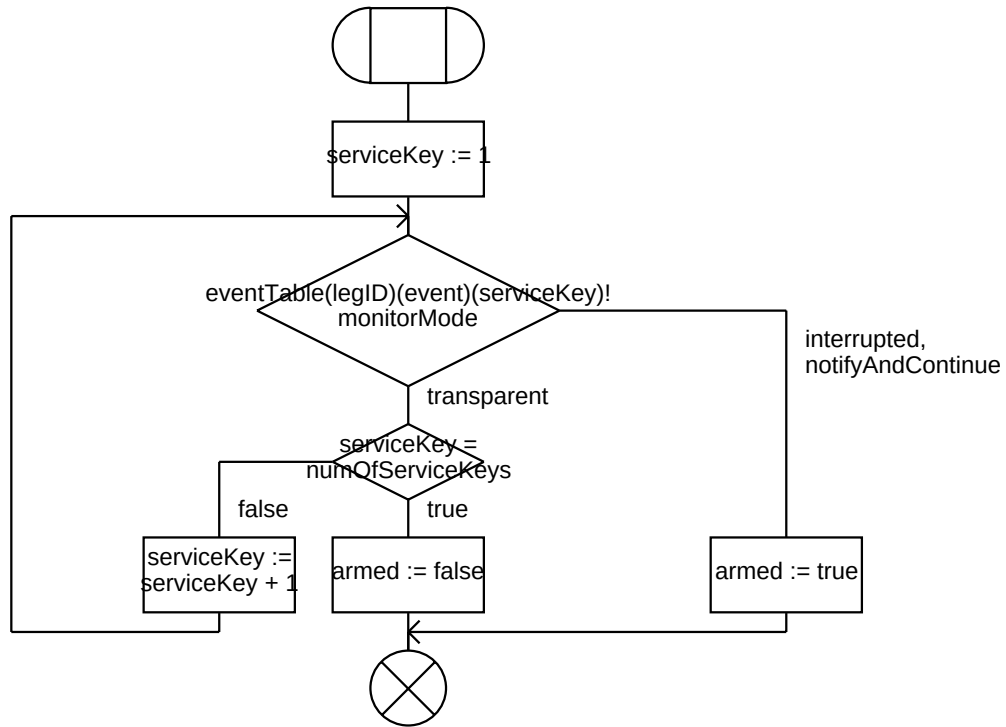
DCL
legId LegType,
event EventTypeBCSM;



Procedure IsDPArmed

1(1)

FPAR
IN legID LegType,
IN event EventTypeBCSM;
returns armed Boolean;

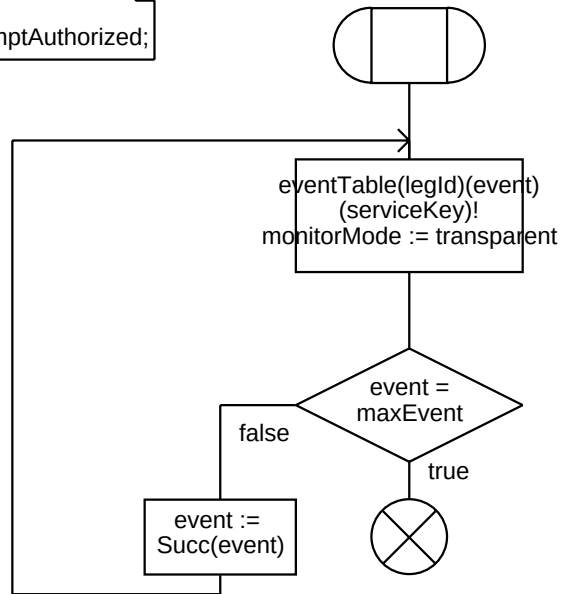


Procedure DisarmDPs

1(1)

fpar
in legId LegType,
in maxEvent EventTypeBCSM;

DCL
event EventTypeBCSM := origAttemptAuthorized;



Procedure DPArmed

1(1)

```

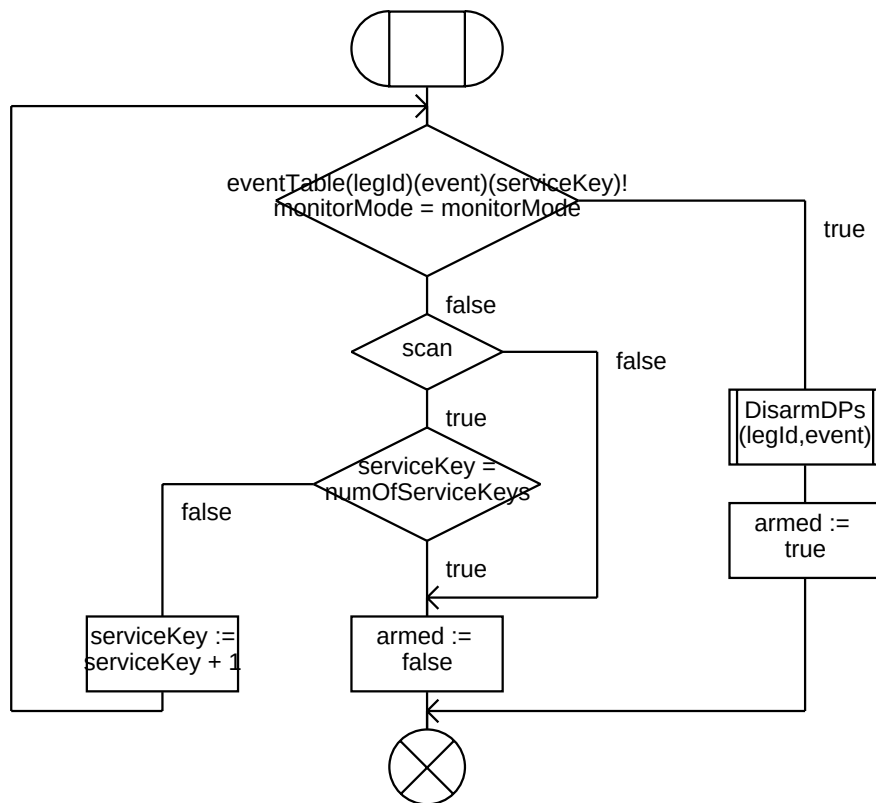
fpar
in legId LegType,
in event EventTypeBCSM,
in monitorMode MonitorMode,
in scan Boolean;
returns armed Boolean

```

```

/*
The procedure searches for the first
armed instance of the DP. If an armed
instance is encountered the instance
is disarmed and the procedure returns
true, else the procedure returns false.
*/

```

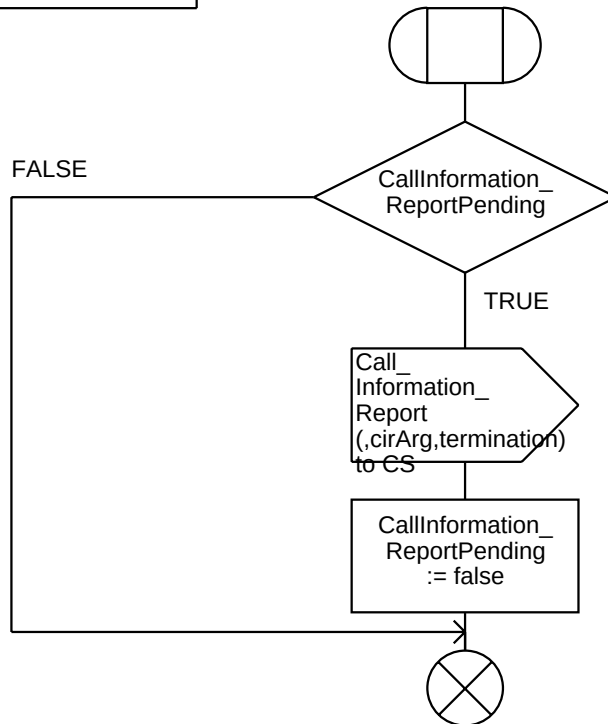


Procedure CallInformationReportPending

1(1)

;Fpar
In termination Boolean;

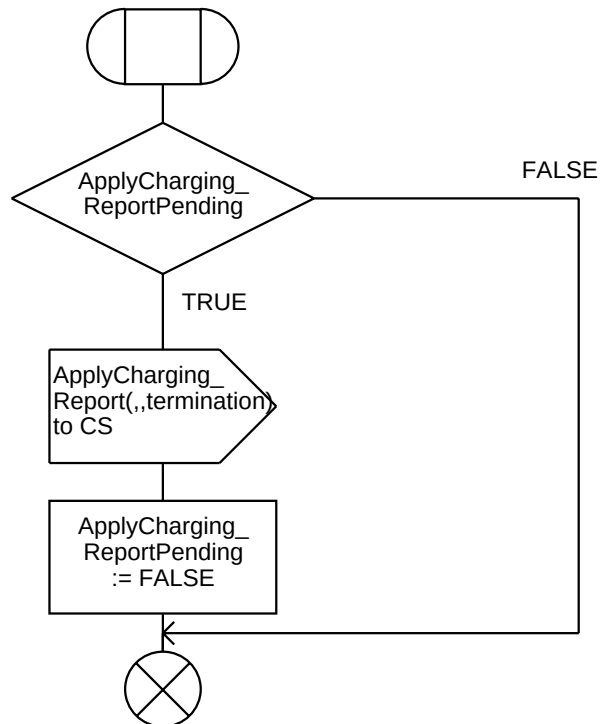
DCL
CIRArg CallInformationReportArg;



Procedure ApplyChargingReportPending

1(1)

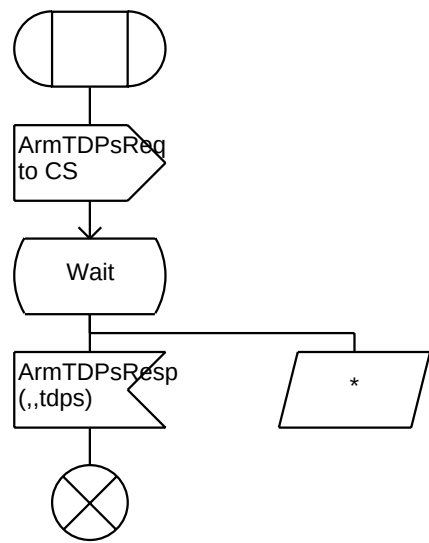
;Fpar
In termination Boolean;



Procedure <<Process Type SSF_FSM>> ArmTDPs

1(1)

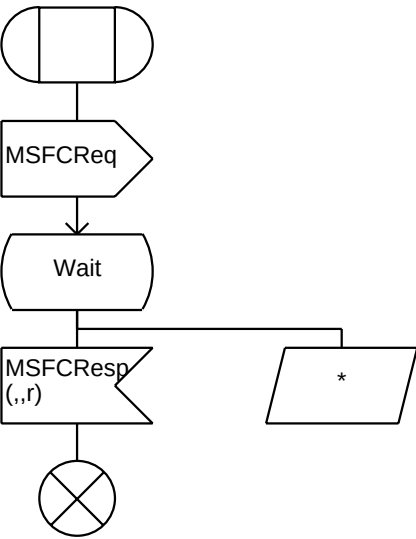
;returns tdp EventTableType;



Procedure <<Process Type SSF_FSM>> MatchingServiceFilteringCriteria

1(1)

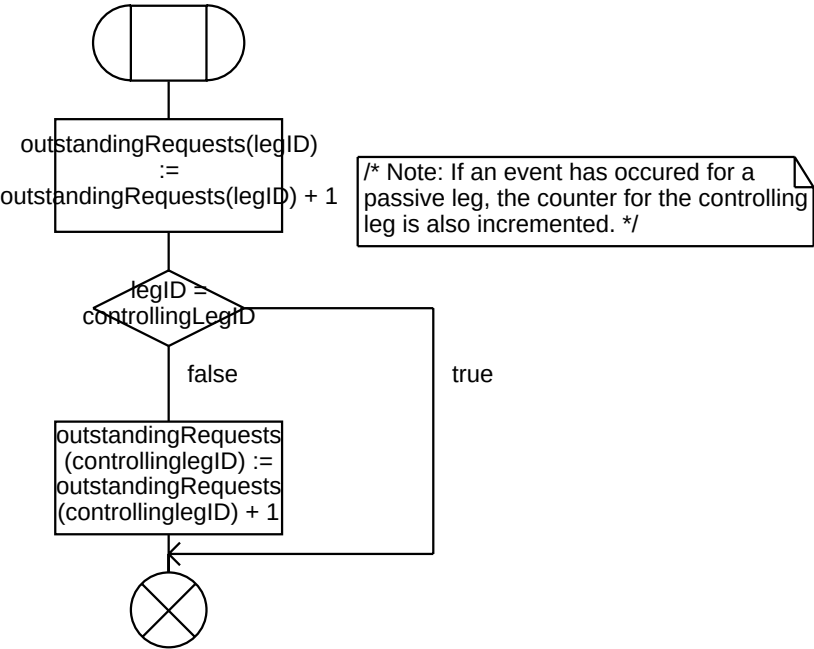
;returns r Boolean;



Procedure IncrementOutstandingRequests

1(1)

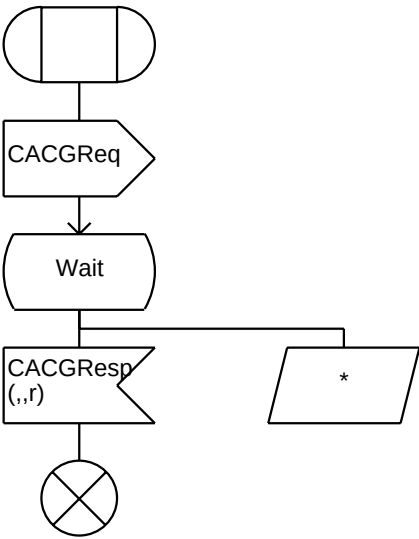
;FPAR
IN legID LegType;



Procedure <<Process Type SSF_FSM>> CheckACG

1(1)

;returns r Boolean;

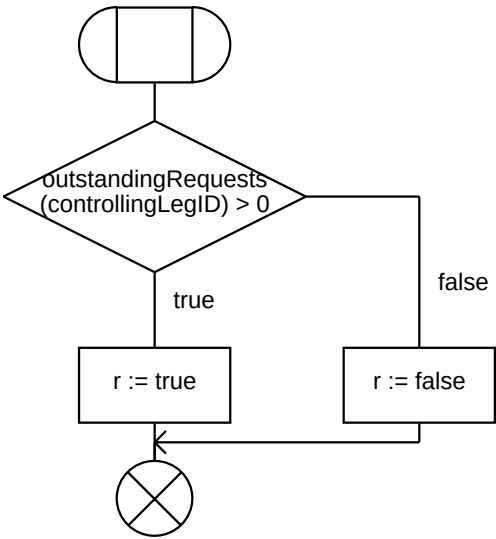


Procedure OutstandingRequests

1(1)

;returns r Boolean;

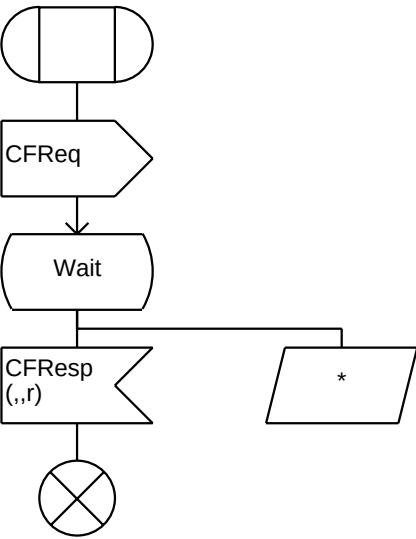
DCL
legID LegType,
i INTEGER;



Procedure <<Process Type SSF_FSM>> CallFiltered

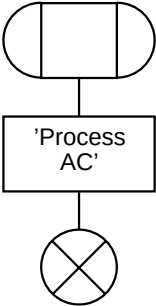
1(1)

;returns r Boolean;



Procedure ProcessApplyCharging

1(1)



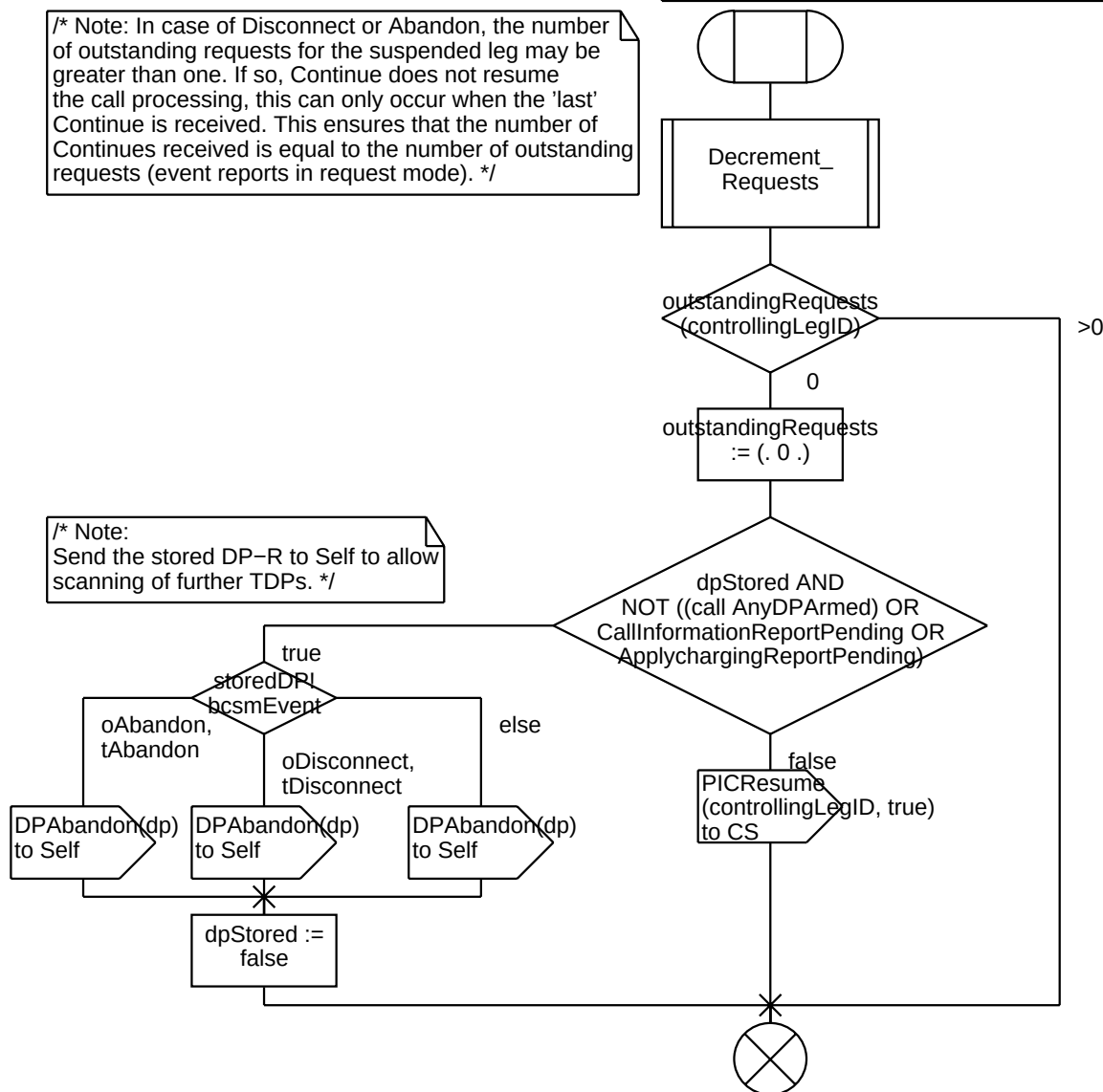
Procedure ProcessContinue

1(1)



/* Note: In case of Disconnect or Abandon, the number of outstanding requests for the suspended leg may be greater than one. If so, Continue does not resume the call processing, this can only occur when the 'last' Continue is received. This ensures that the number of Continues received is equal to the number of outstanding requests (event reports in request mode). */

/* Note:
If the SSF-FSM will transit to Idle as a result of the Continue, then the last DP-R that was processed as a TDP-R or EDP-R will be resend to the SSF-FSM to allow retriggering. */



/* Note:
Send the stored DP-R to Self to allow
scanning of further TDPs. */

Annex A: ProcessRequestReportBCSMEvent

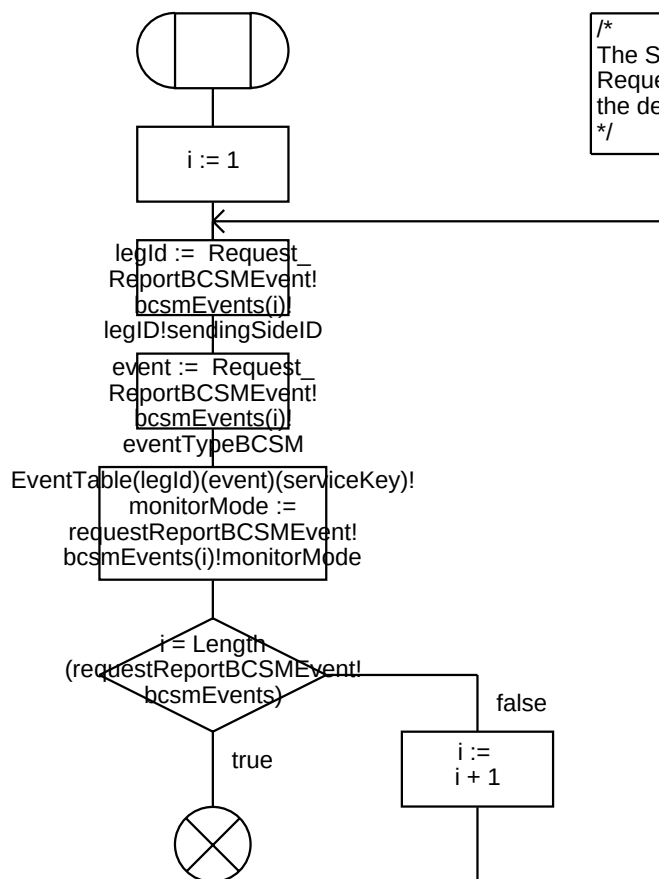
Procedure ProcessRequestReportBCSMEvent

FPAR

IN/OUT requestReportBCSMEvent RequestReportBCSMEventArg;

DCL

legId LegType,
event EventTypeBCSM,
i INTEGER;



/*

The procedure assumes that the EventReportTypes can be in a different order to the internal EventTable (Reasonable assumption).

*/

1(1)

/*

The ServiceKey is not specified in the RequestReportBCSMEvent operation. Therefore the default value is used.

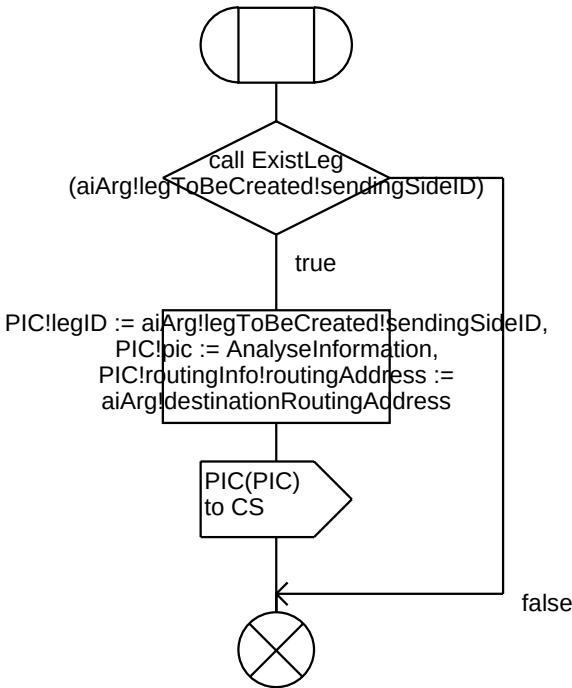
*/

Procedure ProcessAnalyseInformation

1(1)

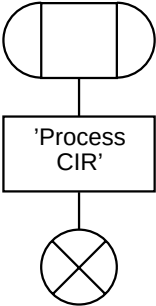
fpar
in aiArg AnalyseInformationArg;

/*
If the leg does not exist this implies
that a new BCSM has been created which
will begin processing at the AnalyseInformation PIC
PIC. The SSF does not need to resume processing.
*/



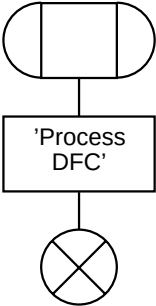
Procedure ProcessCallInformationRequest

1(1)



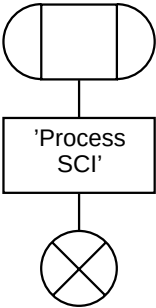
Procedure ProcessDisconnectForwardConnection

1(1)



Procedure ProcessSendChargingInformation

1(1)

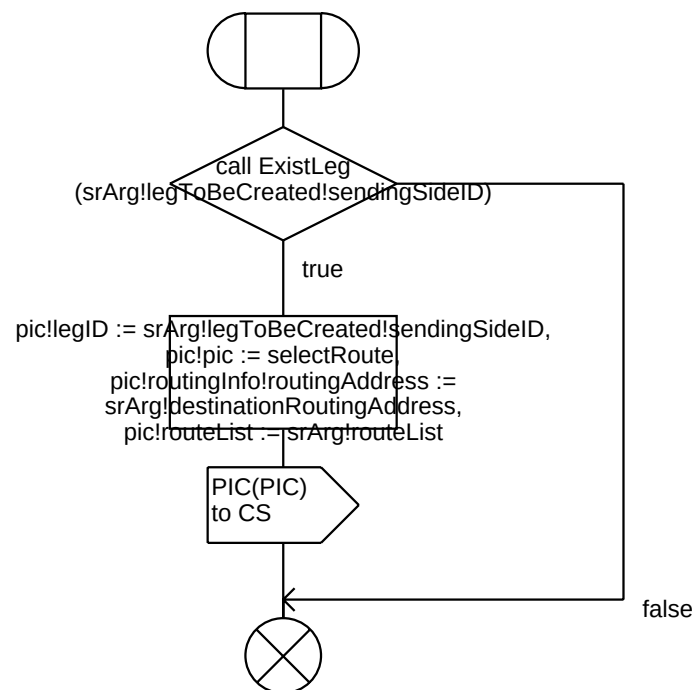


Procedure ProcessSelectRoute

1(1)

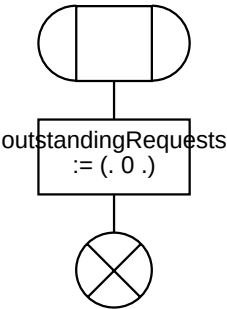
fpar
in srArg SelectRouteArg;

/*
If the leg does not exist this implies
that a new BCSM has been created which
will begin processing at the SelectRoute
PIC. The SSF does not need to resume processing.
*/



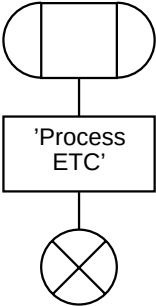
Procedure ProcessCancel

1(1)



Procedure ProcessEstablishTemporaryConnection

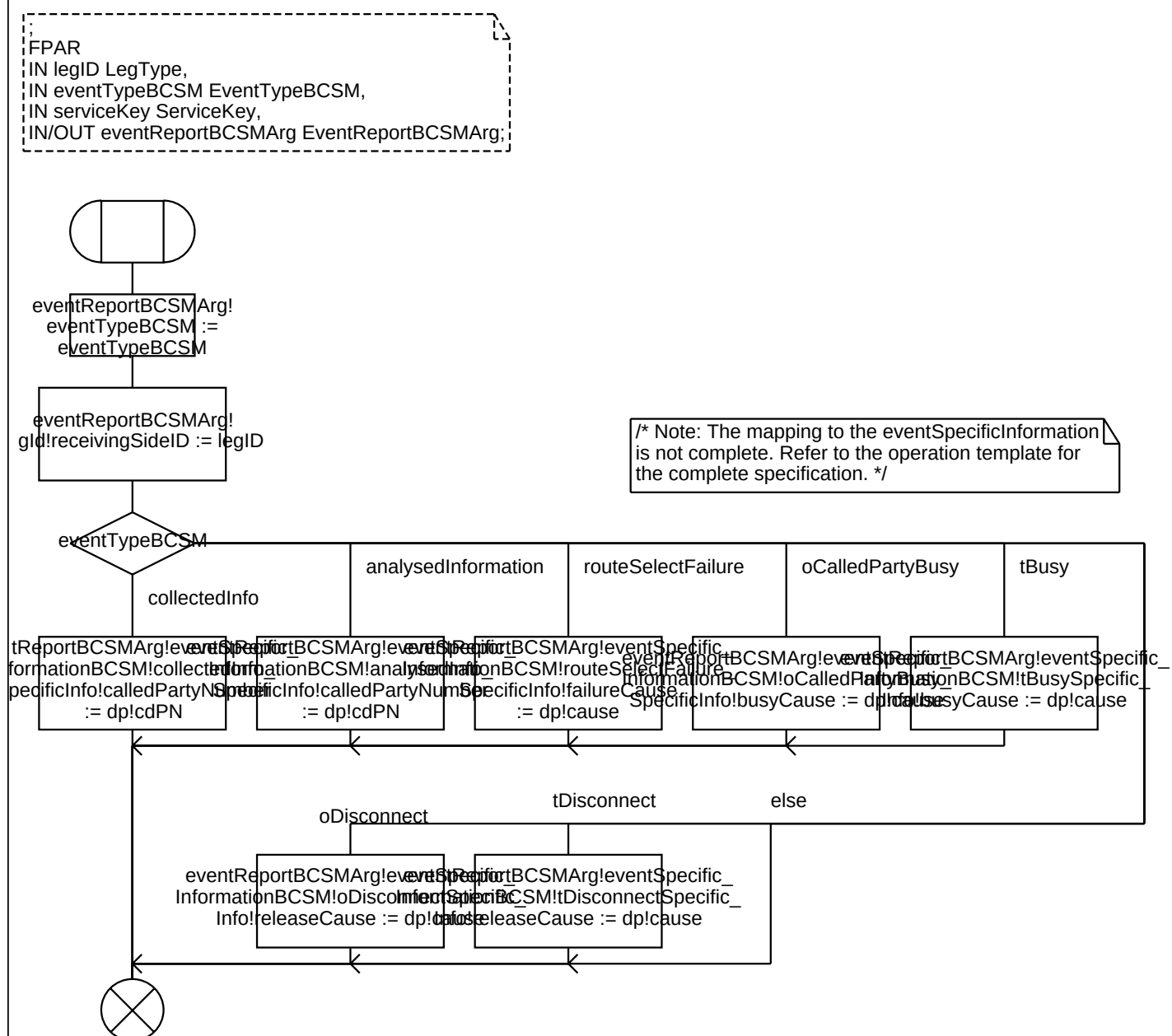
1(1)



Annex A: ProcessEventReportBCSM

Procedure ProcessEventReportBCSM

1(1)

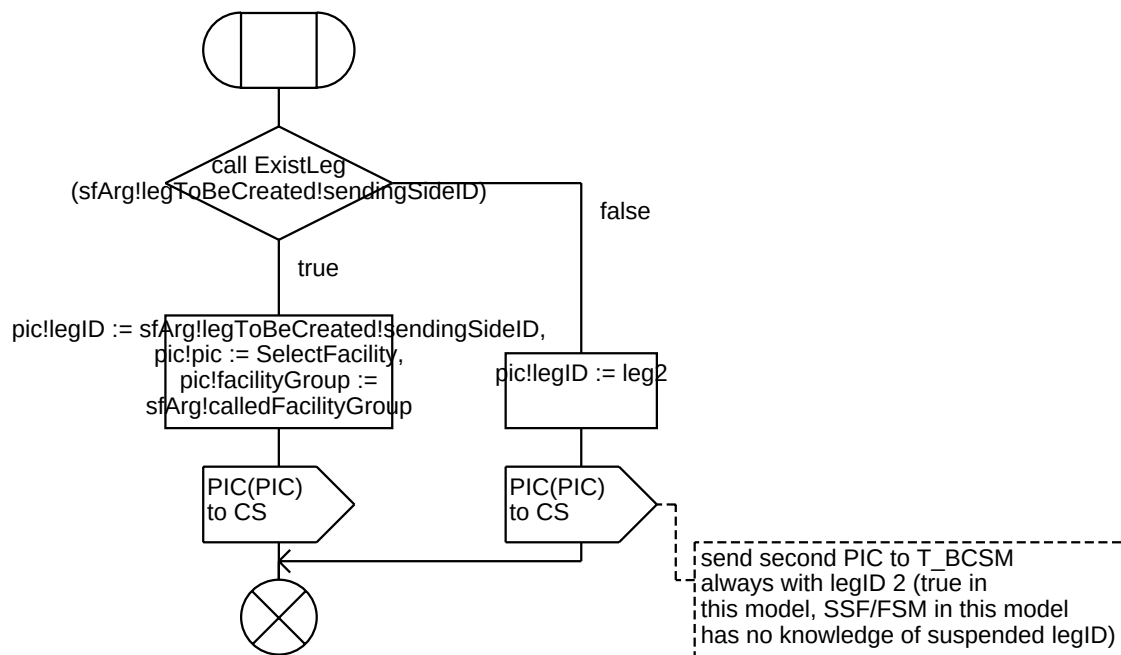


Procedure ProcessSelectFacility

1(1)

```
fpar
in sfArg SelectFacilityArg;
```

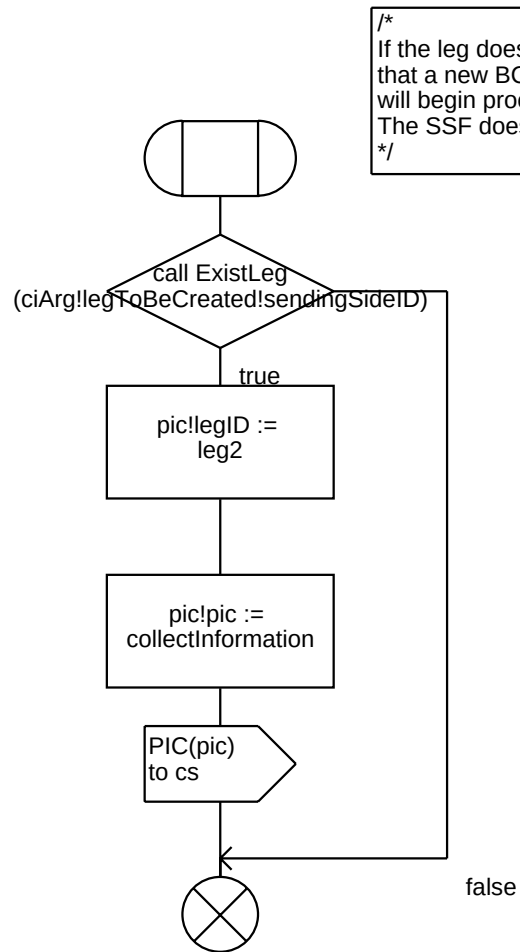
```
/*
If the leg does not exist this implies
that a new O-BCSM has been created which
will begin processing at the ONull
PIC. The SSF does not need to resume processing
for O-BCSM, but does for T-BCSM.
*/
```



Procedure ProcessCollectInformation

1(1)

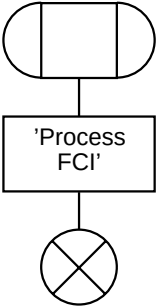
par
in ciArg CollectInformationArg;



/*
If the leg does not exist this implies
that a new BCSM has been created which
will begin processing at the CollectInformation PIC.
The SSF does not need to resume processing.
*/

Procedure ProcessFurnishChargingInformation

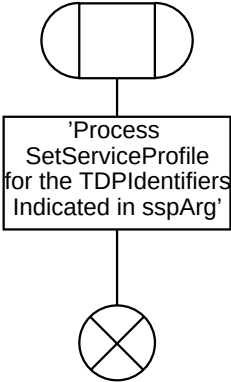
1(1)



Procedure ProcessSetServiceProfile

1(1)

;FPAR
IN sspArg SetServiceProfileArg;

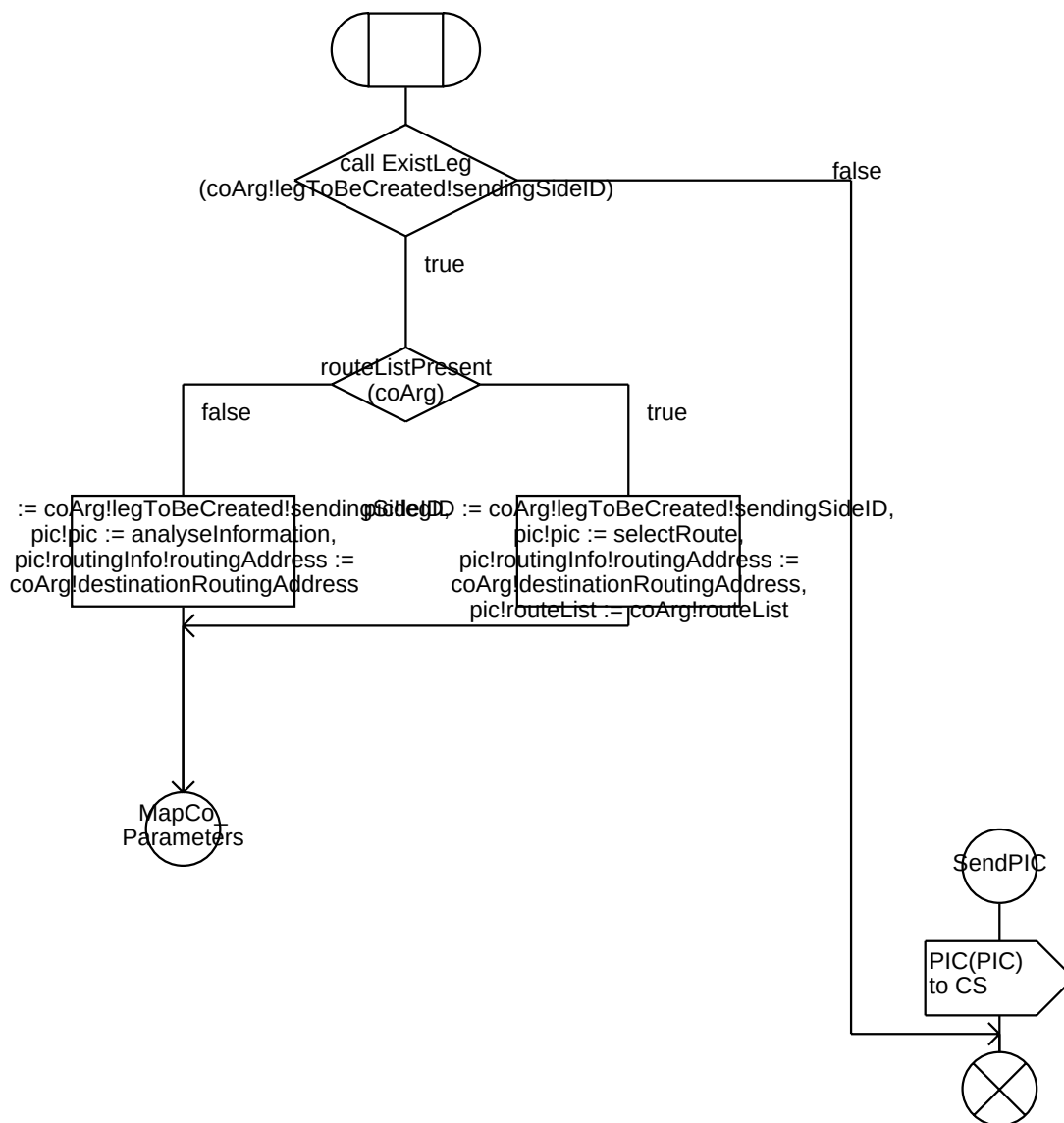


Procedure ProcessConnect

1(2)

fpar
in coArg ConnectArg;

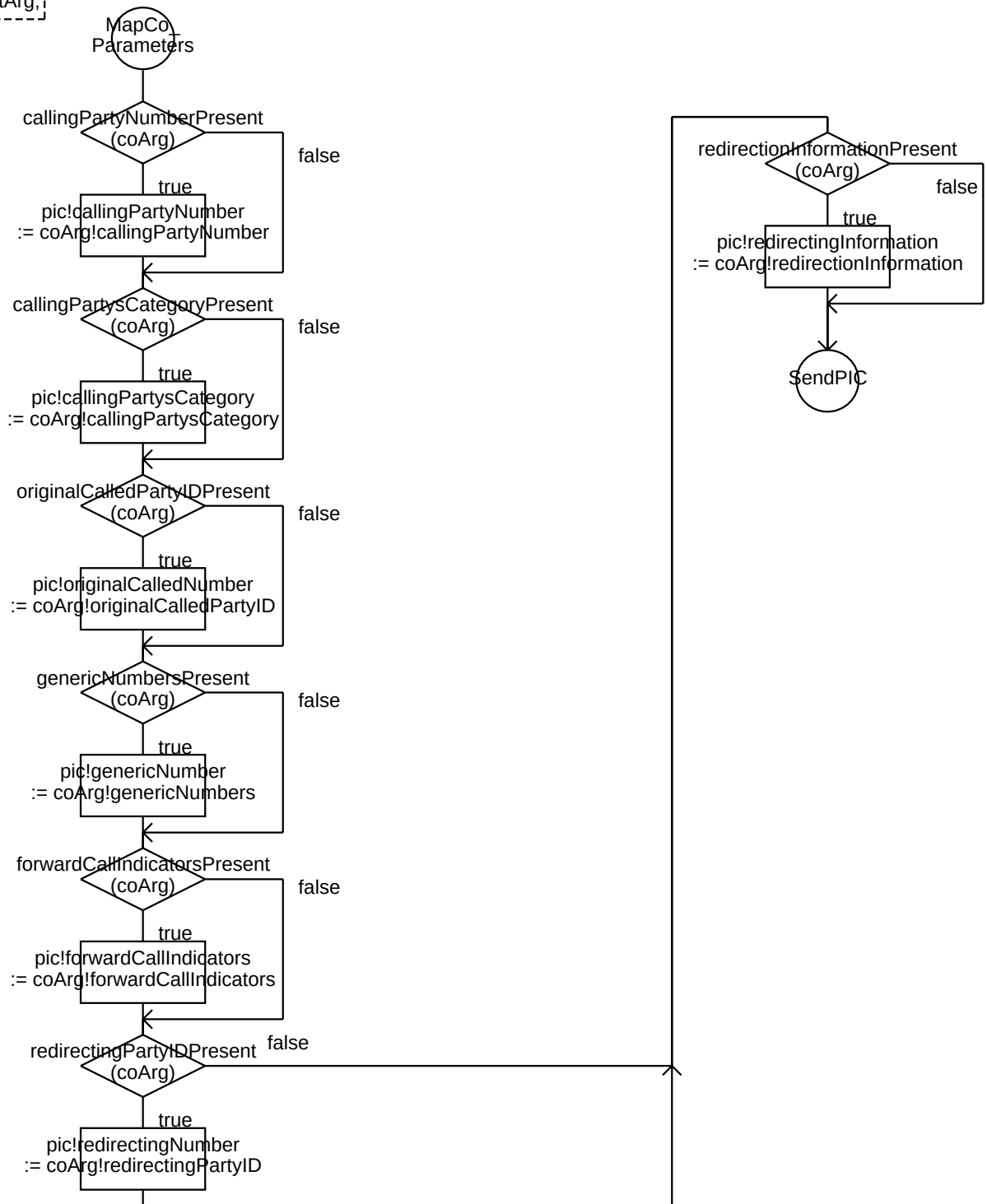
/*
If the leg does not exist this implies
that a new BCSM has been created which
will begin processing at the select route
PIC. The SSF does not need to resume processing.
*/



Procedure ProcessConnect

2(2)

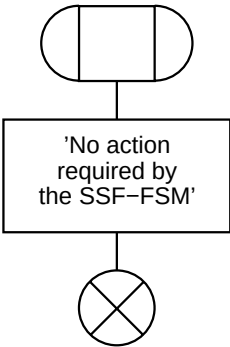
fpar
in coArg ConnectArg;



Procedure ProcessInitiateCallAttempt

1(1)

fpar
in ICAArg InitiateCallAttemptArg;



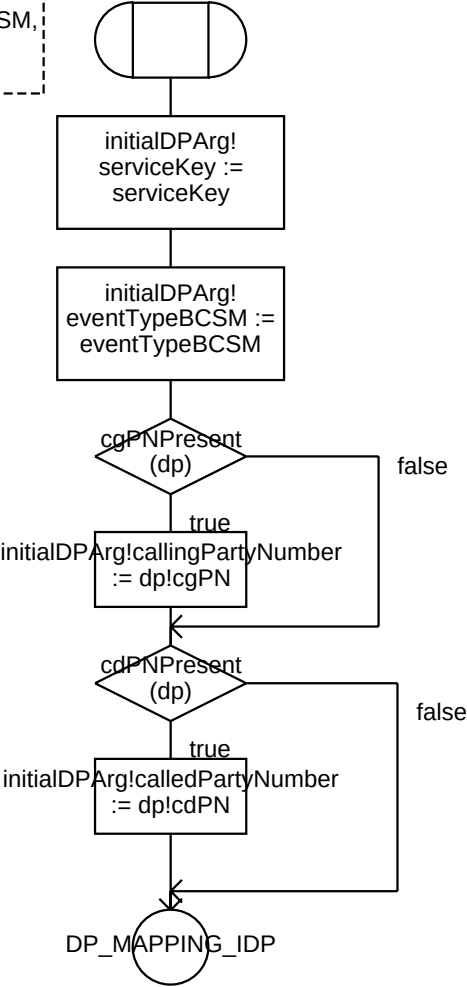
/*
The Call Segment creates an instance of an
Originating BCSM which is suspended at DP1.
*/

Procedure ProcessInitialDP

1(2)

FPAR
IN legID LegType,
IN eventTypeBCSM EventTypeBCSM,
IN serviceKey ServiceKey,
IN/OUT initialDPArg InitialDPArg;

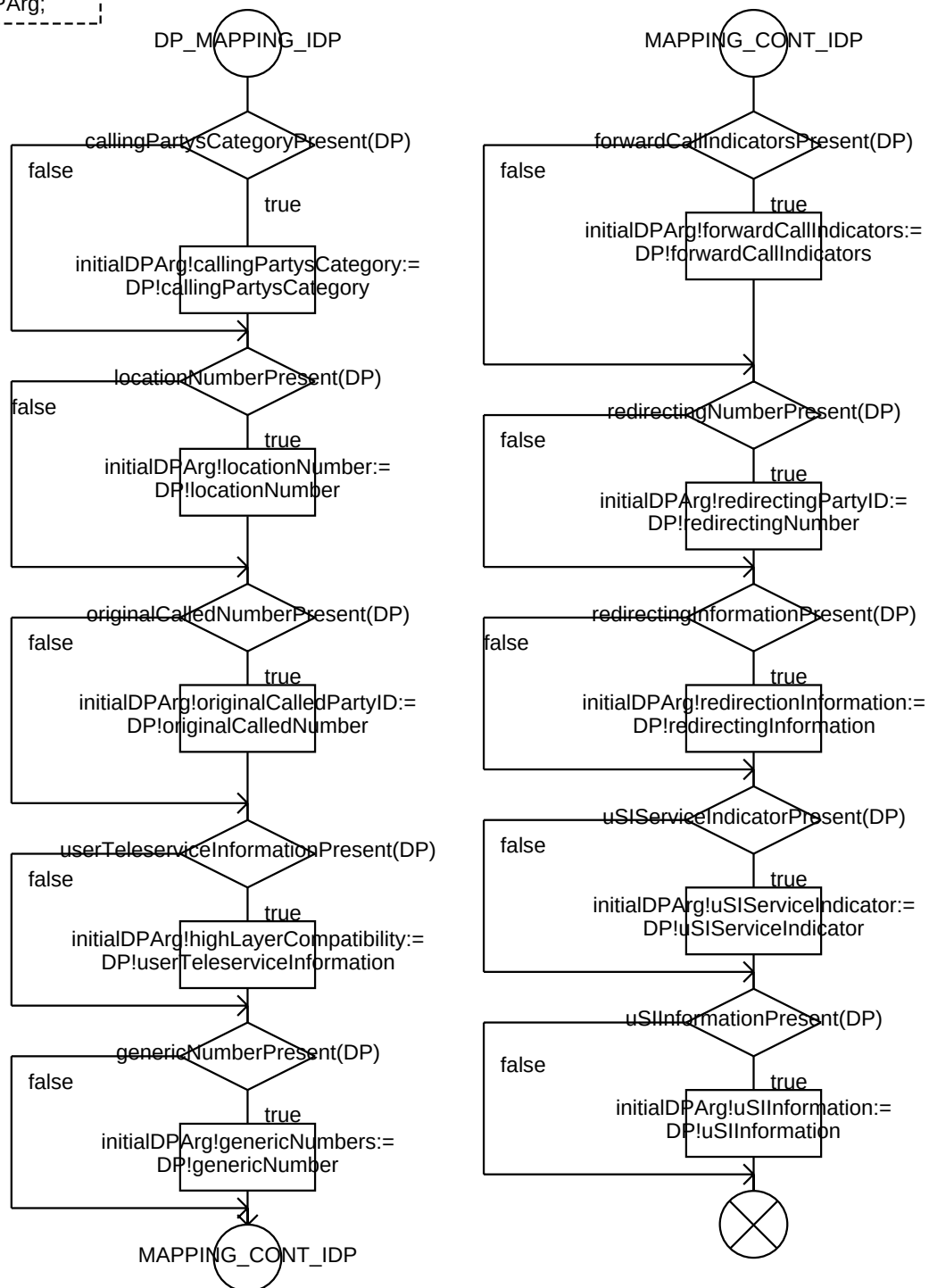
/*
The LegId is not specified in the
InitialDP operation.
*/



Procedure ProcessInitialDP

2(2)

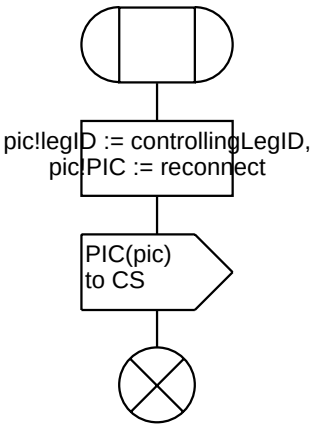
FPAR
IN legID LegType,
IN eventTypeBCSM EventTypeBCSM,
IN serviceKey ServiceKey,
IN/OUT initialDPArg InitialDPArg;



Procedure ProcessReconnect

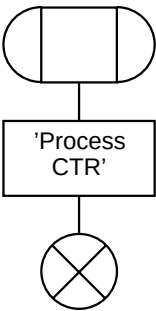
1(1)

;FPAR
IN reconArg ReconnectArg;



Procedure ProcessConnectToResource

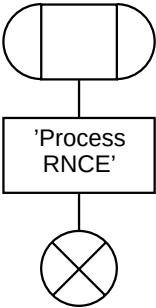
1(1)



Procedure ProcessRequestNotificationChargingEvent

1(1)

fpar
in RNCEArg RequestNotificationChargingEventArg;



Procedure ProcessDPSpecific

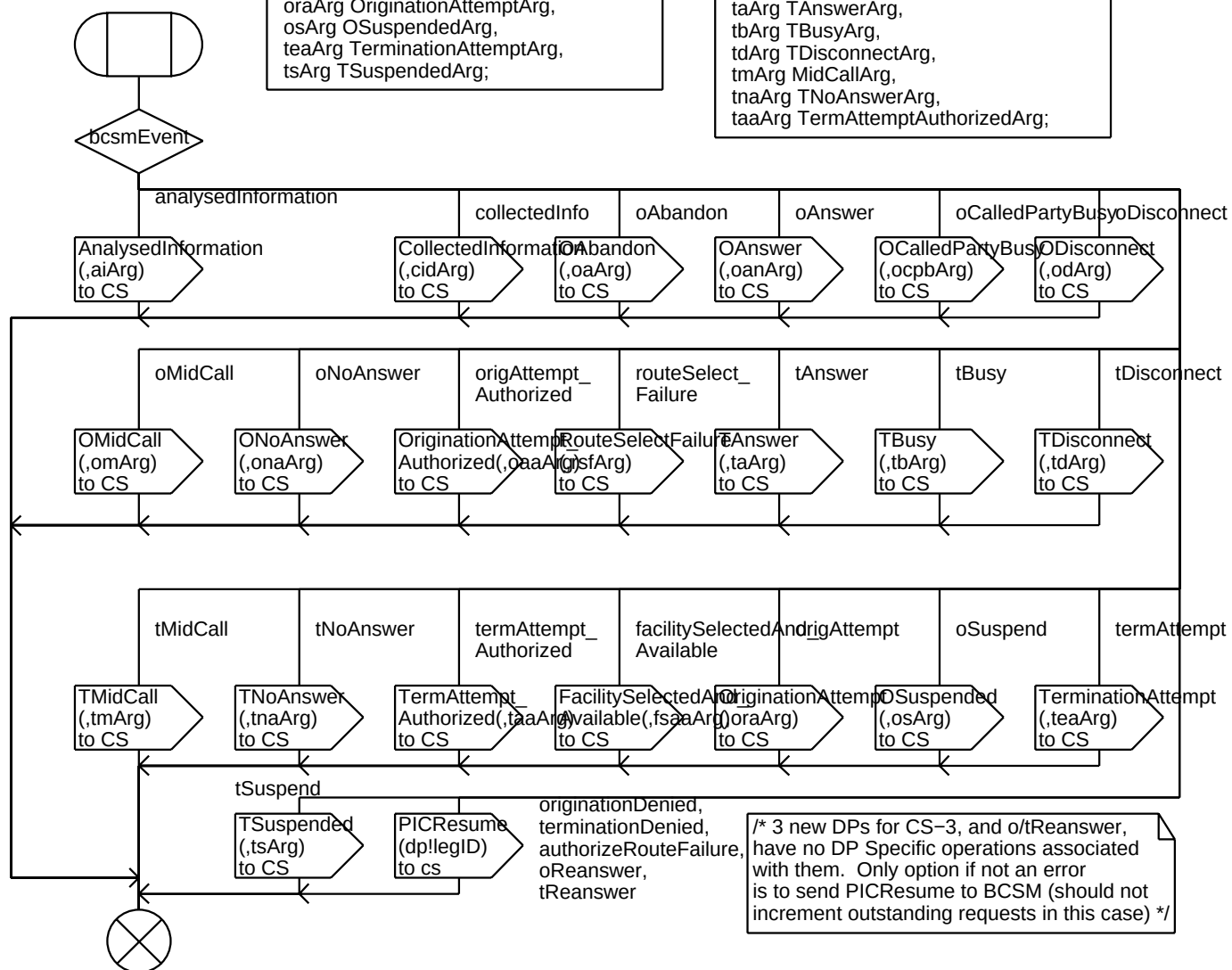
1(1)

FPAR
IN legID LegType,
IN bcsmEvent EventTypeBCSM,
IN serviceKey ServiceKey;

/* Note: The operation arguments
are assigned as defined in Q.1238-2
section 4. */

DCL
fsaaArg FacilitySelectedAndAvailableArg,
oraArg OriginationAttemptArg,
osArg OSuspendedArg,
teaArg TerminationAttemptArg,
tsArg TSuspendedArg;

DCL
aiArg AnalysedInformationArg,
cidArg CollectedInformationArg,
oaArg OAbandonArg,
oanArg OanswerArg,
ocpbArg OCalledPartyBusyArg,
odArg ODisconnectArg,
omArg MidCallArg,
onaArg ONoAnswerArg,
oaaArg OriginationAttemptAuthorizedArg,
rsfArg RouteSelectFailureArg,
taArg TAnswerArg,
tbArg TBusyArg,
tdArg TDisconnectArg,
tmArg MidCallArg,
tnaArg TNoAnswerArg,
taaArg TermAttemptAuthorizedArg;



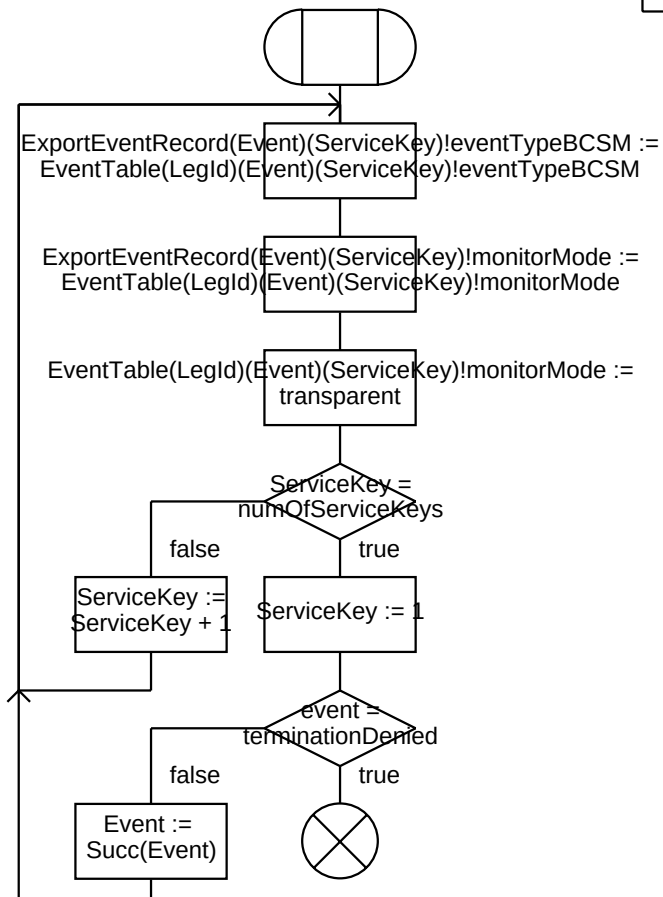
Procedure ExportEventRecord

1(1)

FPAR
IN LegId LegType,
IN/OUT ExportEventRecord EventRecordType;

DCL

event EventTypeBCSM := origAttemptAuthorized,
serviceKey ServiceKey := 1;



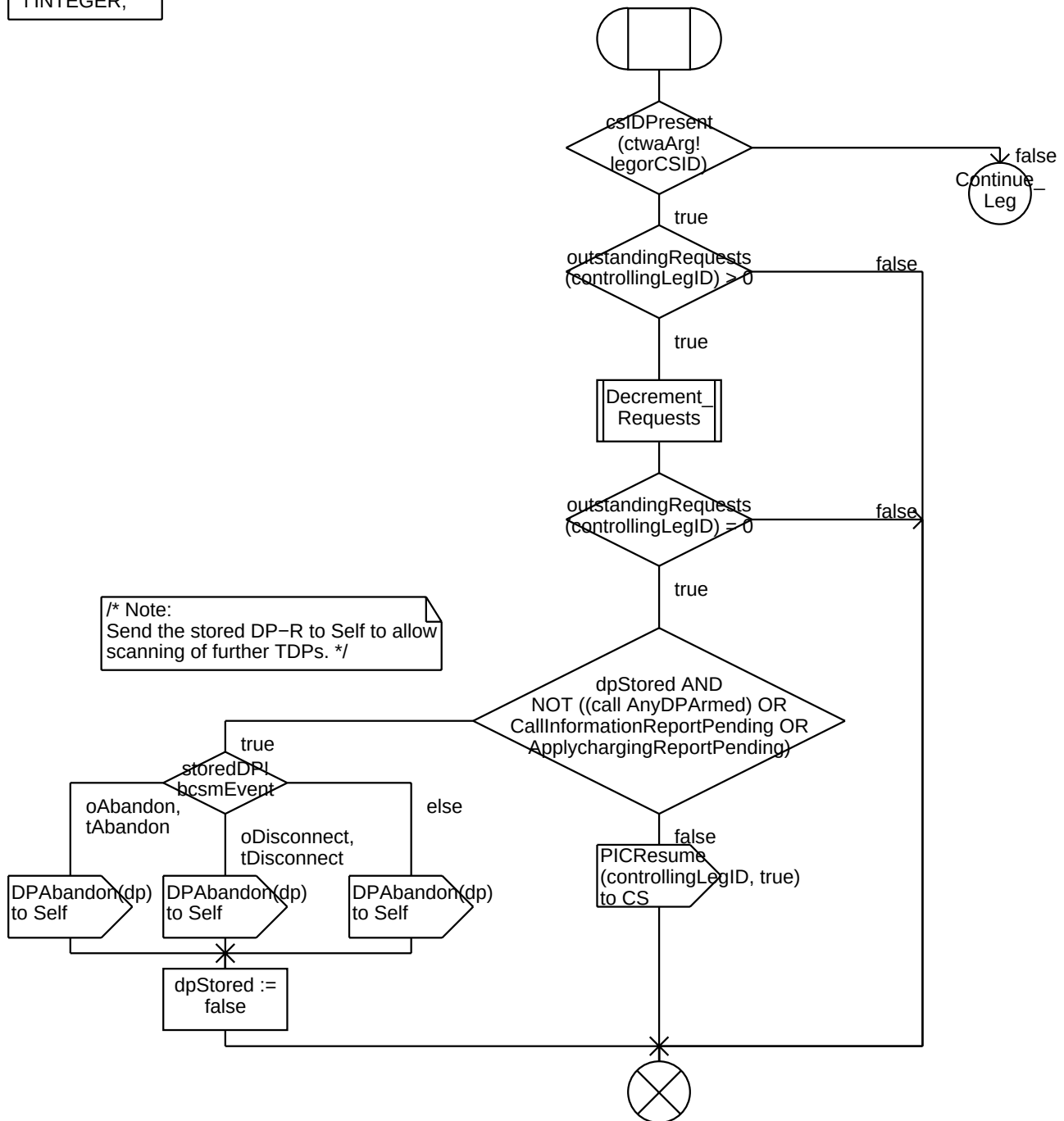
Procedure ProcessContinueWithArgument

1(2)

FPAR
IN ctwaArg ContinueWithArgumentArg;

DCL
legID LegType,
i INTEGER;

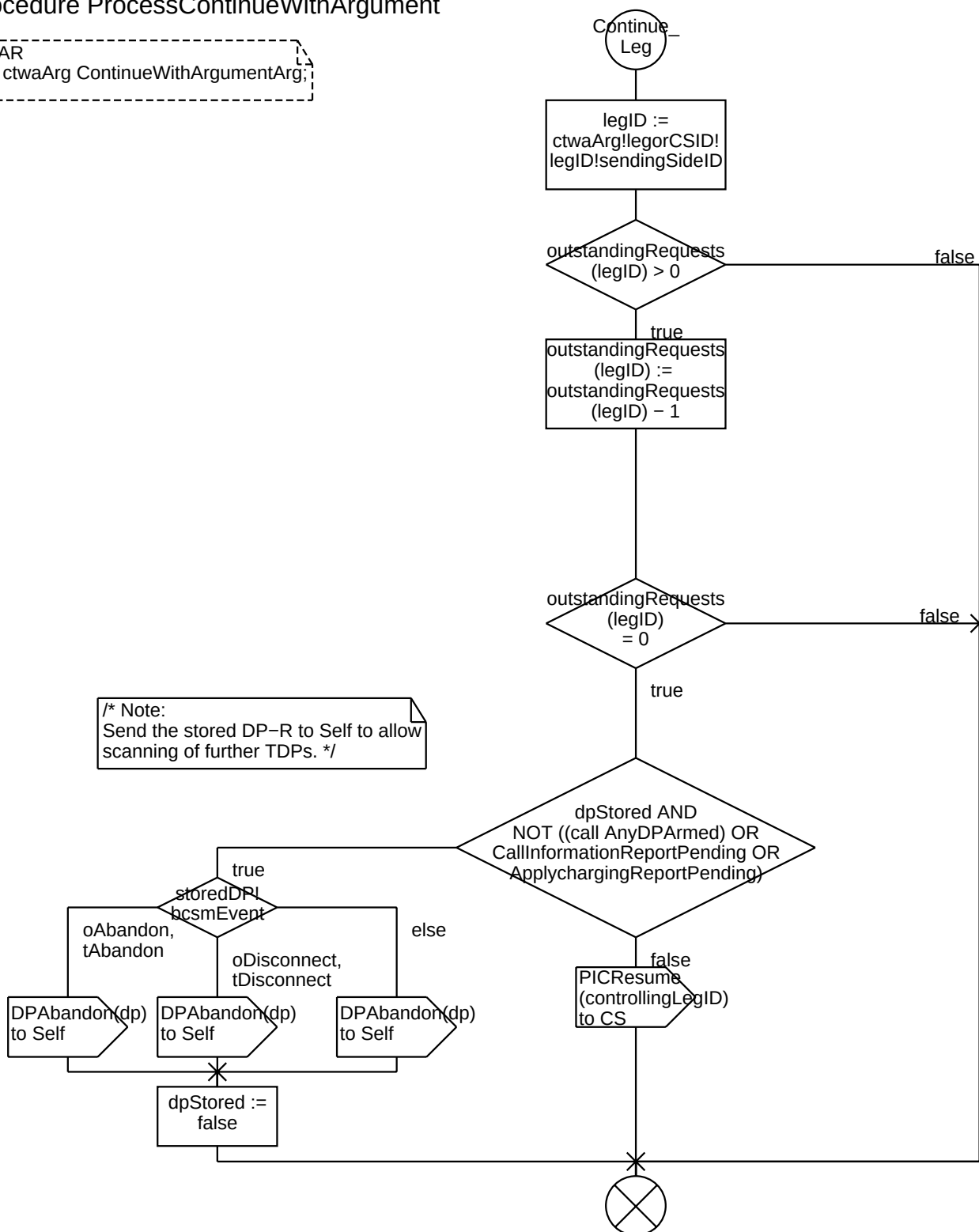
/* Note:
Send the stored DP-R to Self to allow
scanning of further TDPs. */



Procedure ProcessContinueWithArgument

2(2)

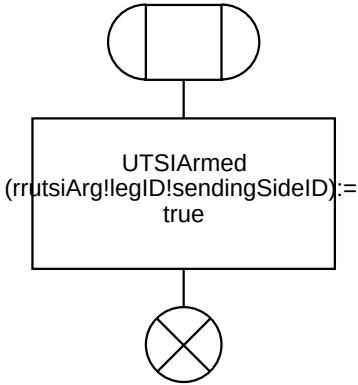
FPAR
IN ctwaArg ContinueWithArgumentArg;



Procedure ProcessRequestReportUTSI

1(1)

;FPAR
rrutsiArg RequestReportUTSIArg;

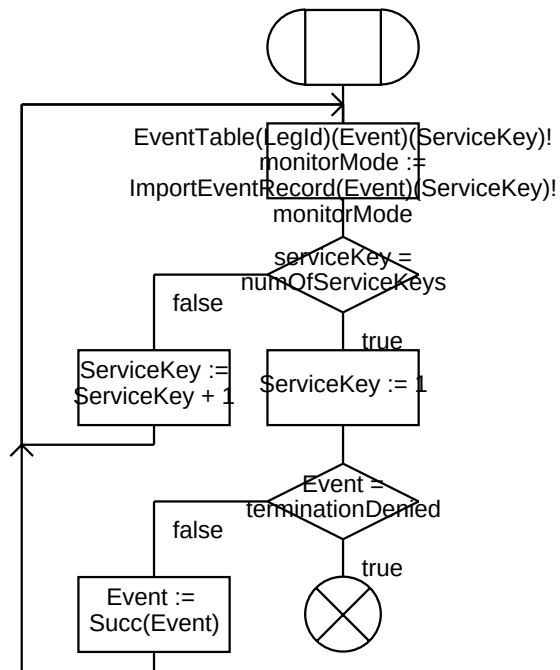


Procedure ImportEventRecord

1(1)

FPAR
IN legID LegType,
IN/OUT ImportEventRecord EventRecordType;

DCL
event EventTypeBCSM := origAttemptAuthorized,
serviceKey ServiceKey := 1;



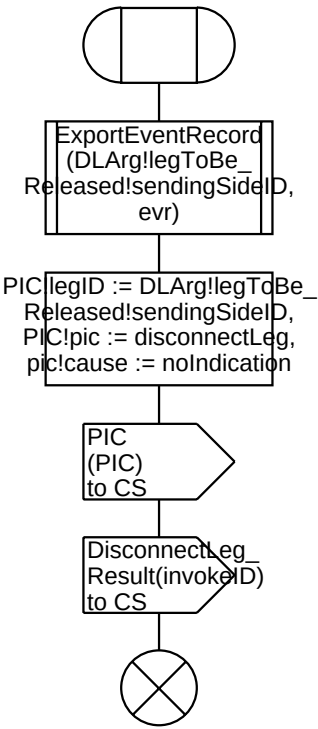
Procedure ProcessDisconnectLeg

1(1)

fpar
DLArg DisconnectLegArg;

DCL
evr EventRecordType;

/*
The SSF notifies the BCSM to terminate processing.
*/

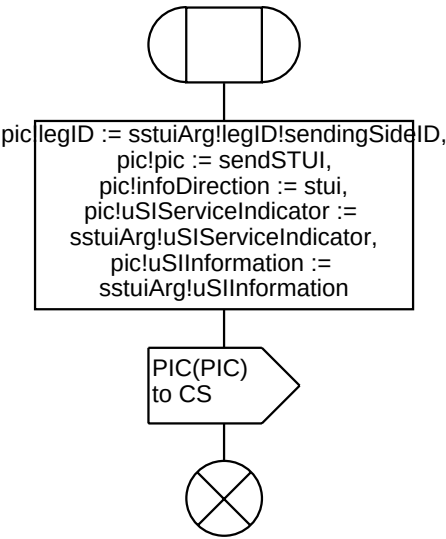


Procedure ProcessSendSTUI

1(1)

FPAR
sstuiArg SendSTUIArg;

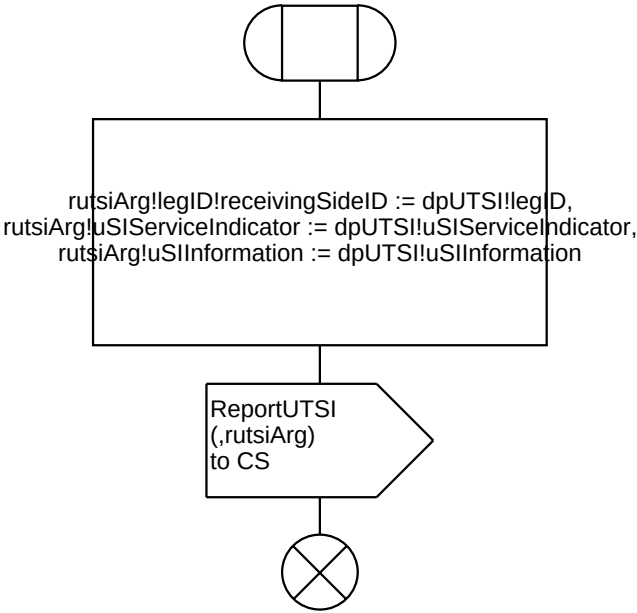
DCL
pic PICArg;



Procedure ProcessReportUTSI

1(1)

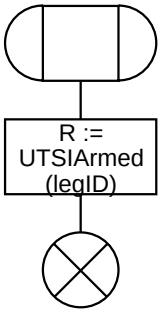
;FPAR
IN dpUTSI DPUTSiArg;



Procedure IsUTSIArmed

1(1)

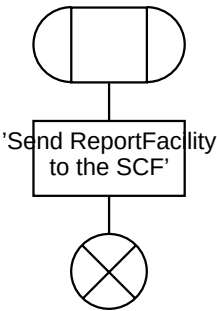
;FPAR
IN legID LegType;
RETURNS R Boolean;



Procedure ProcessReportFacility

1(1)

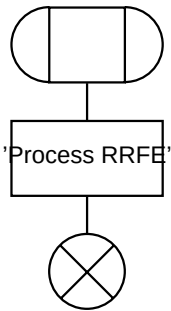
;fpar
IN dpFacility DPFacilityArg;



Procedure ProcessRequestReportFacilityEvent

1(1)

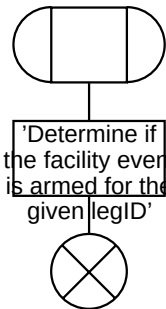
;fpar
IN rrfArg RequestReportFacilityEventArg;



Procedure IsFacilityArmed

1(1)

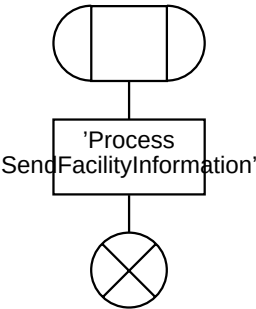
;FPAR
IN legID LegType;
RETURNS R Boolean;



Procedure ProcessSendFacility

1(1)

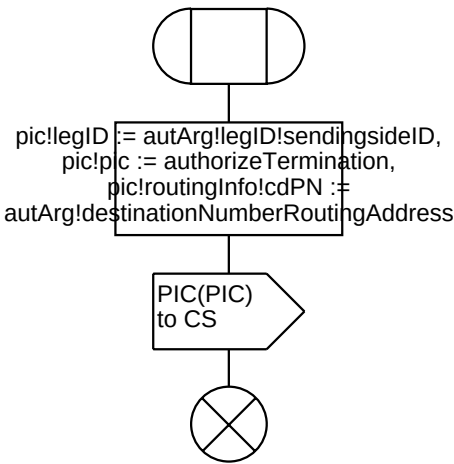
;fpar
IN sefArg SendFacilityInformationArg;



Procedure ProcessAuthorizeTermination

1(1)

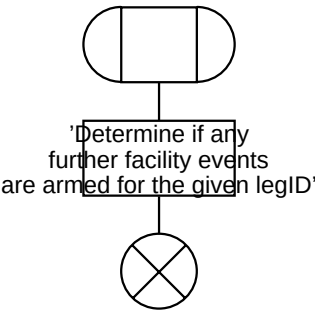
fpar
in autArg AuthorizeTerminationArg;



Procedure AnyFacilityArmed

1(1)

;FPAR
IN legID LegType;
RETURNS R Boolean;

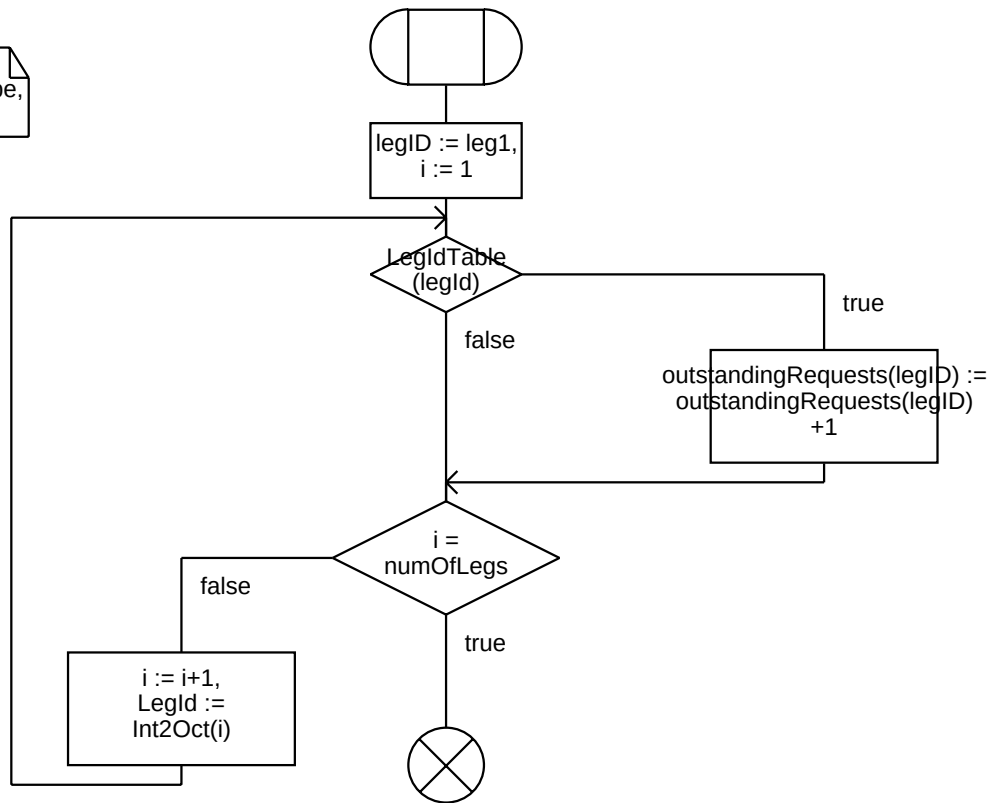


Procedure IncrementRequests

1(1)



DCL
legID LegType,
i INTEGER;

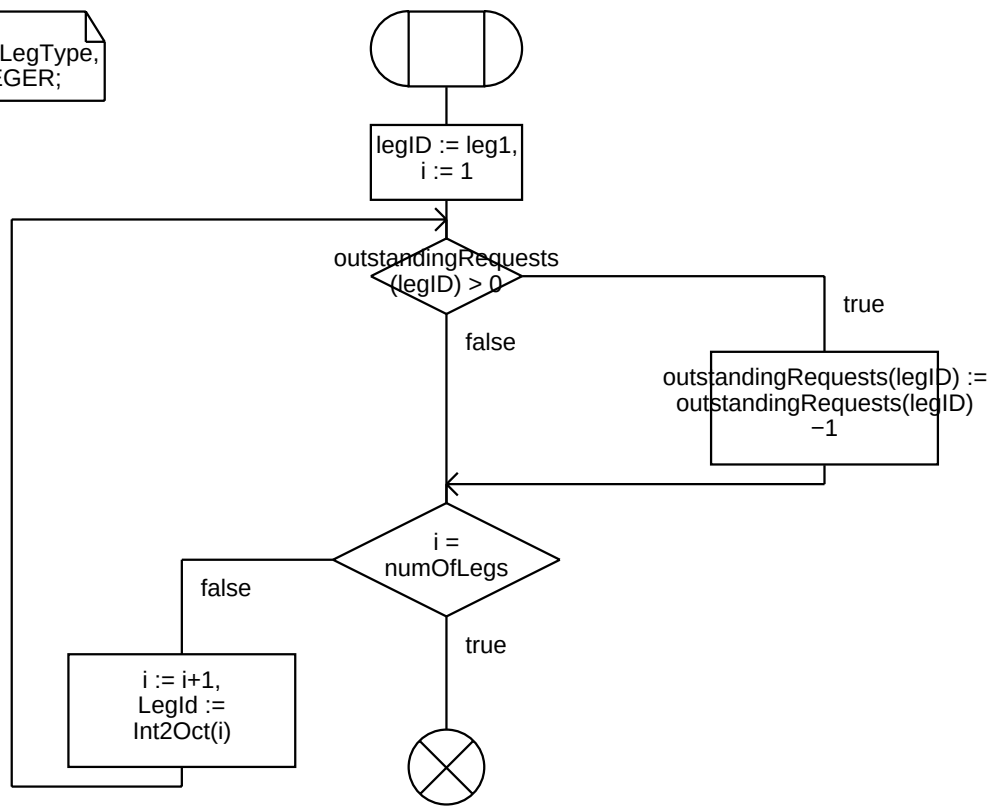


Procedure DecrementRequests

1(1)



DCL
legID LegType,
i INTEGER;



Process Type OriginatingBCSM

1(19)

;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */

/***** O-BCSM FOR INAP CS-3. *****/

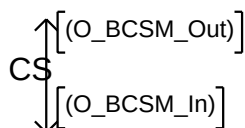
/***** DATA TYPE DEFINITIONS *****/

```
NEWTYPE DPResultType
LITERALS
Resume,
CollectInformation,
AnalyseInformation,
SelectRoute,
Exception,
ARelease,
BRelease,
SCFRelease, /* Used with ReleaseCall */
Answer,
Alerting,
Reanswer,

DL_A, DL_B, /* Used with DisconnectLeg */
CS_Stop
ENDNEWTYPE;

NEWTYPE PICResultType
LITERALS
Success,
Failure,
Select_next_Route,
Analyse_marked_CDPN,
Alerting,
Answer,
NoAnswer,
Busy,
MidCall,
Exception,
ARelease,
BRelease,
SCFRelease, /* Used with ReleaseCall */

NotReachable,
DL_A, DL_B, /* Used with DisconnectLeg */
Reanswer,
Suspended
ENDNEWTYPE;
```



Process Type OriginatingBCSM 2(19)

;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */

/**** VARIABLE AND TIMER DECLARATIONS *****/

TIMER
NoAnswerT := 3600000; /* Value for simulation purposes. */

DCL
/* Pointer to the call segment. */
CS PId,

/* SigCon ID of the signalling entity associated with this BCSM. */
SigConId callRef,

/* Used to indicate the status of the call processing. */
alertingSent boolean := false, /* Indicates if CallProgress(bptyAlerted) sent to the calling party. */
setupSent boolean := false, /* Indicates if SetupReqInd is sent to the called party, i.e. is the T-BCSM created. */

/* Call info */
cdPNIx INTEGER := 1, /* Index to marked cdPN */
numOfCDPNs INTEGER := 1, /* Number of CDPNs in the routingAddress. */
routingAddress DestinationRoutingAddress, /* List of CDPNs */
routeIx INTEGER := 1, /* Index to selected route */

/* Other variables. */
PICResult PICResultType,
DPResult DPResultType,
cause Cause:= noIndication,
PIC PICArg,
DP DPArg,

/* Variables for simulation purposes. */
Enbloc Boolean := true,

digits Digits;

DCL
/* SigCon primitive parameters. */
AEArg AddressEndType,
CPArg CallProgressType,
FArg FailureType,
RArg ReleaseType,
SFtArg ServiceFeatureType,
SIRArg SetupIRType,
SCRArg SetupCRType,
SAArg SubsequentAddressType;

TIMER
SuspendT := 100,
RetentionO := 300;

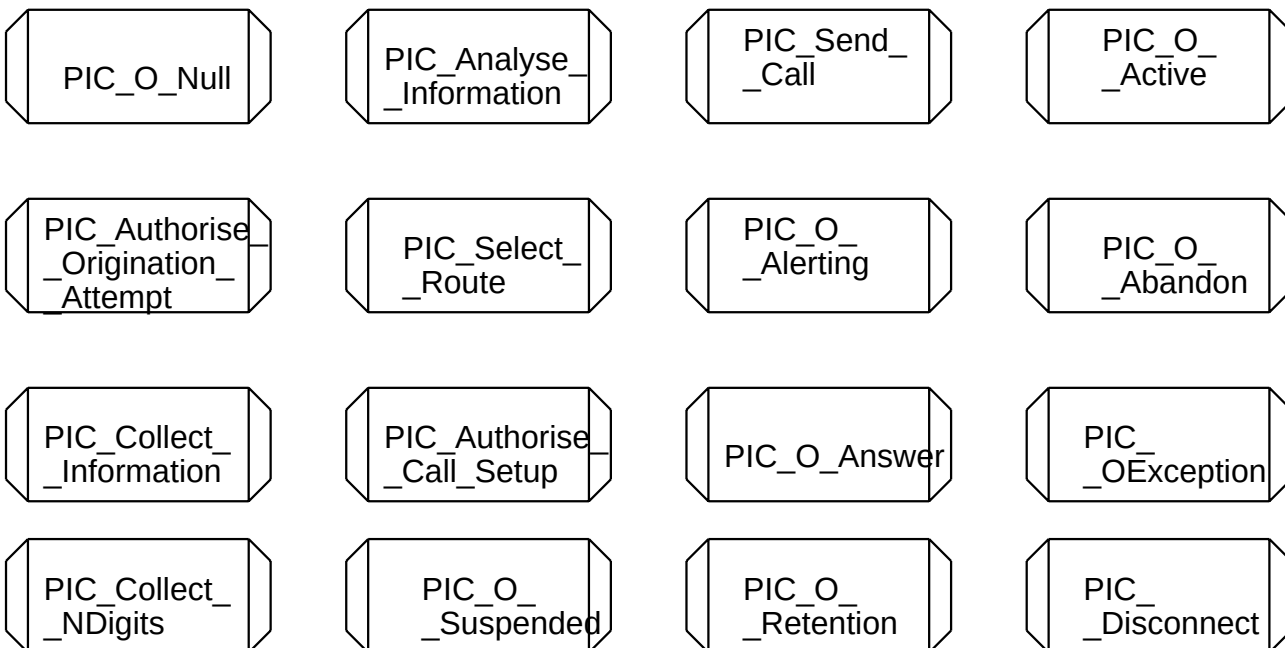
DCL
currentLegID, newLegID LegType,
nsrArg NetworkSRType,
udArg UserData Type,
dpUTSIArg DPUTSIArg;

Process Type OriginatingBCSM

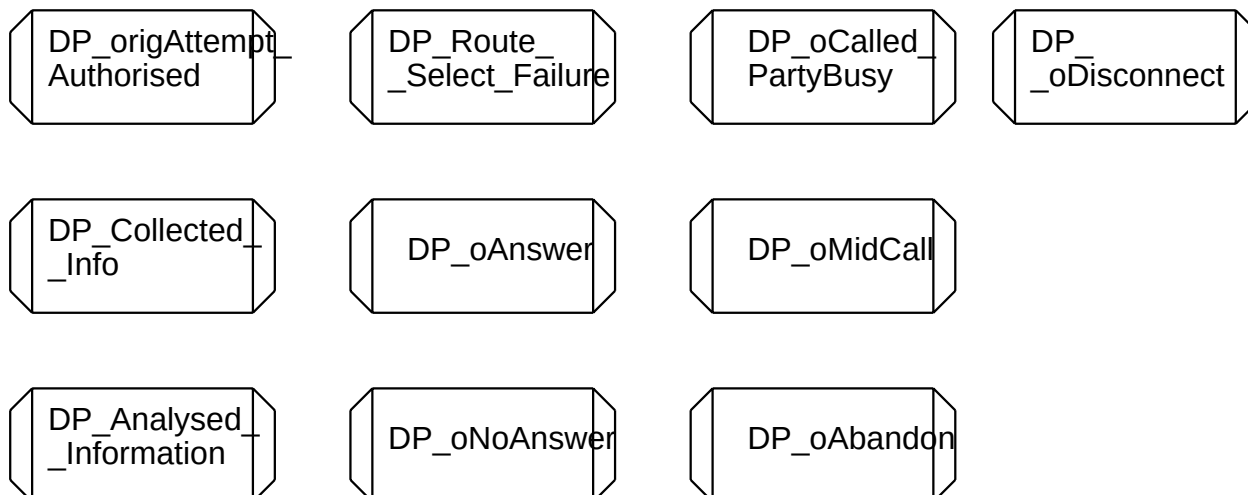
3(19)

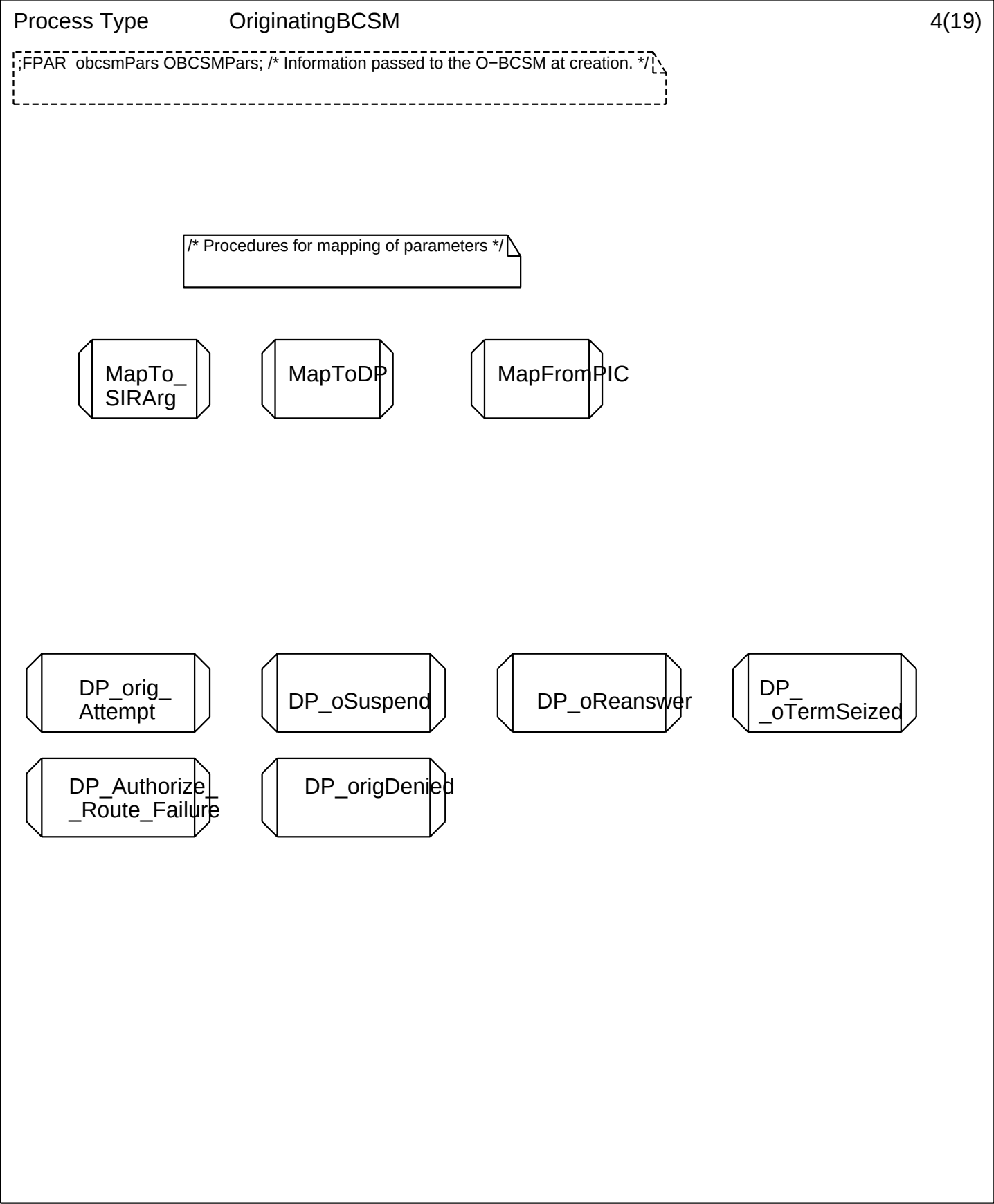
;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */

/* Procedure definitions for Points In Call (PIC). */



/* Procedure definitions for Detection Points (DPs) */





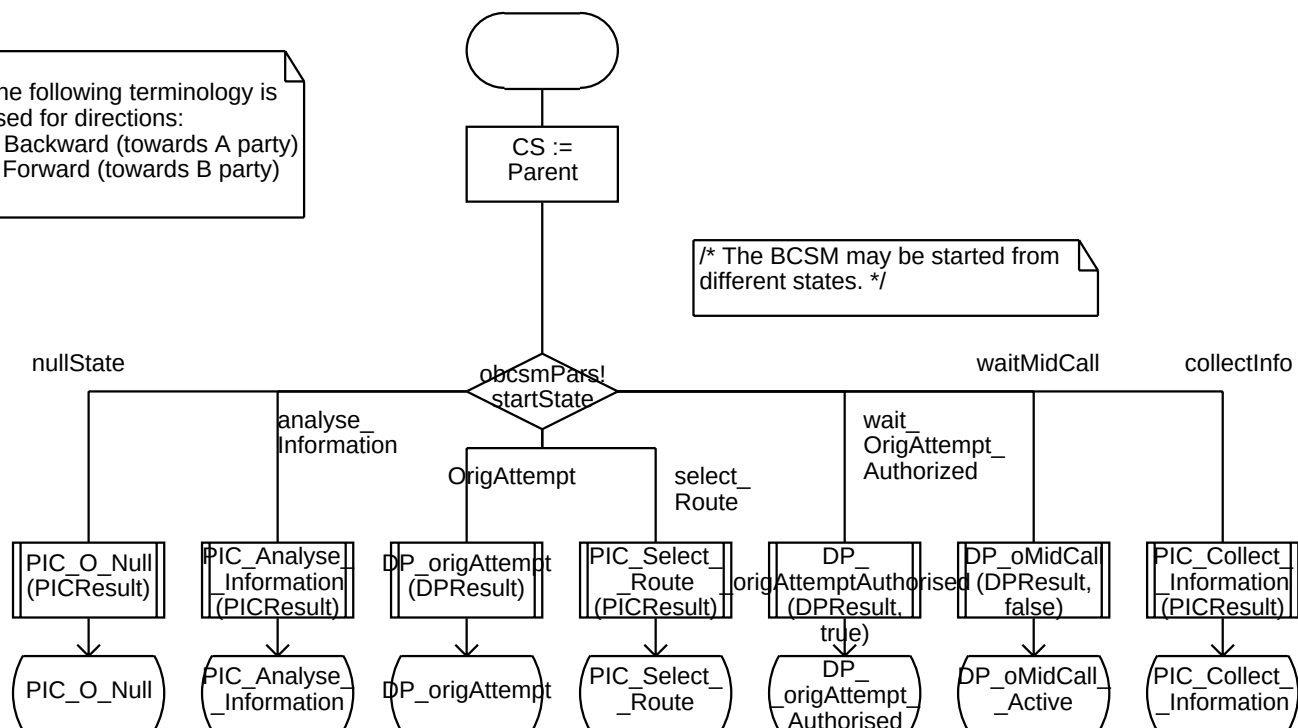
Process Type OriginatingBCSM

5(19)

;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */\n

/*
The following terminology is
used for directions:
B Backward (towards A party)
F Forward (towards B party)
*/

/* The BCSM may be started from
different states. */

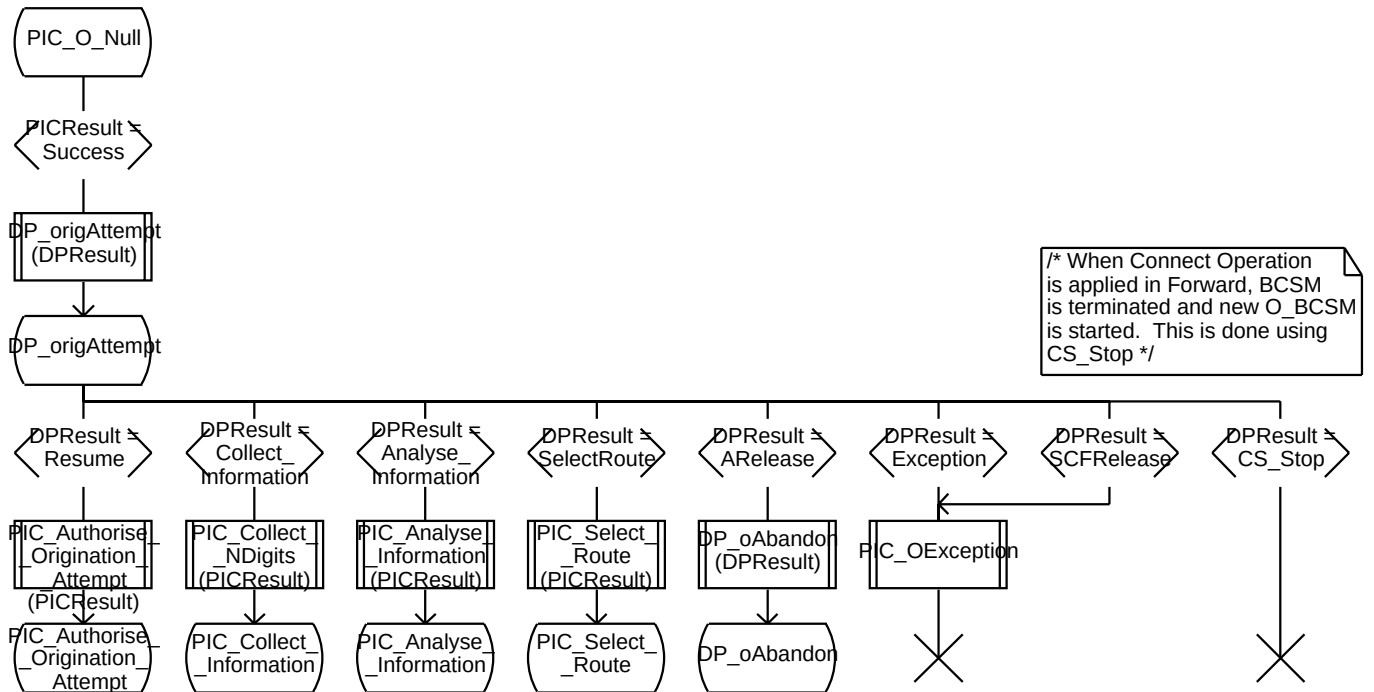


Process Type

OriginatingBCSM

6(19)

;FPAR obcsmpars OBCSMPars; /* Information passed to the O-BCSM at creation. */

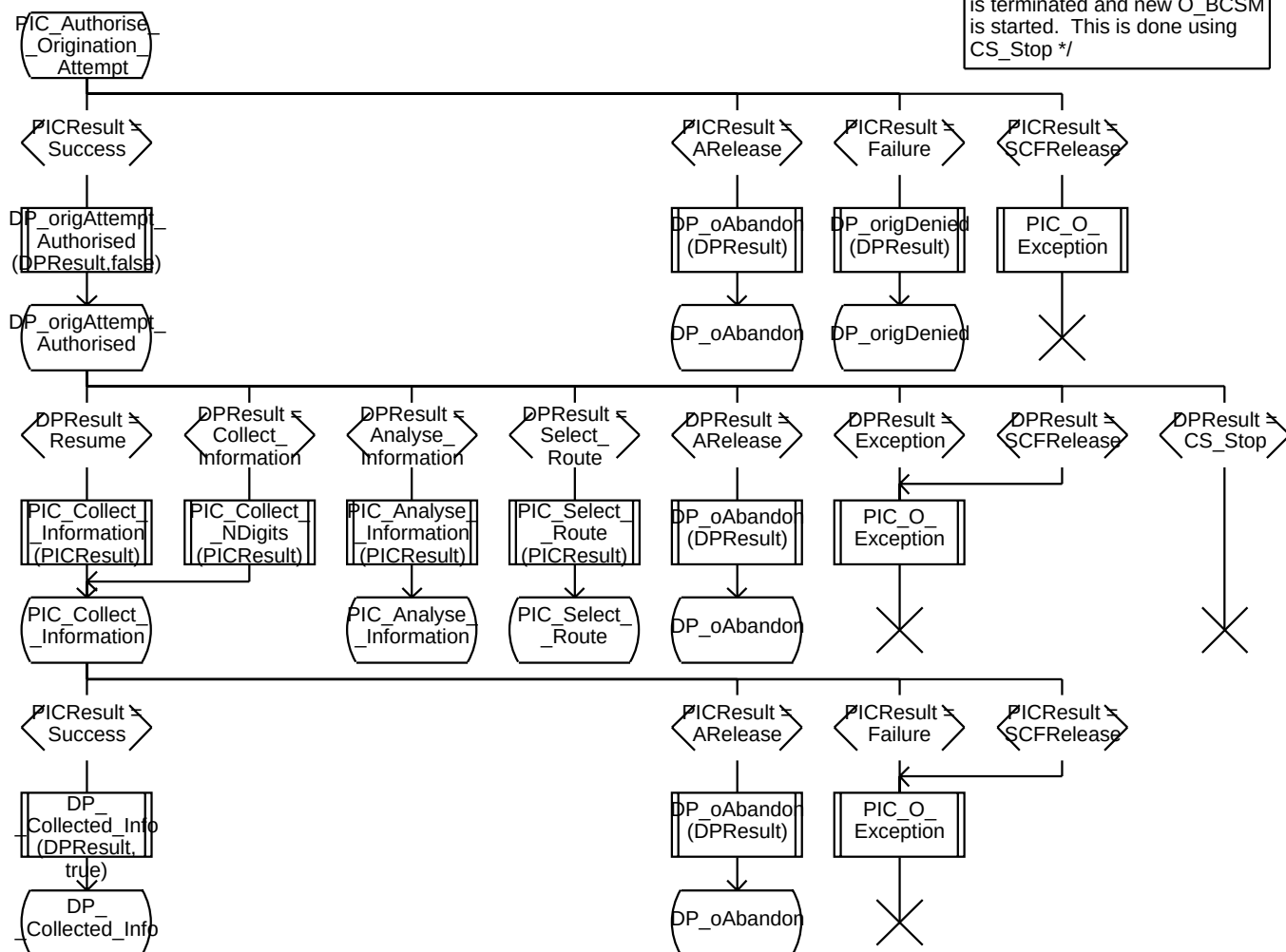


Process Type OriginatingBCSM

7(19)

;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */

/* When Connect Operation is applied in Forward, BCSM is terminated and new O-BCSM is started. This is done using CS_Stop */



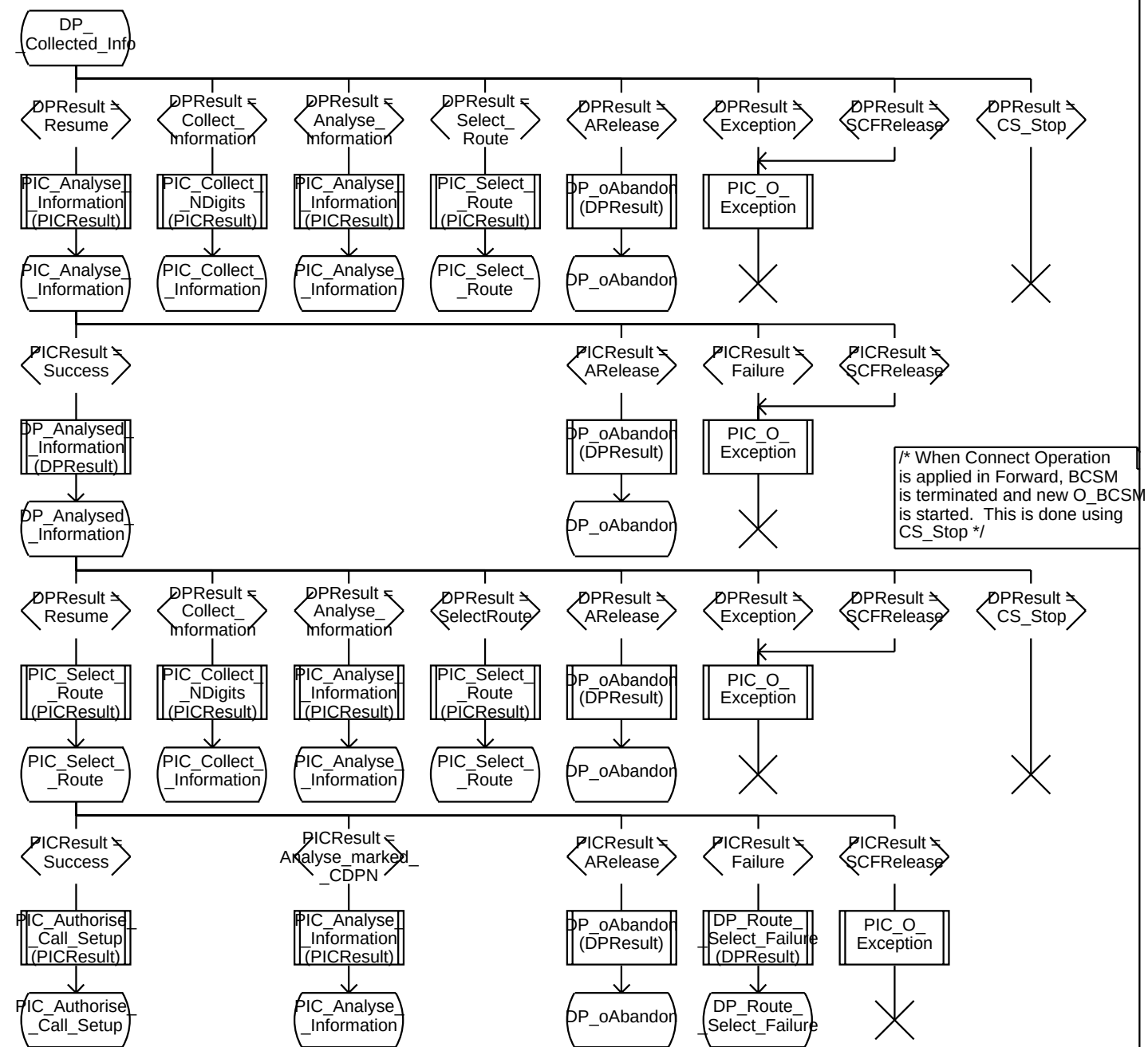
Process Type	OriginatingBCSM

8(19)

```

[;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */\,

```

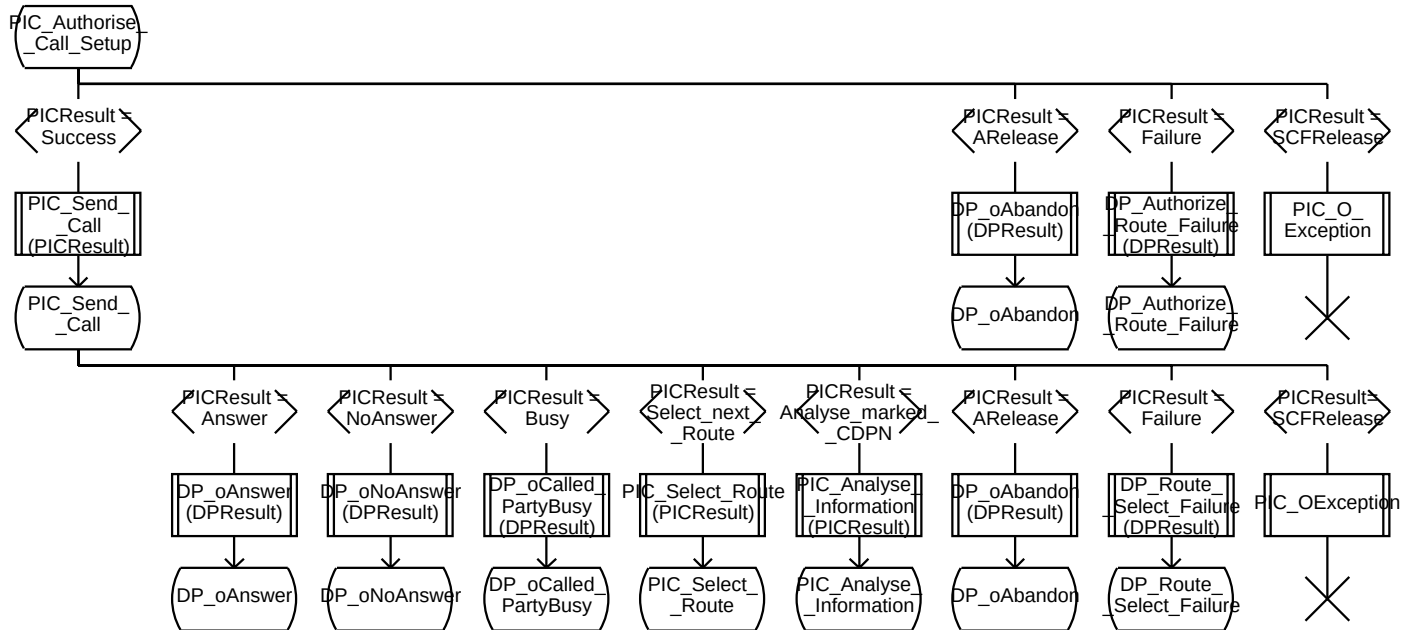


Process Type

OriginatingBCSM

9(19)

/* FPAR obcsmpars OBSCMPars; /* Information passed to the O-BCSM at creation. */

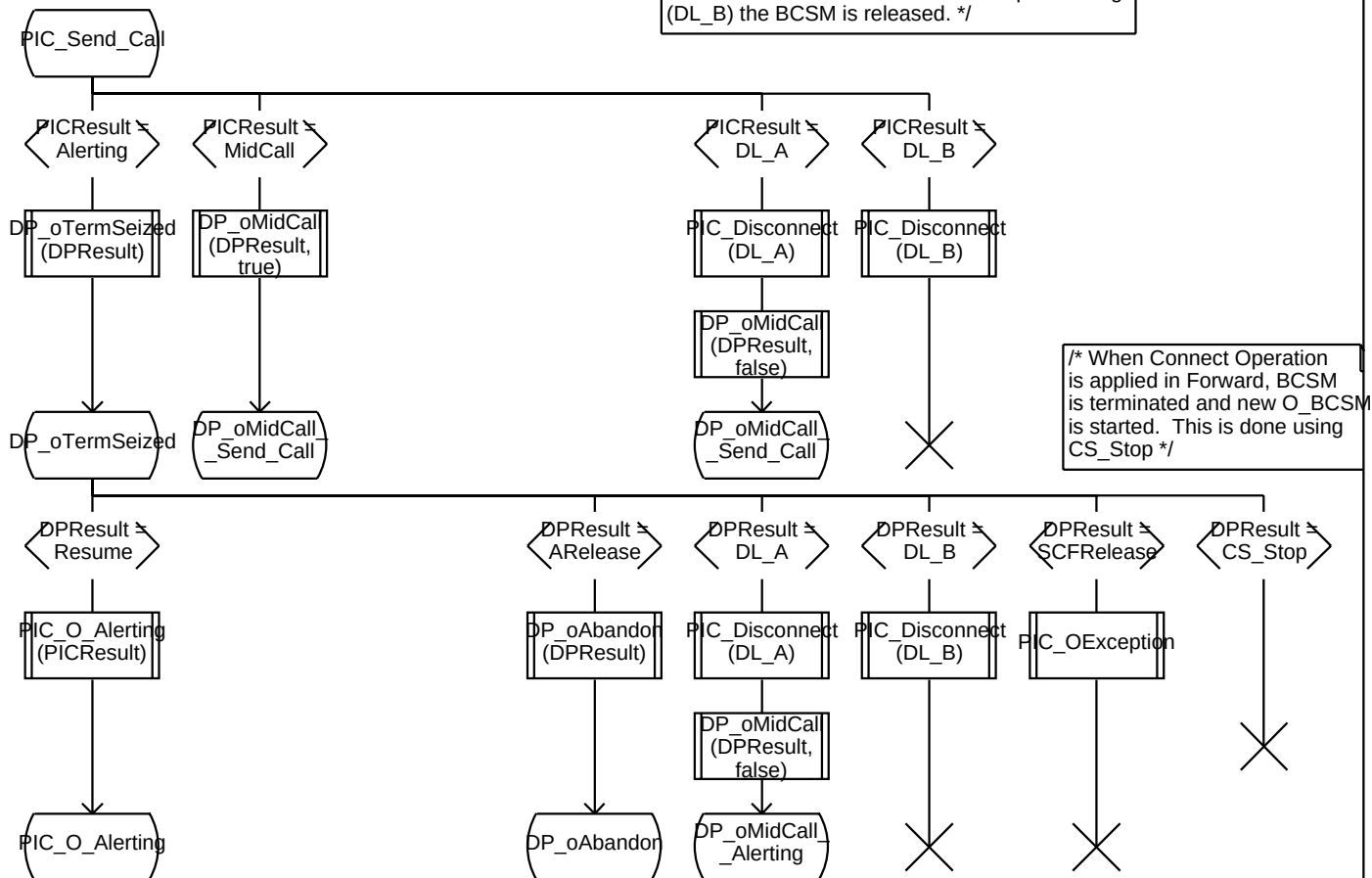


Process Type OriginatingBCSM

10(19)

;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */

/* The SCF may apply the DisconnectLeg operation. In case of DL on the controlling leg (DL_A) the BCSM must transit to the wait state of the MidCall DP. In case of DL on the passive leg (DL_B) the BCSM is released. */



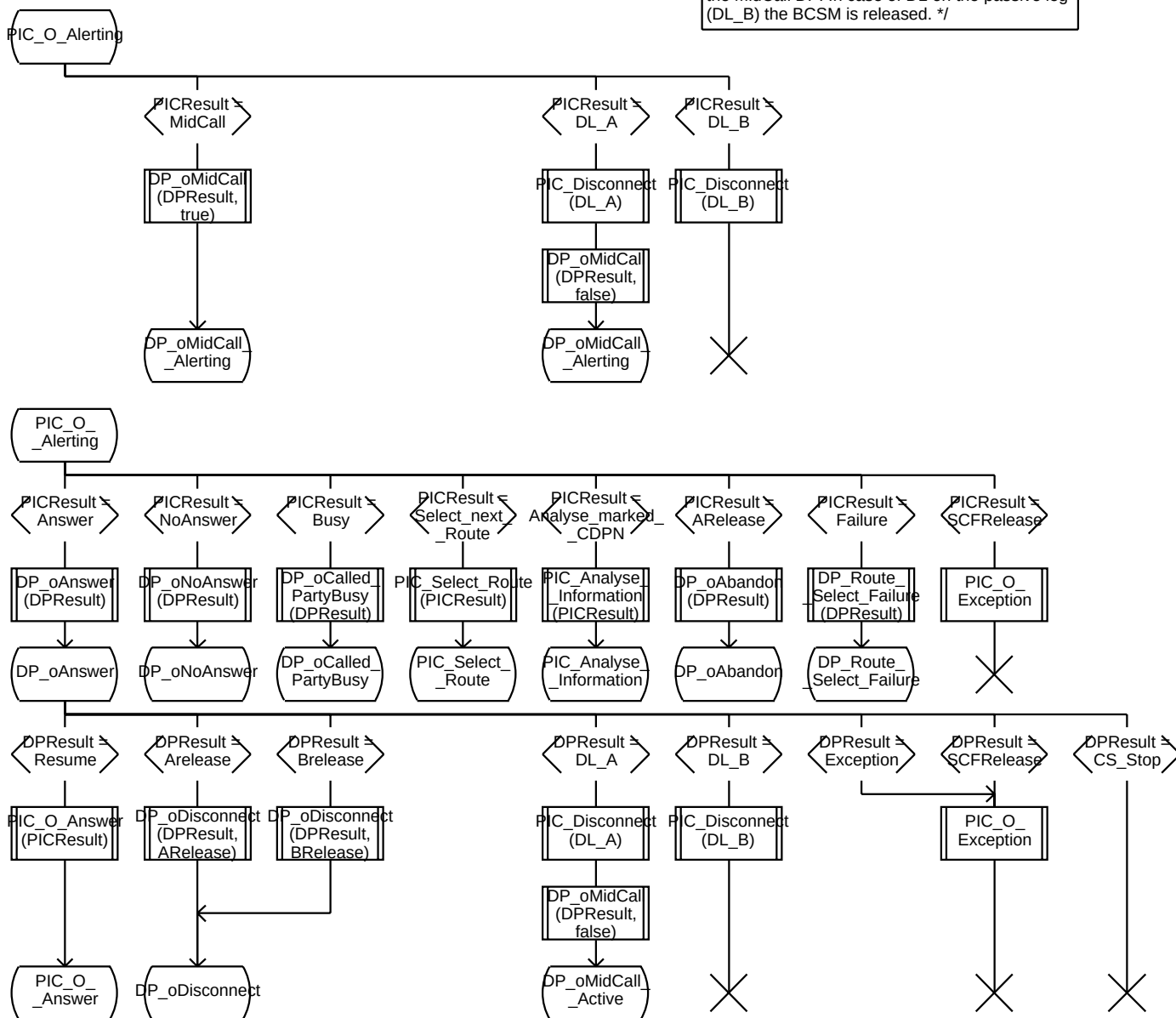
Process Type

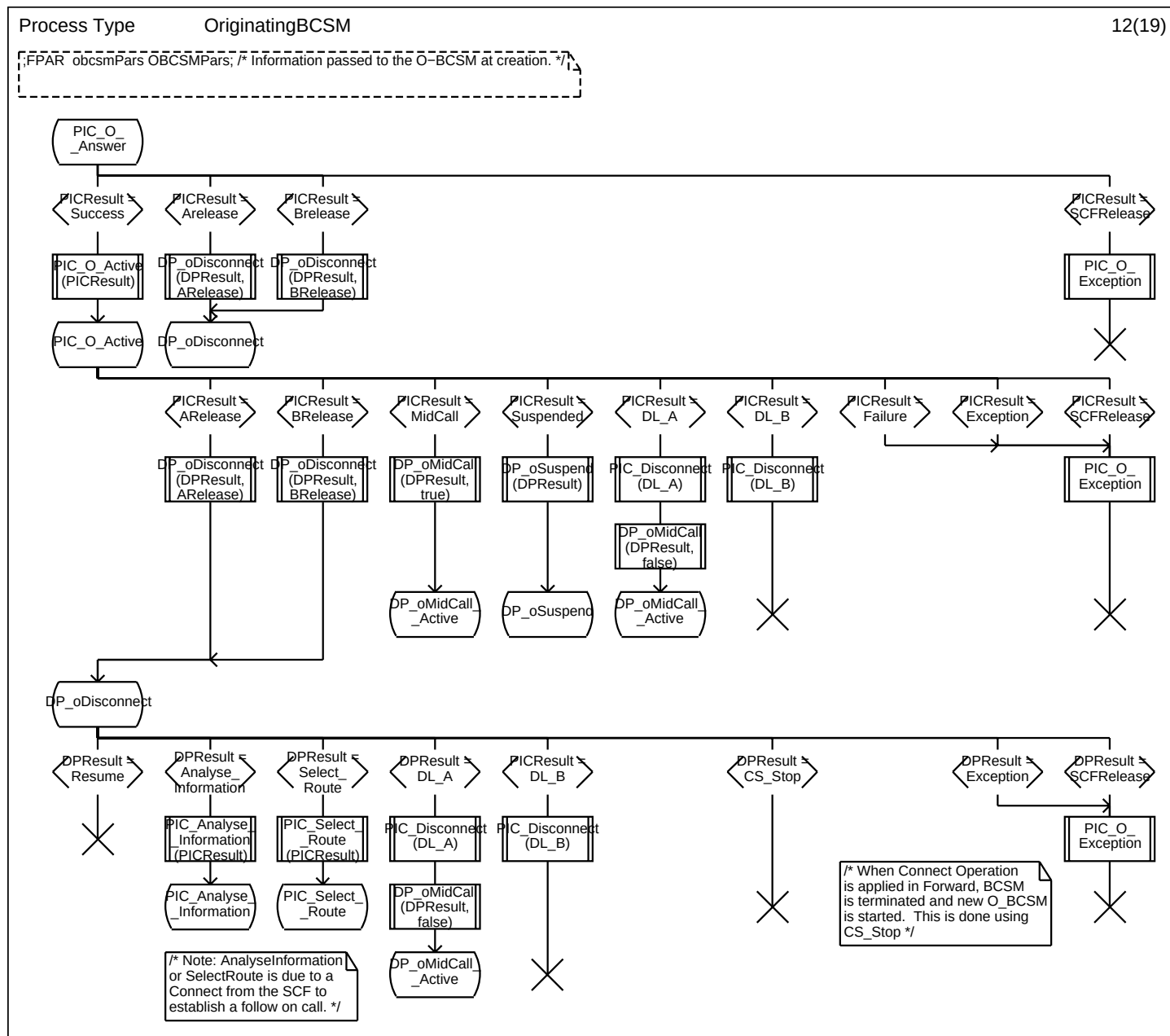
OriginatingBCSM

11(19)

/* FPAR obcsmpars OBSCMPars; /* Information passed to the O-BCSM at creation. */

/* The SCF may apply the DisconnectLeg operation. In case of DL on the controlling leg (DL_A) the BCSM must transit to the wait state of the MidCall DP. In case of DL on the passive leg (DL_B) the BCSM is released. */

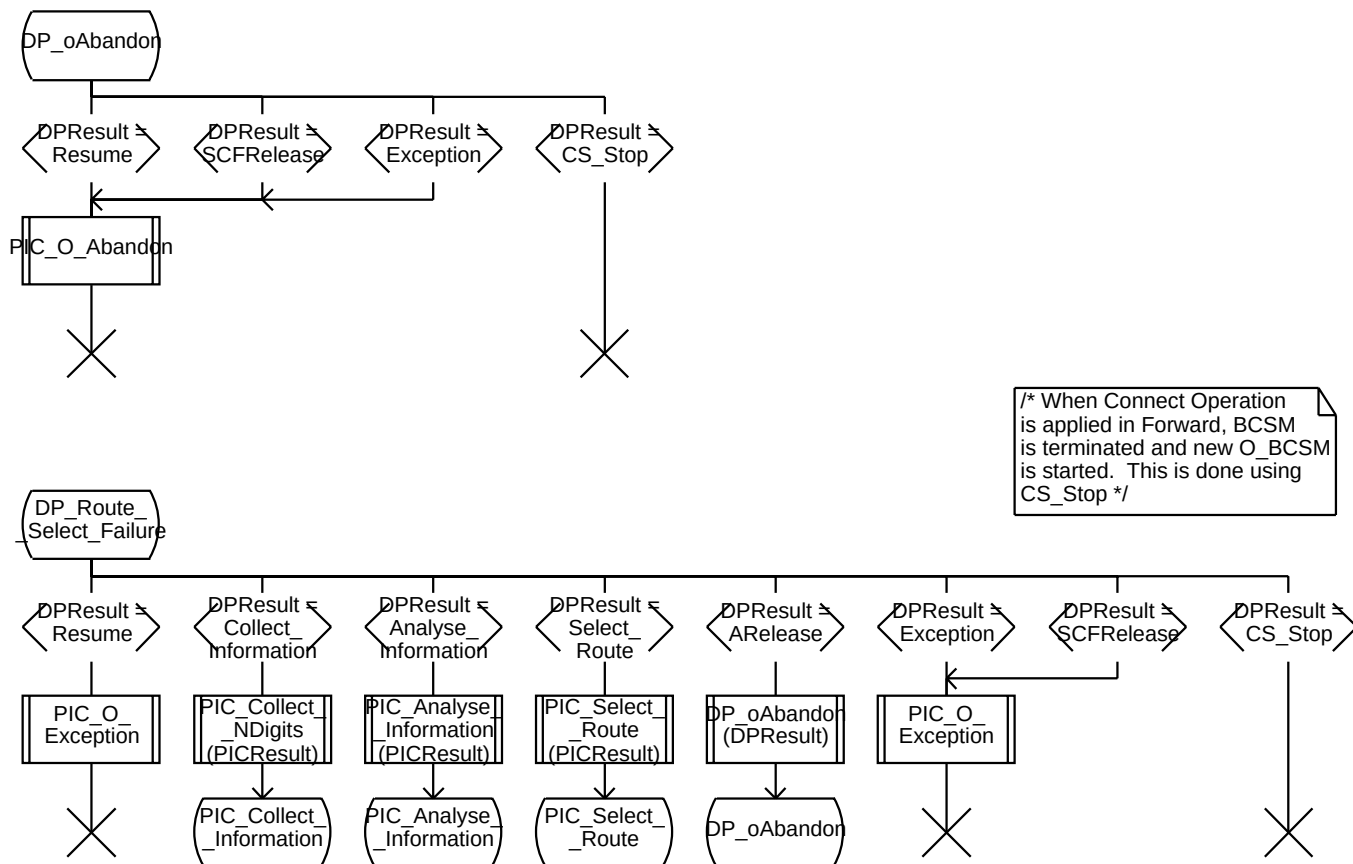




Process Type OriginatingBCSM

13(19)

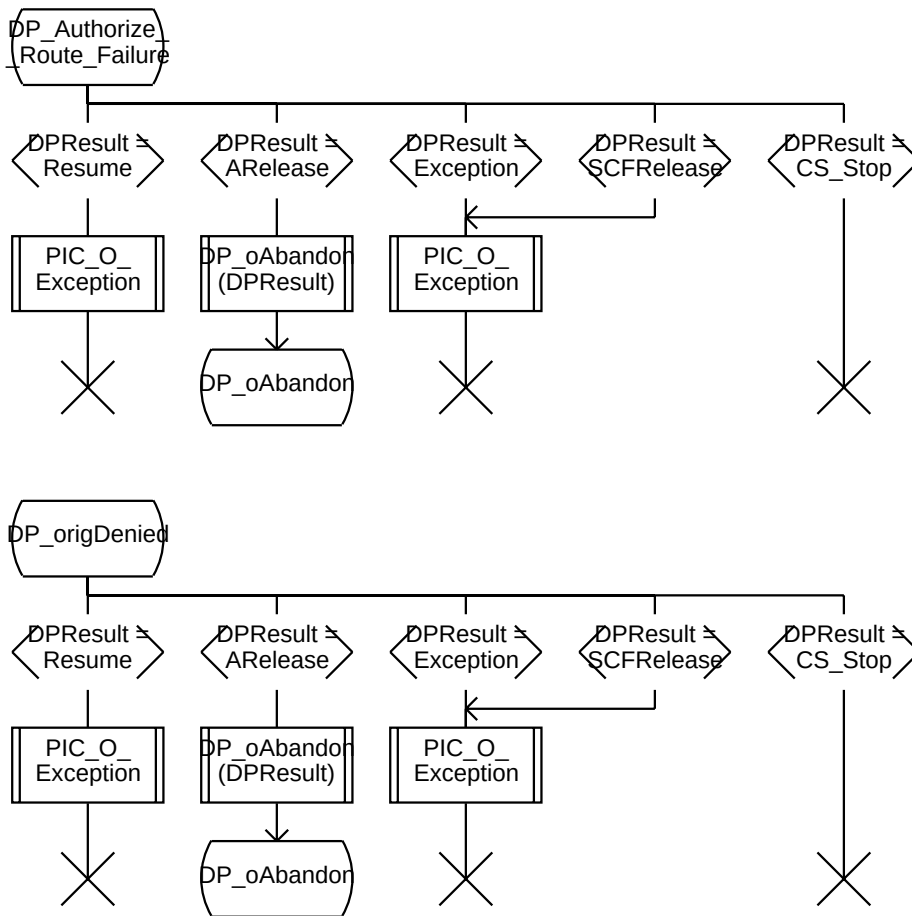
;FPAR obcsmPars OBcsmPars; /* Information passed to the O-BCSM at creation. */



Process Type OriginatingBCSM

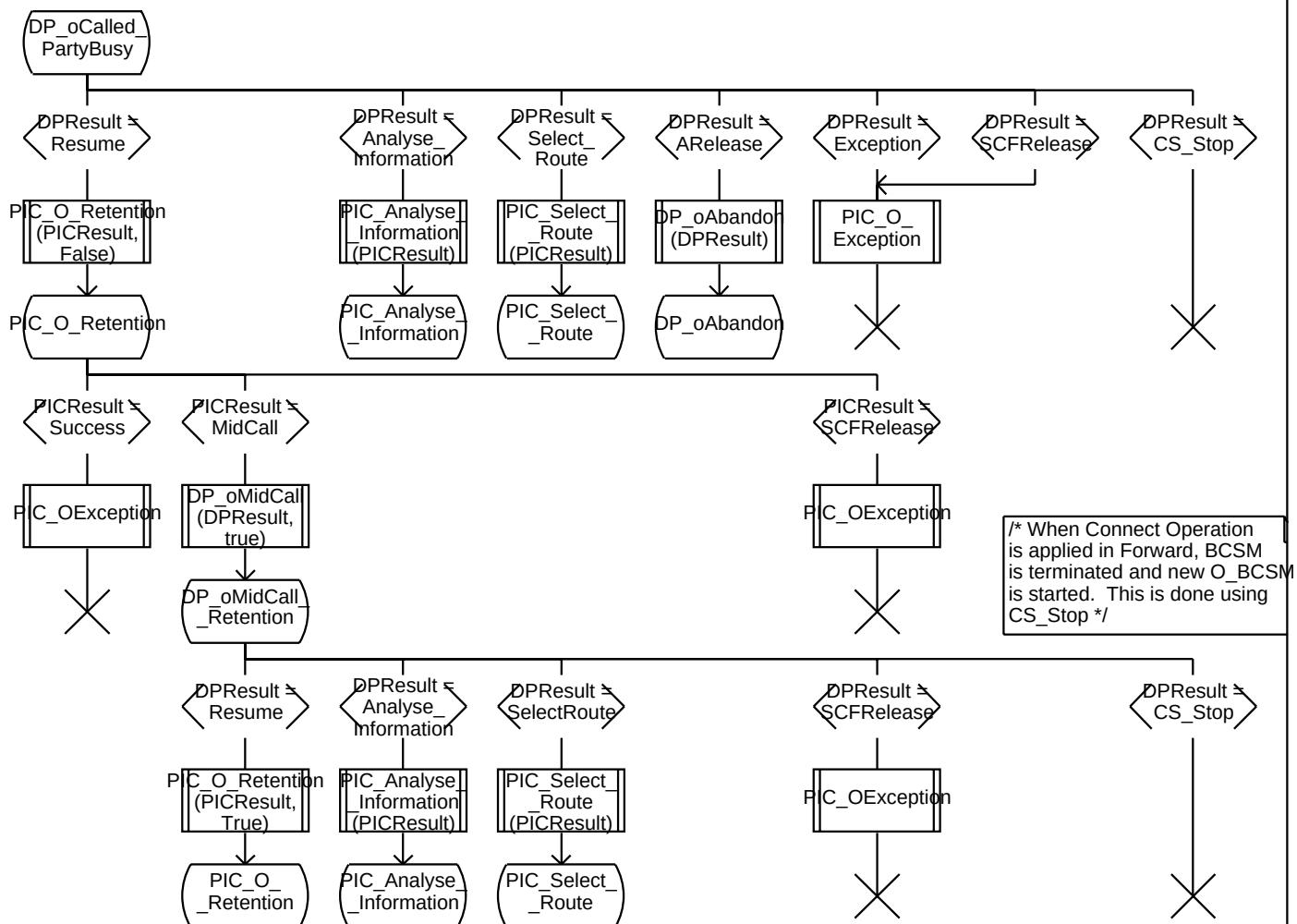
14(19)

;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */\



Process Type OriginatingBCSM 15(19)

;FPAR obcsmpars OBCSMPars; /* Information passed to the O-BCSM at creation. */

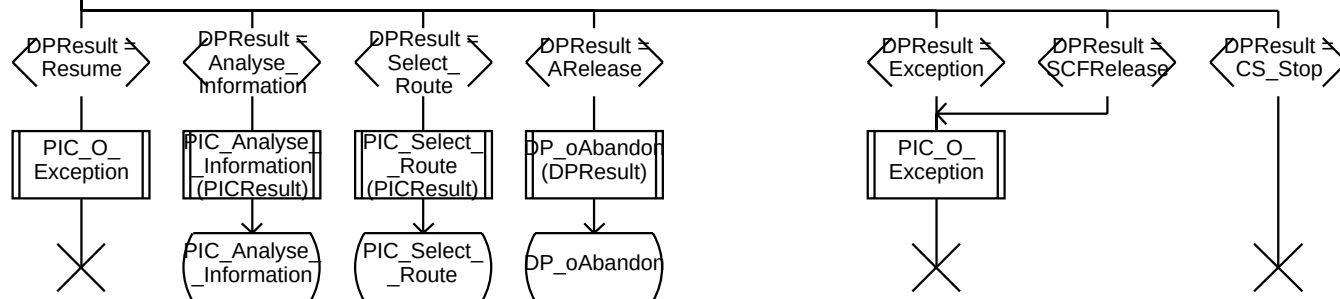


Process Type OriginatingBCSM

16(19)

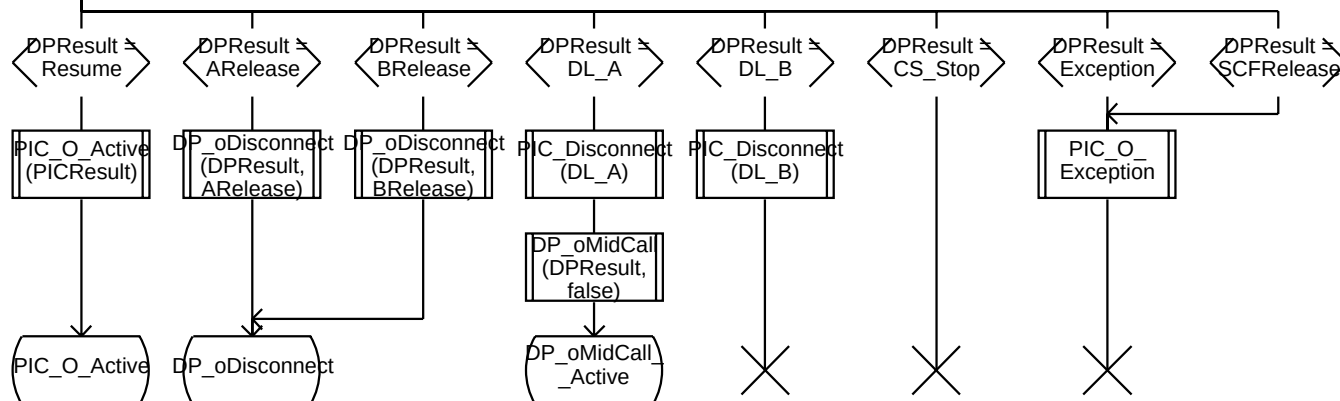
;FPAR obcsmpars OBCSMPars; /* Information passed to the O-BCSM at creation. */

DP_oNoAnswer



/* When the controlling leg is 'stranded', a BCSM is created for the leg. This BCSM is in the wait state of the MidCall DP. When e.g. a Connect operation is applied by the SCF, the BCSM associated with the controlling leg is released. This is in the SDLs done via the CS-Stop instruction from the SSF-FSM.*/

DP_oMidCall_Active



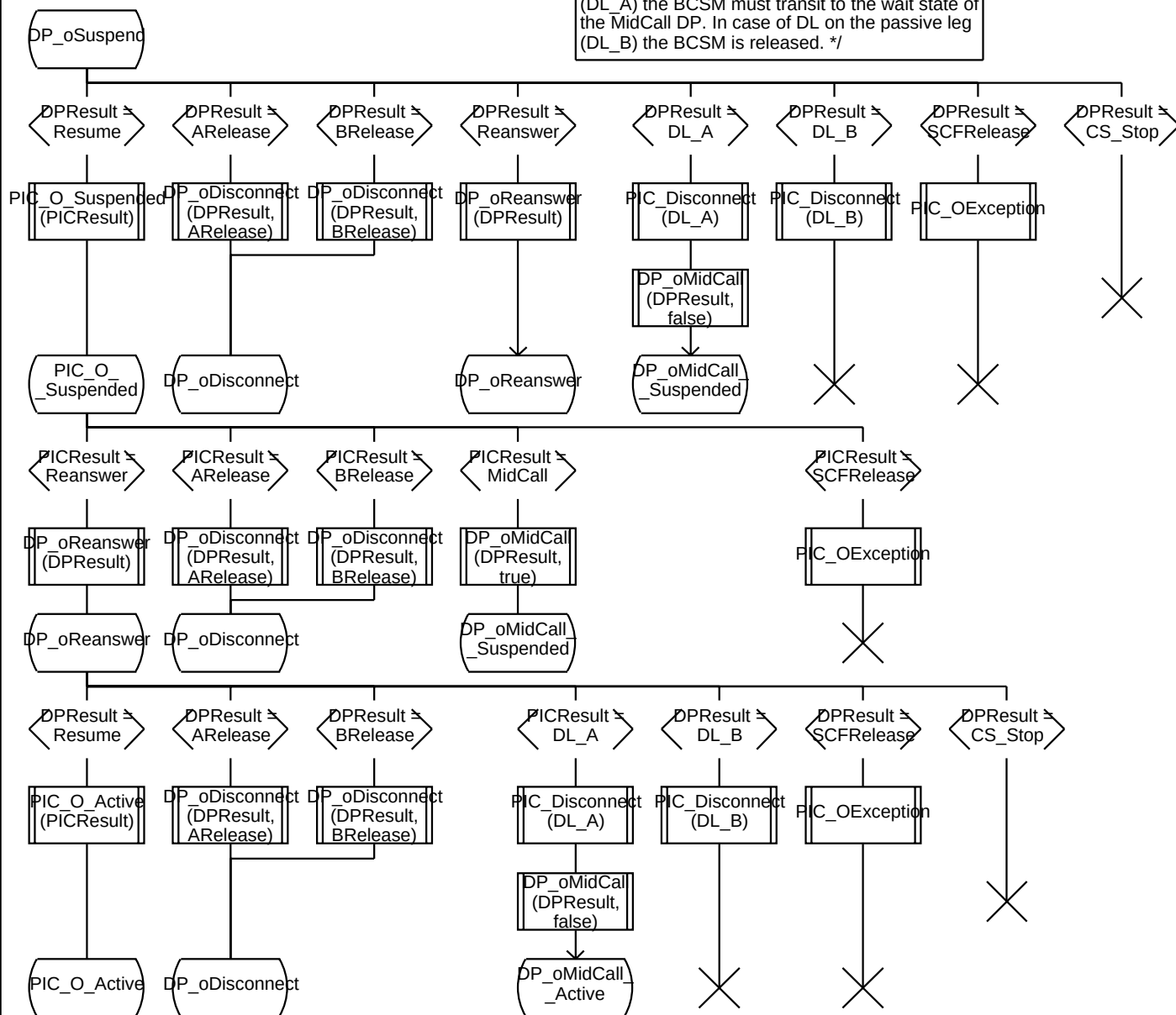
Process Type

OriginatingBCSM

17(19)

;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */

/* The SCF may apply the DisconnectLeg operation. In case of DL on the controlling leg (DL_A) the BCSM must transit to the wait state of the MidCall DP. In case of DL on the passive leg (DL_B) the BCSM is released. */

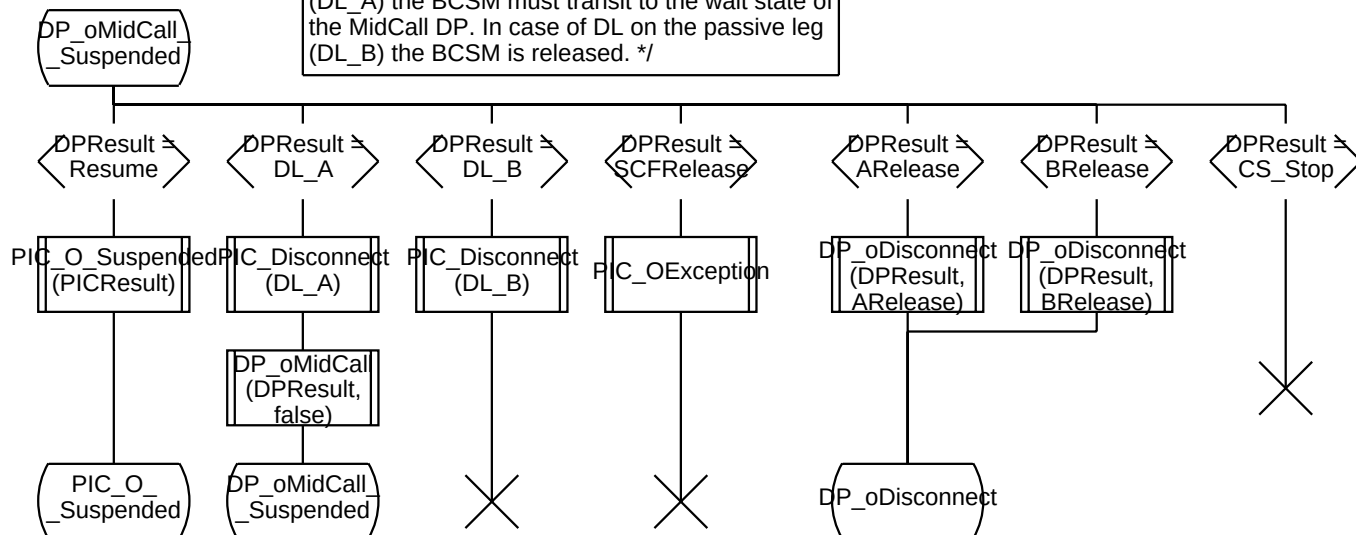


Process Type OriginatingBCSM

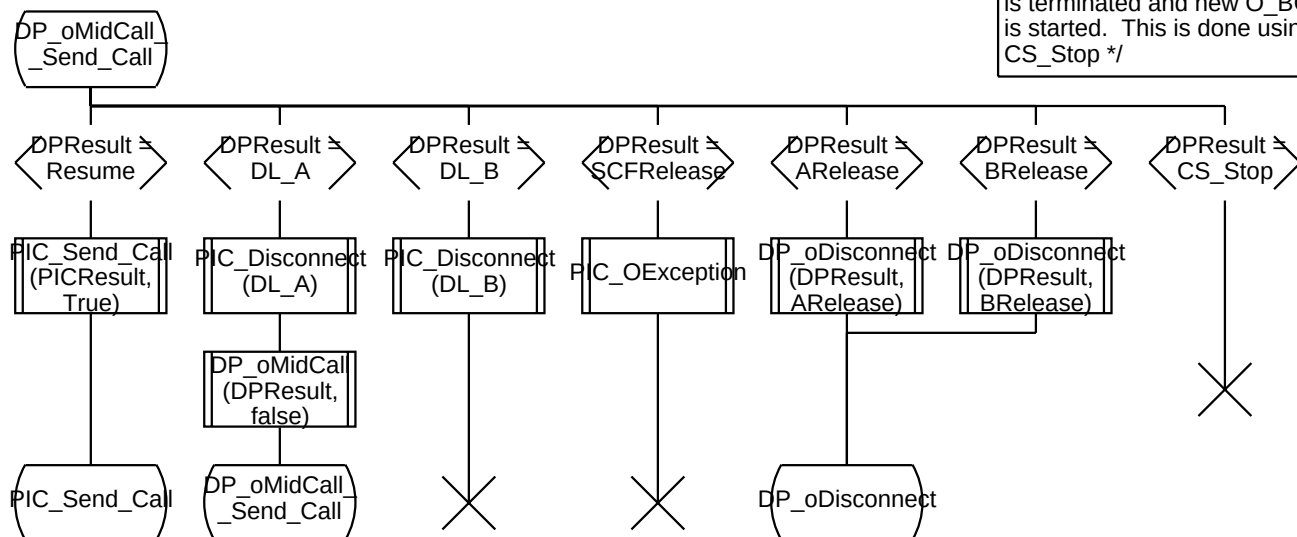
18(19)

;FPAR obcsmPars OBCSMPars; /* Information passed to the O-BCSM at creation. */

/* The SCF may apply the DisconnectLeg operation. In case of DL on the controlling leg (DL_A) the BCSM must transit to the wait state of the MidCall DP. In case of DL on the passive leg (DL_B) the BCSM is released. */



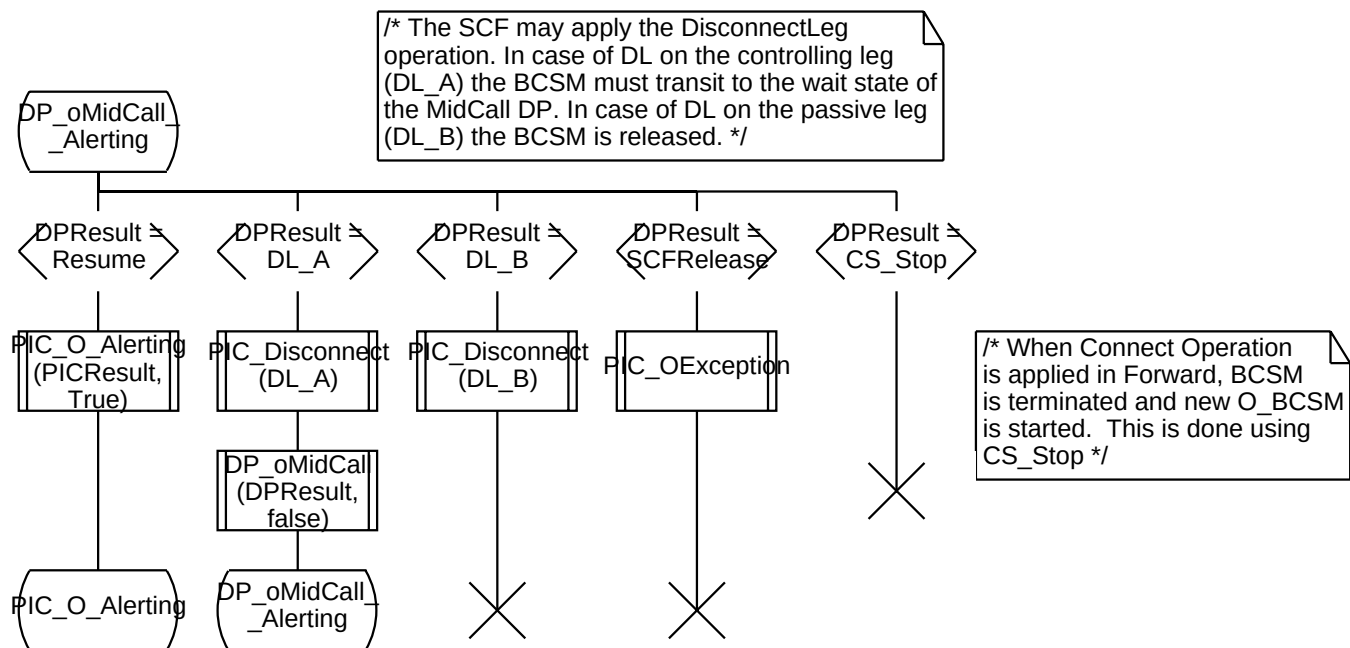
/* When Connect Operation is applied in Forward, BCSM is terminated and new O_BCSM is started. This is done using CS_Stop */



Process Type OriginatingBCSM

19(19)

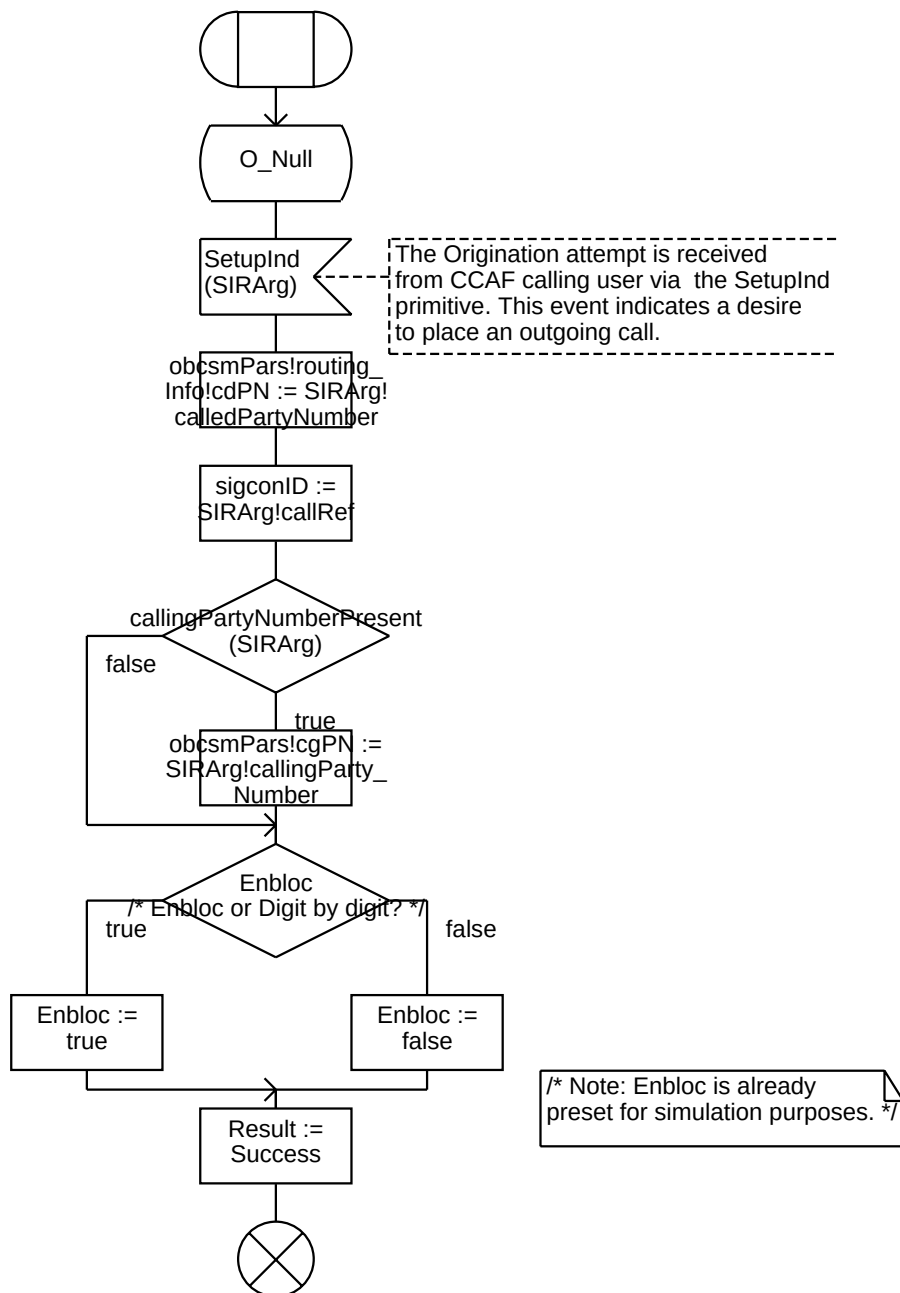
;FPAR obcsmpars OBCSMPars; /* Information passed to the O-BCSM at creation. */



Procedure PIC_O_Null

1(1)

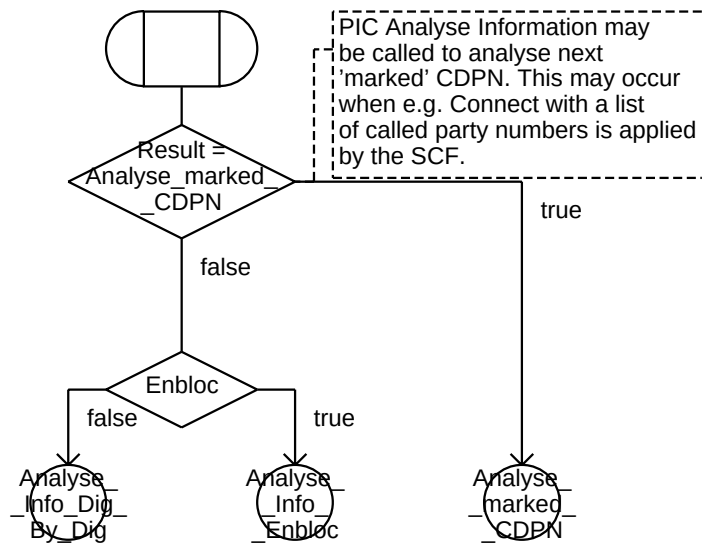
FPAR
IN/OUT Result PICResultType;



Procedure PIC_Analyse_Information

1(5)

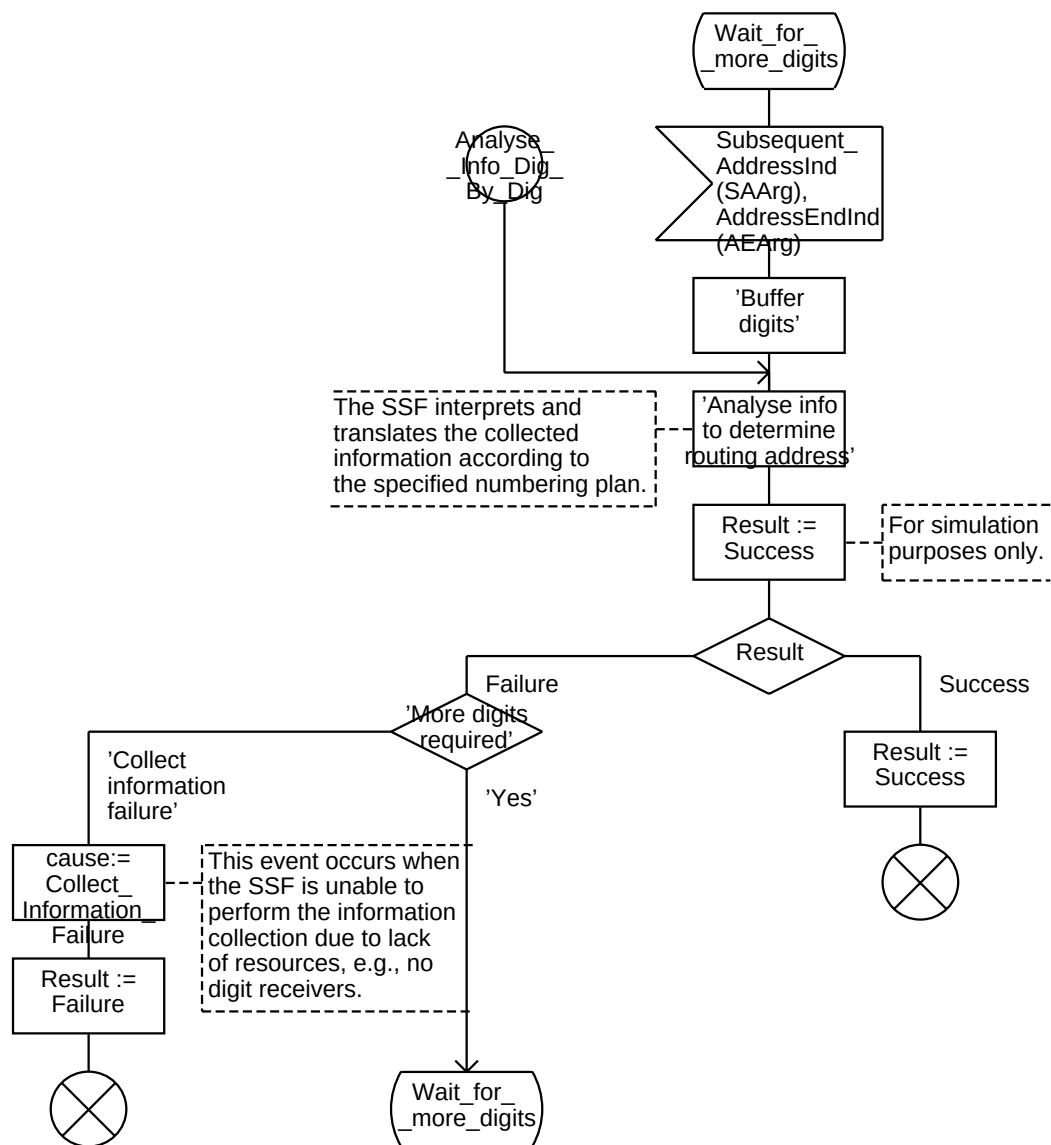
FPAR
IN/OUT Result PICResultType;



Procedure PIC_Analyse_Information

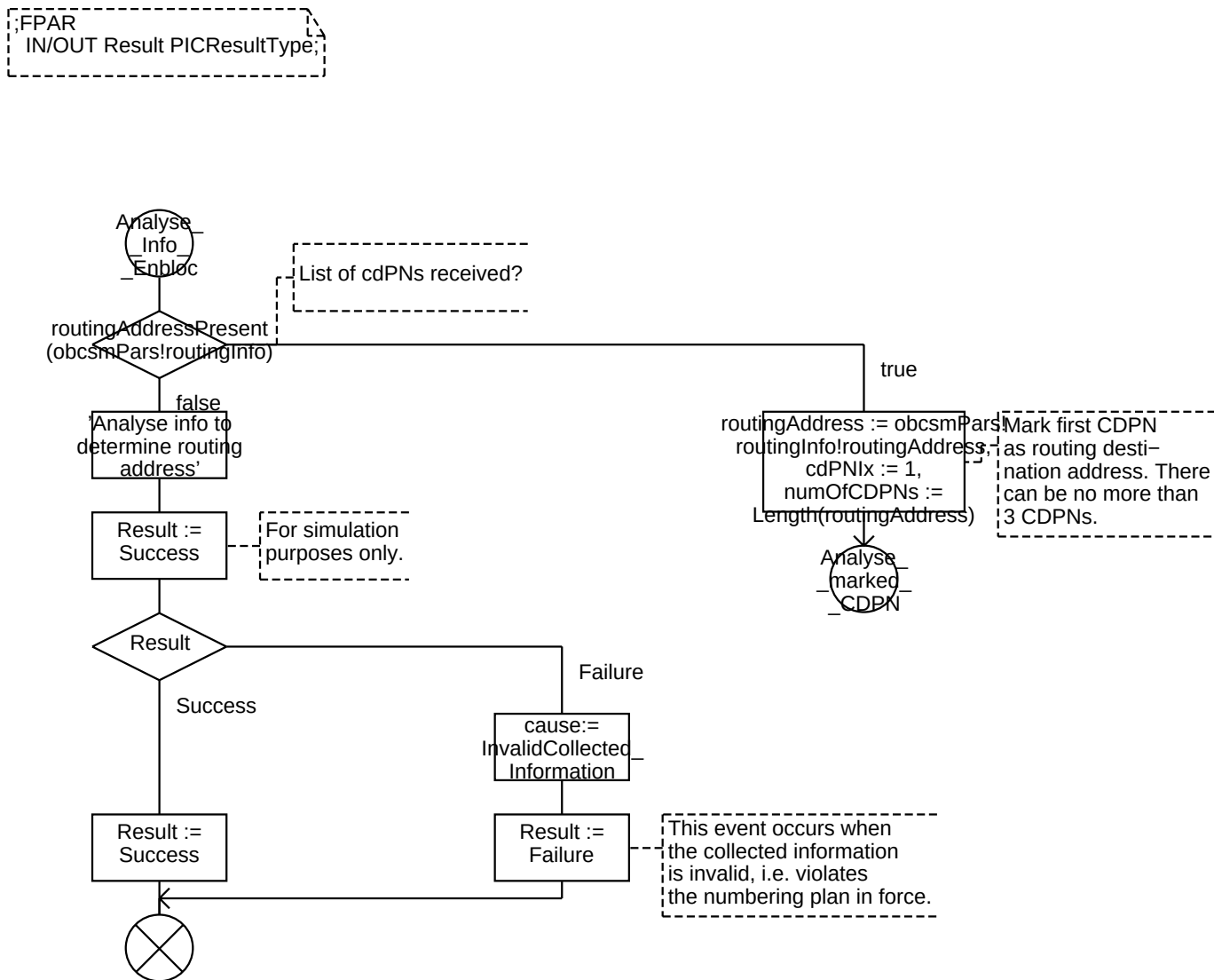
2(5)

FPAR
 IN/OUT Result PICResultType;



Procedure PIC_Analyse_Information

3(5)



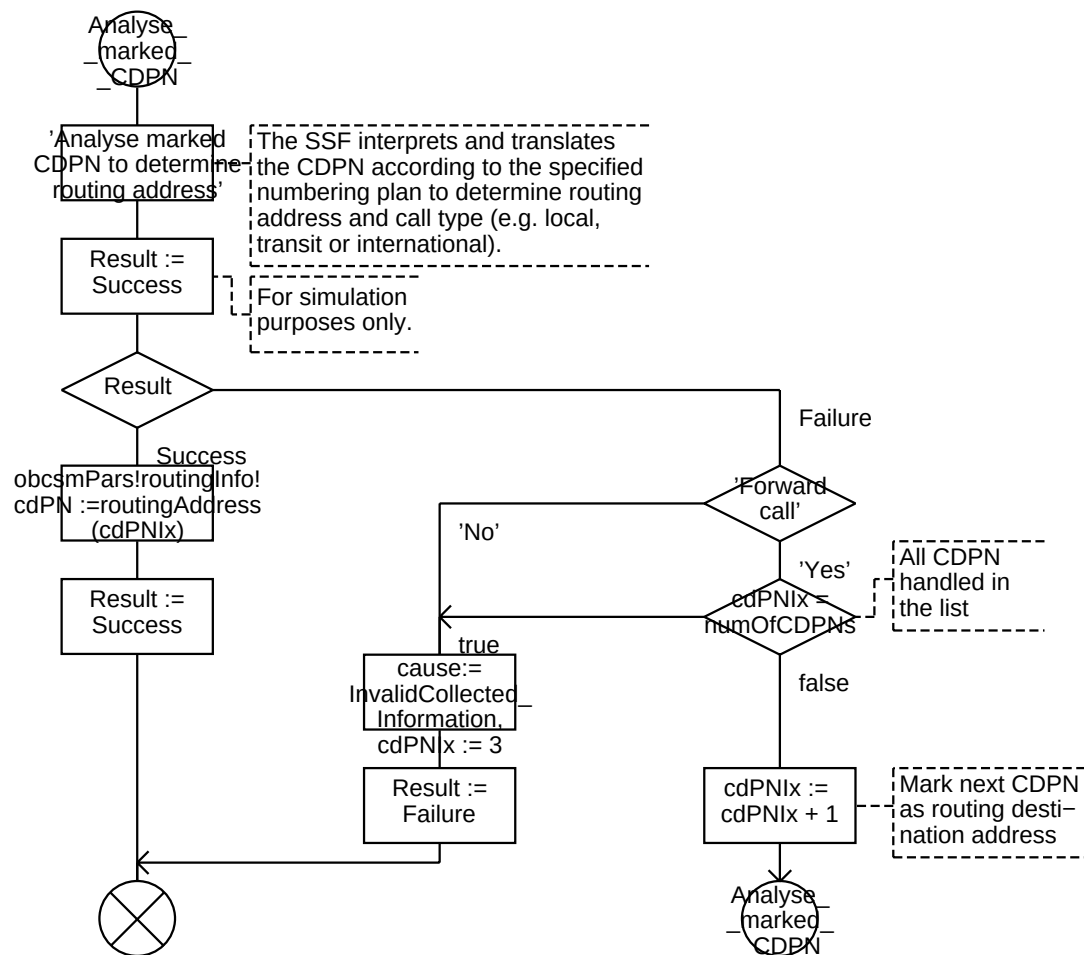
Procedure PIC_Analyse_Information

4(5)

```

;FPAR
IN/OUT Result PICResultType;

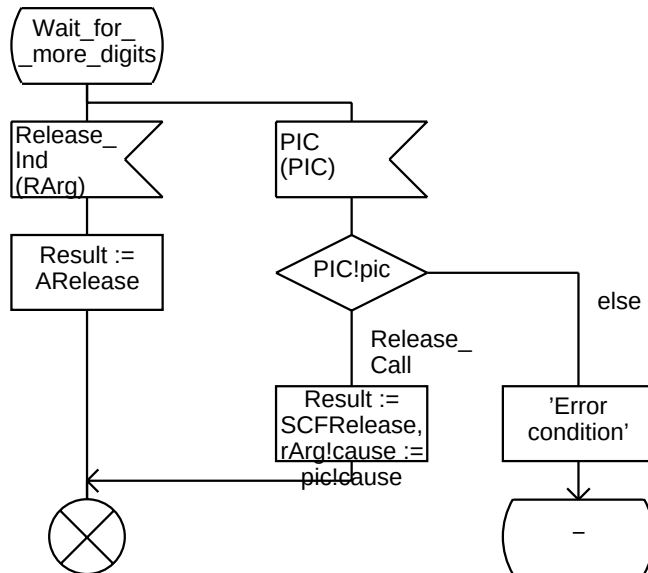
```



Procedure PIC_Analyse_Information

5(5)

FPAR
IN/OUT Result PICResultType;

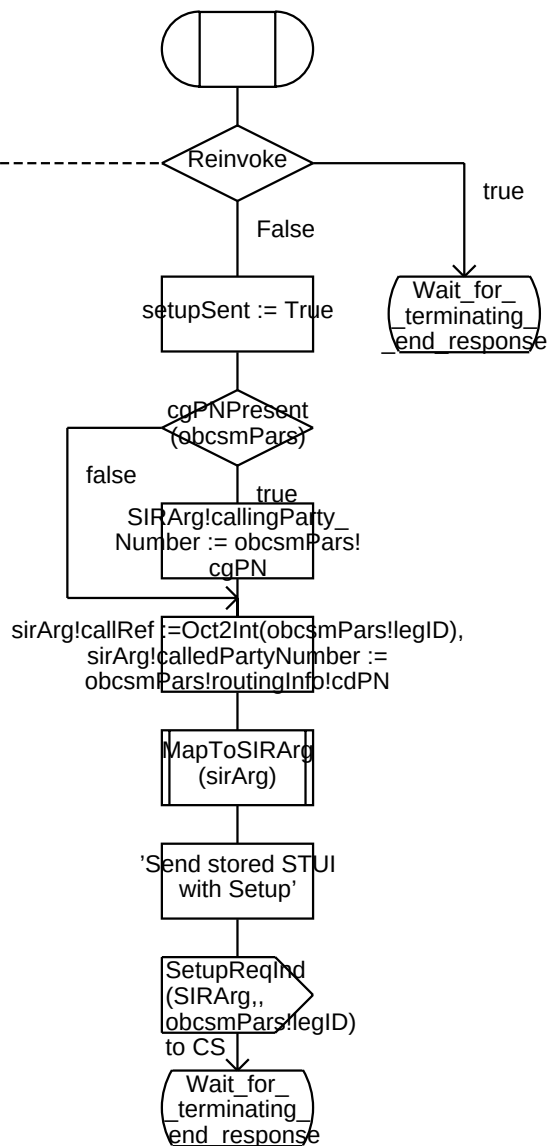


Procedure PIC_Send_Call

1(5)

FPAR
IN/OUT Result PICResultType,
IN Reinvoke Boolean;

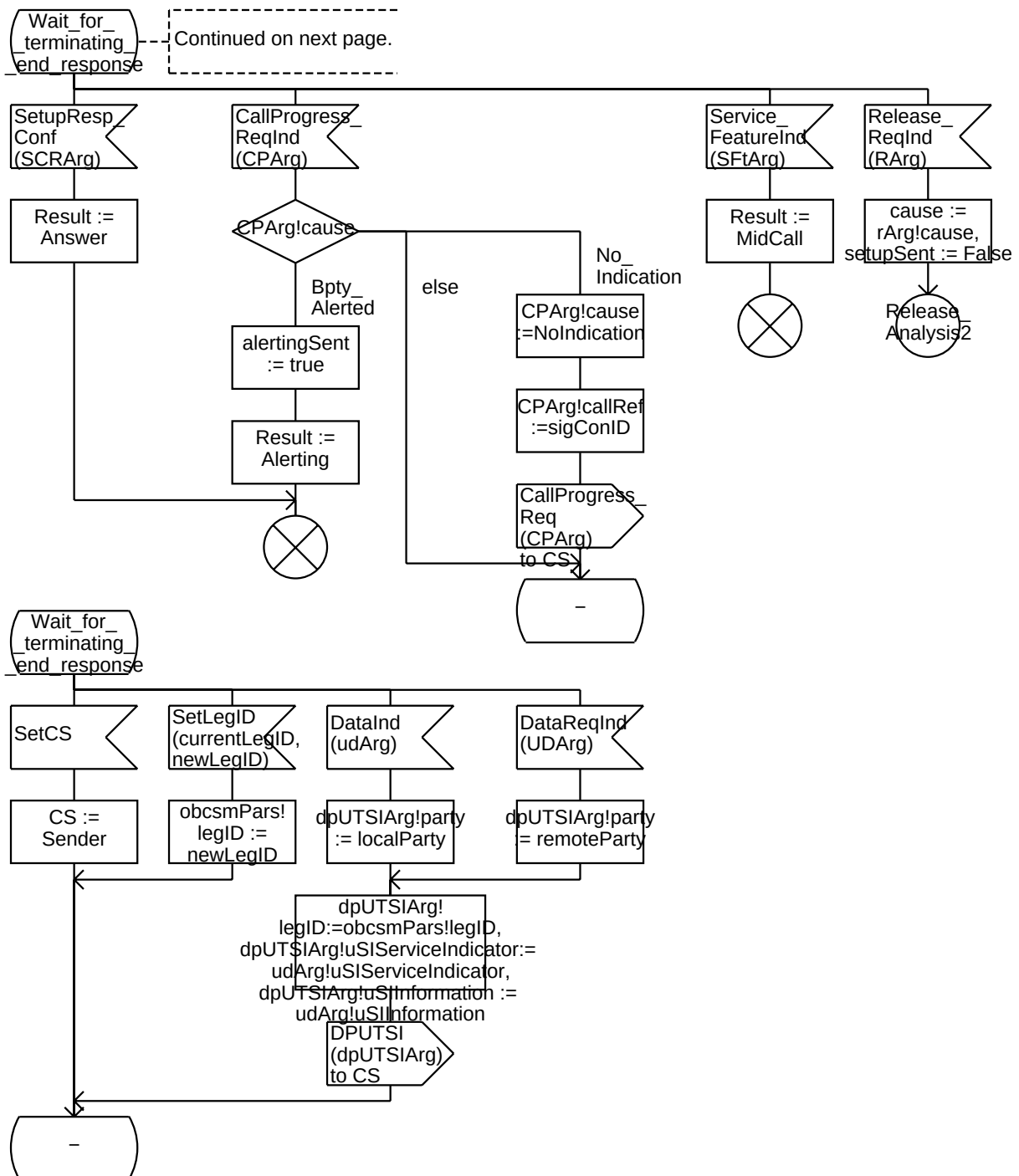
The call processing may temporarily exit the SendCall PIC (e.g. due to a ServiceFeatureInd generating a MidCall DP). When call processing resumes, the PIC SendCall is reinvoked. This should not generate a new SetupReqInd.



Procedure PIC_Send_Call

2(5)

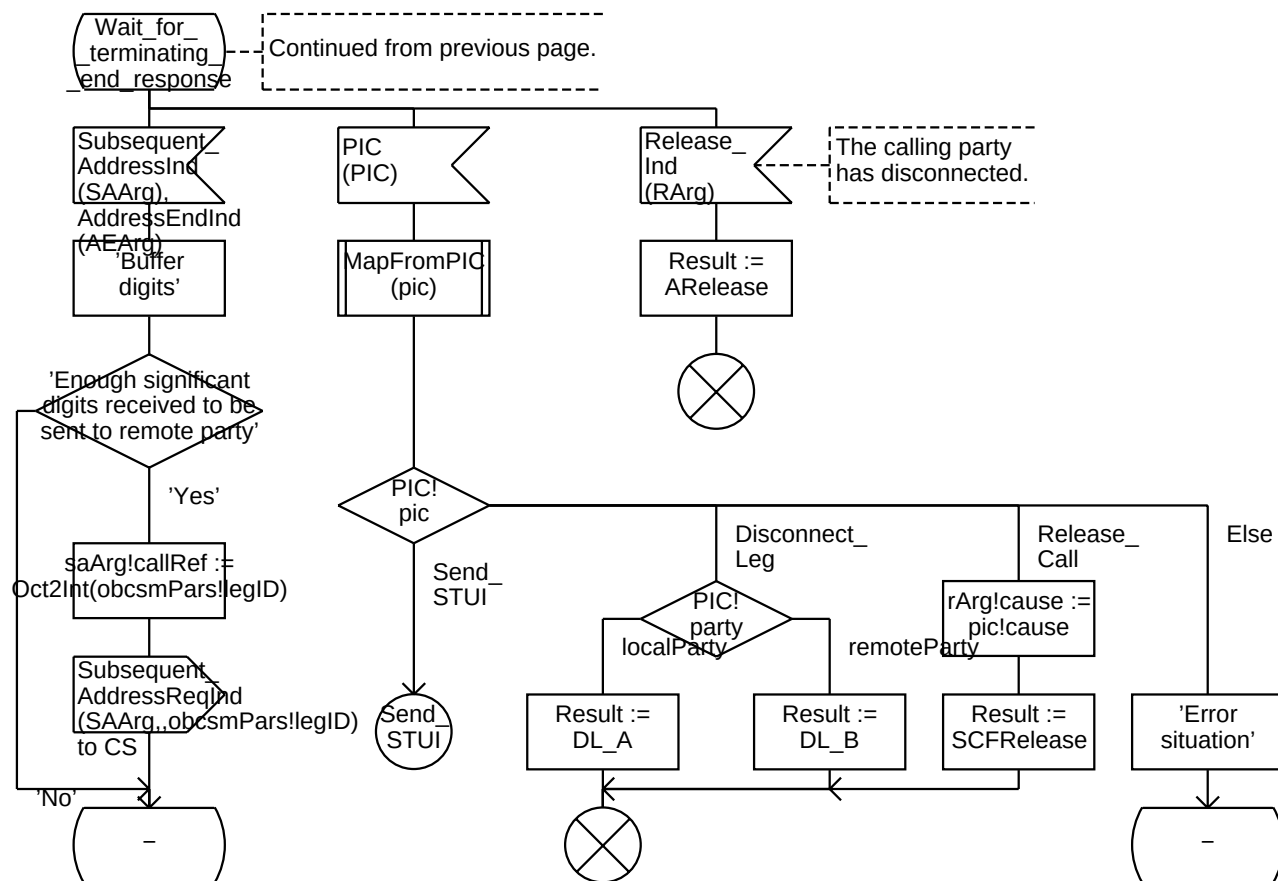
FPAR
IN/OUT Result PICResultType,
IN Reinvoke Boolean;



Procedure PIC_Send_Call

3(5)

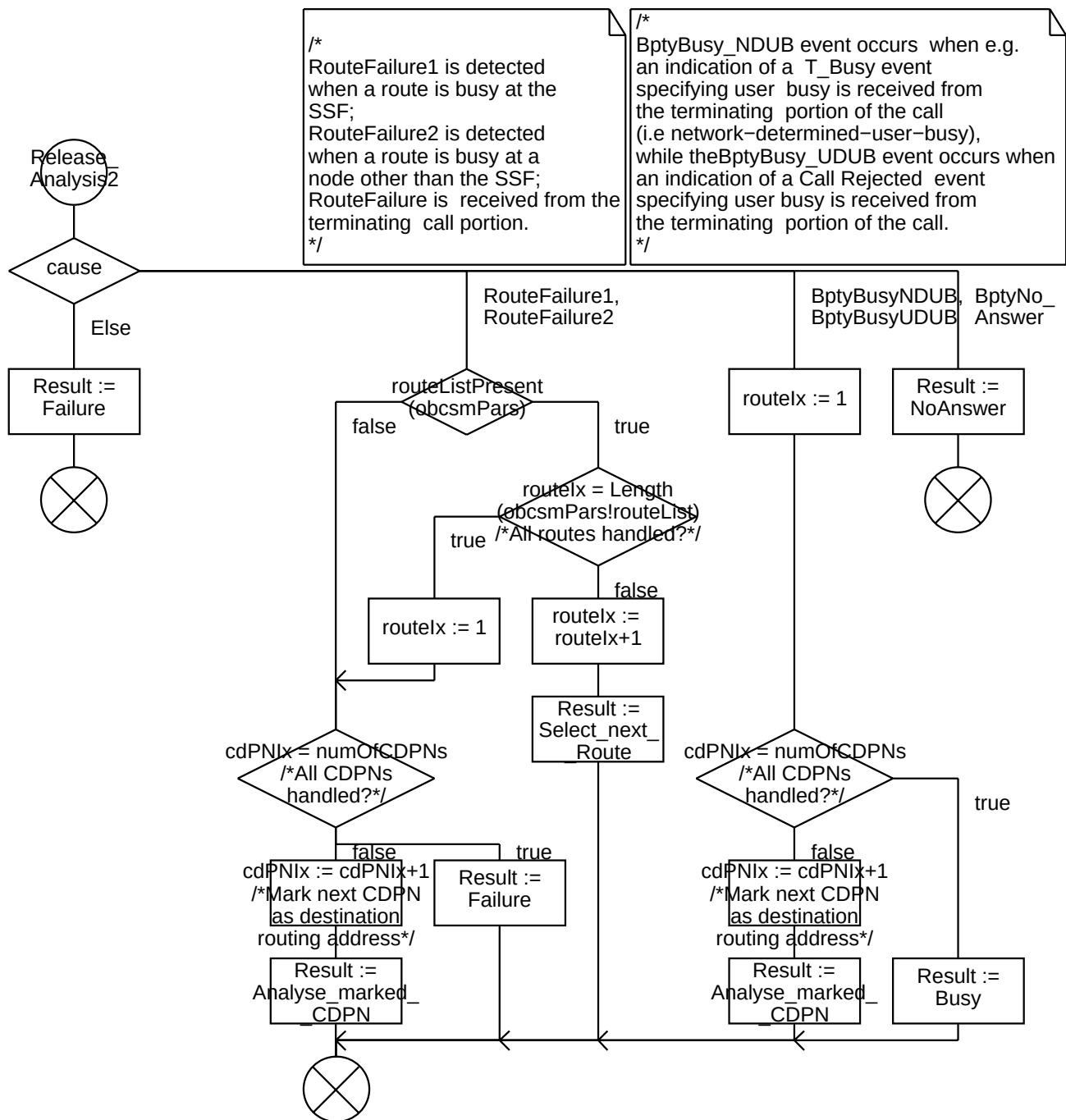
FPAR
IN/OUT Result PICResultType,
IN Reinvoke Boolean;



Procedure PIC_Send_Call

4(5)

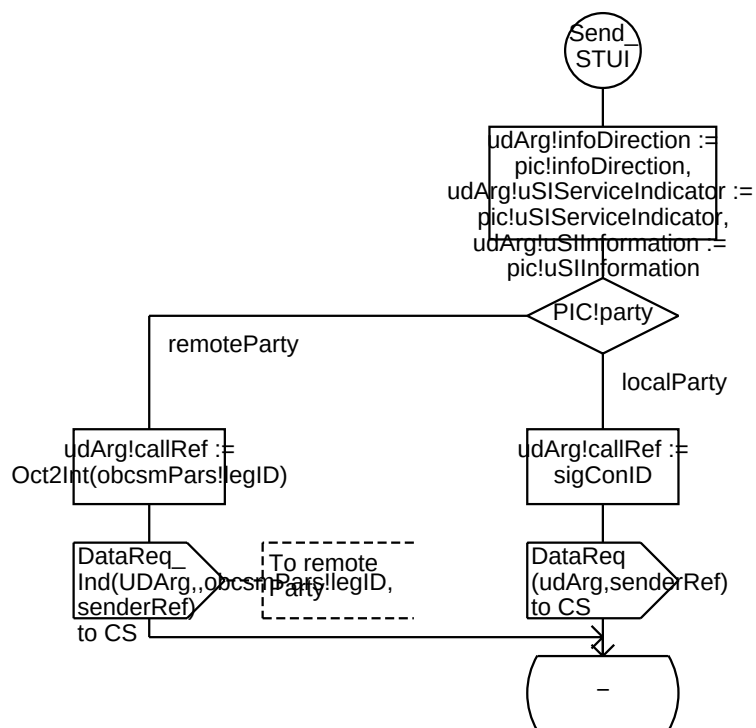
FPAR
IN/OUT Result PICResultType;
IN Reinvoke Boolean;



Procedure PIC_Send_Call

5(5)

FPAR
IN/OUT Result PICResultType,
IN Reinvoke Boolean;



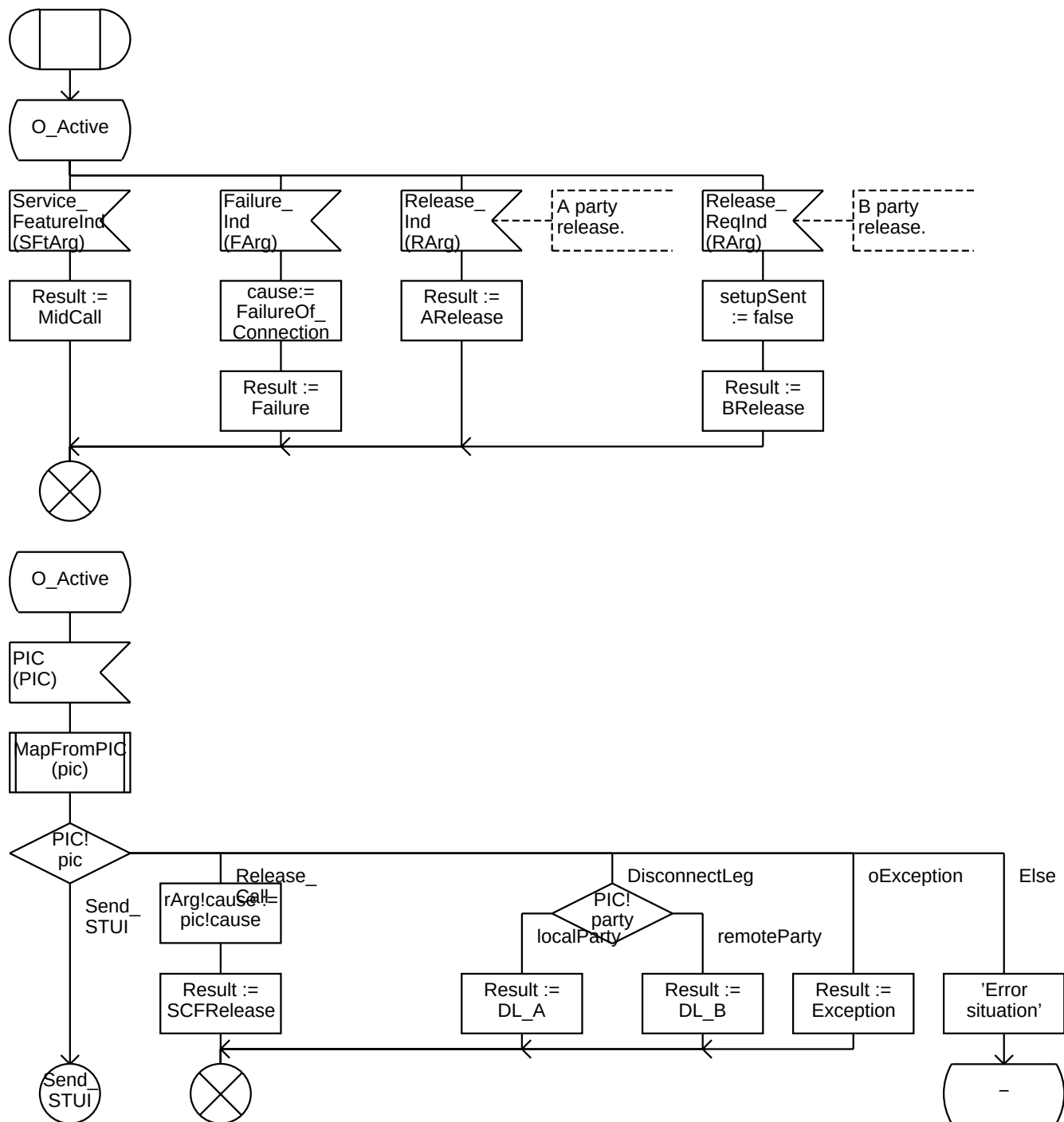
Procedure PIC_O_Active

1(2)

```

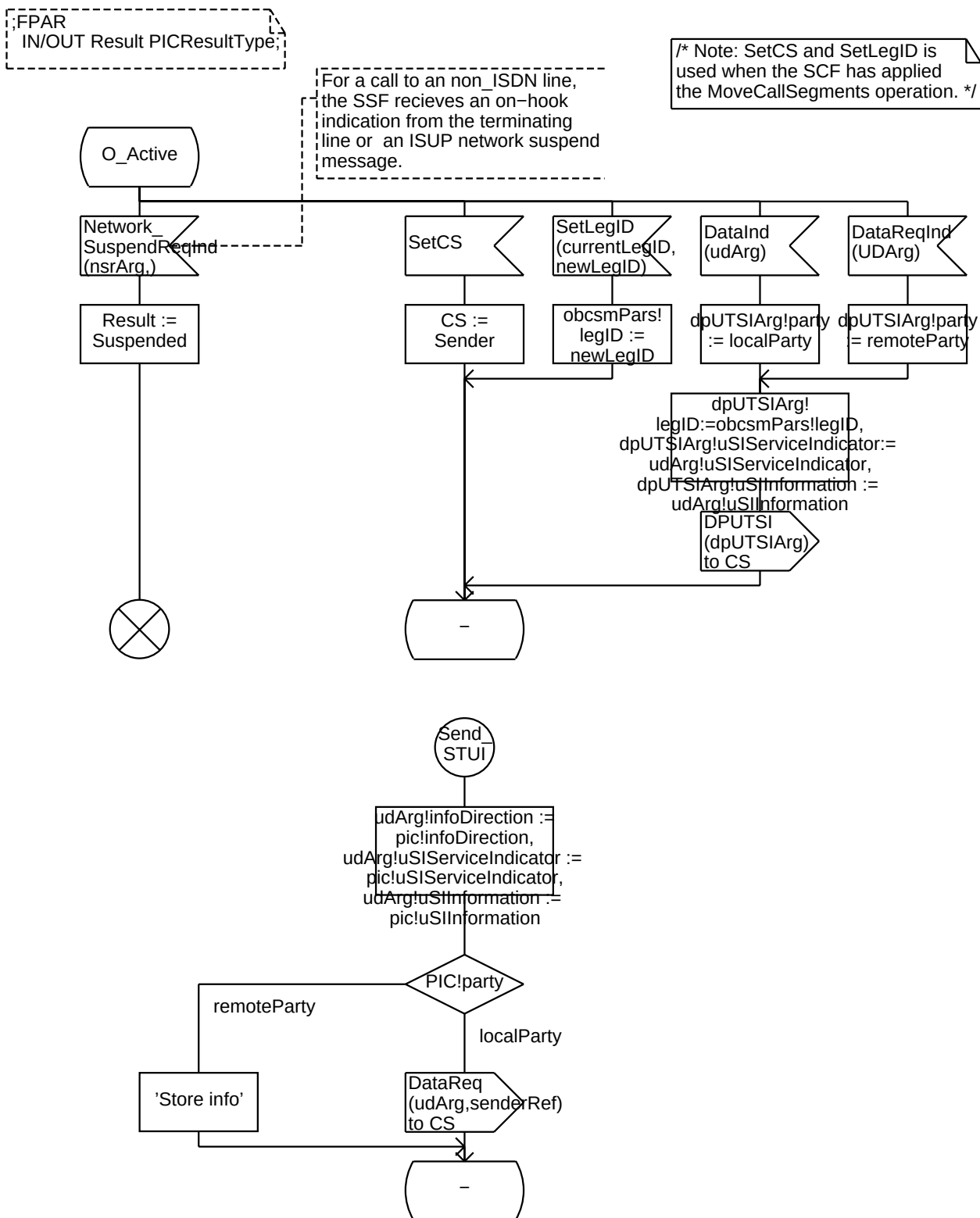
;FPAR
IN/OUT Result PICResultType;

```



Procedure PIC_O_Active

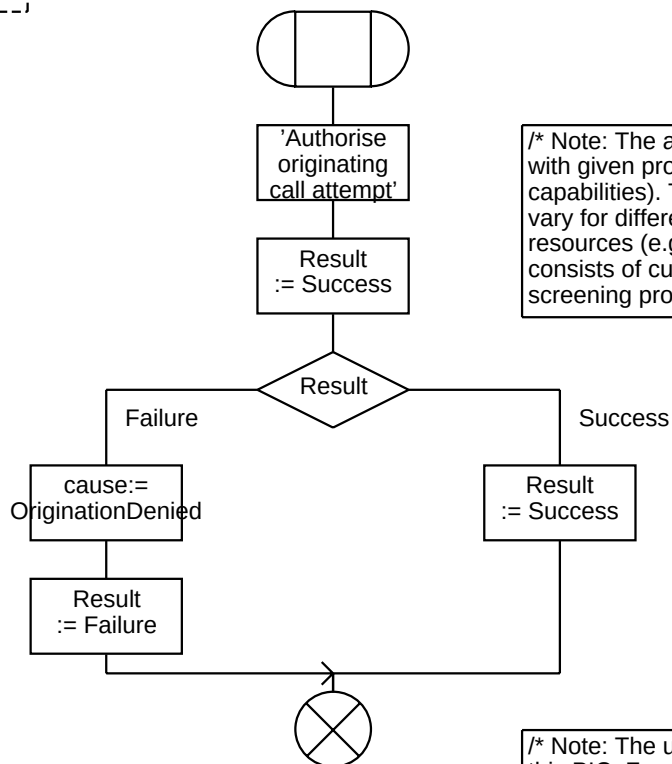
2(2)



Procedure PIC_Authorise_Origination_Attempt

1(1)

FPAR
IN/OUT Result PICResultType;



/* Note: The authority of an user to place a call with given properties is verified (e.g. bearer capabilities). The type of authorisation may vary for different types of originating resources (e.g., for lines or trunks) and consists of current originating authorisation screening procedures. */

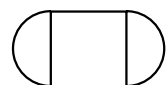
/* Note: The user may abandon the call during this PIC. For reasons of simplicity this is not modelled in the SDLs (it would have required the introduction of a state with two inputs: ReleaseInd and None). */

Procedure PIC_Select_Route

1(1)

```

;FPAR
IN/OUT Result PICResultType;
    
```



'Interpret
routing address
and call type'

To select the outgoing route, the SSF interprets the analysis results. The routing address and call type is analysed. The next route is being selected. This may involve sequential search in a route list, translating a directory number into a physical port address.

Result :=
Success

For simulation
purposes only.

Result

Success

failure

routeListPresent
(obcsmPars)

true

false

routeIx = Length
(obcsmPars!routeList)
/*All routes handled?*/

true

'Select next route
in the route list'

routeIx :=
routeIx+1

cdPNIx = numOfCDPNs
/*All CDPNs handled?*/

true

false

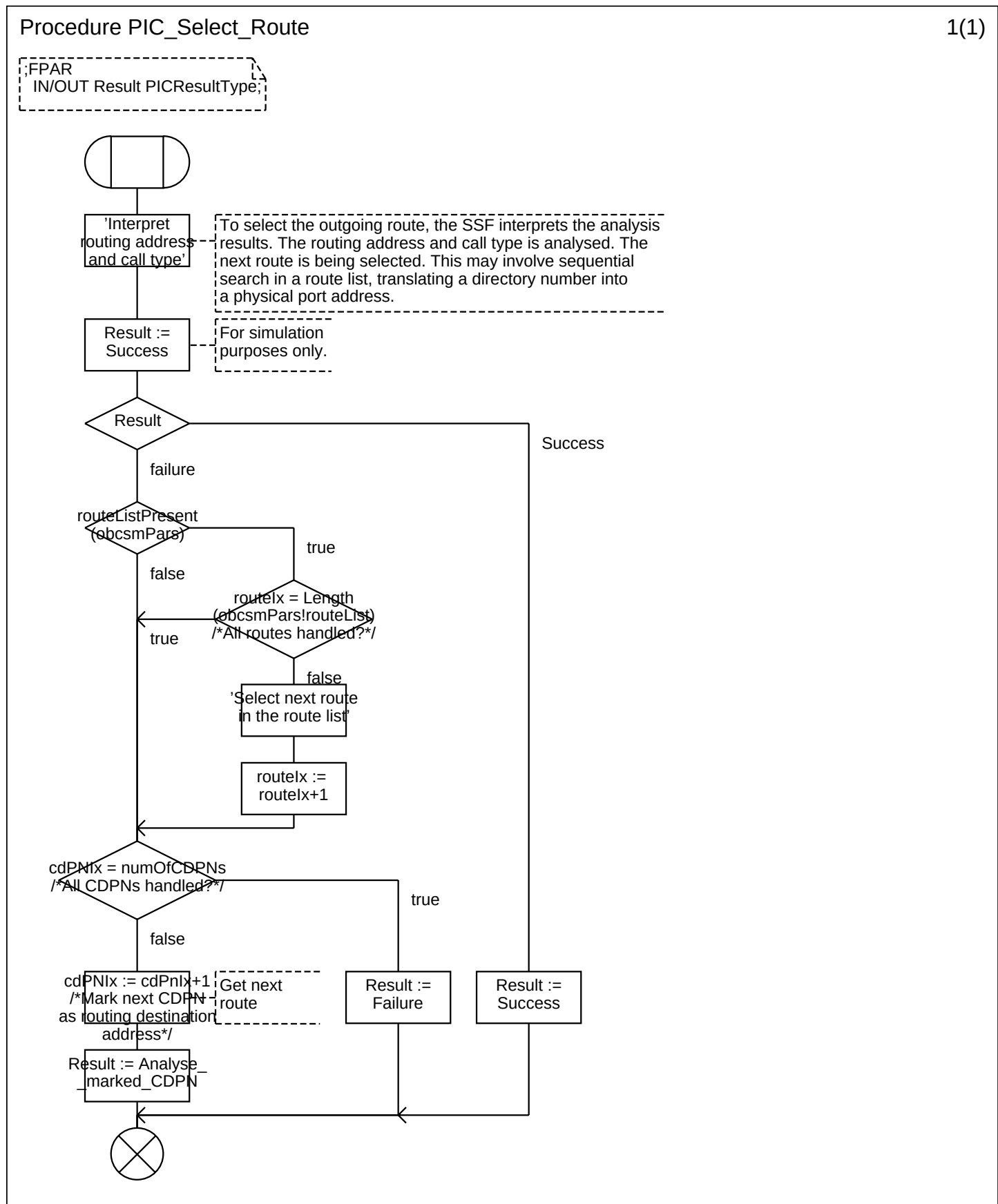
cdPNIx := cdPNIx+1
/*Mark next CDPN
as routing destination
address*/

Get next
route

Result :=
Failure

Result :=
Success

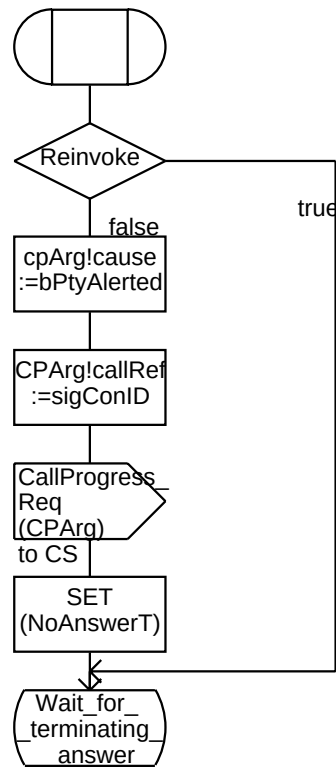
Result := Analyse_
marked_CDPN



Procedure PIC_O_Alerting

1(5)

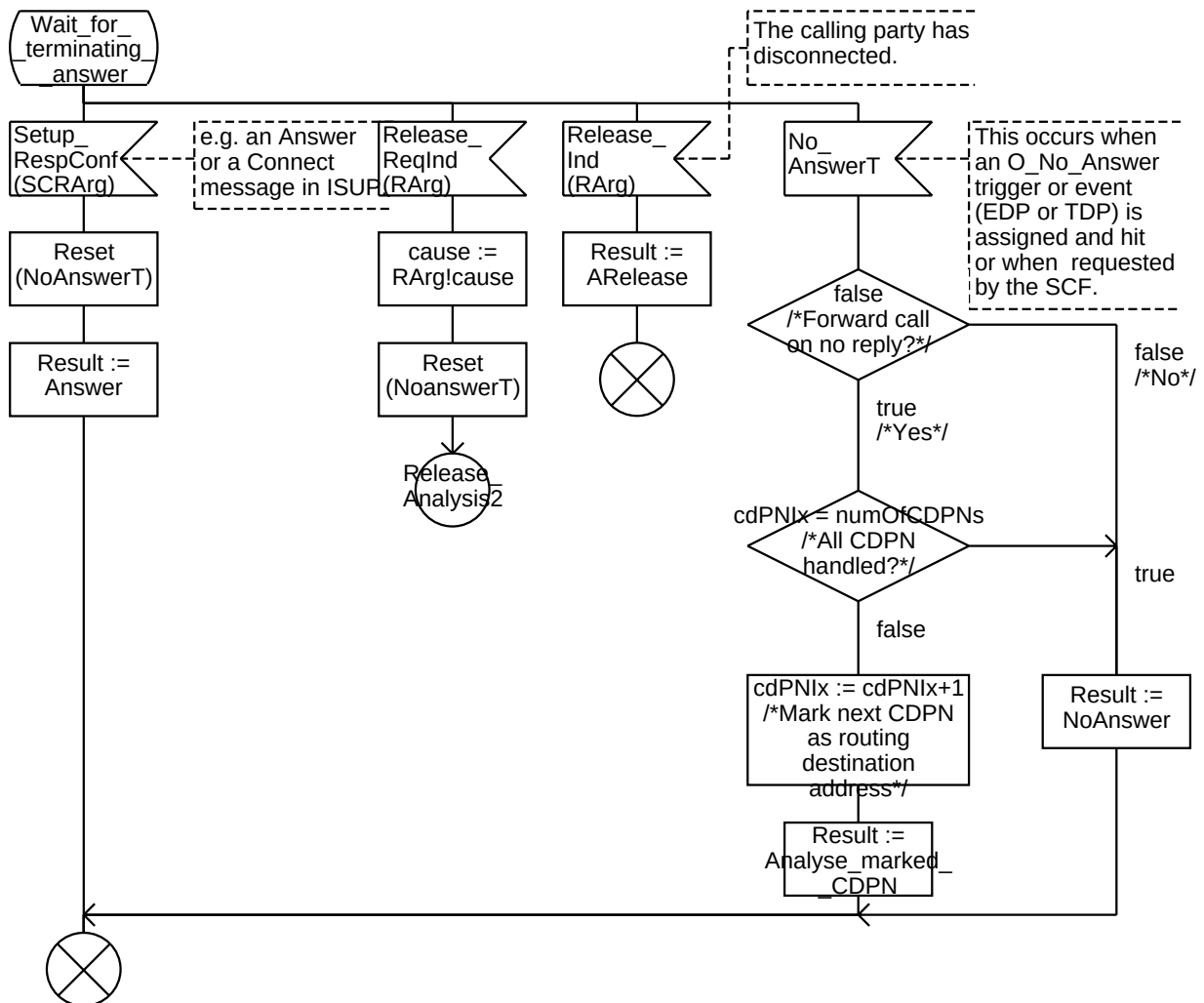
;FPAR
IN/OUT Result PICResultType,
IN Reinvoke Boolean;



Procedure PIC_O_Alerting

2(5)

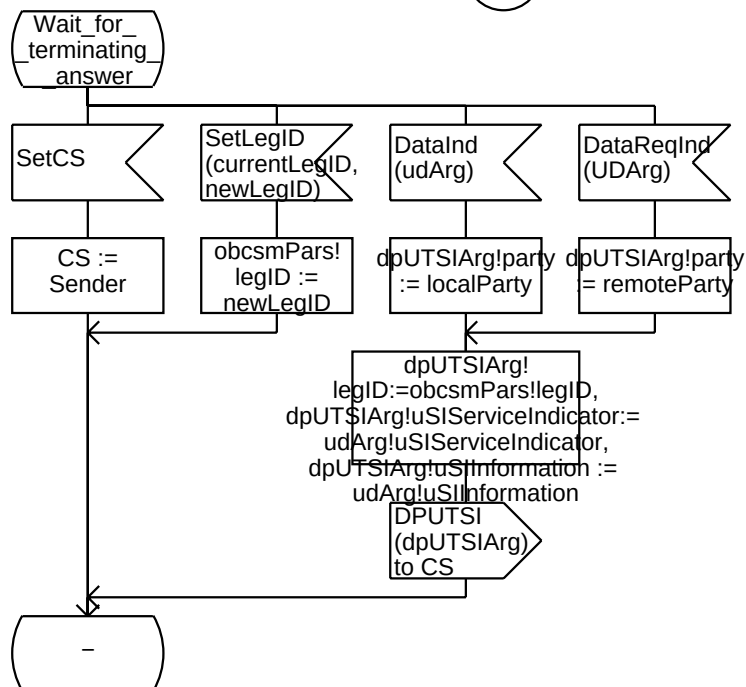
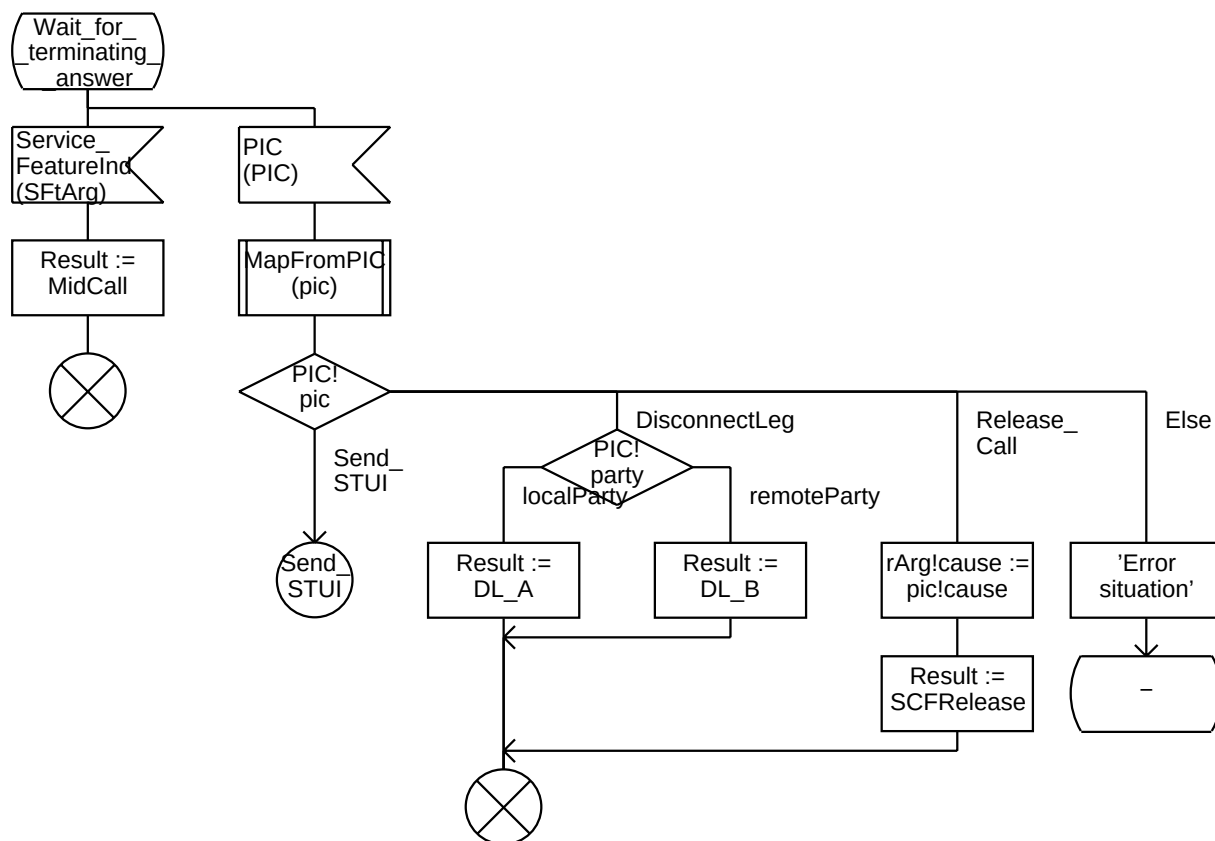
FPAR
IN/OUT Result PICResultType,
IN Reinvoke Boolean;



Procedure PIC_O_Alerting

3(5)

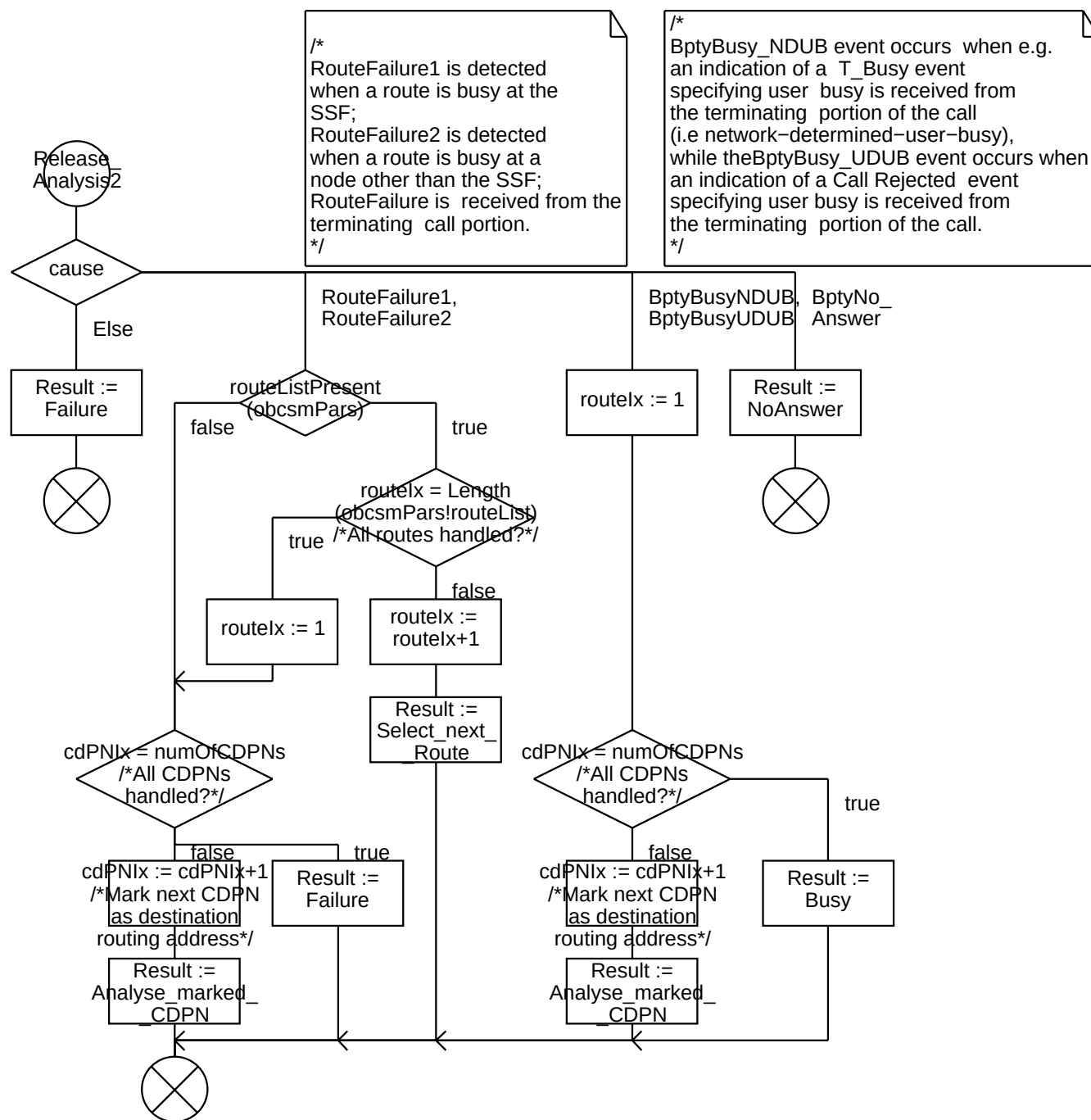
FPAR
IN/OUT Result PICResultType,
IN Reinvoke Boolean;



Procedure PIC_O_Alerting

4(5)

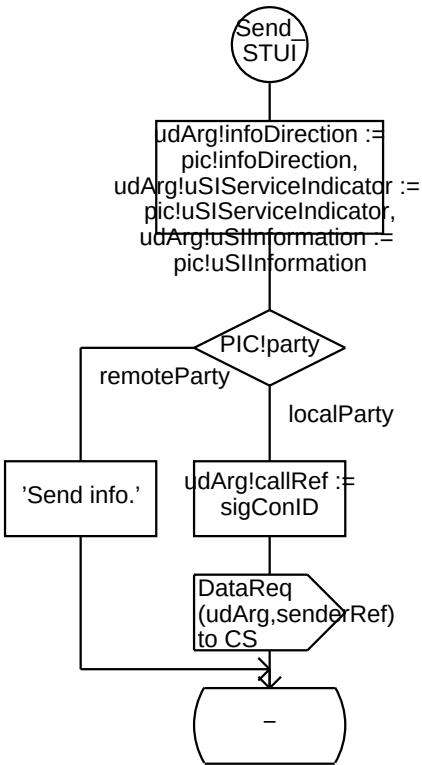
FPAR
IN/OUT Result PICResultType;
IN Reinvoke Boolean;



Procedure PIC_O_Alerting

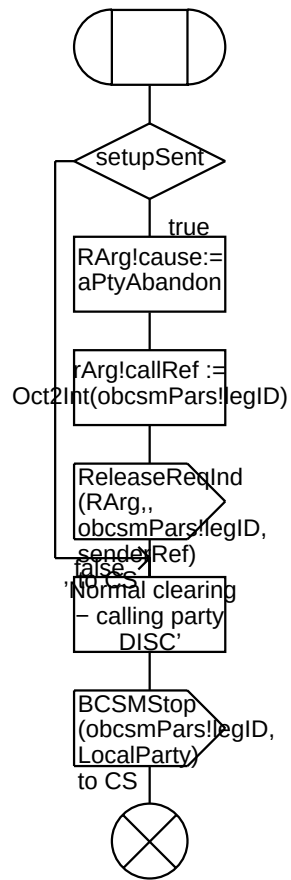
5(5)

FPAR
IN/OUT Result PICResultType,
IN Reinvoke Boolean;



Procedure PIC_O_Abandon

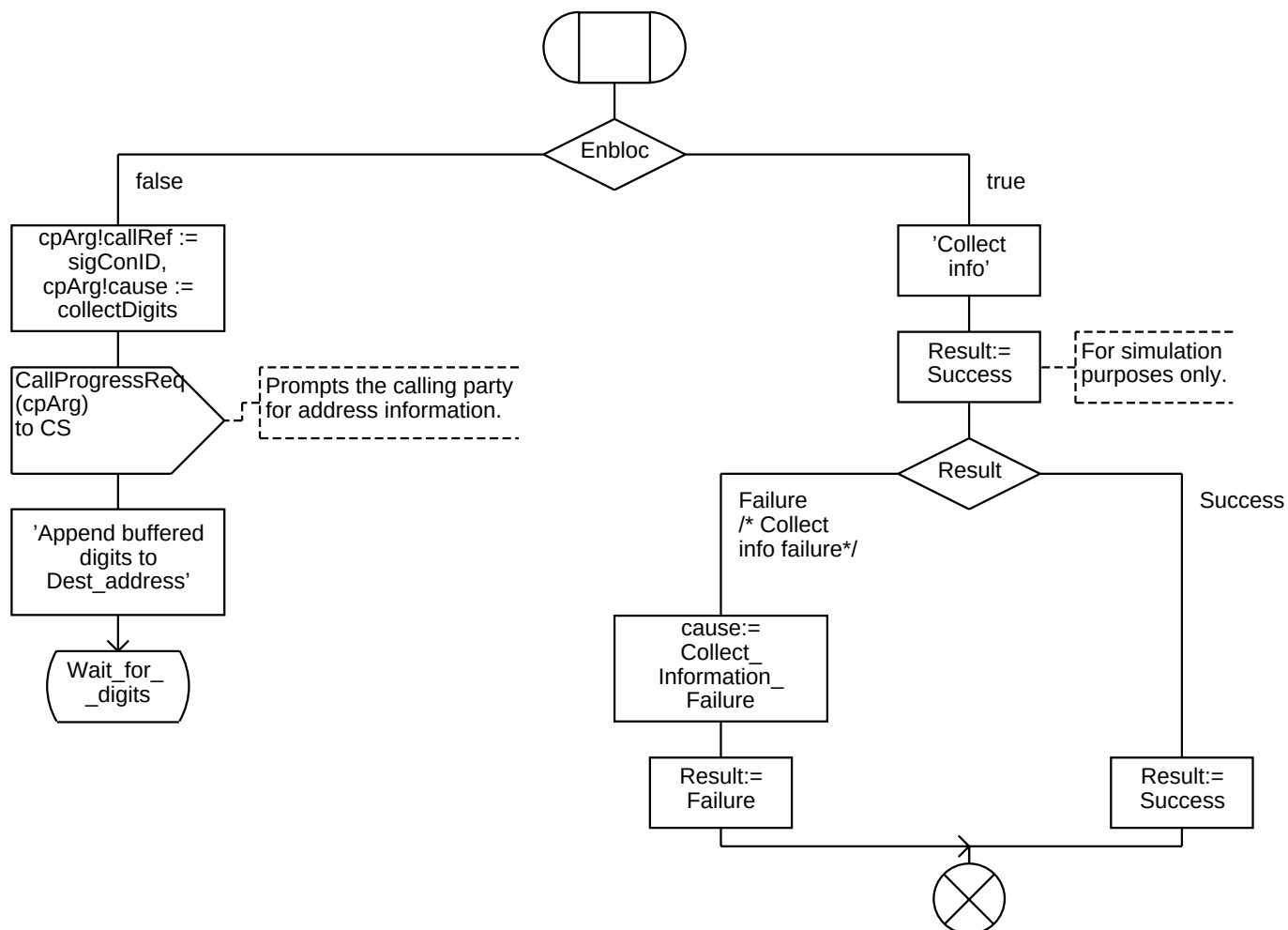
1(1)



Procedure PIC_Collect_Information

1(3)

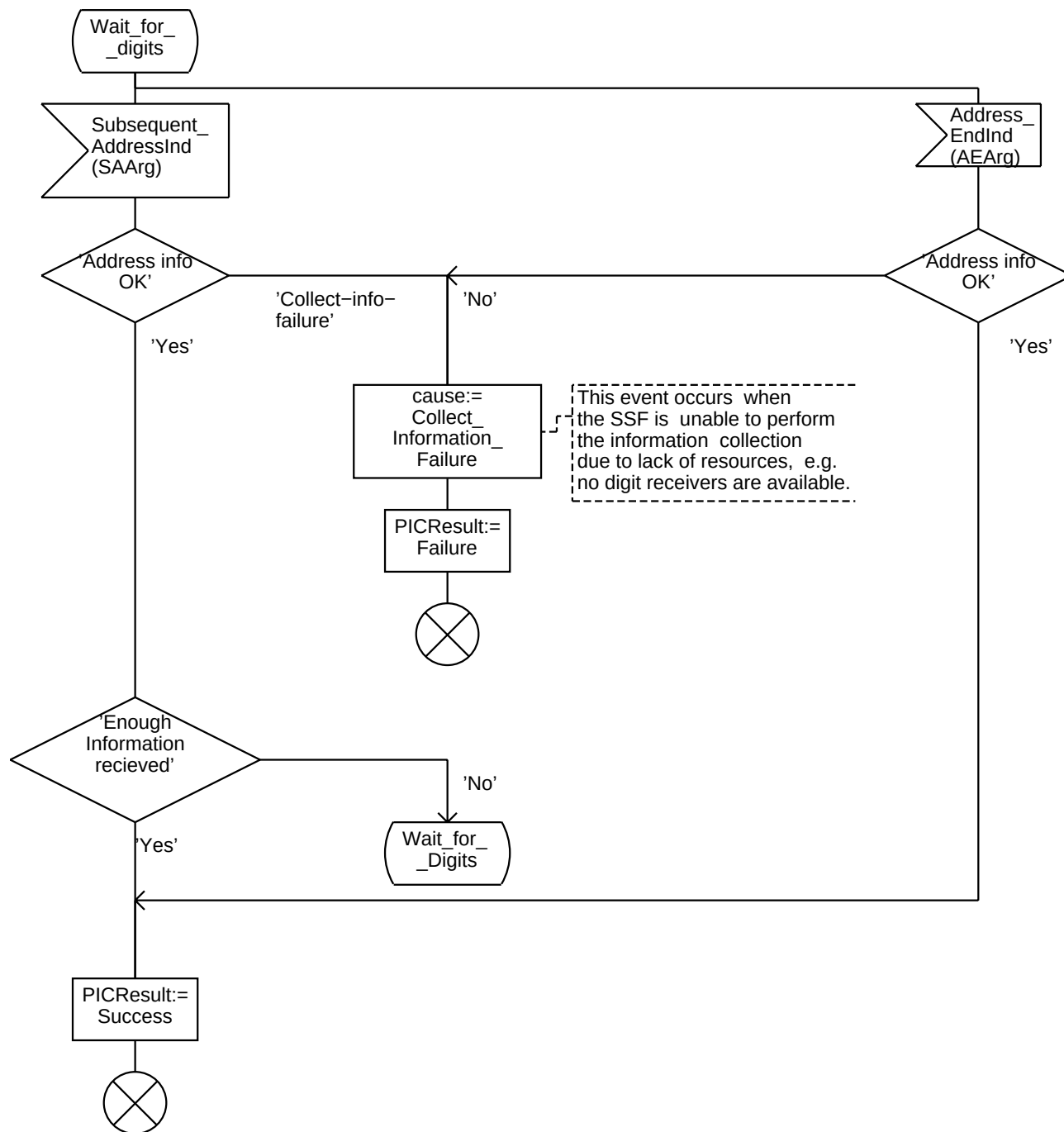
FPAR
 IN/OUT Result PICResultType;



Procedure PIC_Collect_Information

2(3)

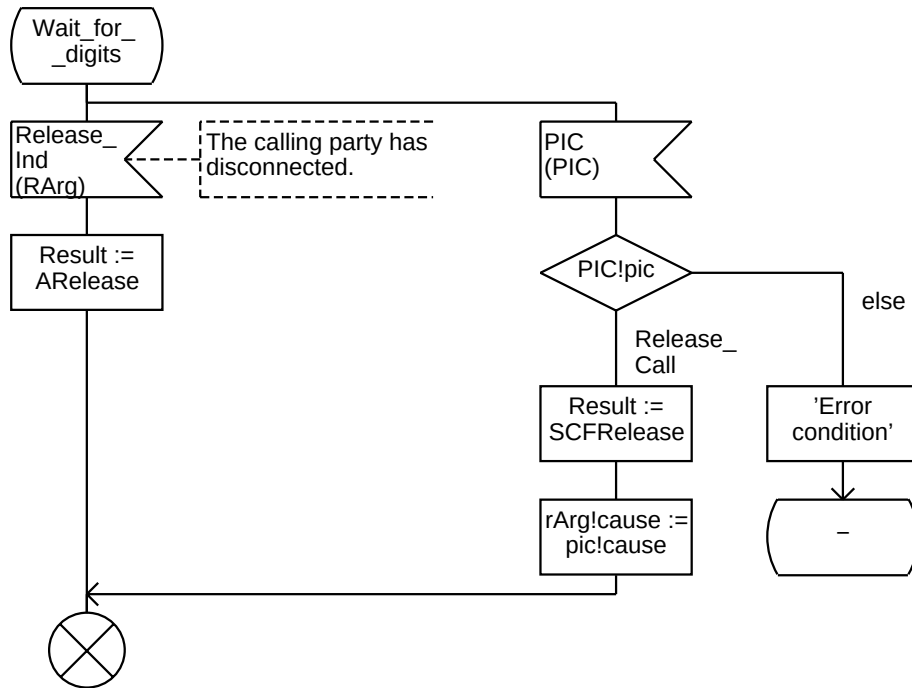
FPAR
IN/OUT Result PICResultType;



Procedure PIC_Collect_Information

3(3)

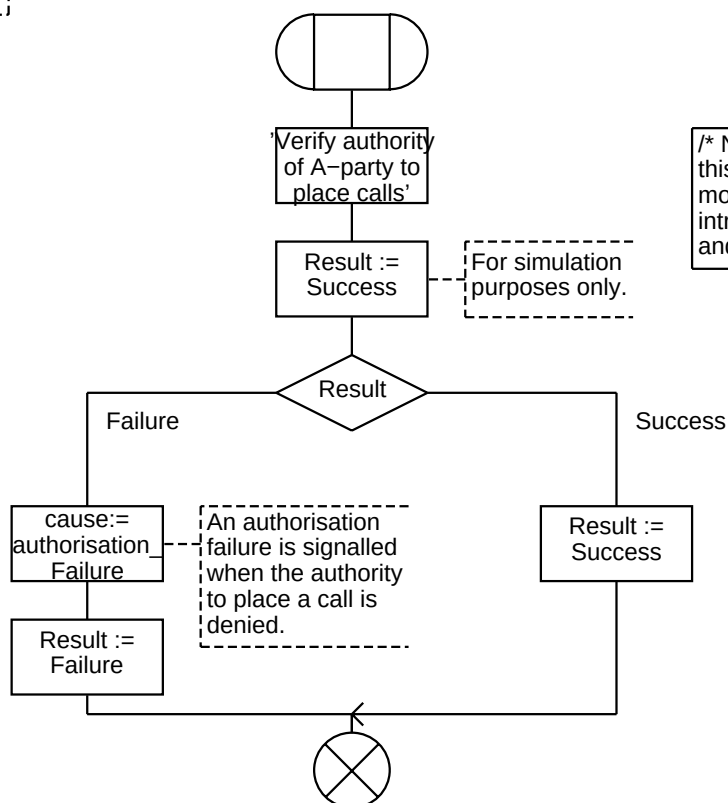
FPAR
IN/OUT Result PICResultType;



Procedure PIC_Authorise_Call_Setup

1(1)

FPAR
 IN/OUT Result PICResultType;

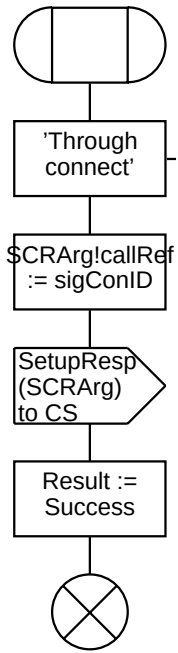


/* Note: The user may abandon the this PIC. For reasons of simplicity th modelled in the SDLs (it would have introduction of a state with two input and None). */

Procedure PIC_O_Answer

1(1)

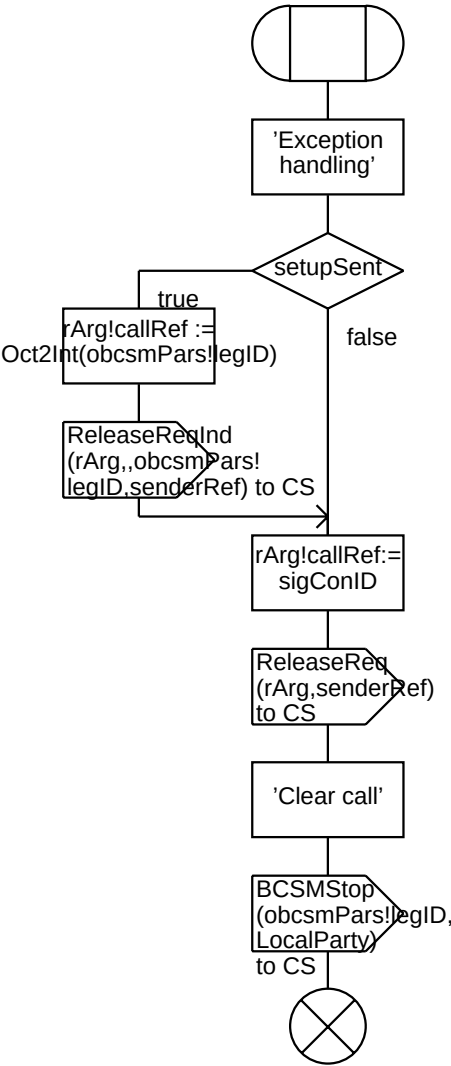
;FPAR
IN/OUT Result PICResultType;



/* Note: The user may abandon the this PIC. For reasons of simplicity th modelled in the SDLs (it would have tion of a state with two input calling and the called parties ie). */
The connection between the is established.

Procedure PIC_OException

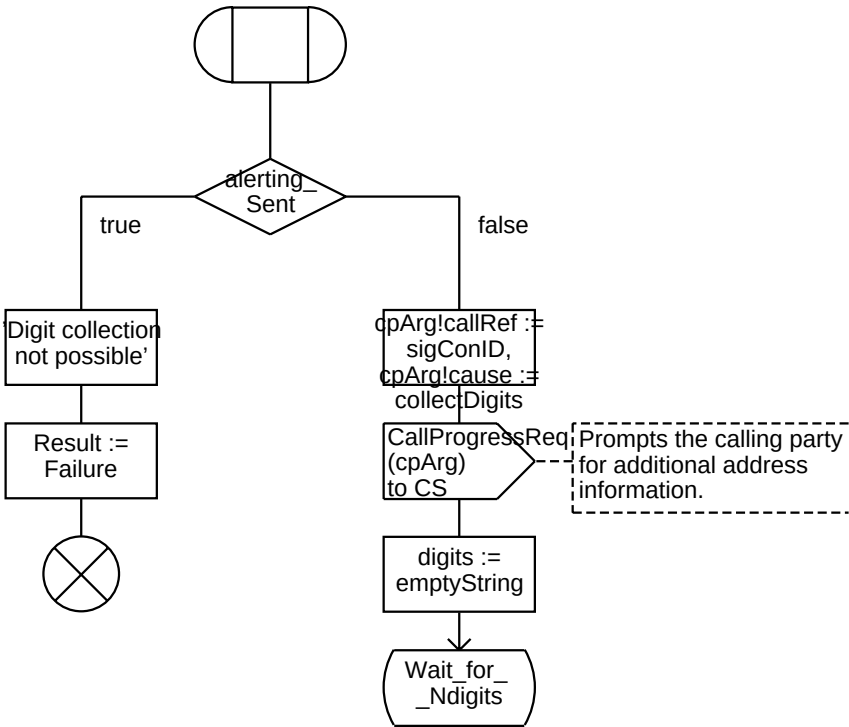
1(1)



Procedure PIC_Collect_NDigits

1(3)

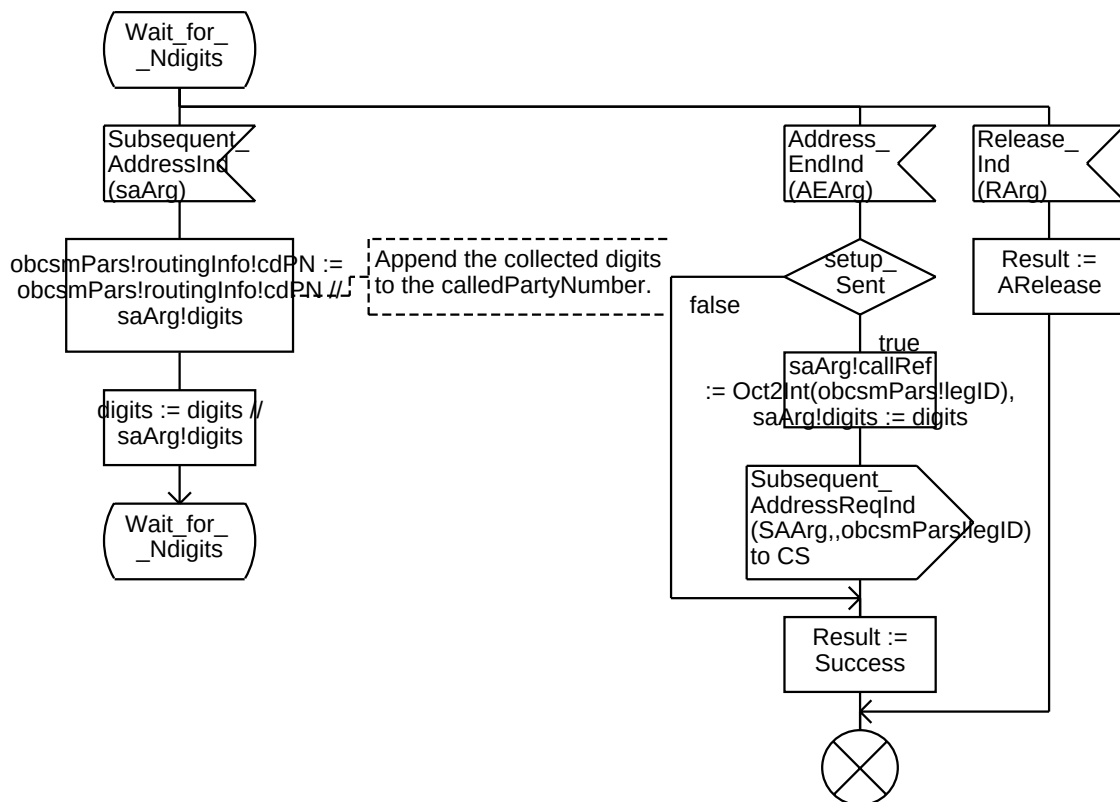
;FPAR
IN/OUT Result PICResultType;



Procedure PIC_Collect_NDigits

2(3)

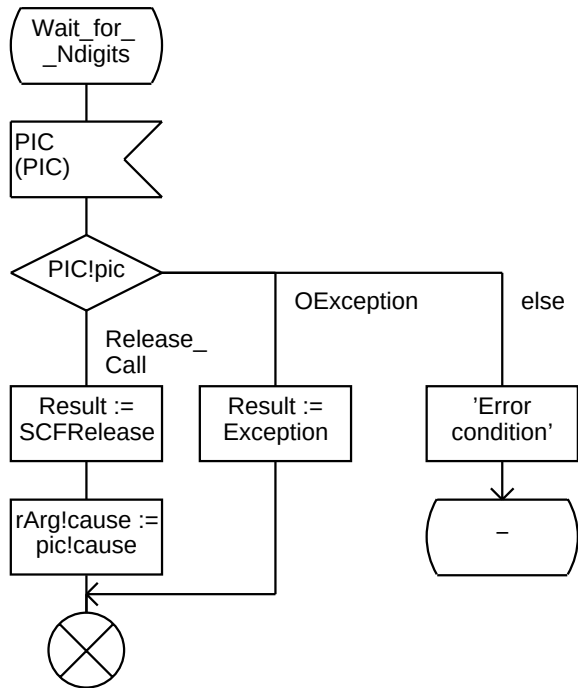
FPAR
IN/OUT Result PICResultType;



Procedure PIC_Collect_NDigits

3(3)

FPAR
IN/OUT Result PICResultType;

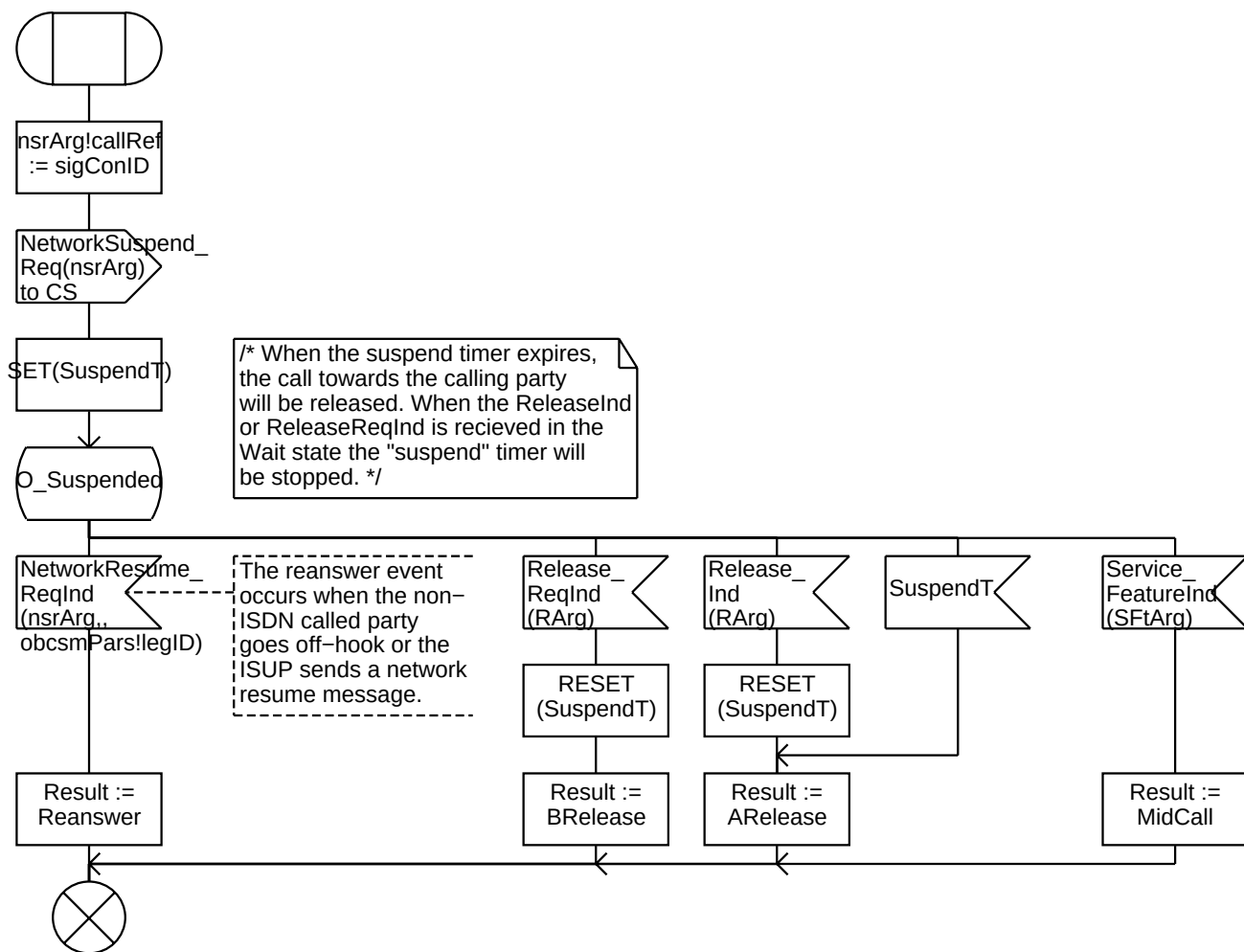


Procedure PIC_O_Suspended

1(3)

```

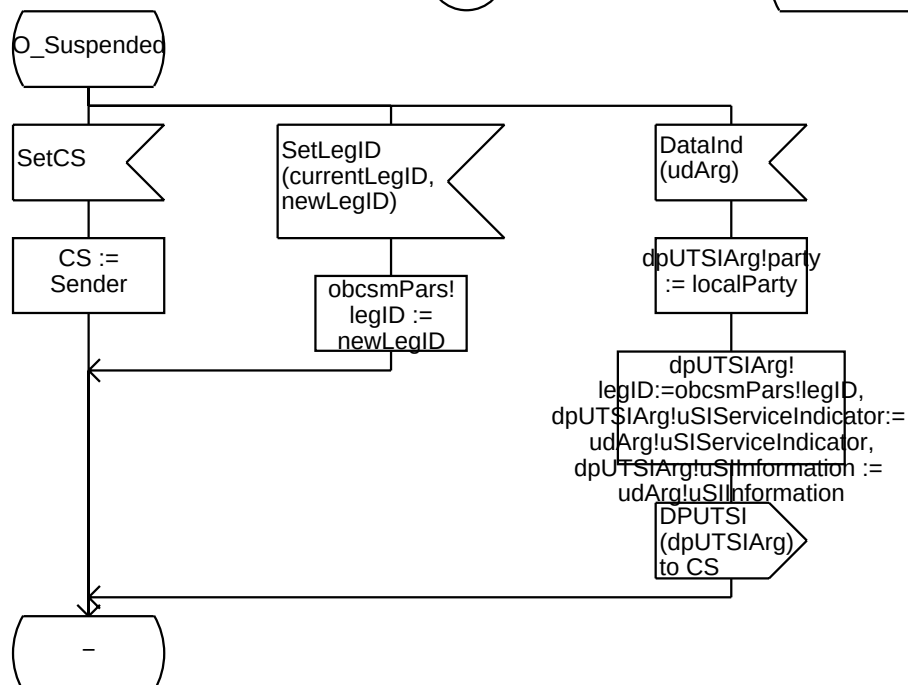
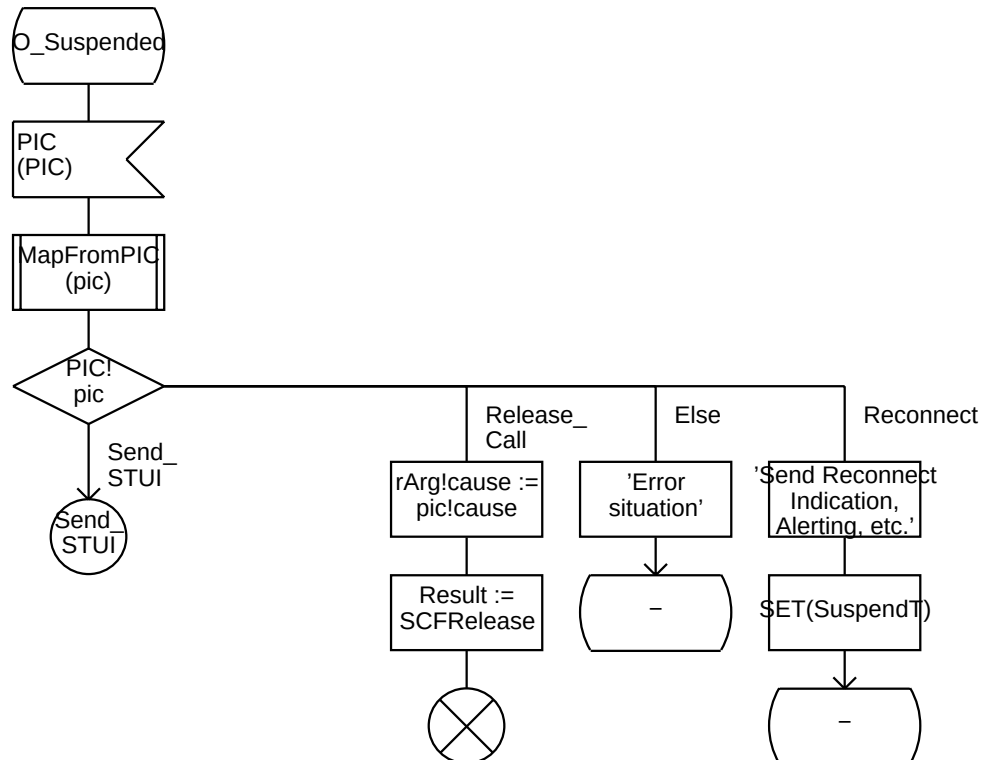
;FPAR
IN/OUT Result PICResultType;
    
```



Procedure PIC_O_Suspended

2(3)

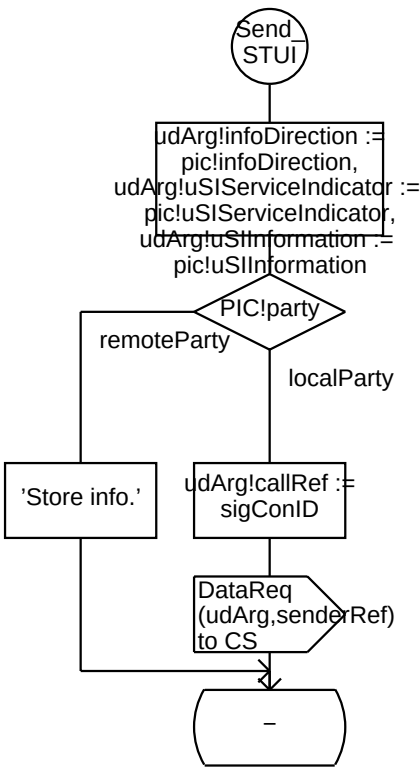
FPAR
IN/OUT Result PICResultType;



Procedure PIC_O_Suspended

3(3)

;FPAR
IN/OUT Result PICResultType;

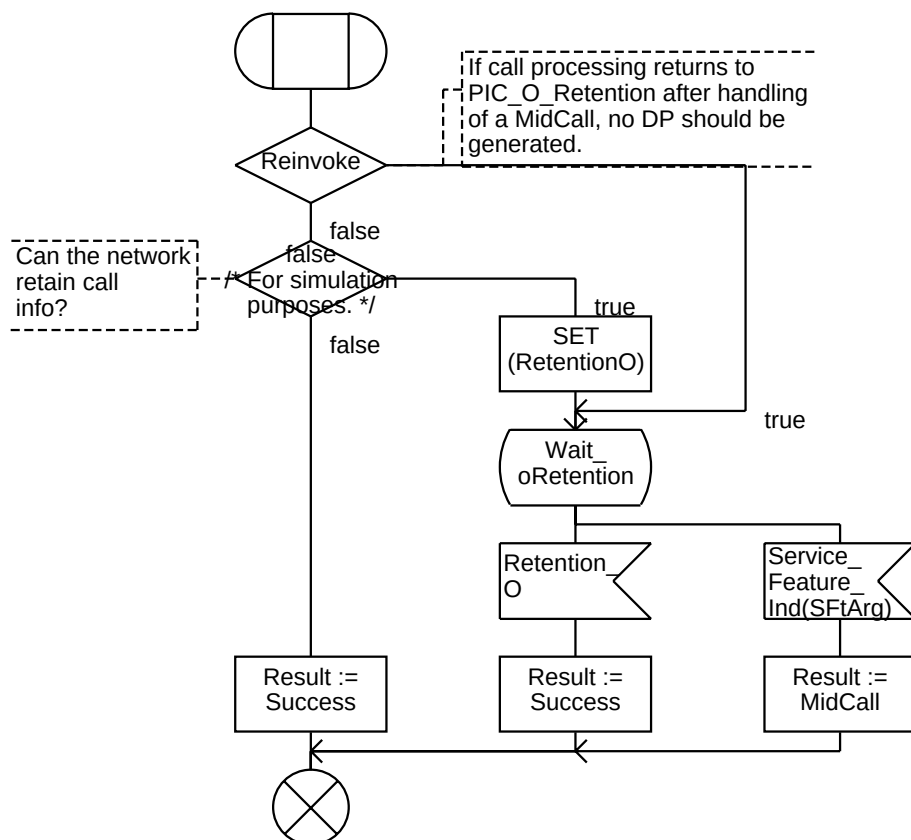


Procedure PIC_O_Retention

1(2)

```

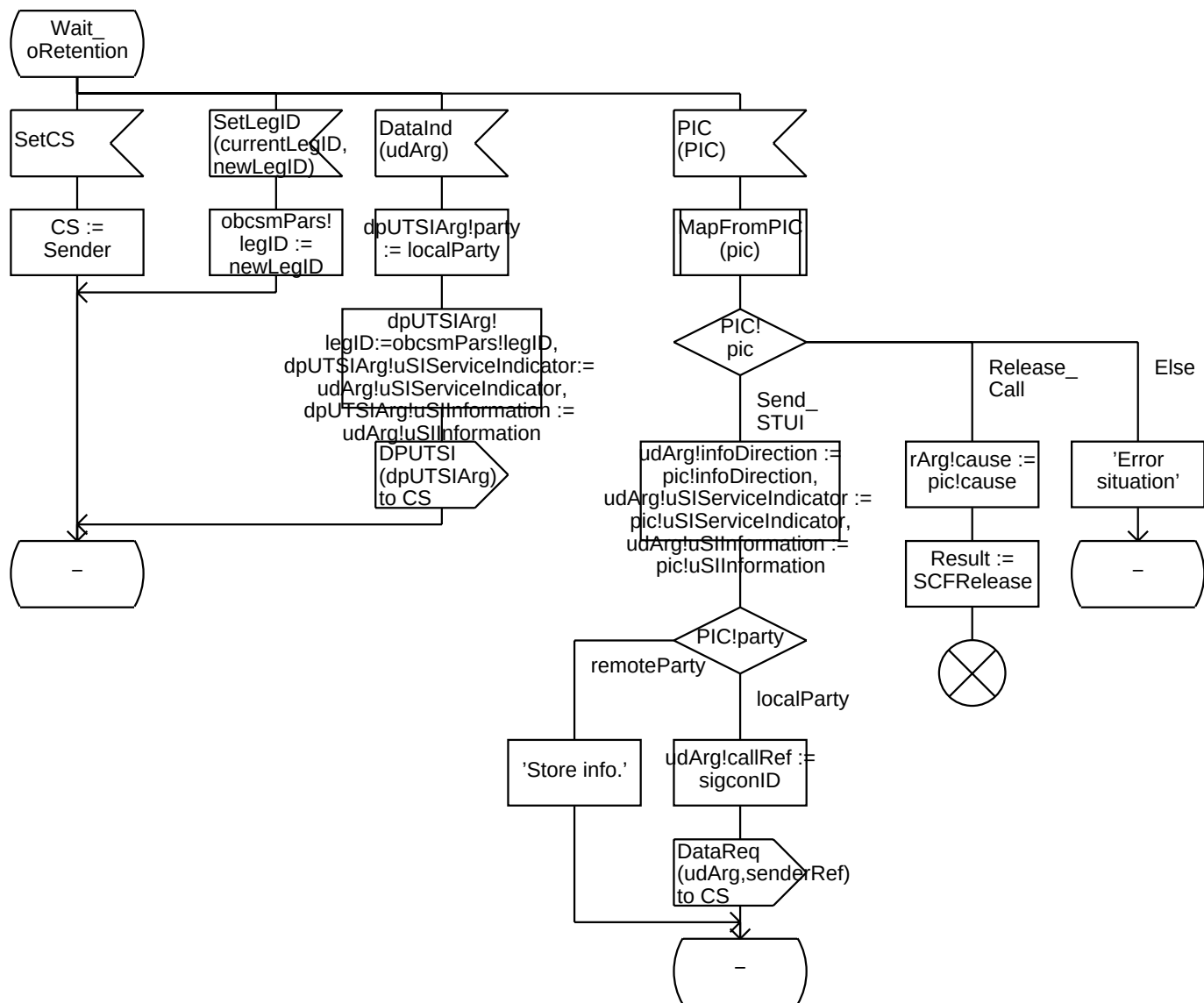
;FPAR
IN/OUT Result PICResultType,
IN   Reinvoke Boolean;
    
```



Procedure PIC_O_Retention

2(2)

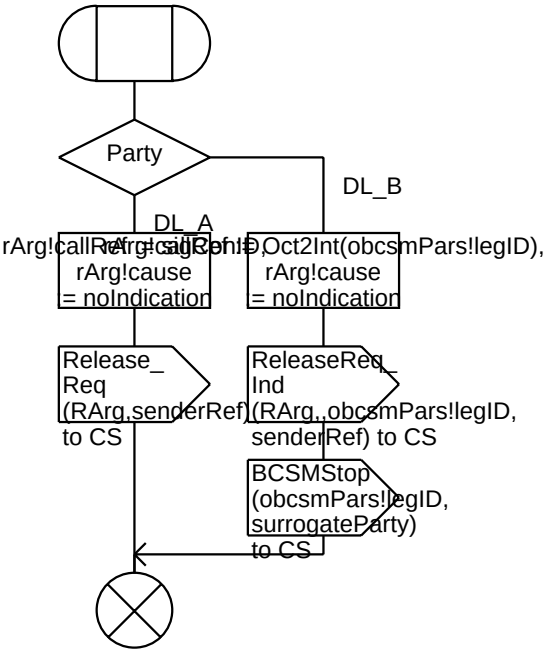
FPAR
IN/OUT Result PICResultType,
IN Reinvoke Boolean;



Procedure <<Process Type OriginatingBCSM>> PIC_Disconnect

1(1)

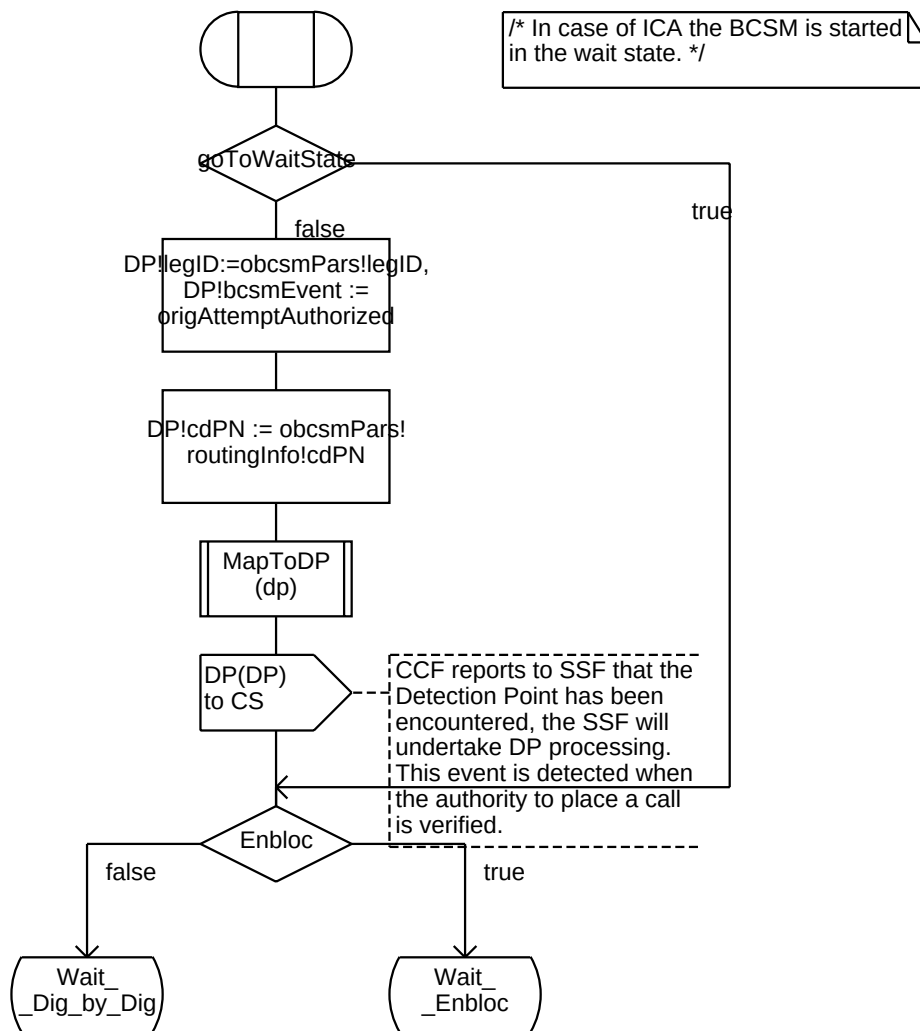
FPAR
IN Party DResultType;



Procedure DP_origAttemptAuthorised

1(4)

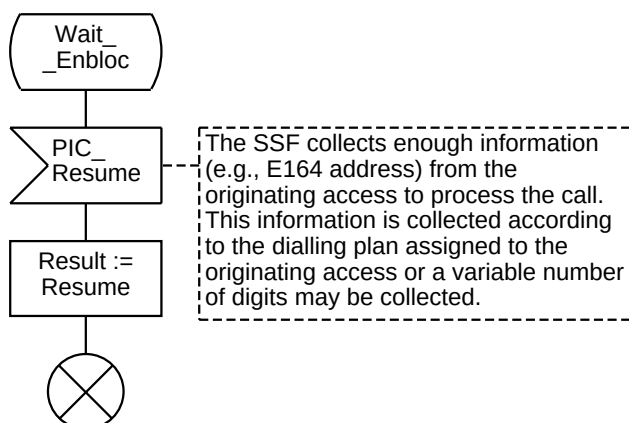
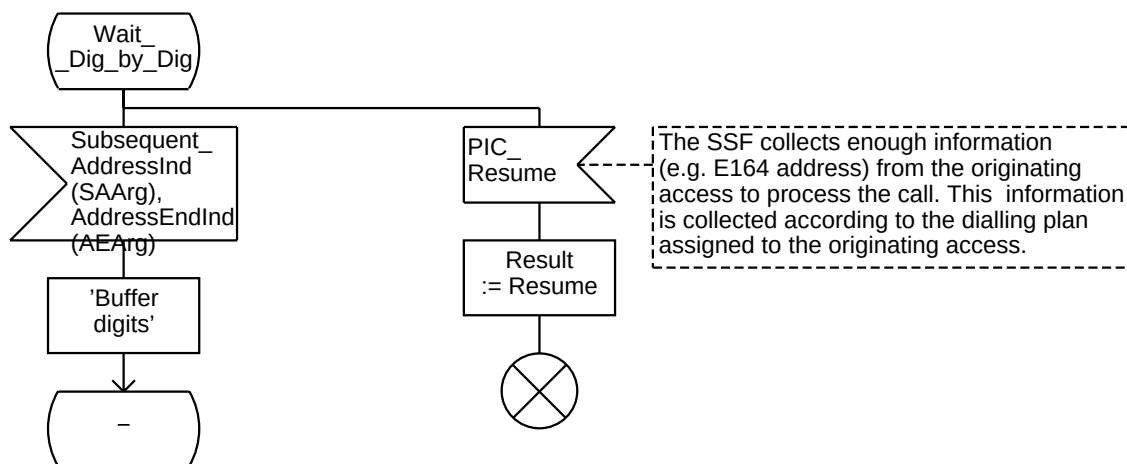
FPAR
 IN/OUT Result DPResultType,
 IN goToWaitState Boolean;



Procedure DP_origAttemptAuthorised

2(4)

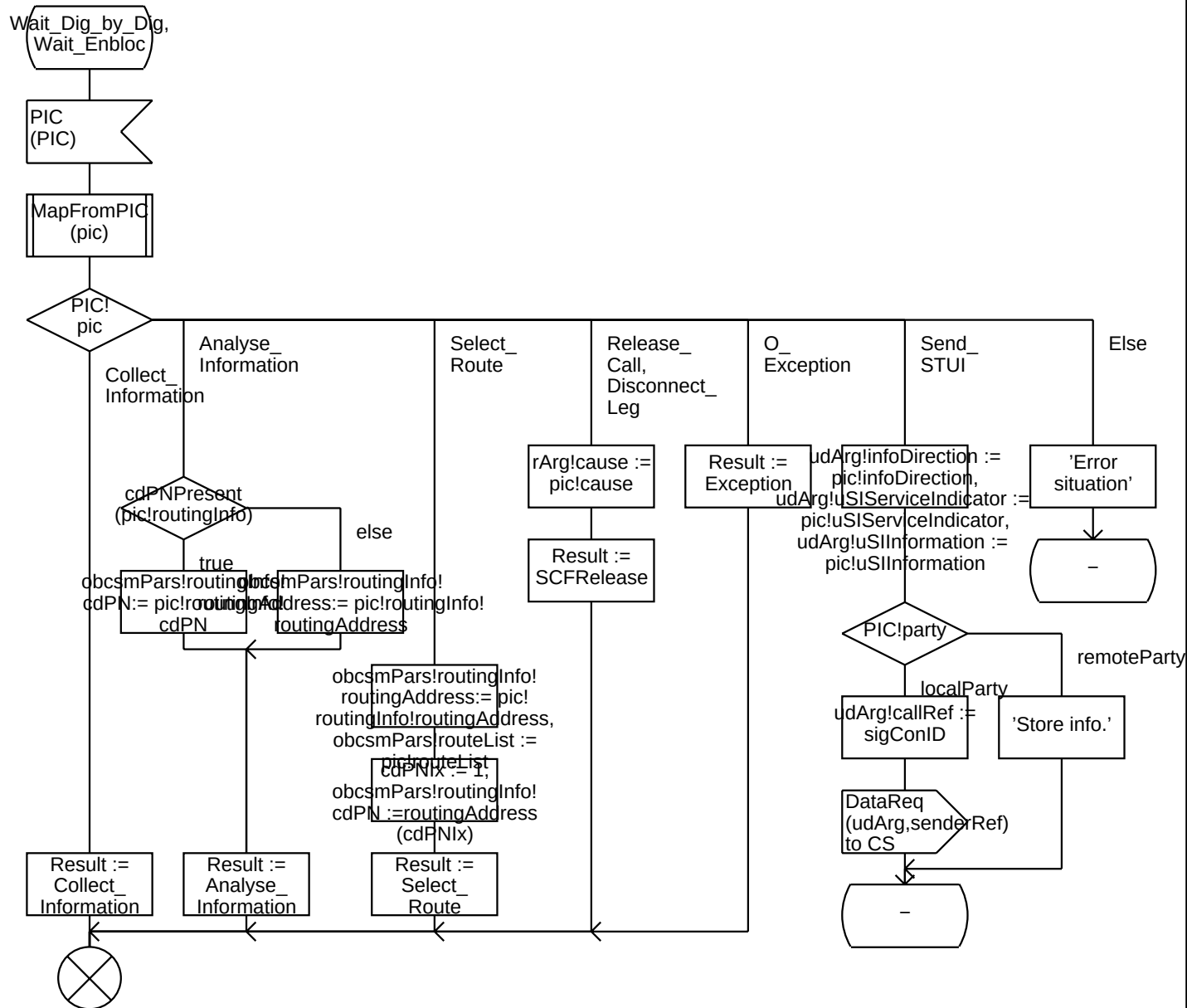
FPAR
IN/OUT Result DPResultType,
IN goToWaitState Boolean;



Procedure DP_origAttemptAuthorised

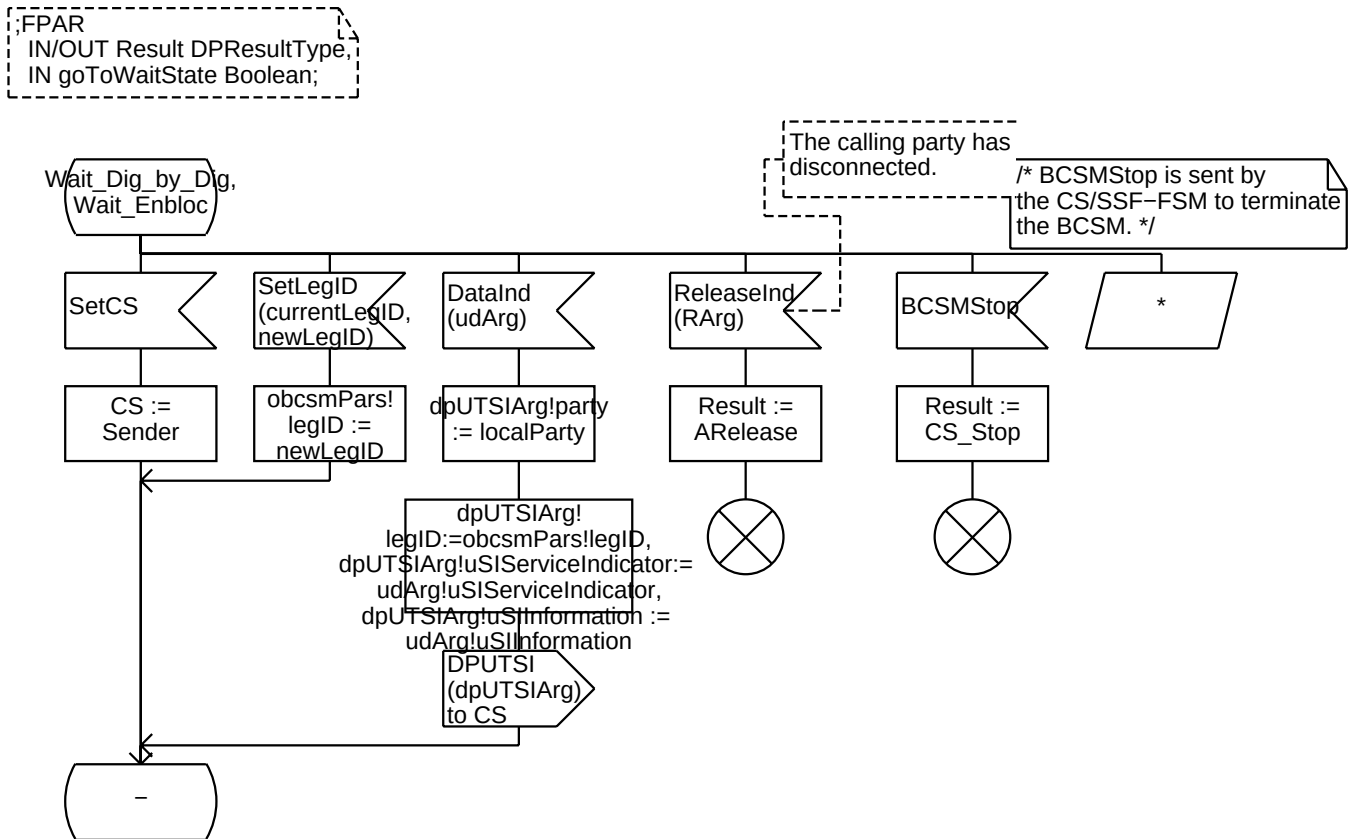
3(4)

FPAR
IN/OUT Result DPResultType,
IN goToWaitState Boolean;



Procedure DP_origAttemptAuthorised

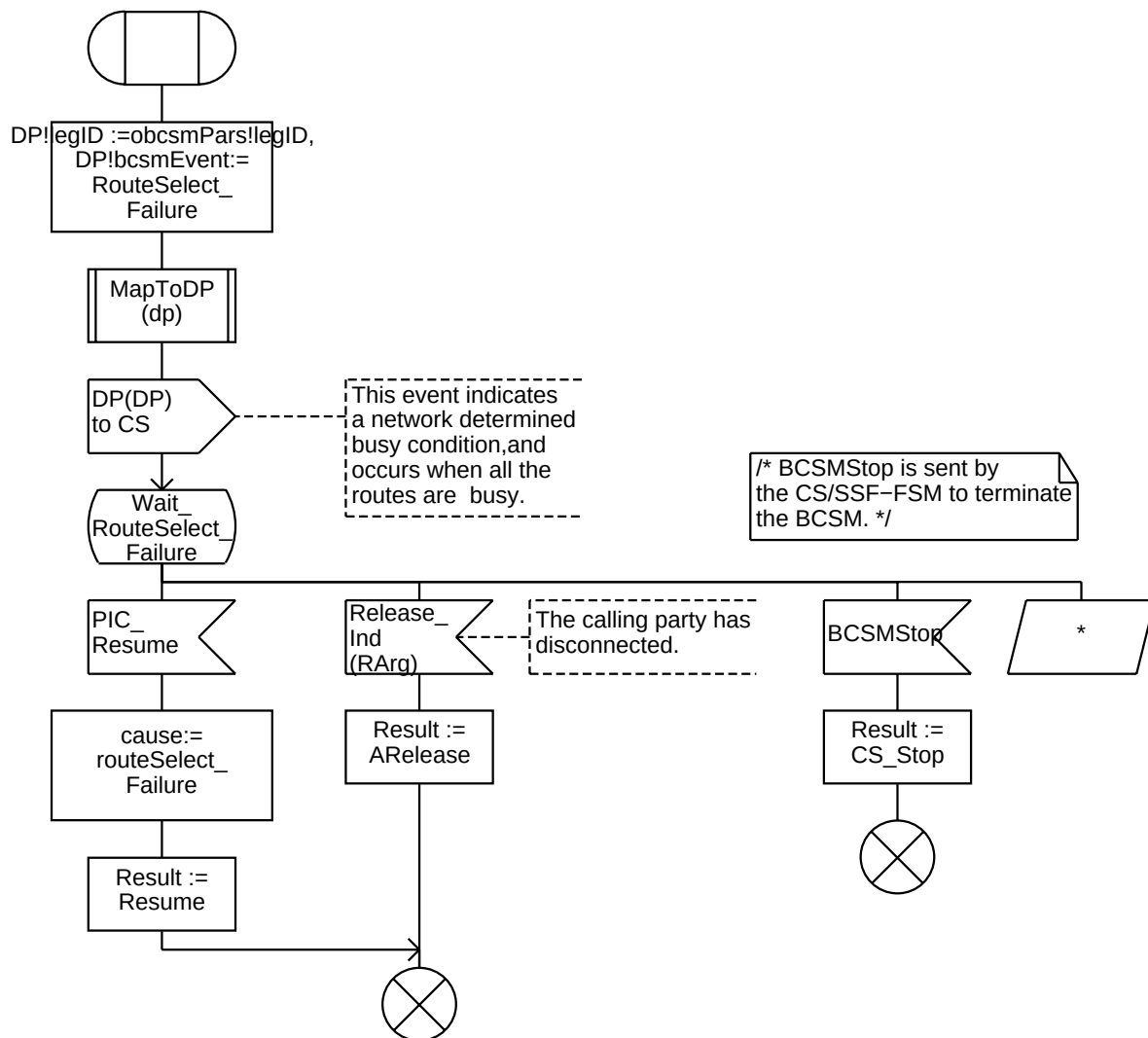
4(4)



Procedure DP_Route_Select_Failure

1(3)

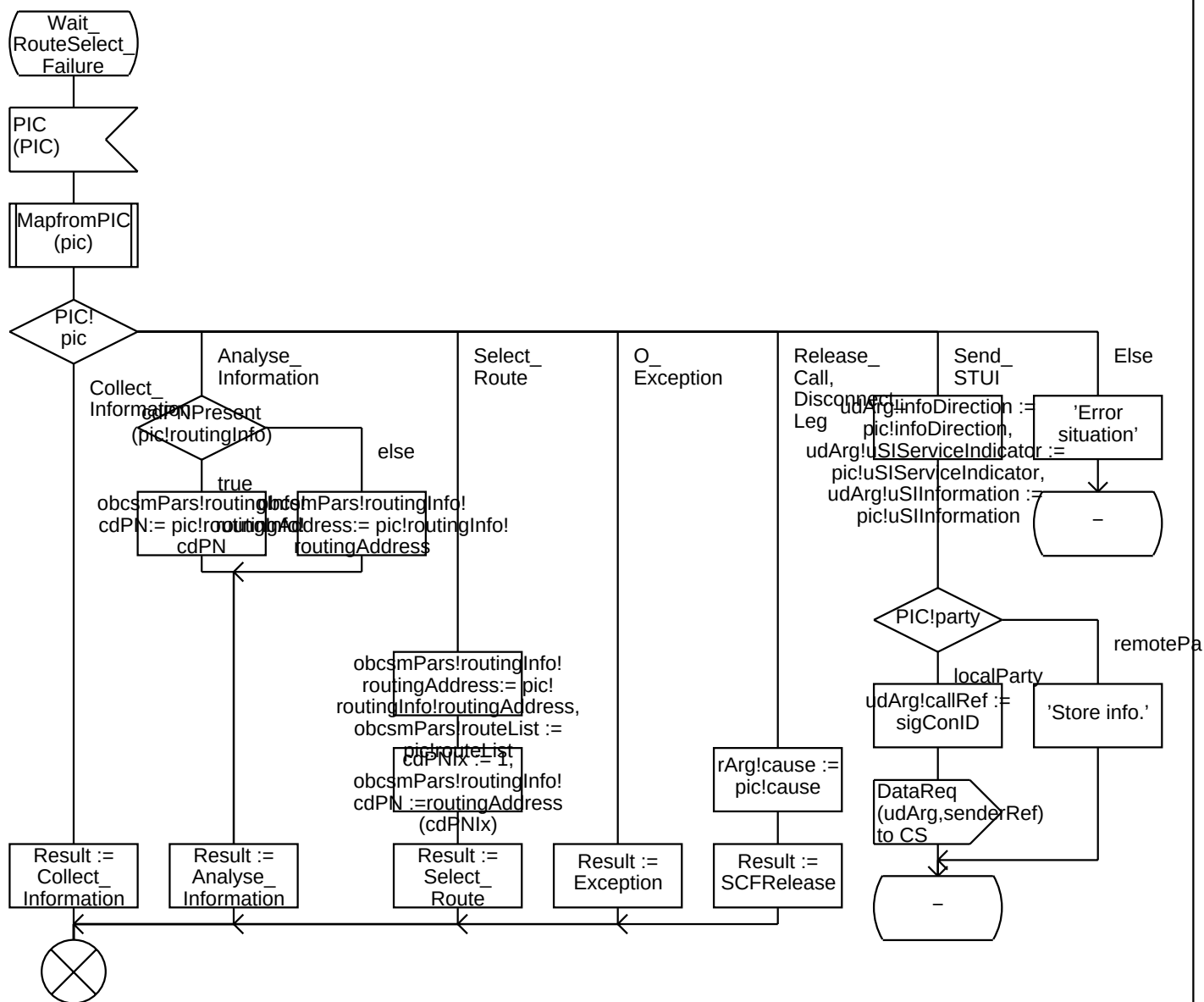
FPAR
IN/OUT Result DPResultType;



Procedure DP_Route_Select_Failure

2(3)

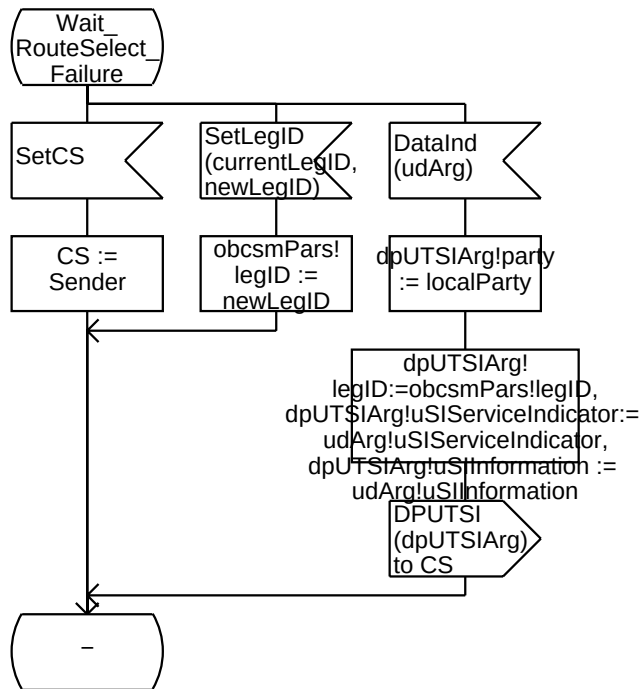
FPAR
IN/OUT Result DPResultType;



Procedure DP_Route_Select_Failure

3(3)

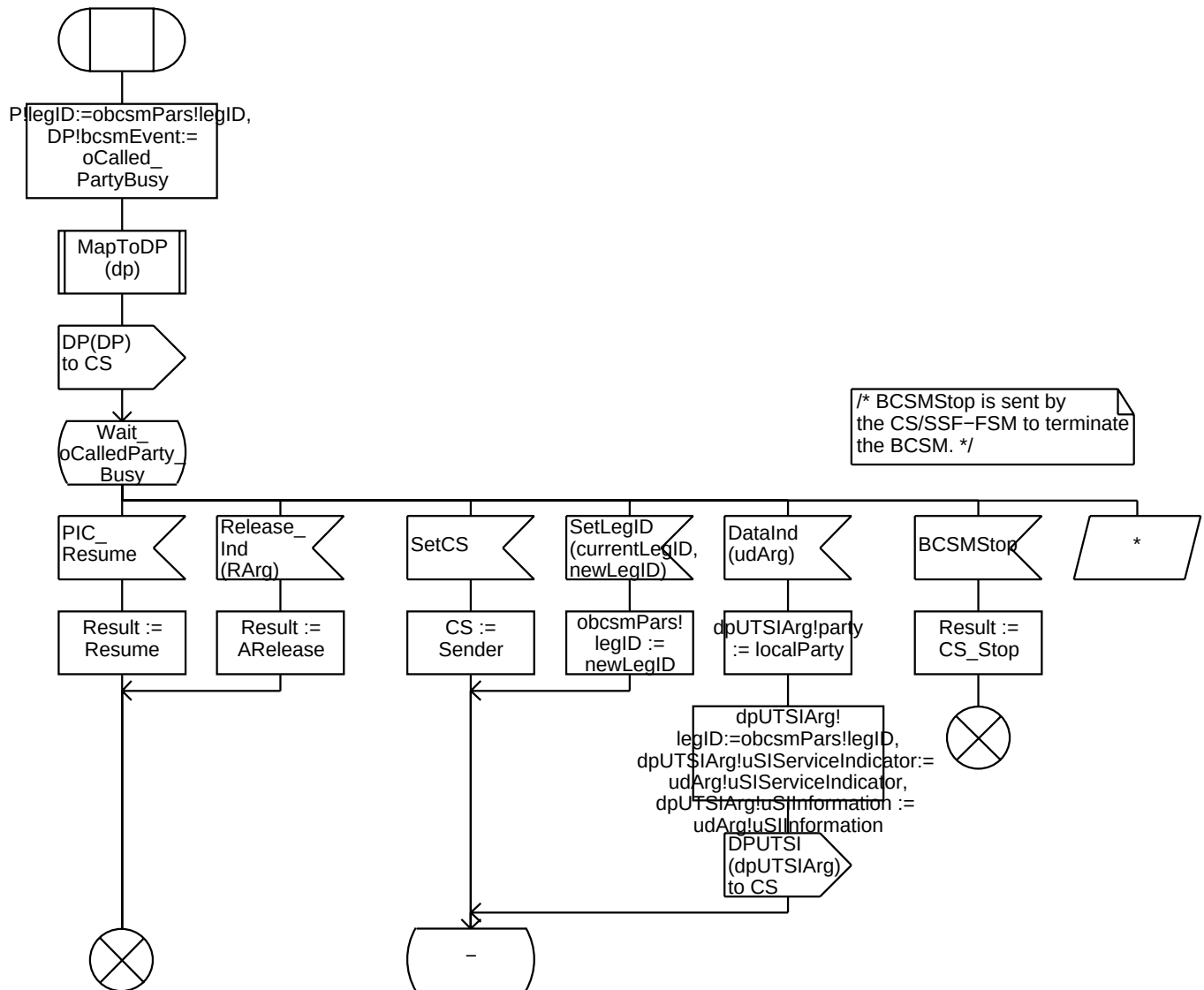
FPAR
IN/OUT Result DPResultType;



Procedure DP_oCalledPartyBusy

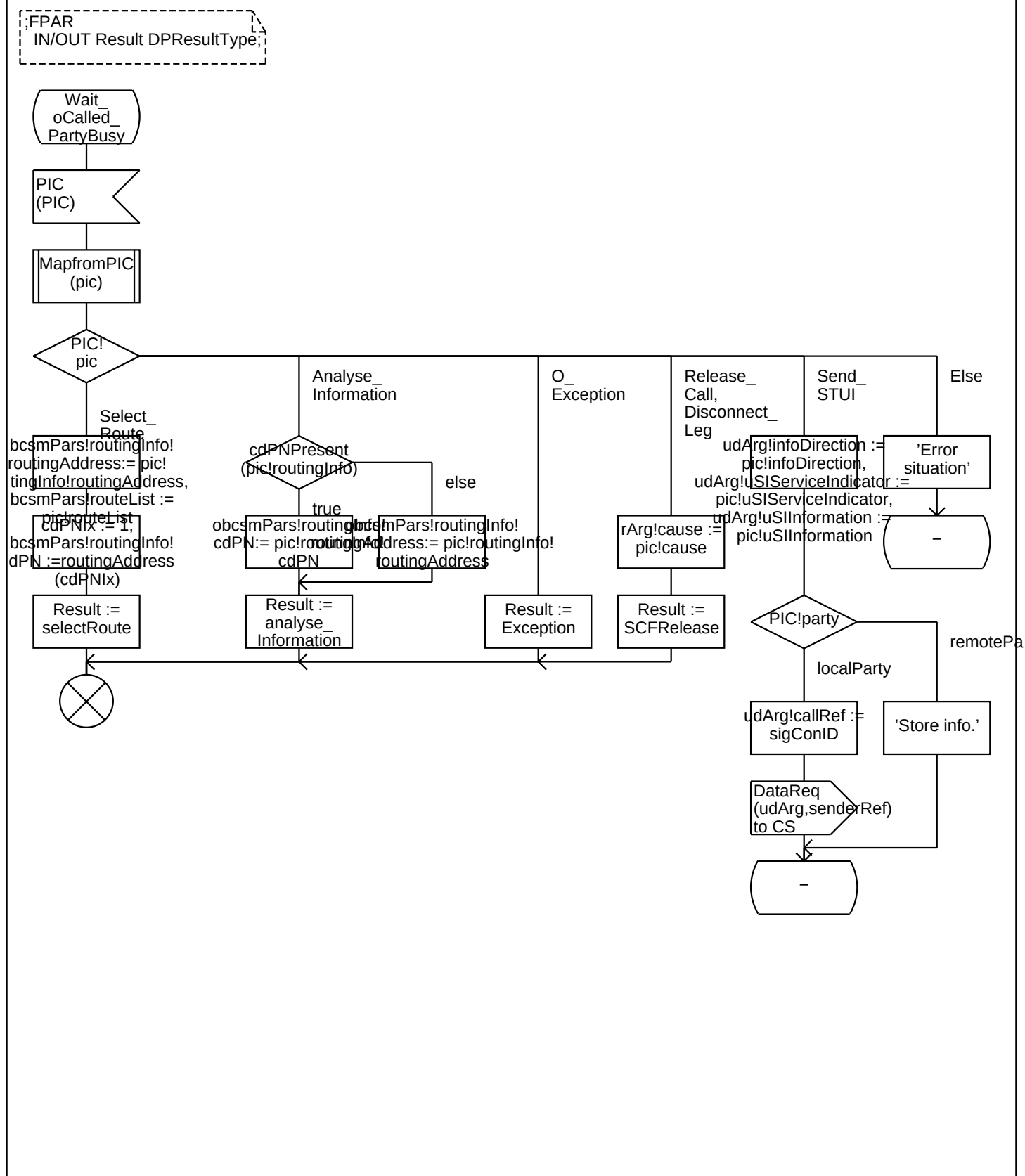
1(2)

FPAR
IN/OUT Result DPResultType;



Procedure DP_oCalledPartyBusy

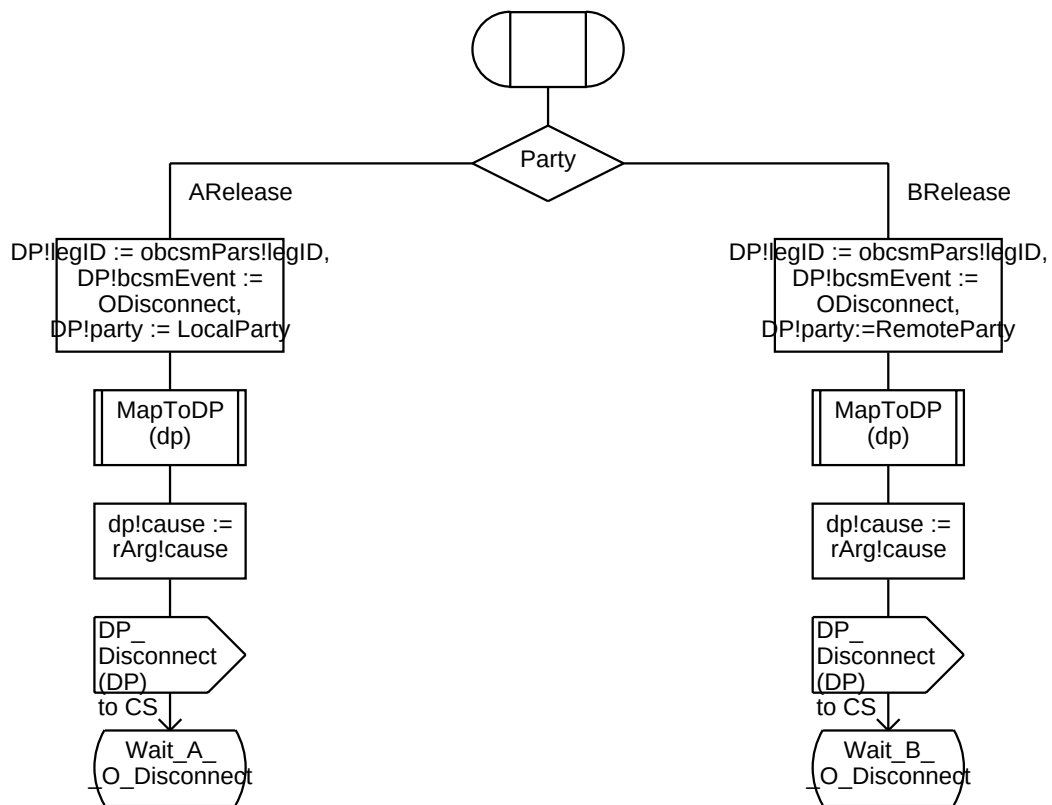
2(2)



Procedure DP_oDisconnect

1(5)

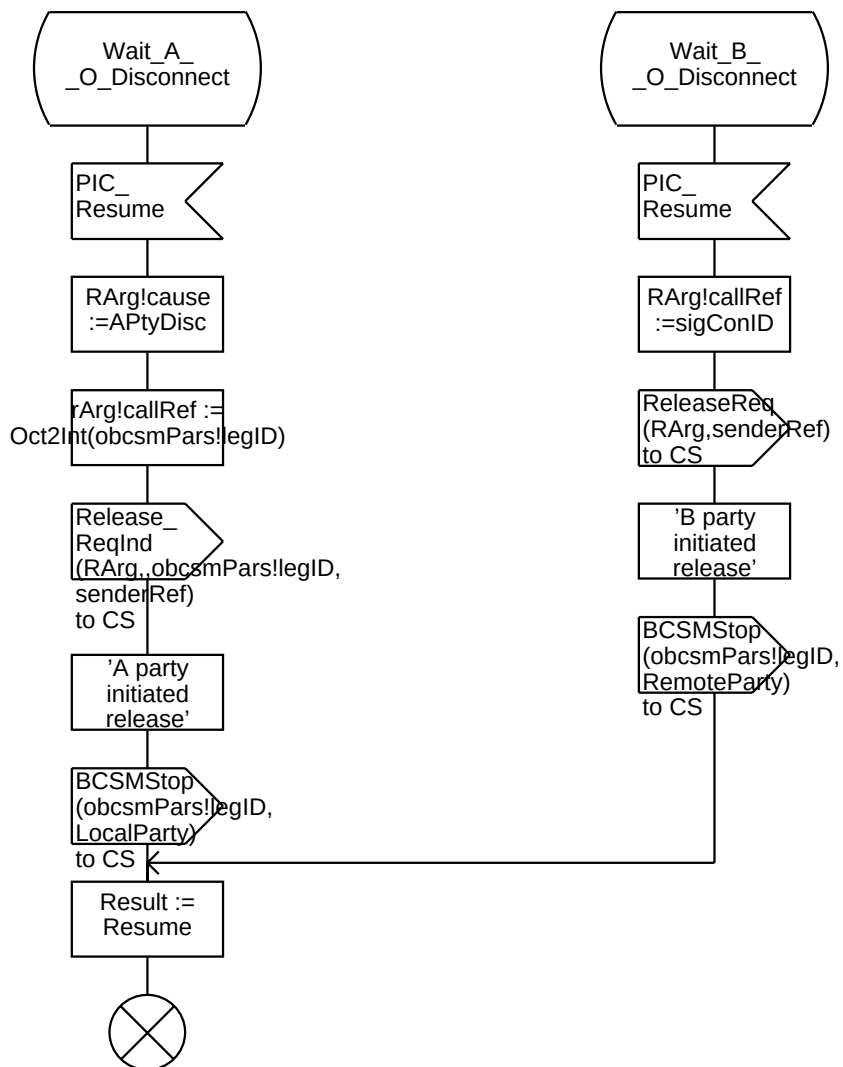
FPAR
IN/OUT Result DPResultType,
IN Party DPResultType;



Procedure DP_oDisconnect

2(5)

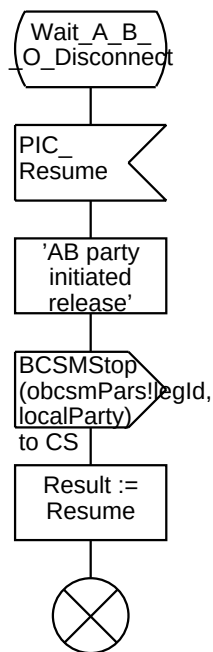
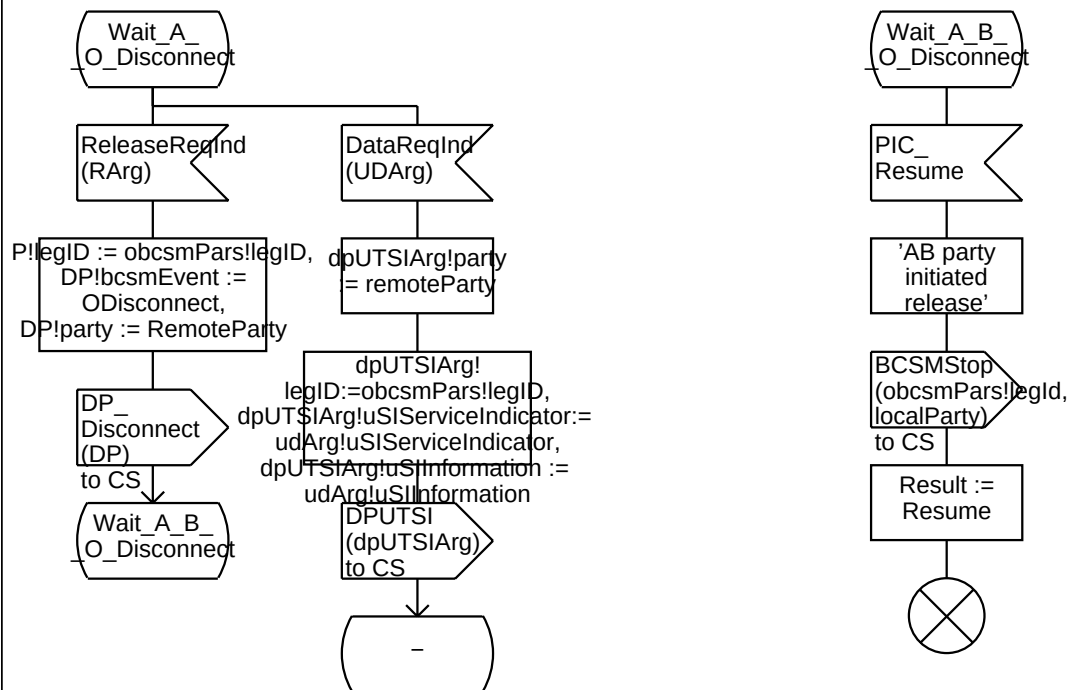
FPAR
IN/OUT Result DPResultType,
IN Party DPResultType;



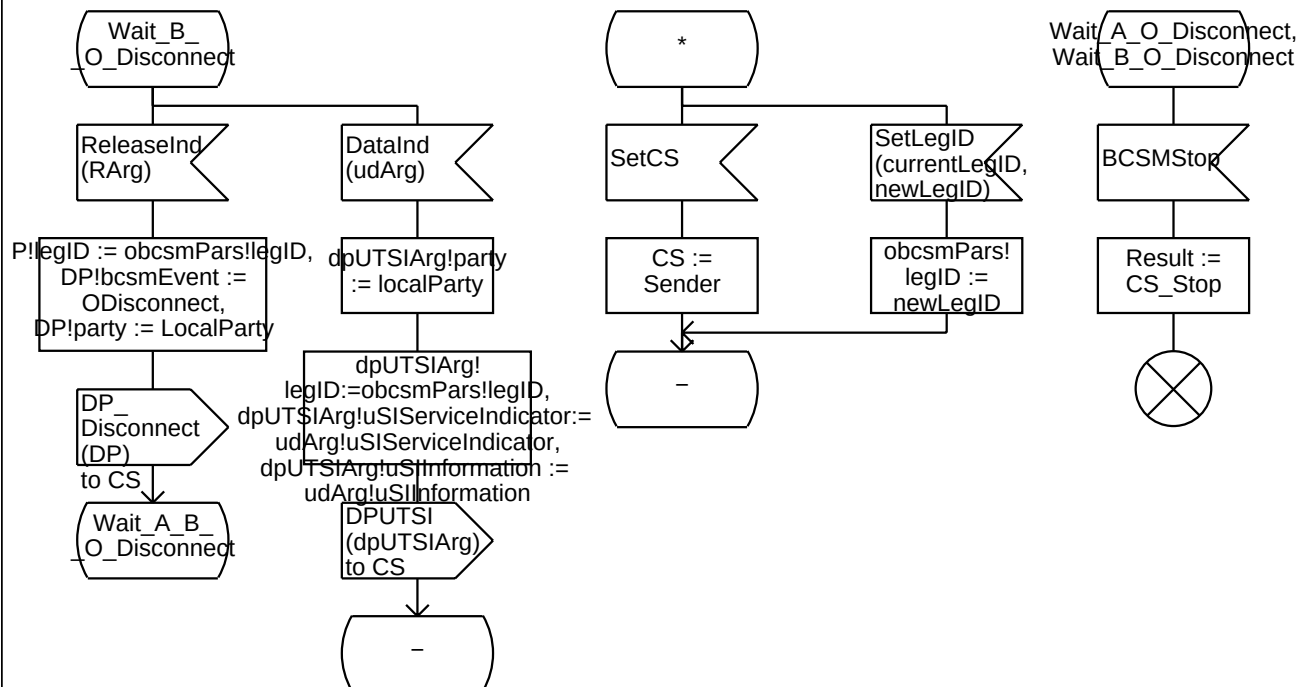
Procedure DP_oDisconnect

3(5)

FPAR
IN/OUT Result DPResultType,
IN Party DPResultType;

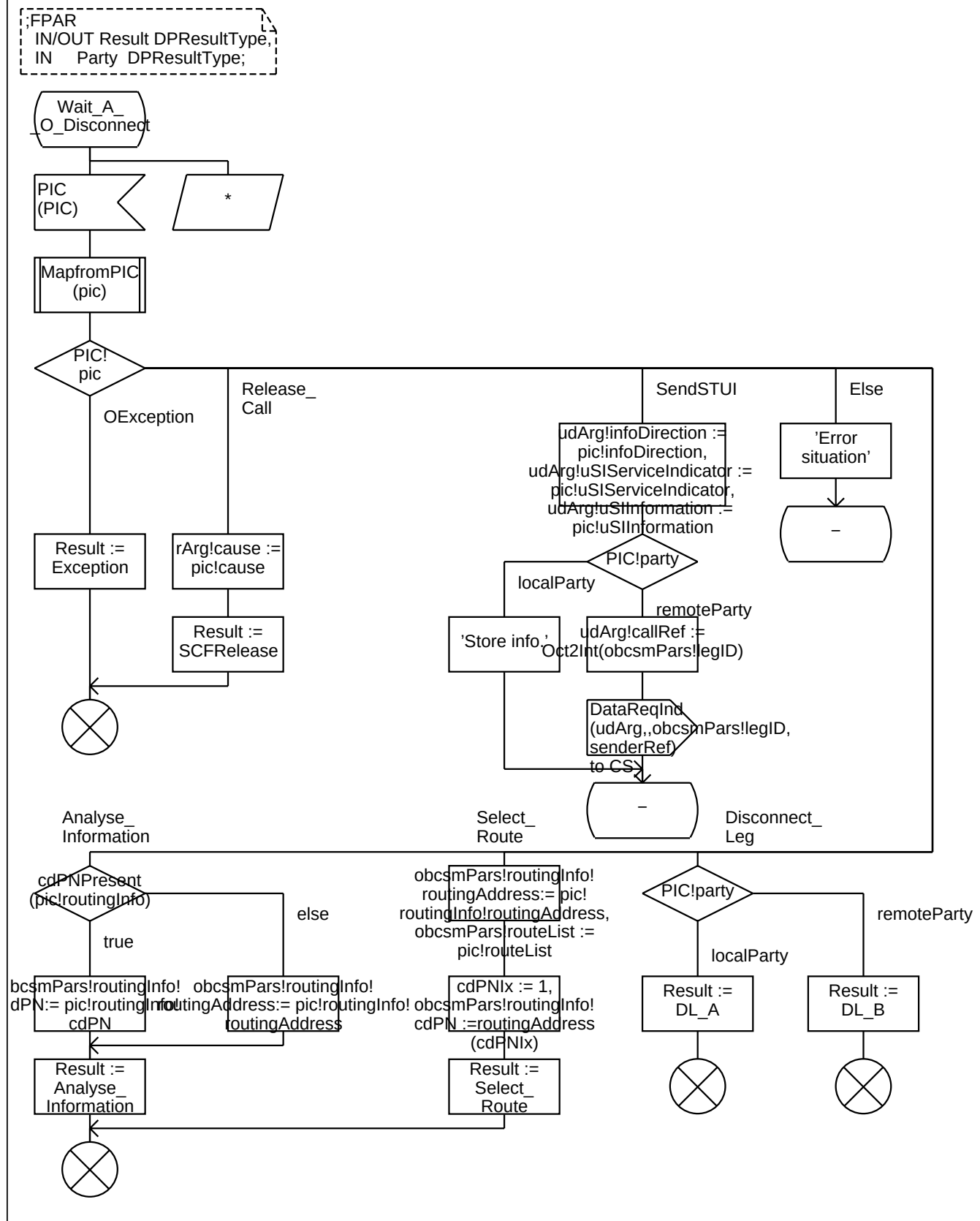


/* BCSMStop is sent by the CS/SSF-FSM to terminate the BCSM. */



Procedure DP_oDisconnect

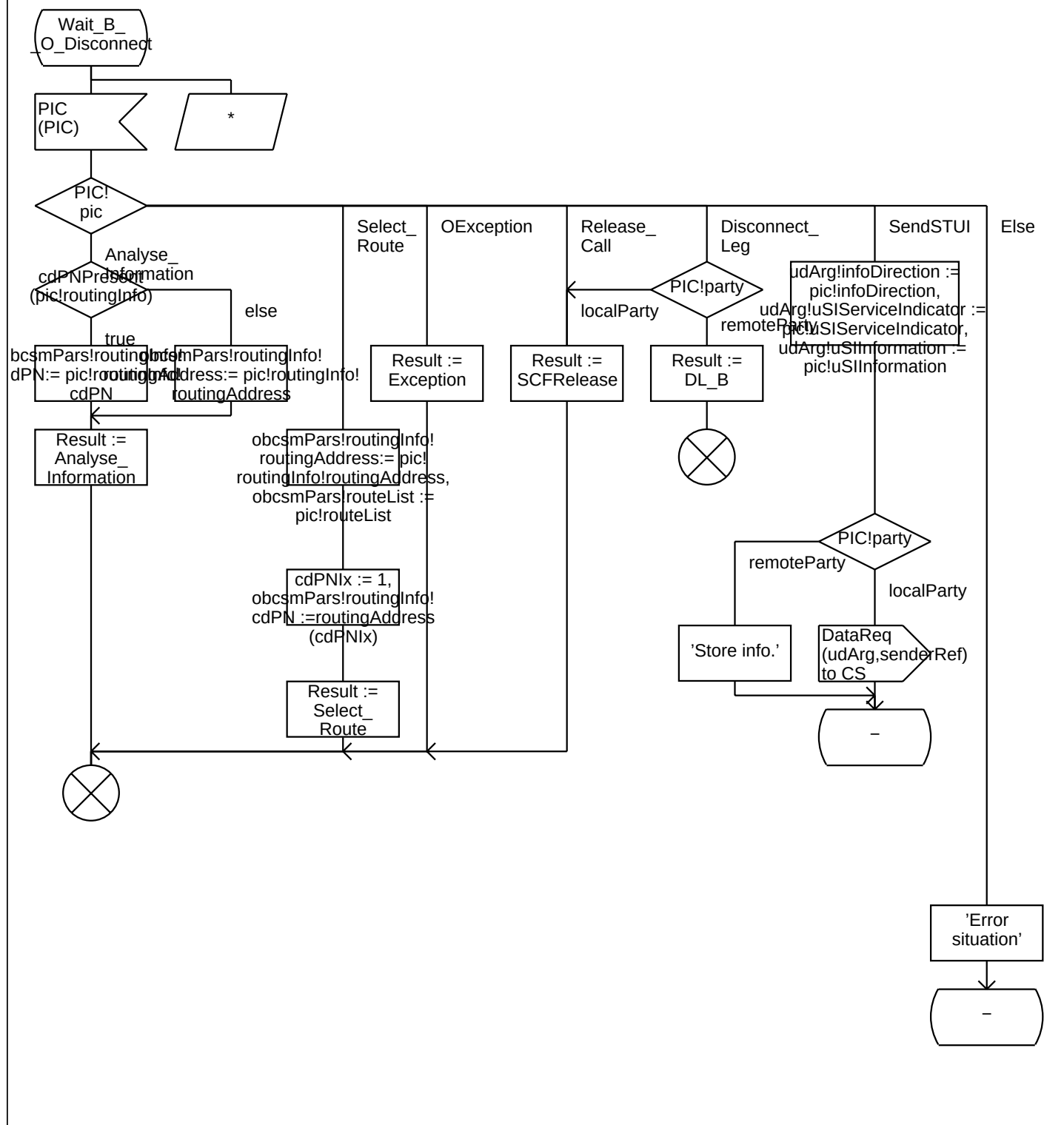
4(5)



Procedure DP_oDisconnect

5(5)

FPAR
IN/OUT Result DPResultType,
IN Party DPResultType;

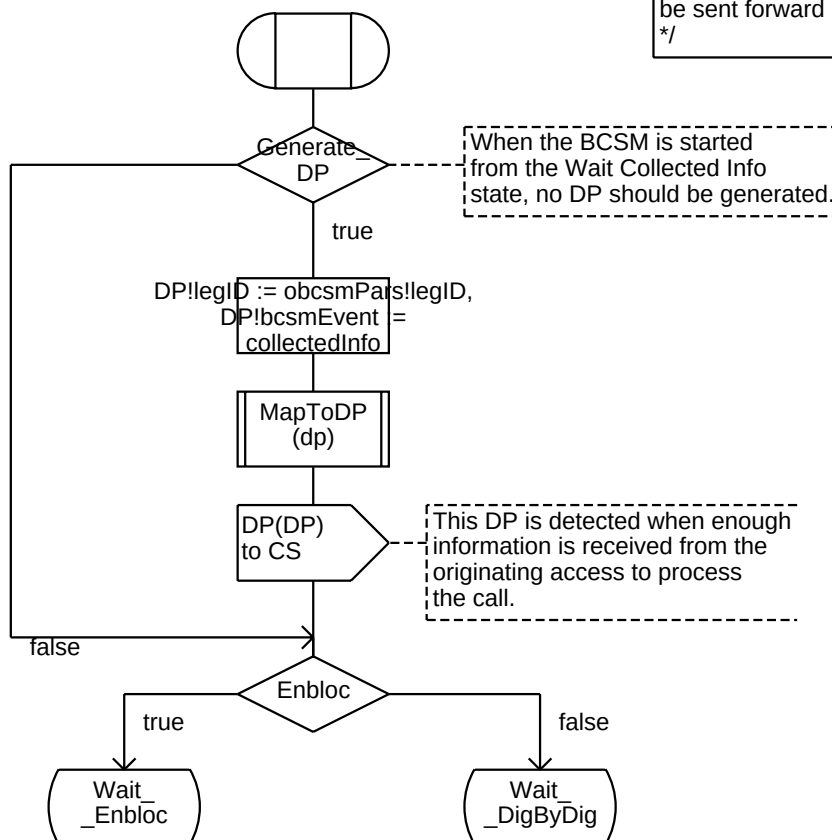


Procedure DP_Collected_Info

1(4)

FPAR
IN/OUT Result DPResultType,
IN GenerateDP Boolean;

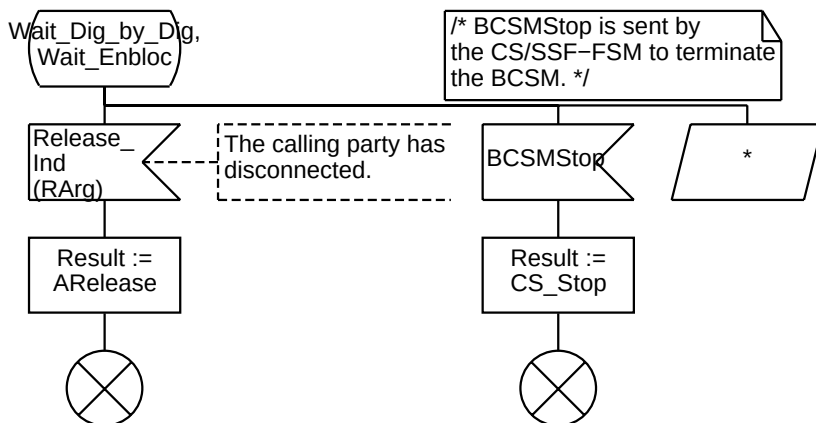
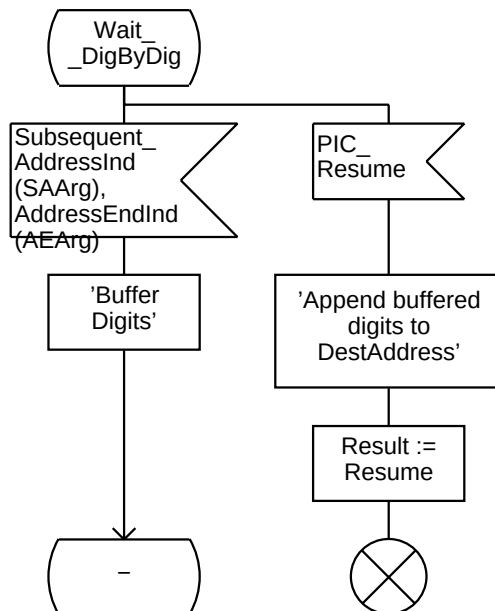
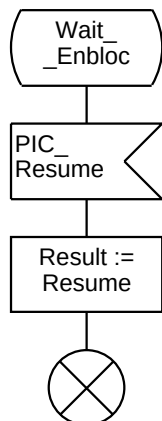
/*
DP Collected_Information occurs when there is
sufficient information available to start outgoing
setup. Outgoing call setup is now possible but
cannot proceed until SCF has adviced SSF either
to continue call setup with dialled digits or to
replace them with digits supplied by SCF. Call
setup commences after Analysed_Information DP.
In digit by digit case, the user may be continuing
to dial further digits and these must be stored, to
be sent forward later.
*/



Procedure DP_Collected_Info

2(4)

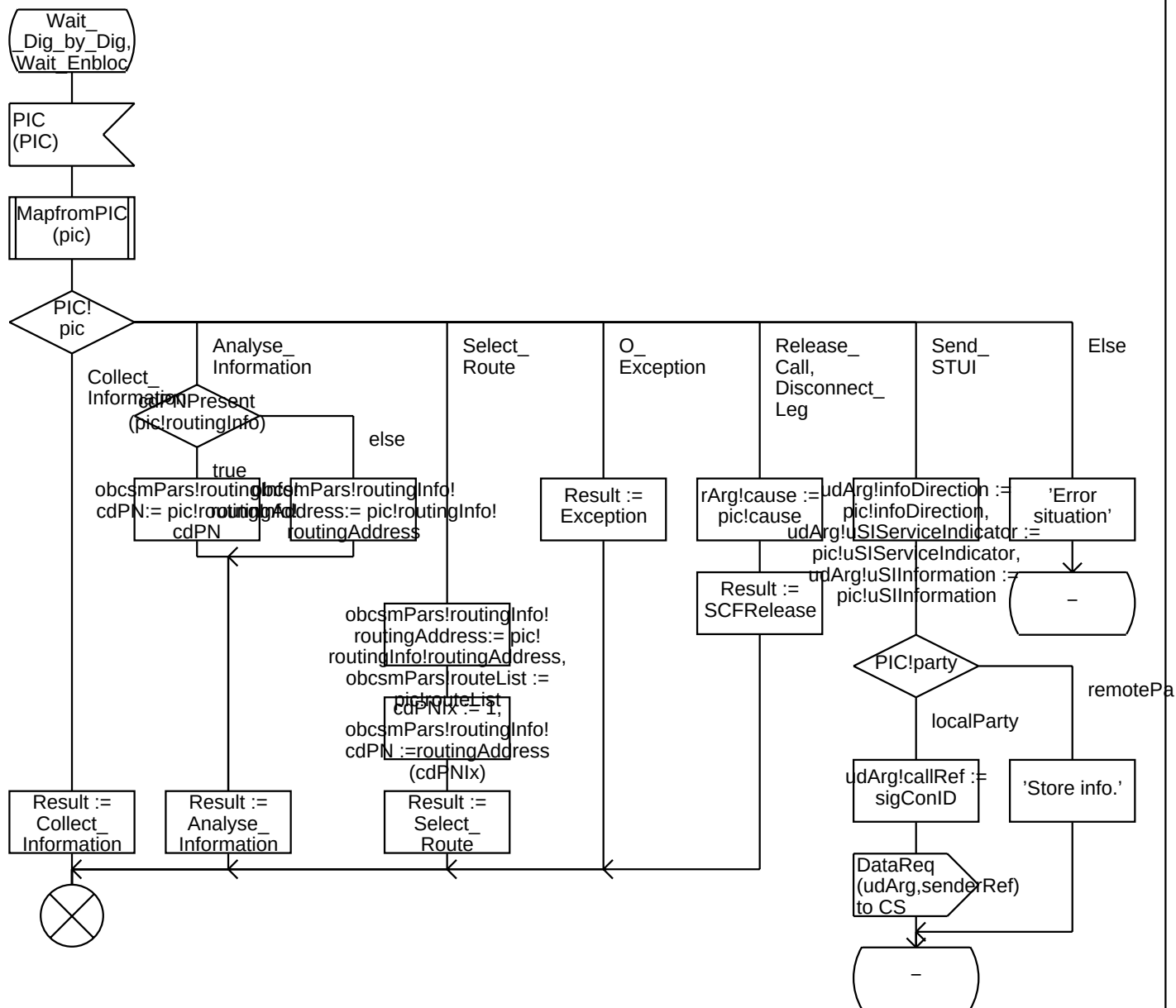
FPAR
IN/OUT Result DPResultType,
IN GenerateDP Boolean;



Procedure DP_Collected_Info

3(4)

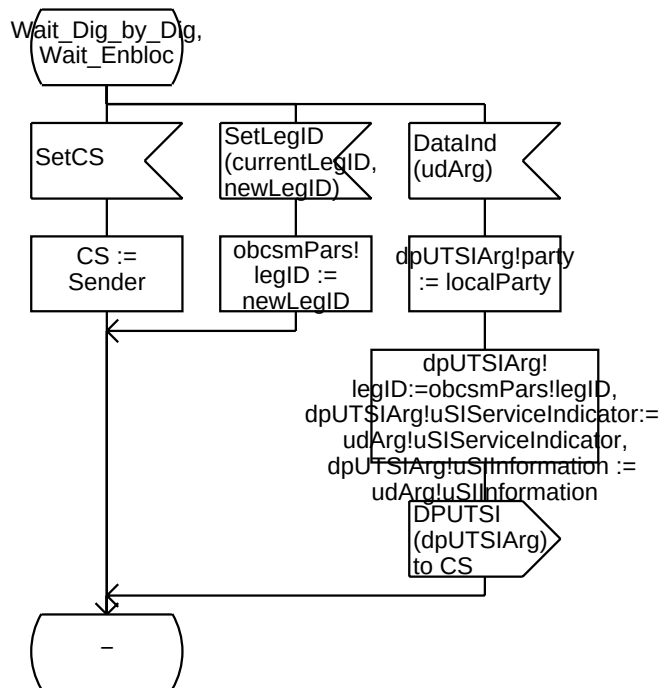
FPAR
IN/OUT Result DPResultType,
IN GenerateDP Boolean;



Procedure DP_Collected_Info

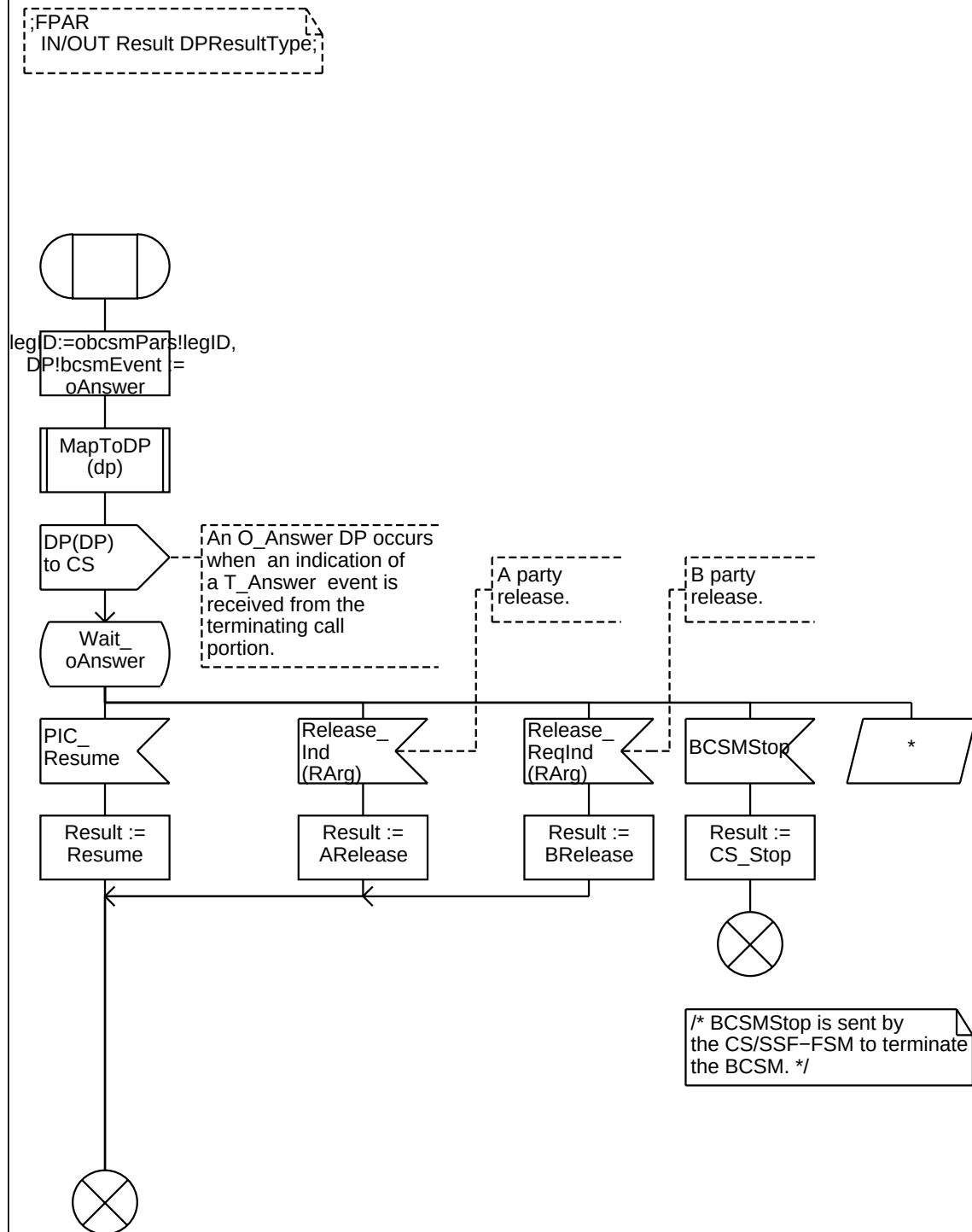
4(4)

FPAR
IN/OUT Result DPResultType,
IN GenerateDP Boolean;



Procedure DP_oAnswer

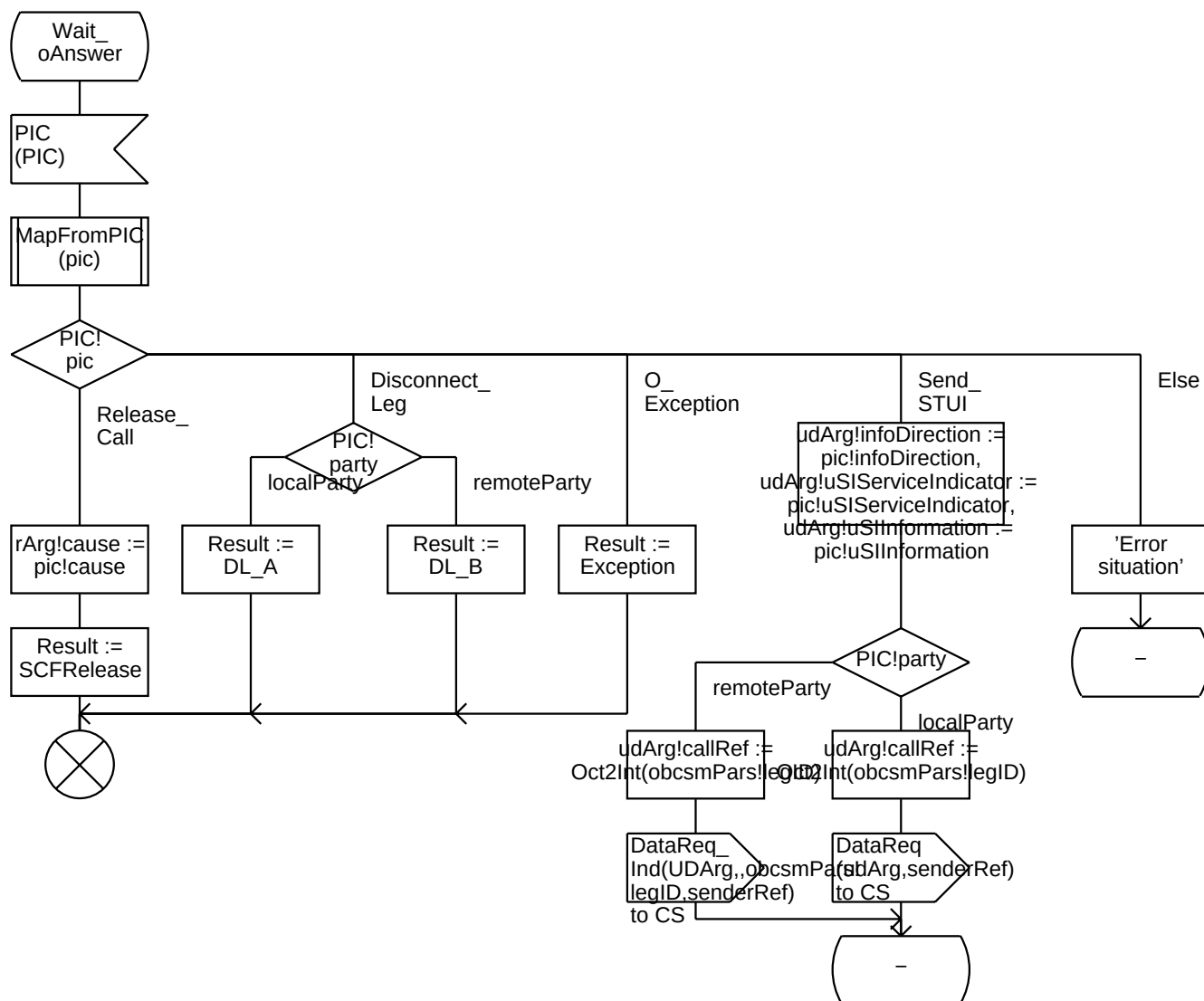
1(3)



Procedure DP_oAnswer

2(3)

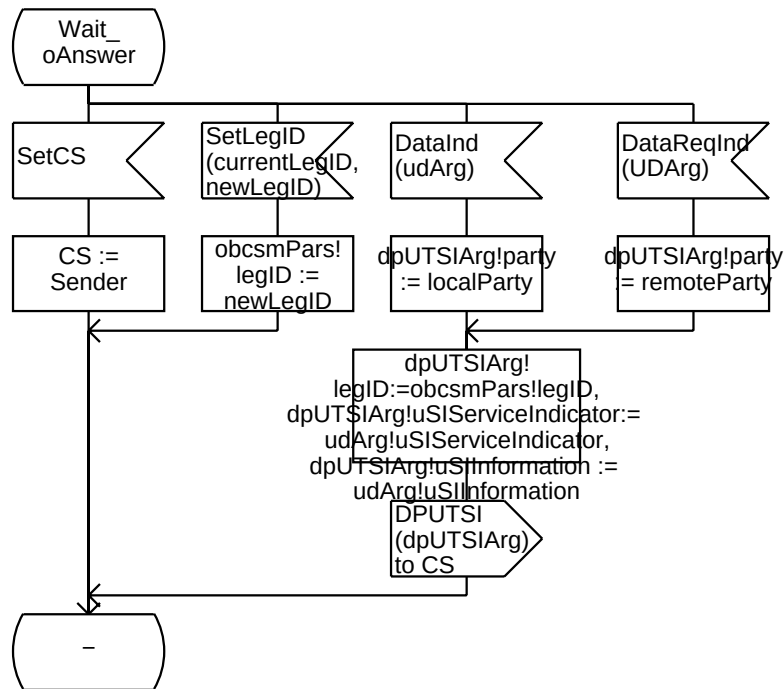
FPAR
IN/OUT Result DPResultType;



Procedure DP_oAnswer

3(3)

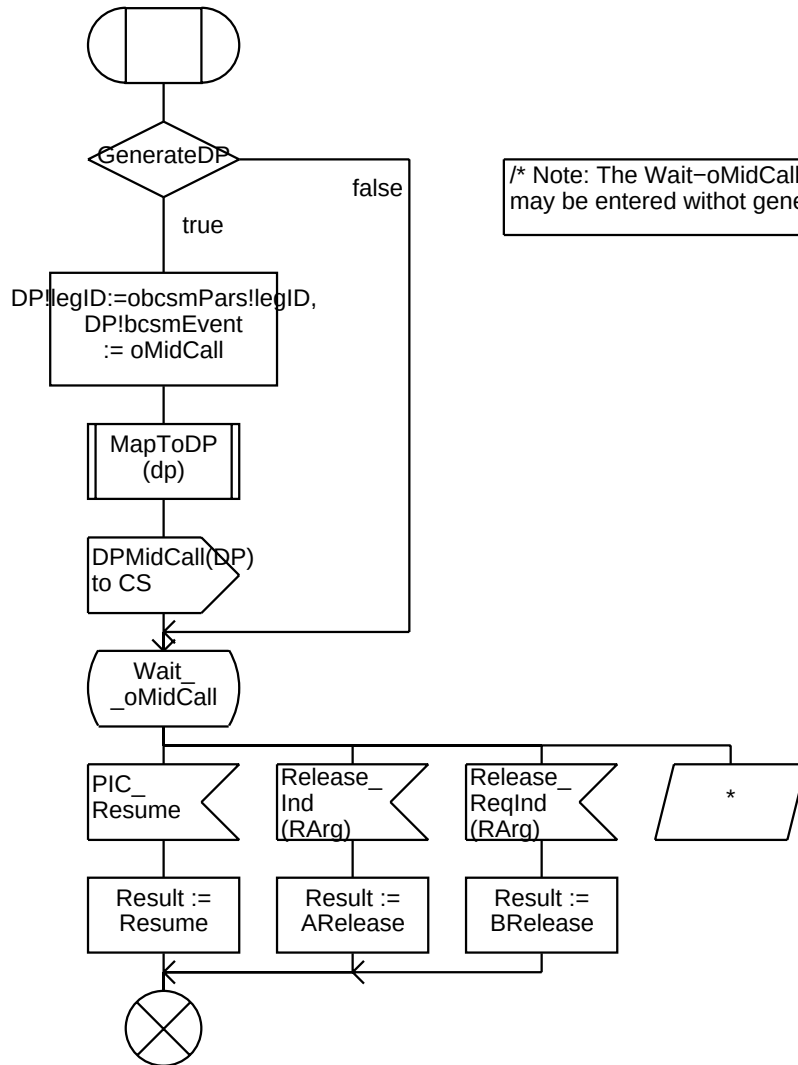
FPAR
 IN/OUT Result DPResultType;



Procedure DP_oMidCall

1(3)

FPAR
IN/OUT Result DPResultType,
IN GenerateDP Boolean;

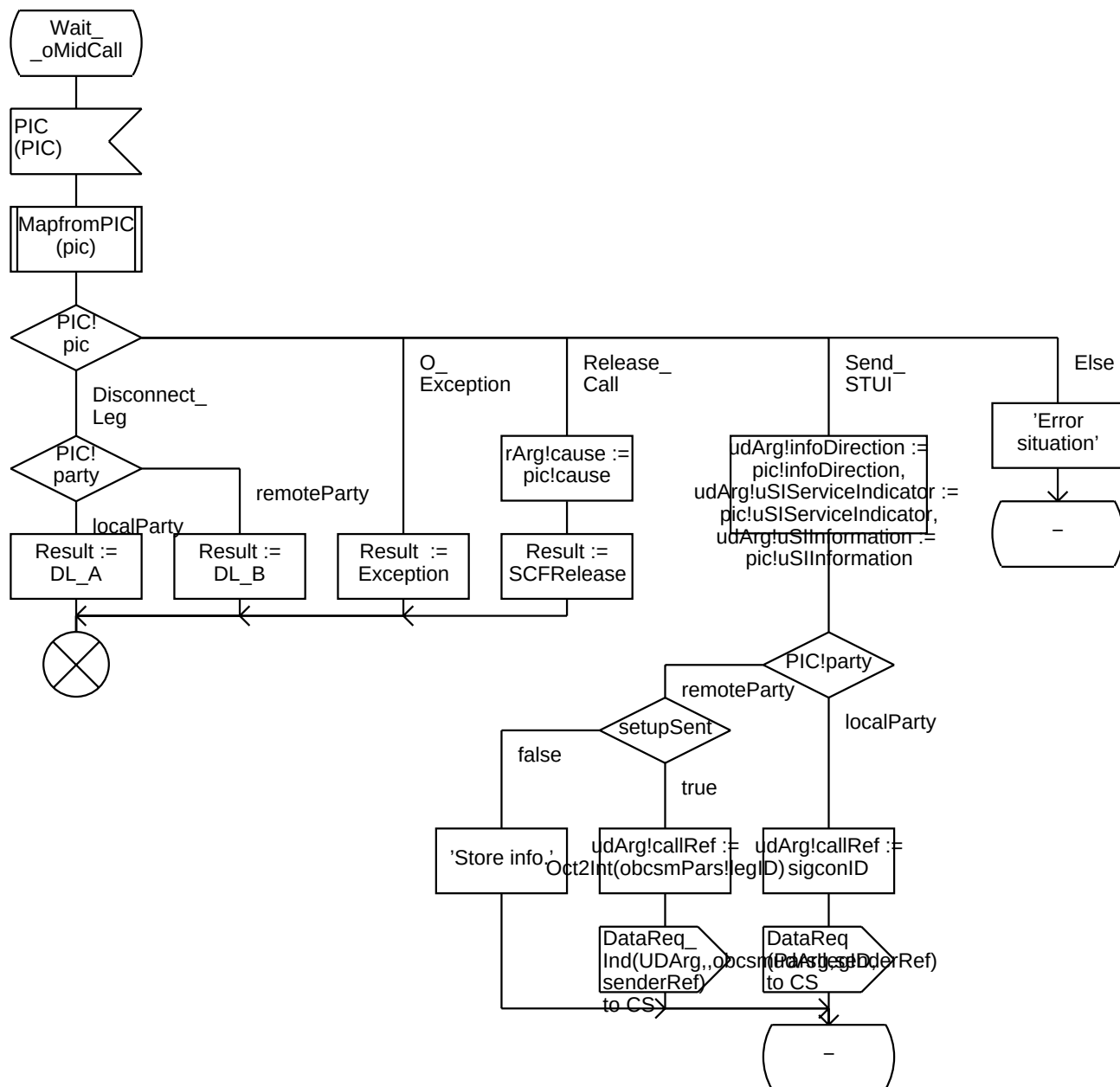


/* Note: The Wait-oMidCall state
may be entered without generating any DP. */

Procedure DP_oMidCall

2(3)

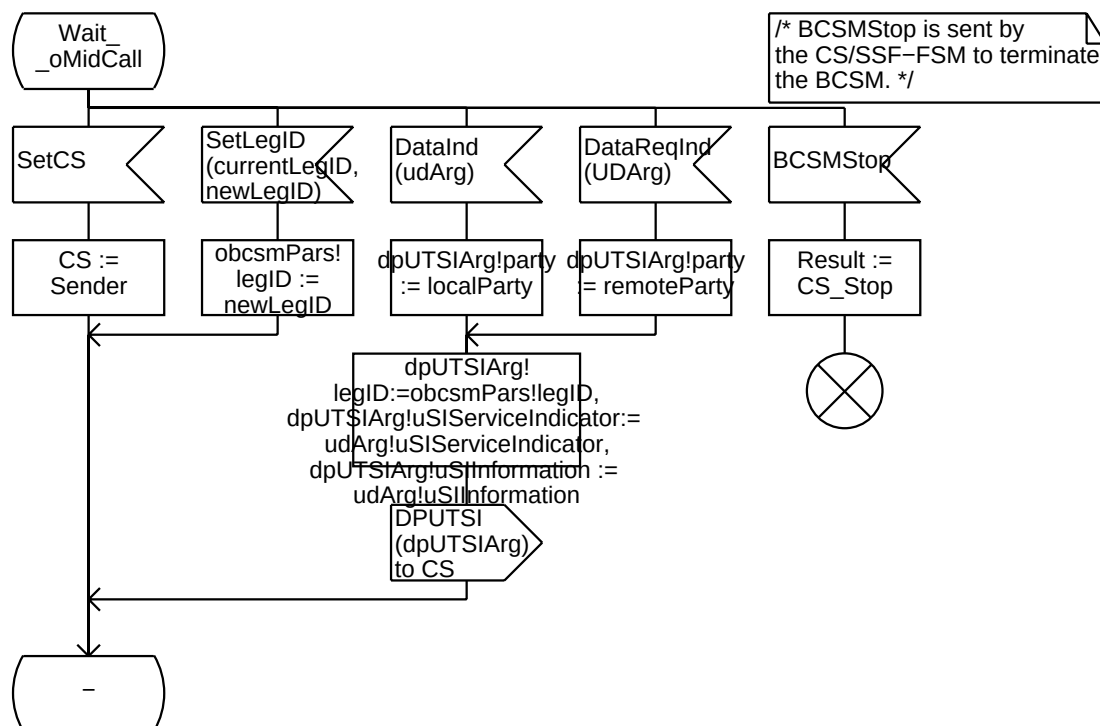
FPAR
IN/OUT Result DPResultType,
IN GenerateDP Boolean;



Procedure DP_oMidCall

3(3)

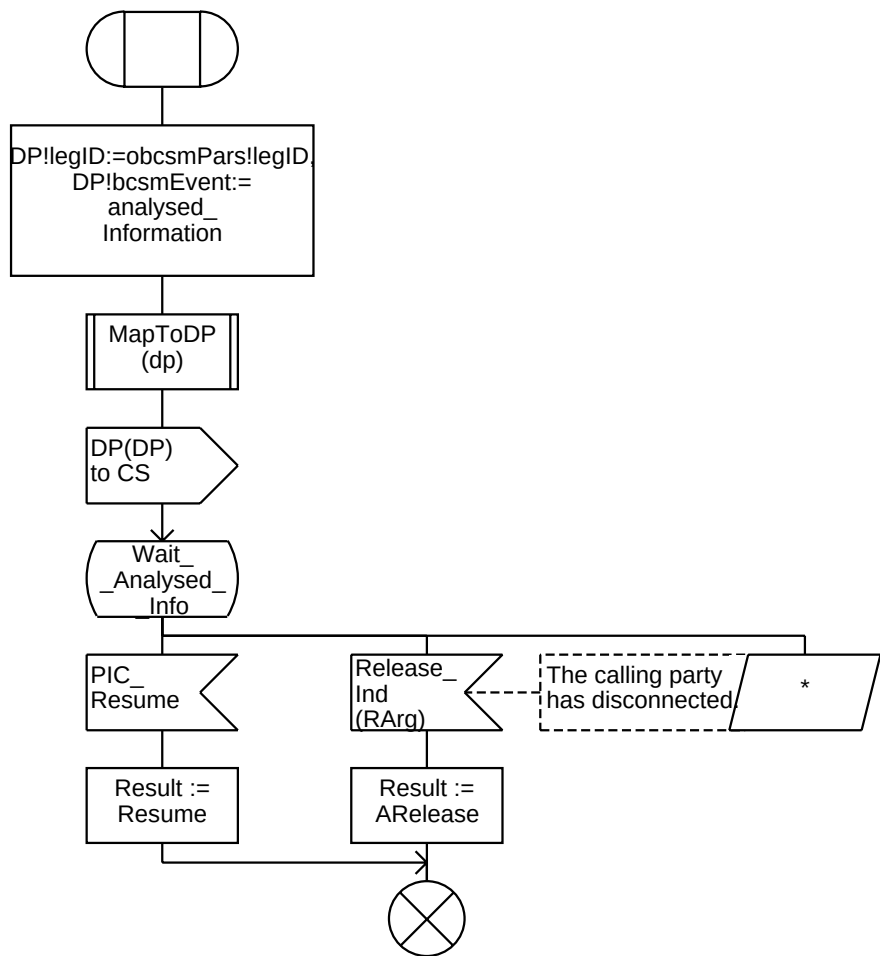
FPAR
IN/OUT Result DPResultType,
IN GenerateDP Boolean;



Procedure DP_Analysed_Information

1(3)

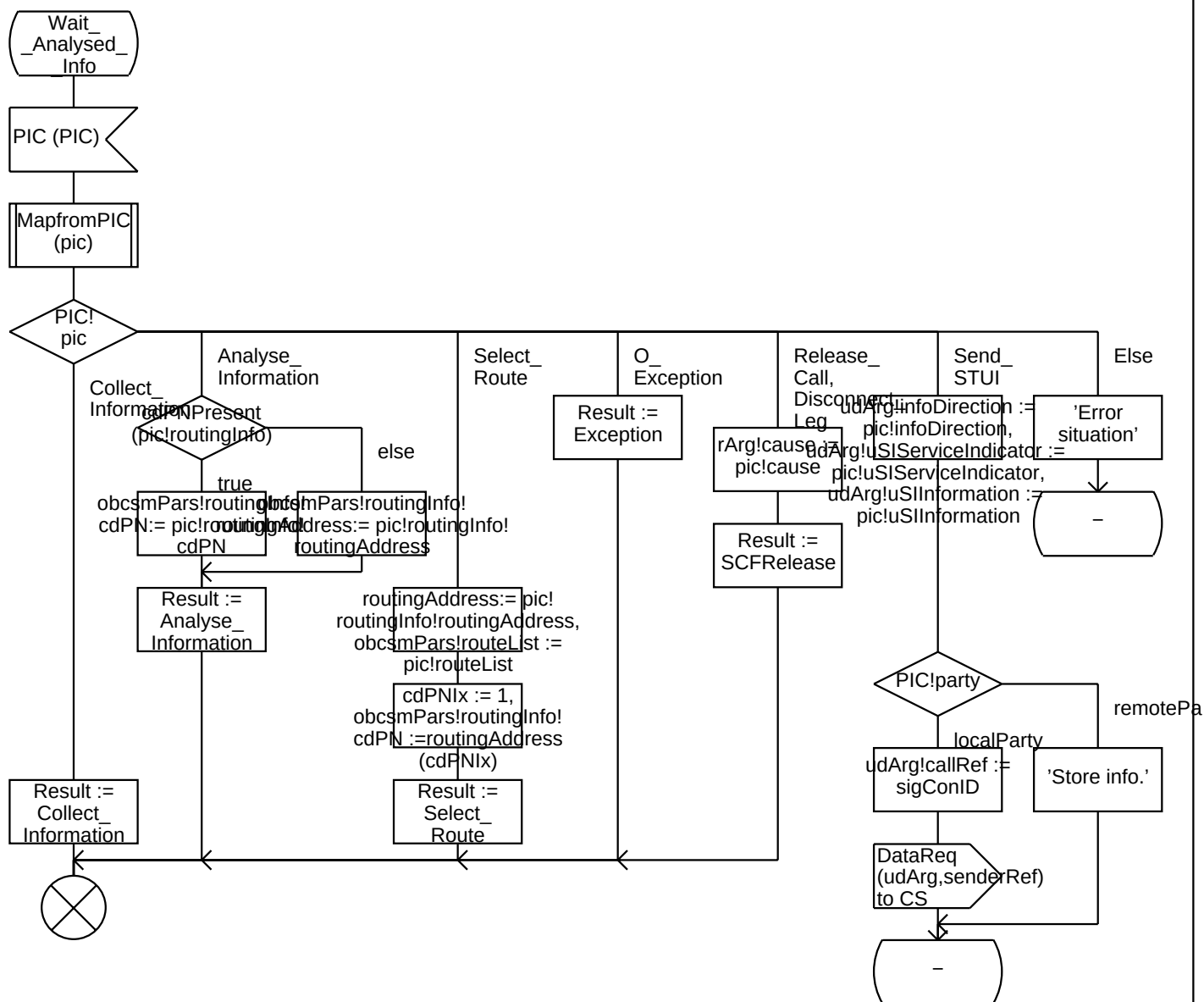
FPAR
IN/OUT Result DPResultType;



Procedure DP_Analysed_Information

2(3)

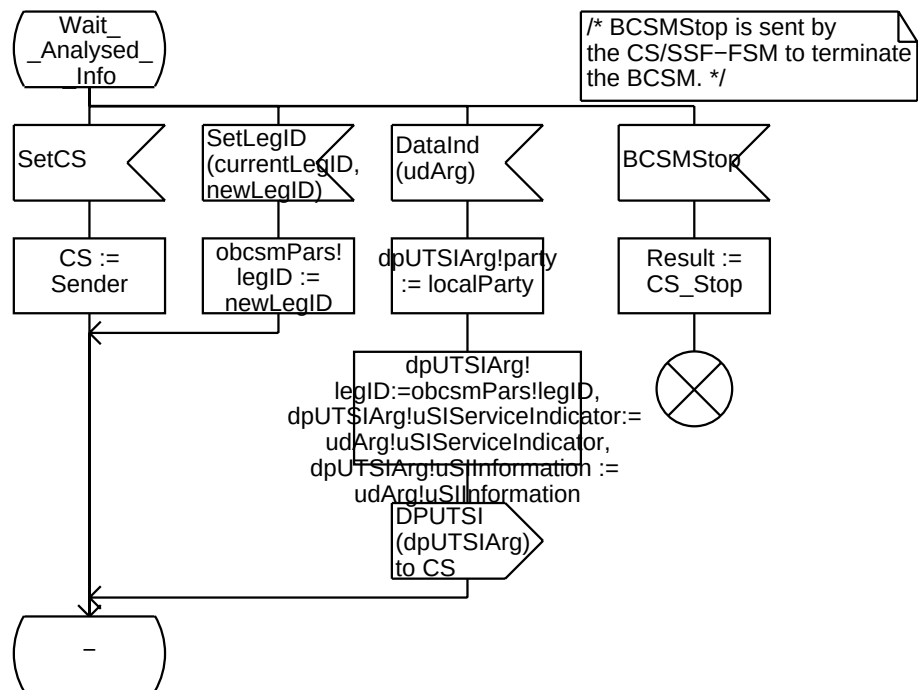
FPAR
IN/OUT Result DPResultType;



Procedure DP_Analysed_Information

3(3)

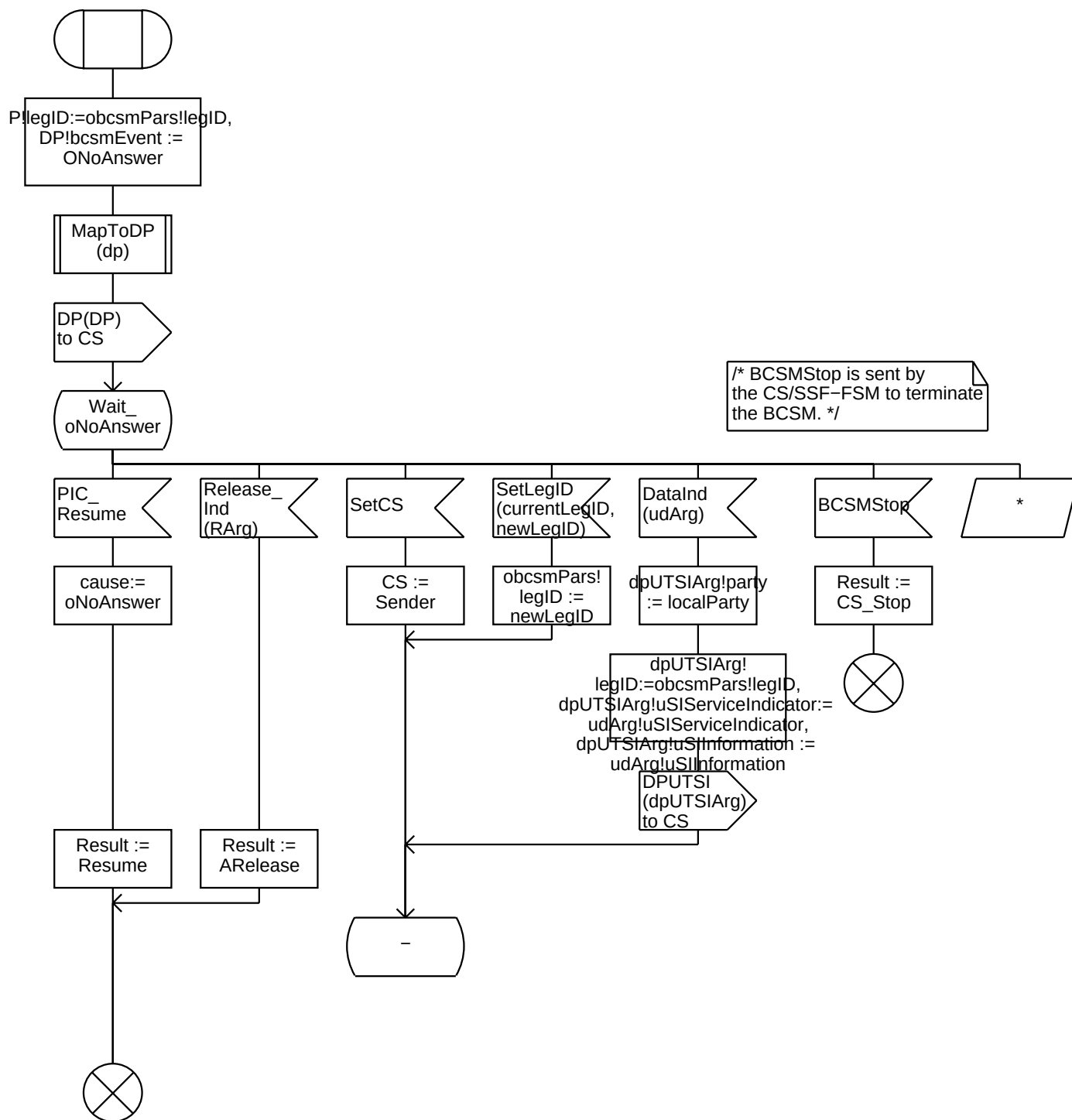
FPAR
IN/OUT Result DPResultType;



Procedure DP_oNoAnswer

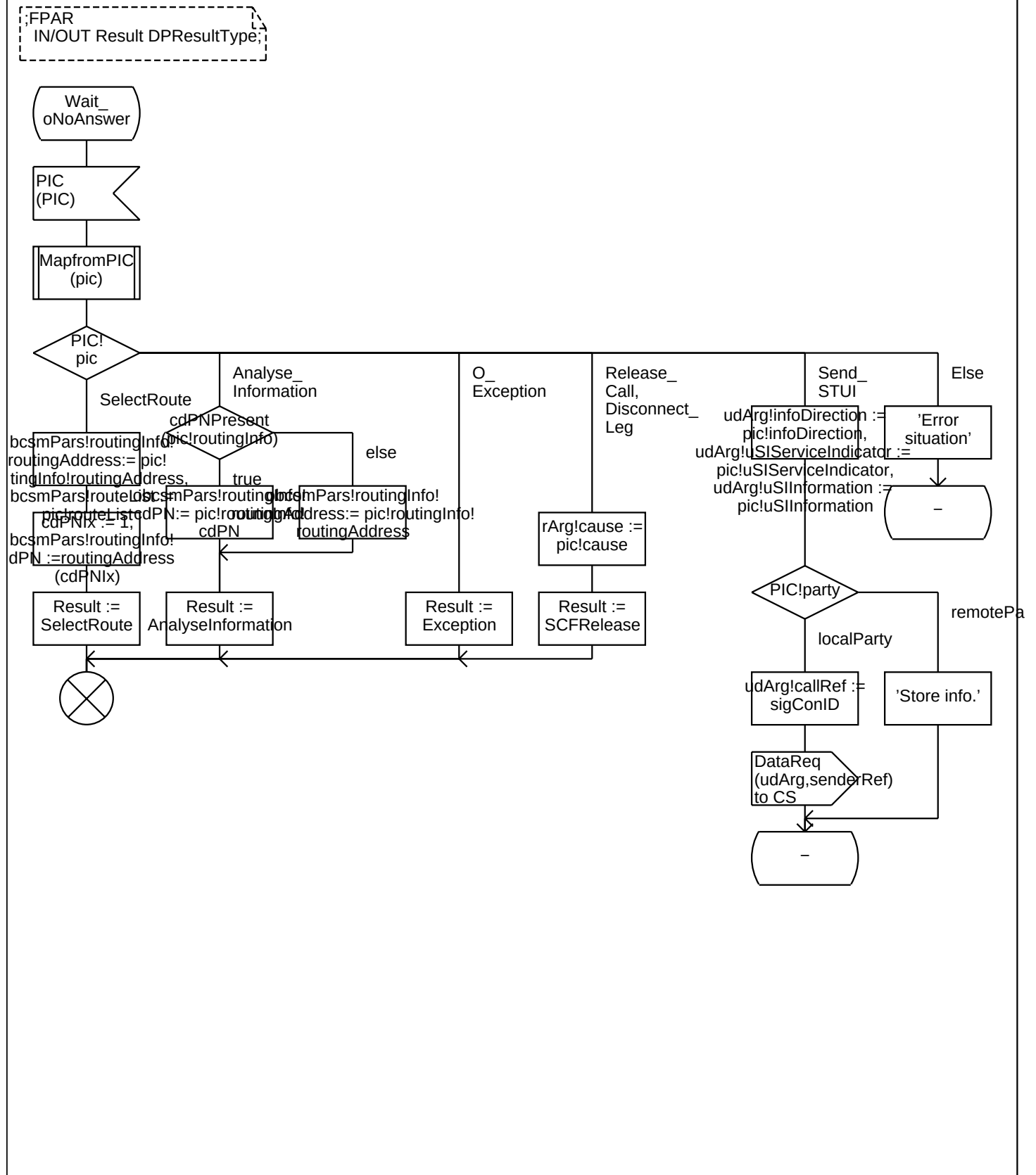
1(2)

FPAR
IN/OUT Result DPResultType;



Procedure DP_oNoAnswer

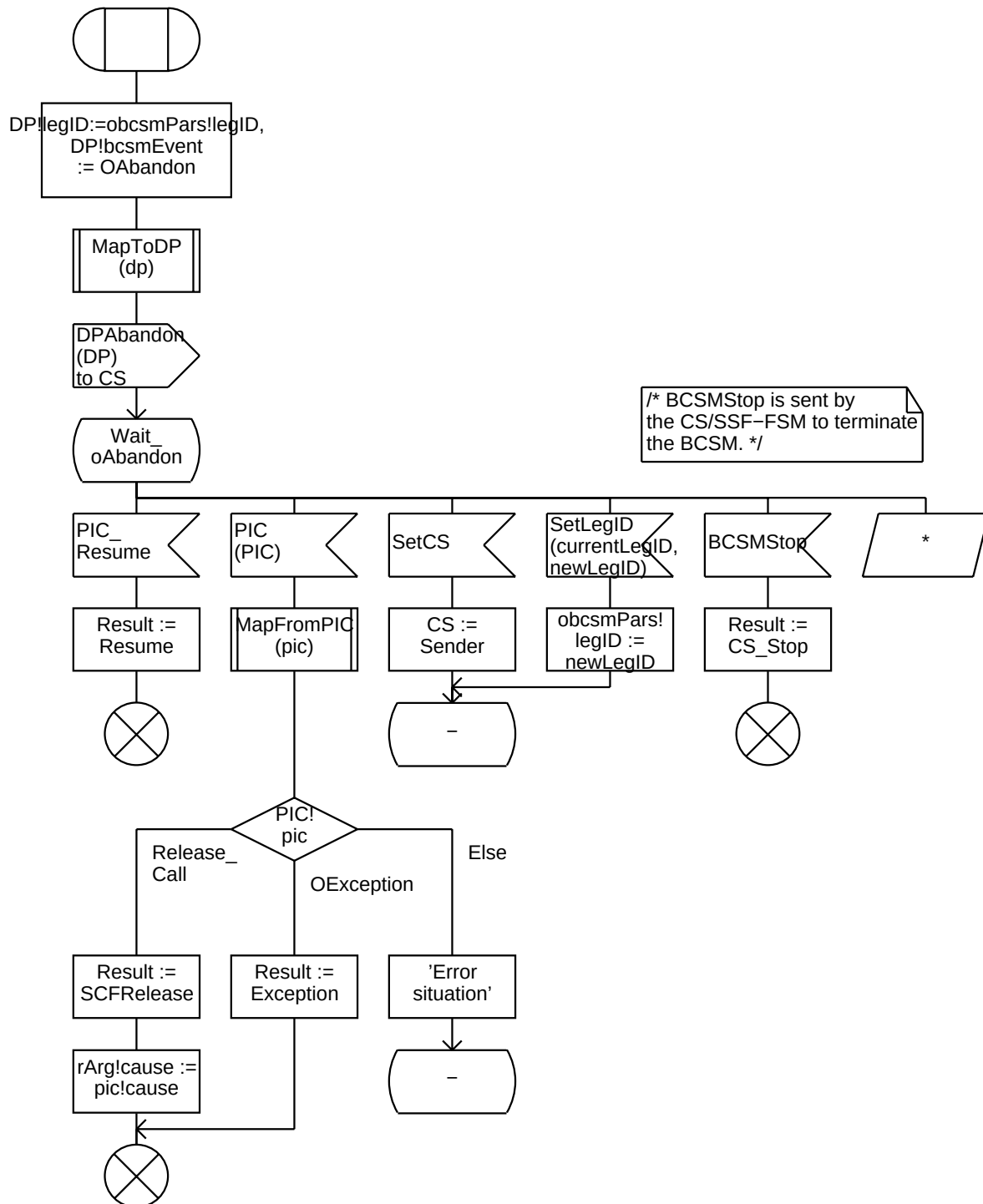
2(2)



Procedure DP_oAbandon

1(1)

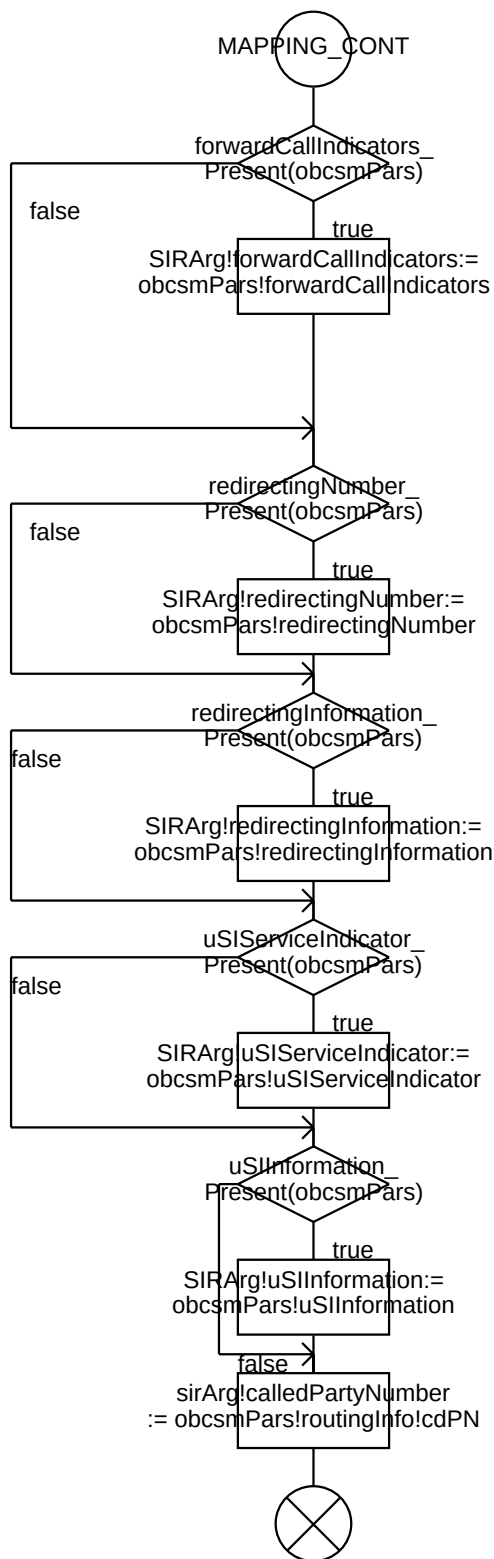
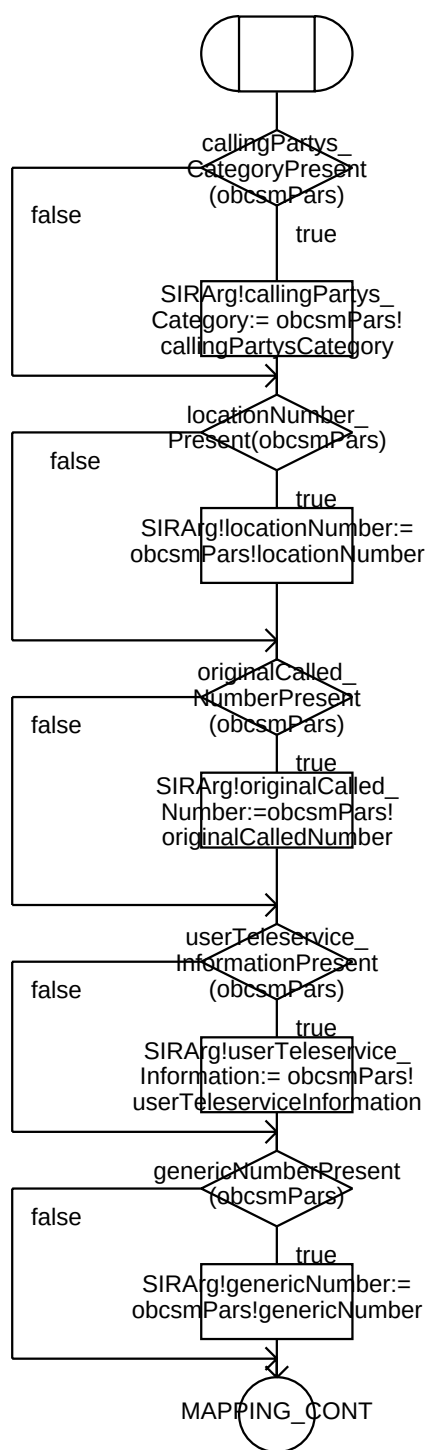
FPAR
IN/OUT Result DPResultType;



Procedure MapToSIRArg

1(1)

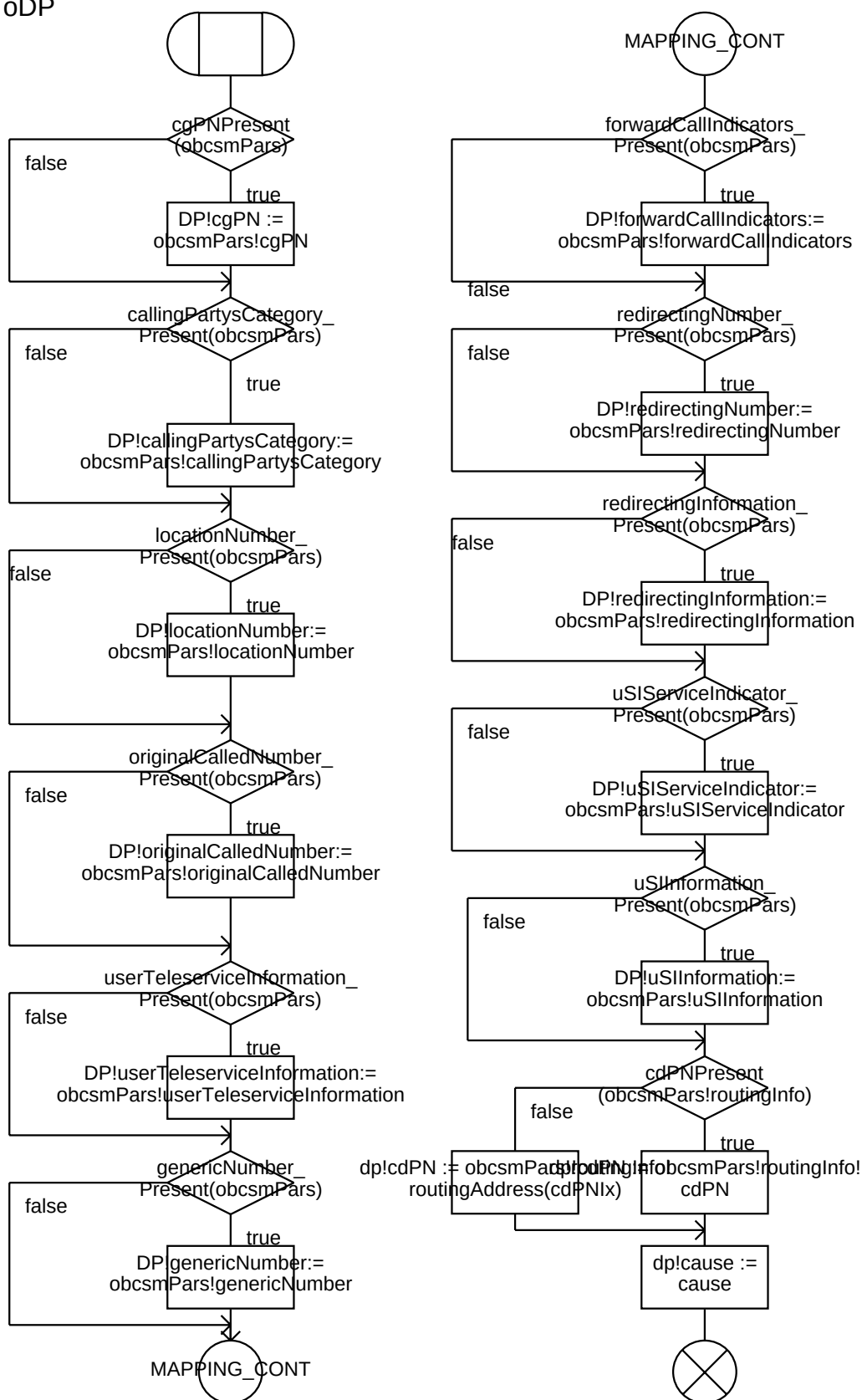
FPAR
IN/OUT sirArg SetupIRType;



Procedure MapToDP

1(1)

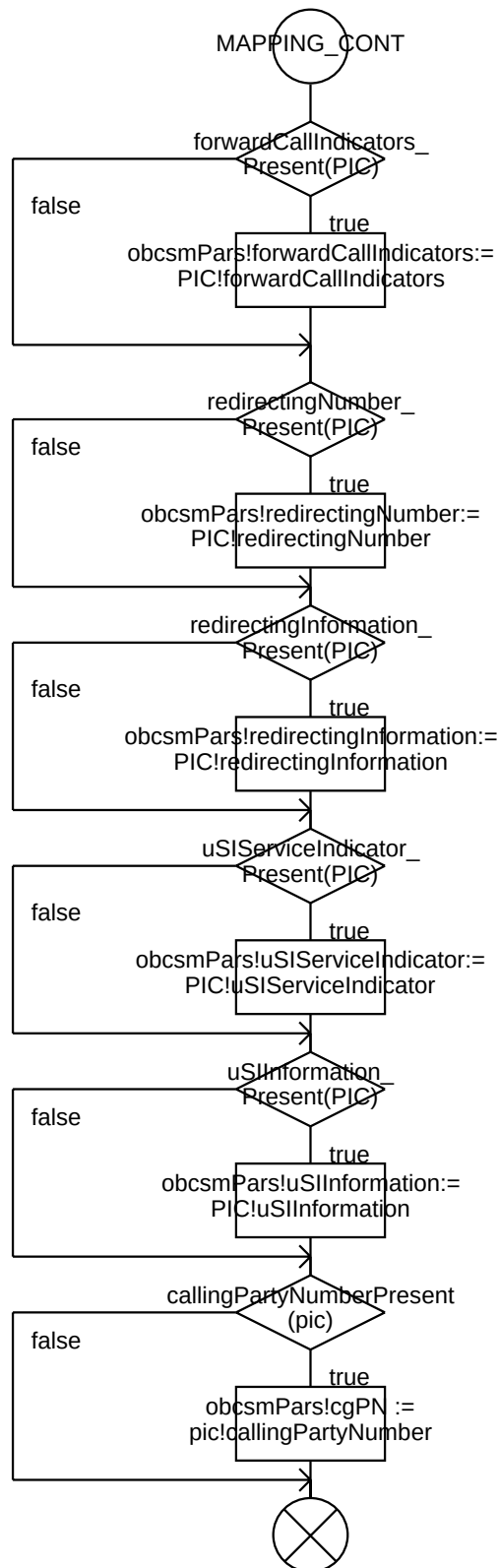
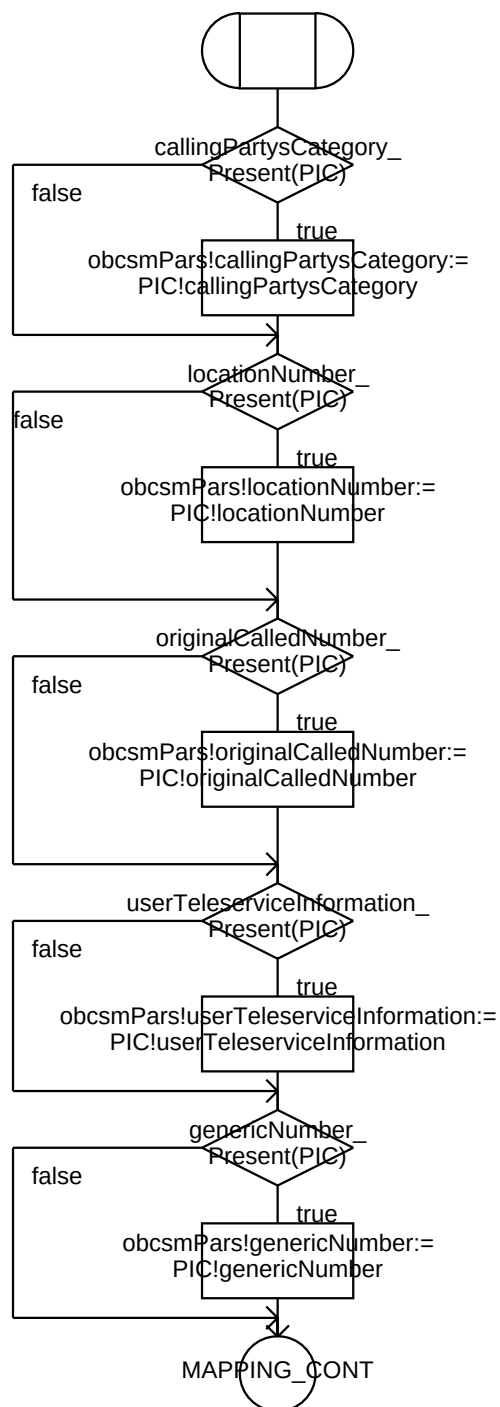
FPAR
IN/OUT dp DPArg;



Procedure MapFromPIC

1(1)

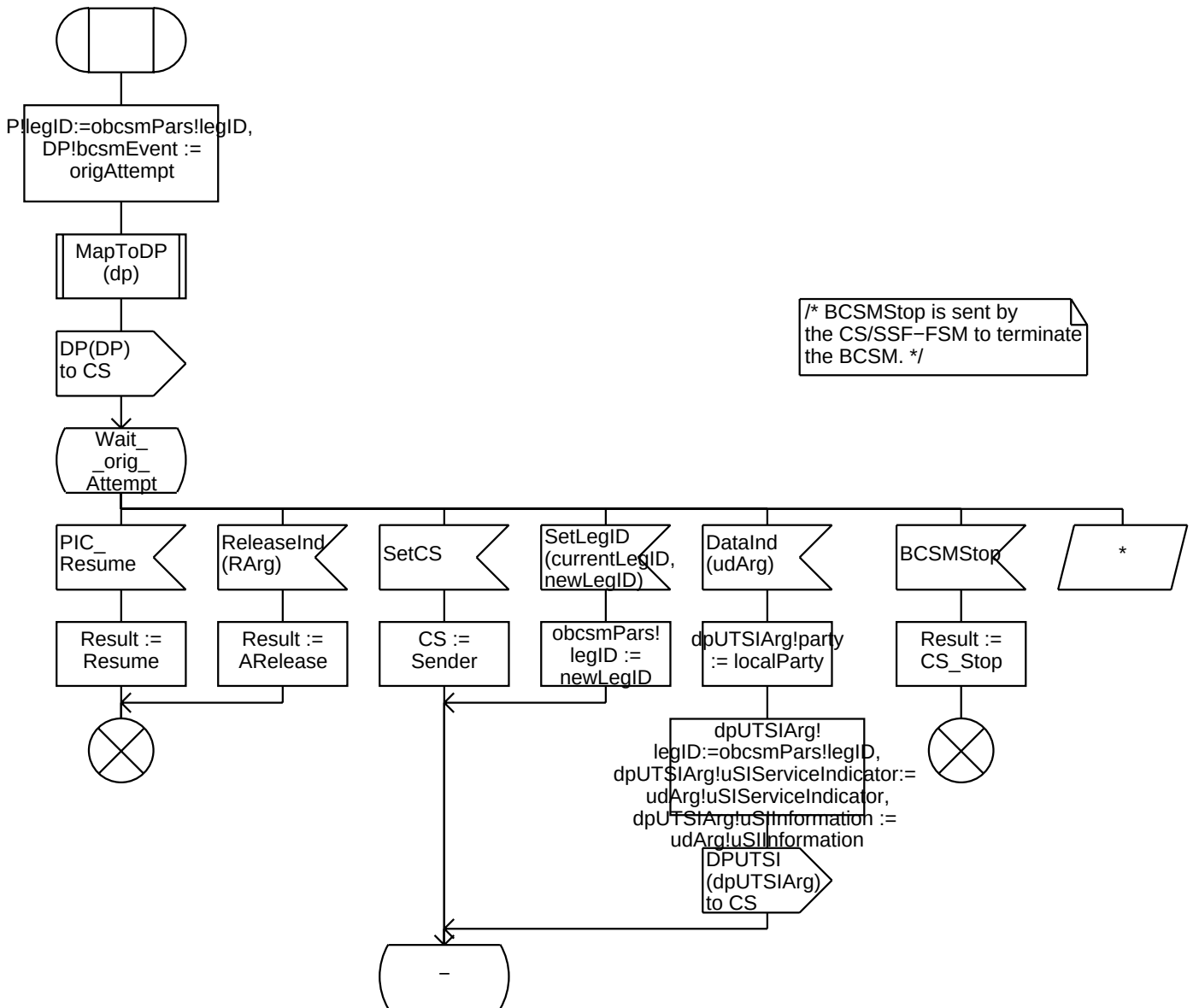
FPAR
IN pic PICArg;



Procedure DP_origAttempt

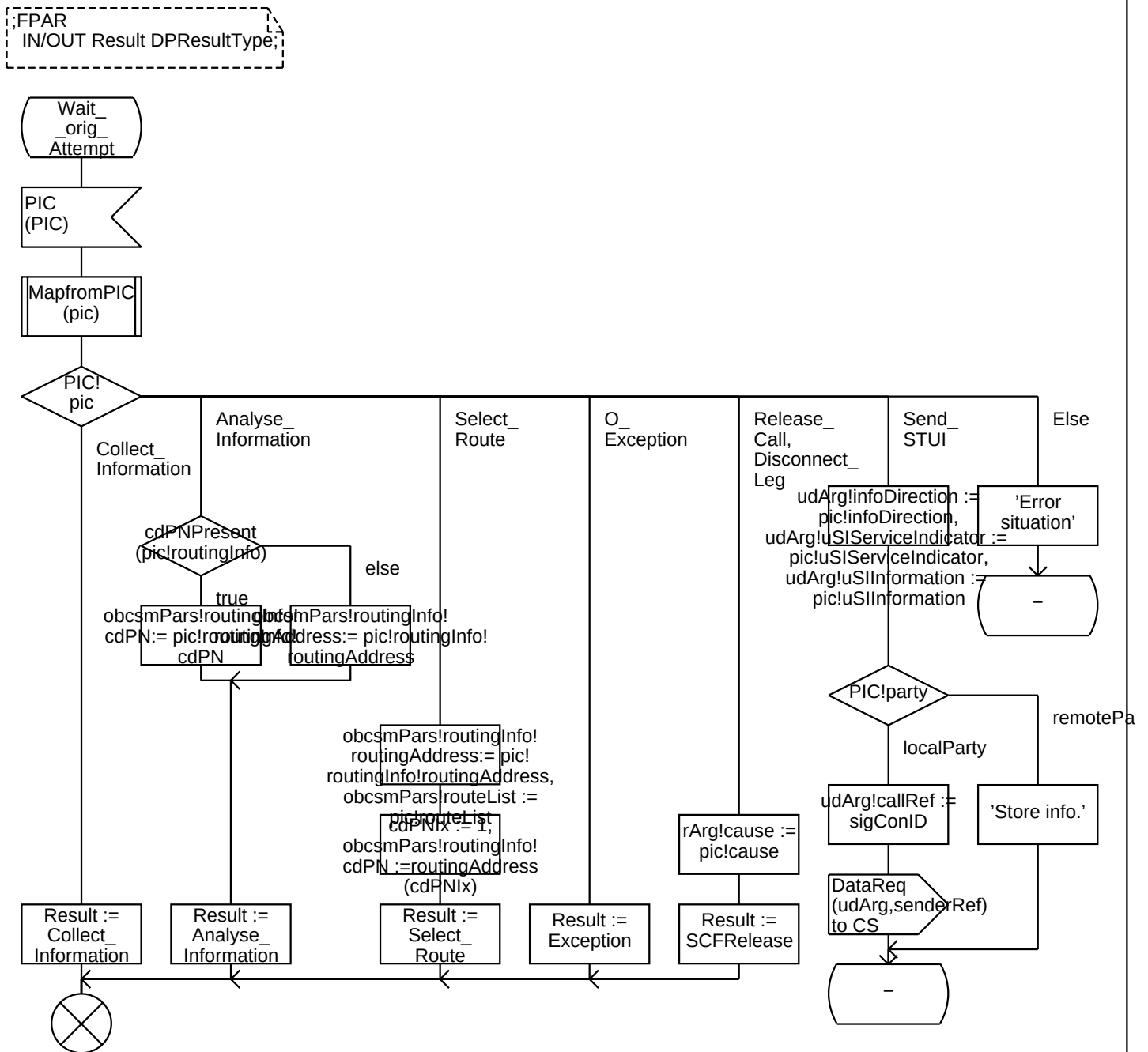
1(2)

FPAR
IN/OUT Result DPResultType;



Procedure DP_origAttempt

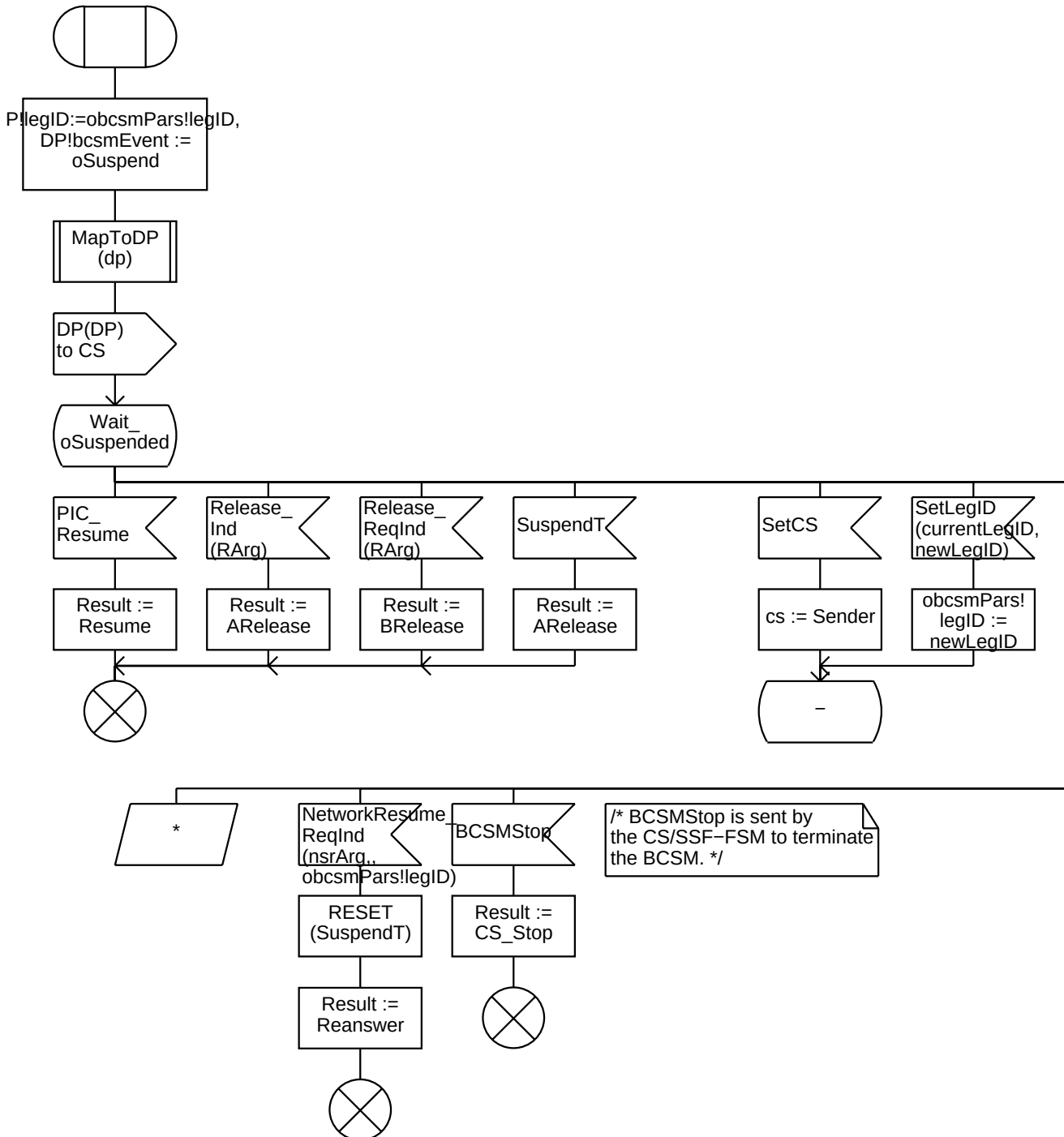
2(2)



Procedure DP_oSuspend

1(2)

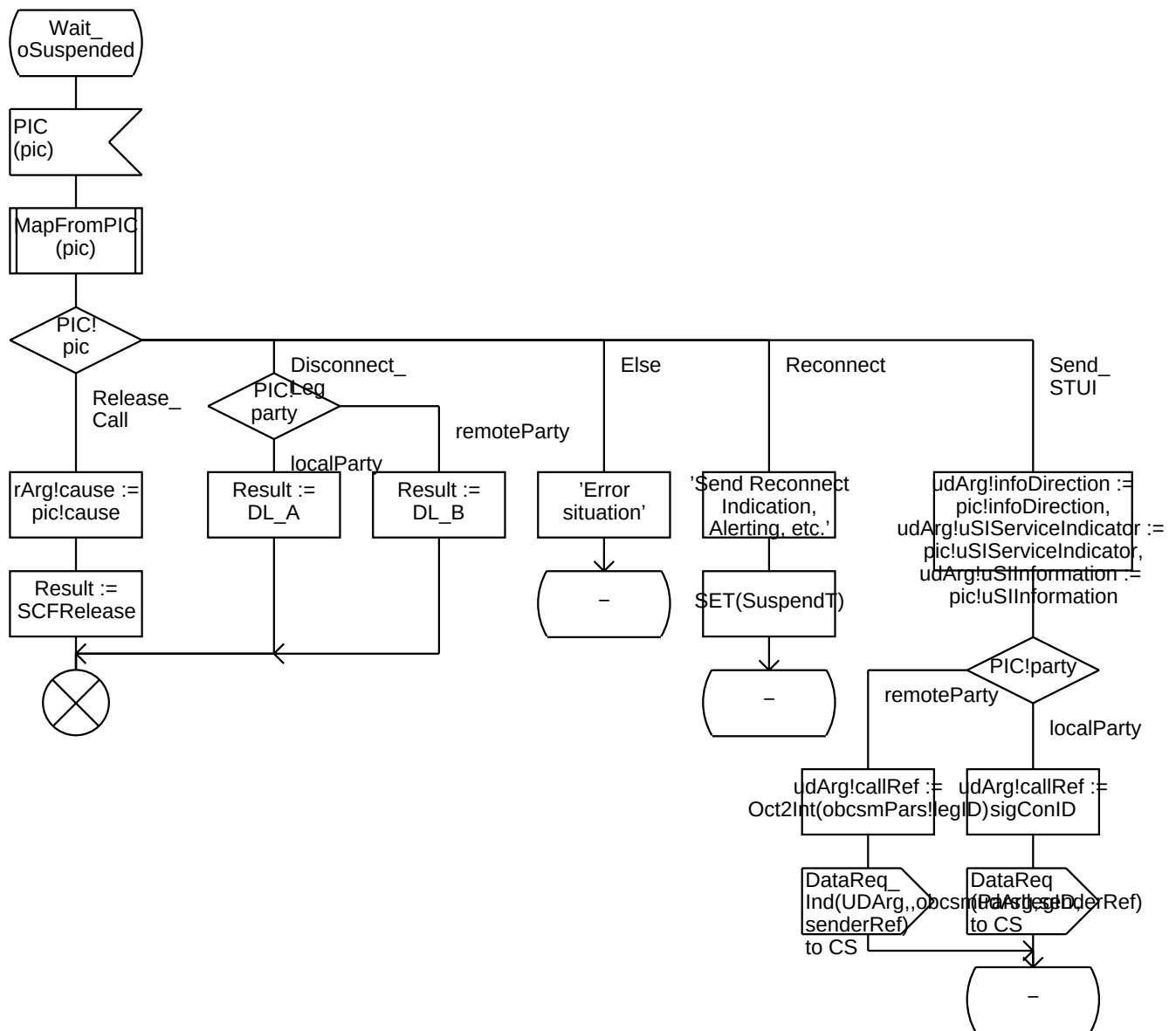
FPAR
IN/OUT Result DPResultType;



Procedure DP_oSuspend

2(2)

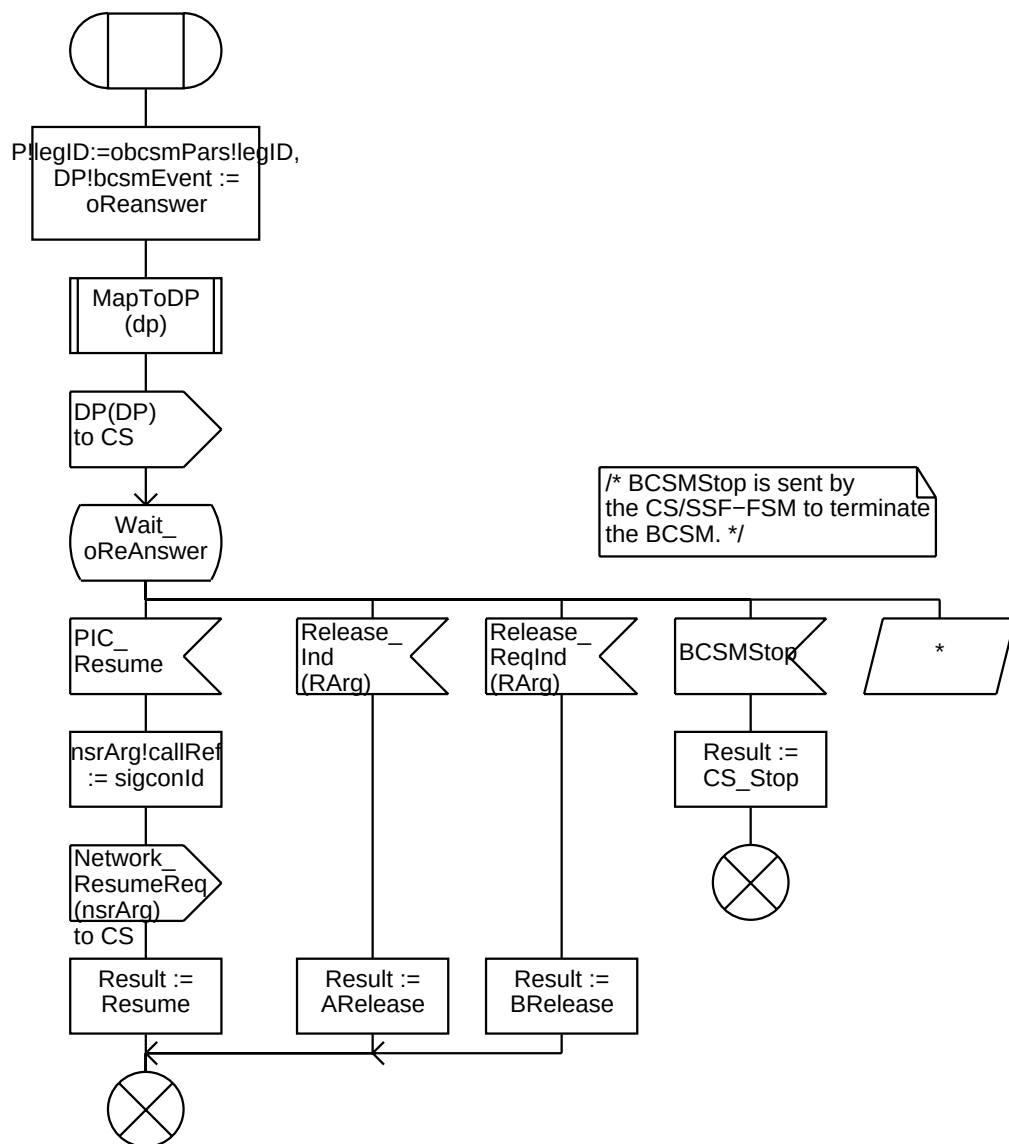
FPAR
IN/OUT Result DPResultType;



Procedure DP_oReanswer

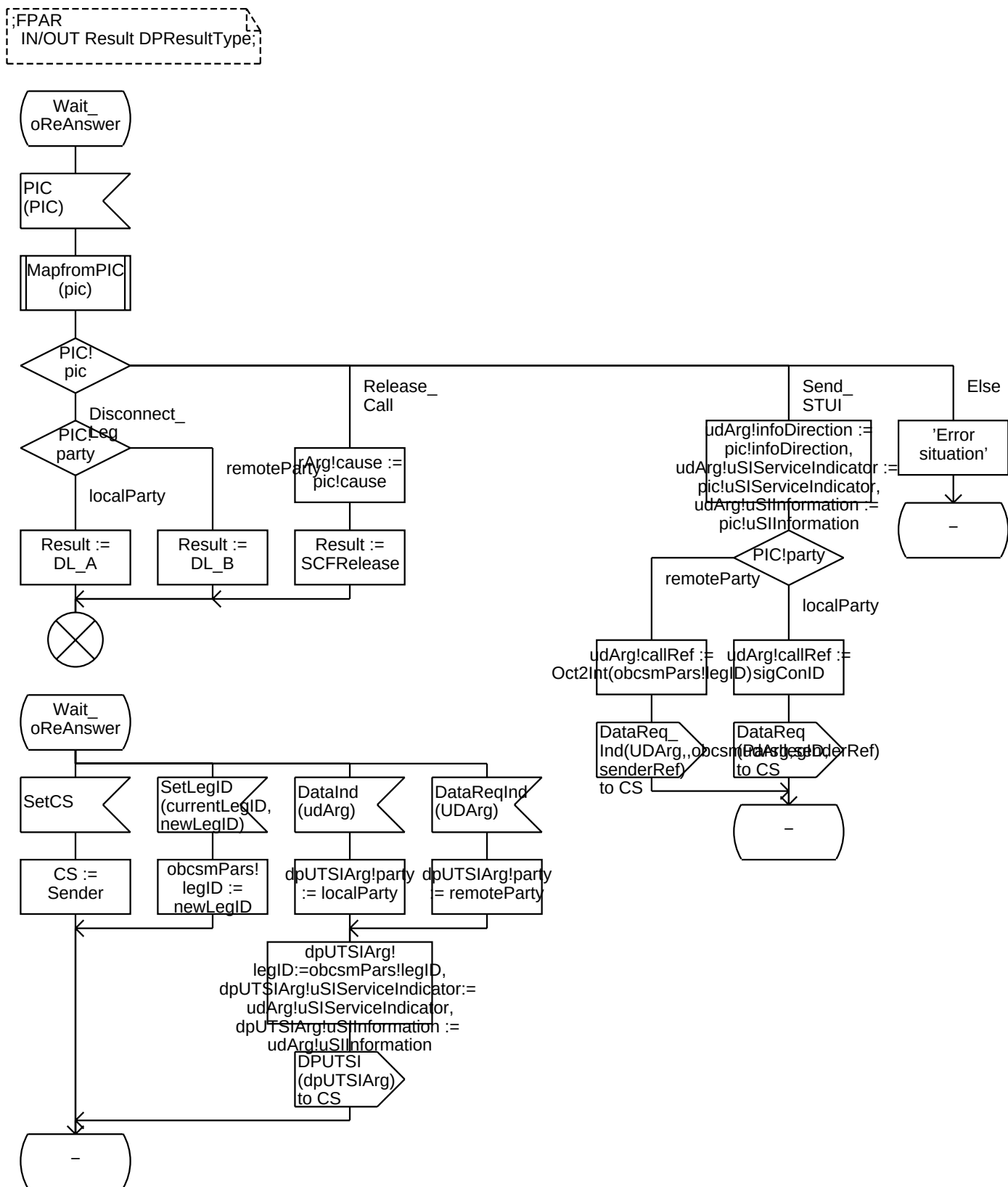
1(2)

FPAR
IN/OUT Result DPResultType;



Procedure DP_oReanswer

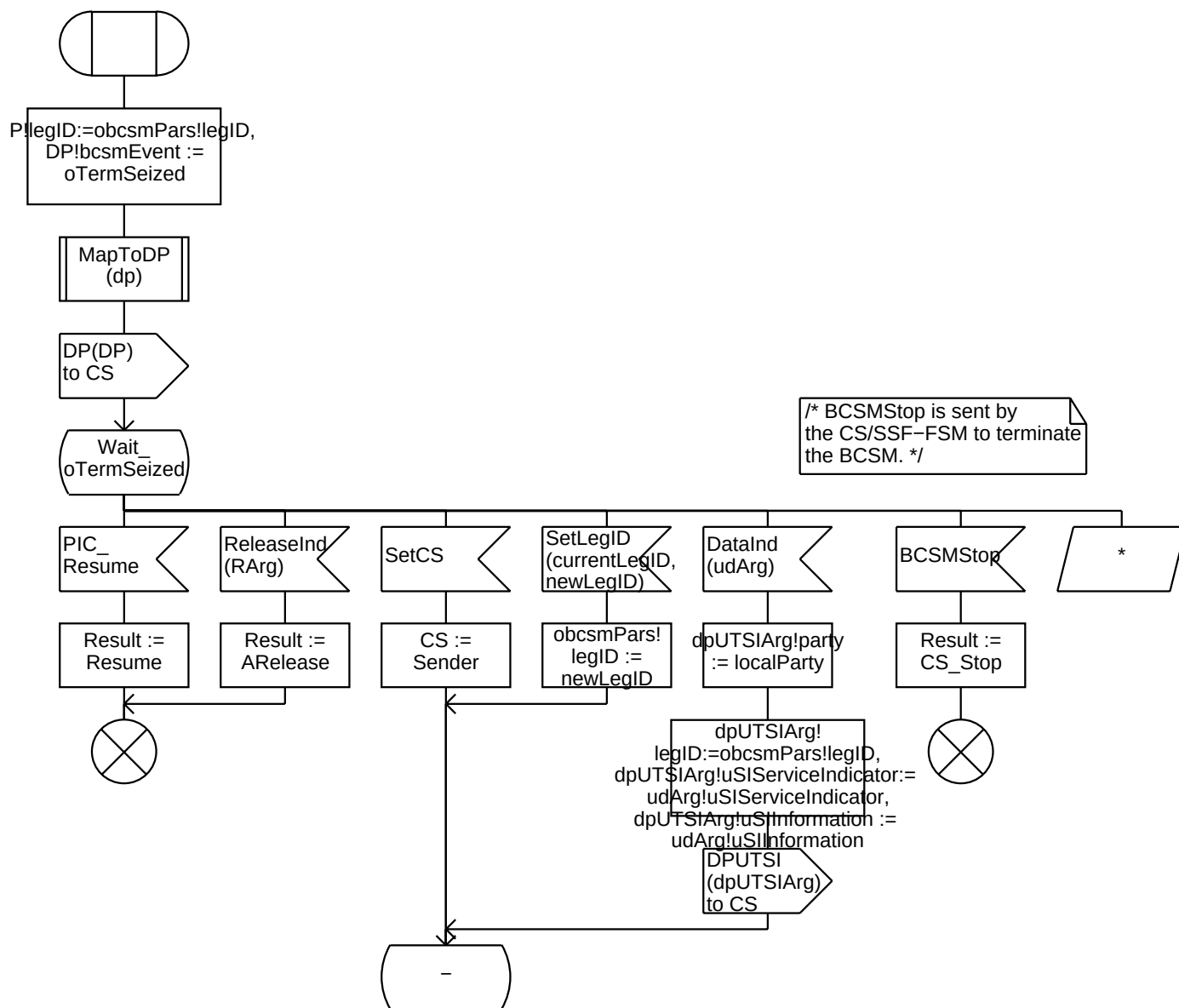
2(2)



Procedure DP_oTermSeized

1(2)

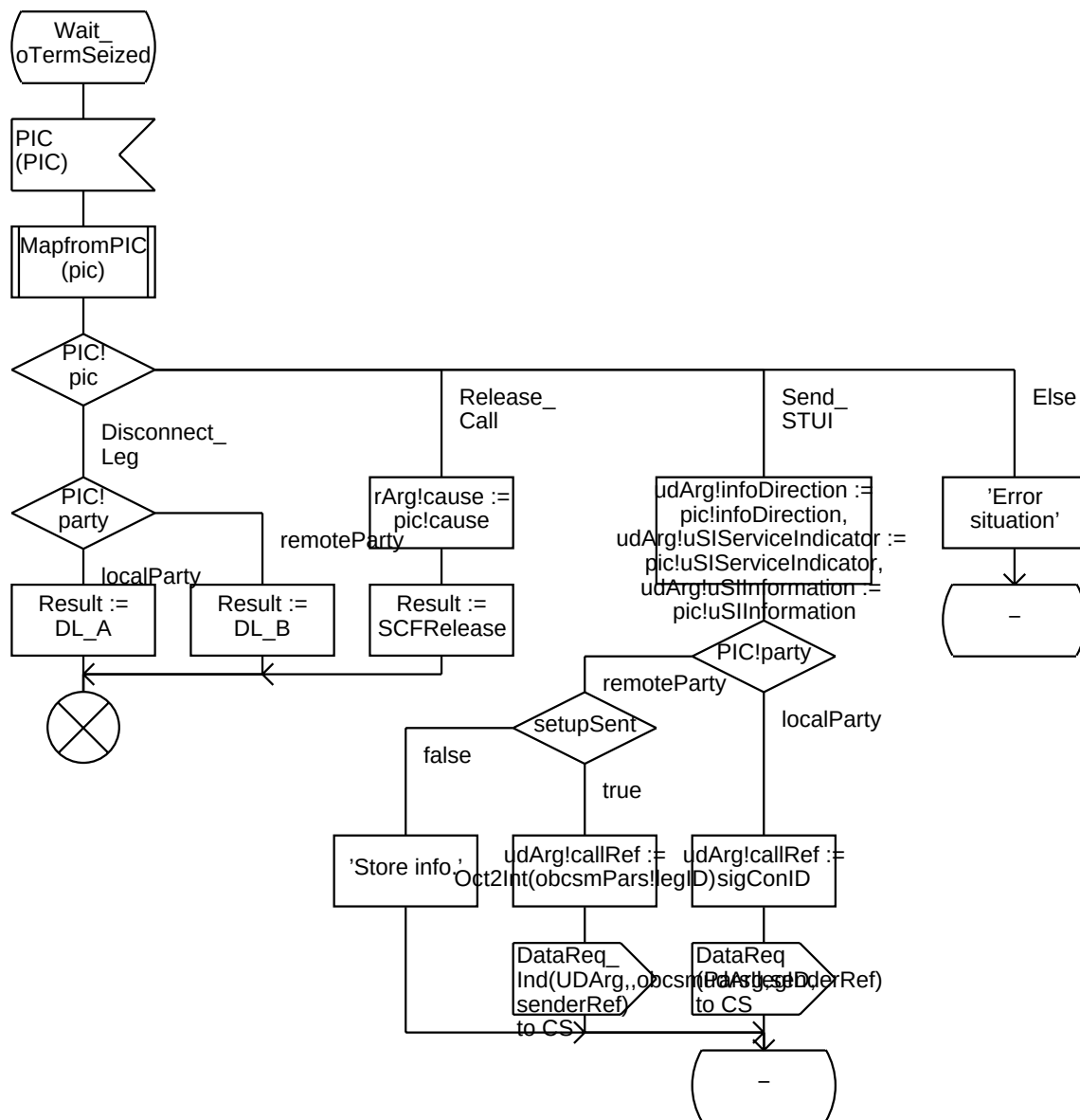
FPAR
IN/OUT Result DPResultType;



Procedure DP_oTermSeized

2(2)

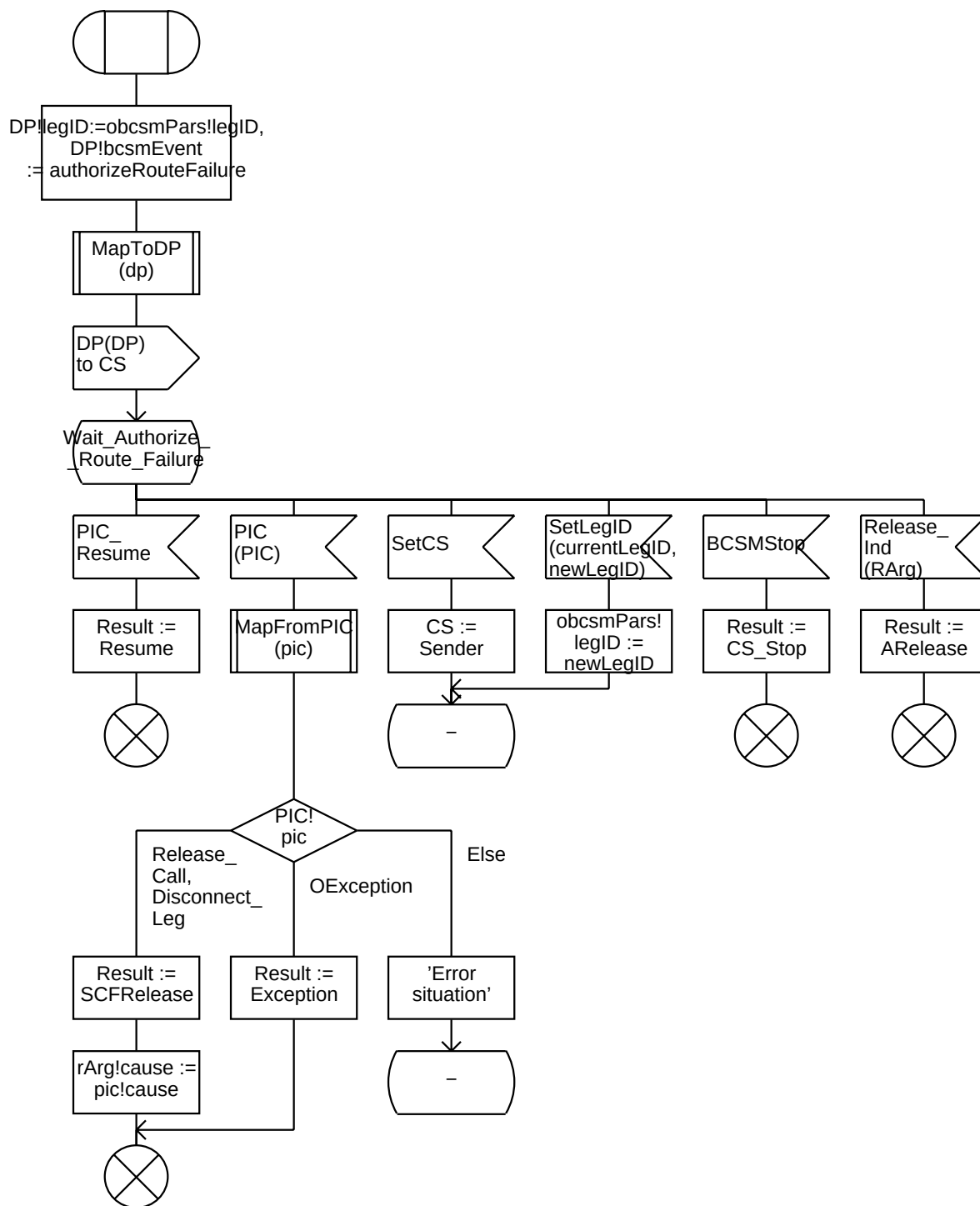
FPAR
IN/OUT Result DPResultType;



Procedure DP_Authorize_Route_Failure

1(1)

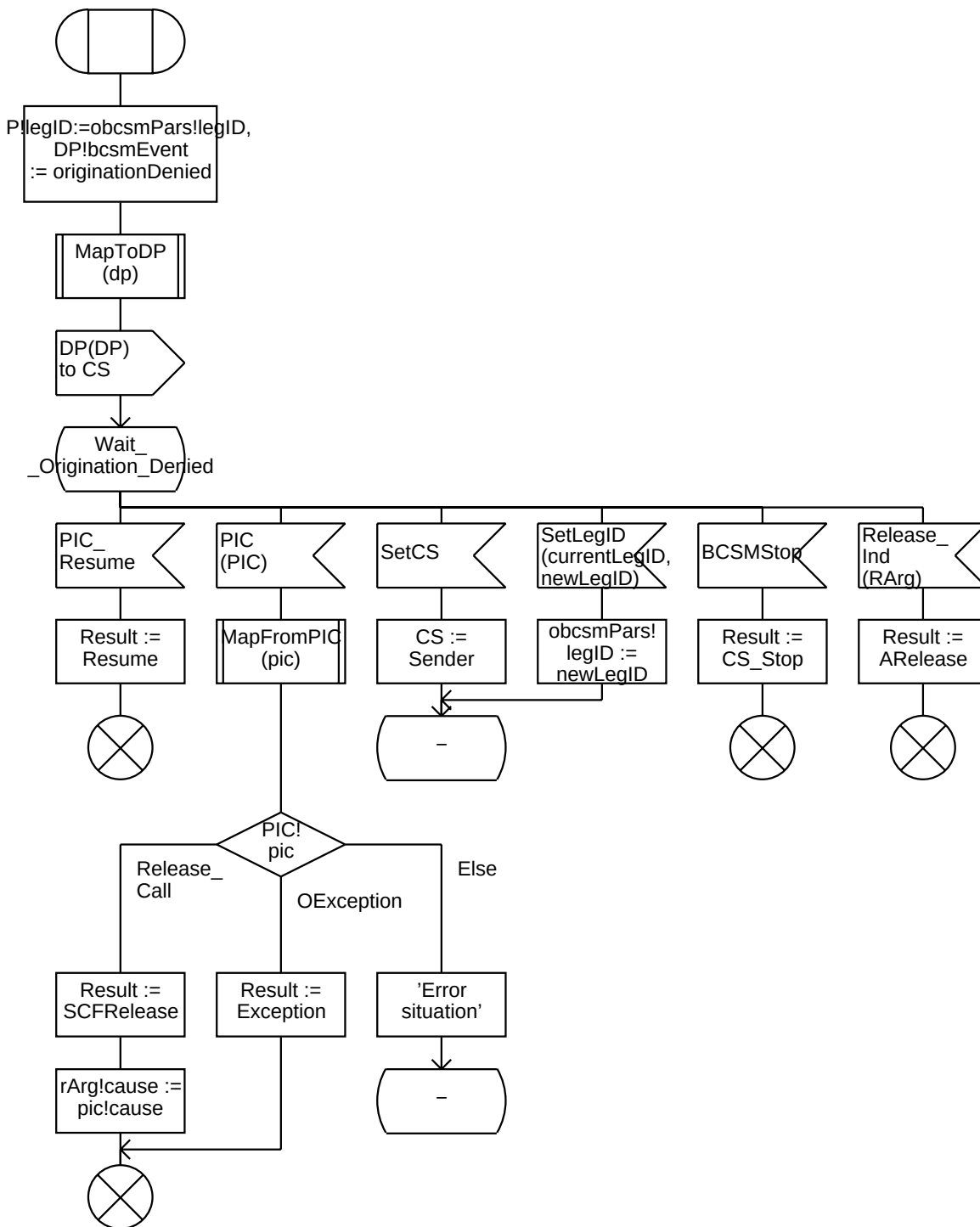
FPAR
IN/OUT Result DPResultType;



Procedure DP_origDenied

1(1)

FPAR
IN/OUT Result DPResultType;



Process Type TerminatingBCSM

1(12)

```
FPAR
StartState BCSMStateType, /* The start state of the BCSM. */
LegID LegType; /* The LegID as assigned by the CS. */
```

```
/***** T-BCSM FOR INAP CS-3. *****/
```

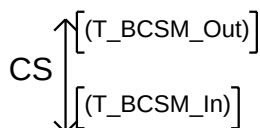
```
/***** DATA TYPE DEFINITIONS *****/
```

```
NEWTYPE DPResultType
LITERALS
Resume,
SelectFacility,
PresentCall,
ARelease,
BRelease,
MidCall,
SCFRelease, /* Used with ReleaseCall. */
Answer,
Alerting,
AuthTermination,
Exception,
Busy,
NoAnswer,
Failure,
Reanswer,

DL_A, DL_B, /* Used with DisconnectLeg */
CS_Stop;
ENDNEWTYPE;

NEWTYPE PICResultType
LITERALS
Failure,
Success,
ARelease,
BRelease,
Alerting,
Busy,
Answer,
NoAnswer,
MidCall,
Exception,
SCFRelease, /* Used with ReleaseCall. */

Reanswer,
Suspended,
NotReachable,
DL_A, DL_B; /* Used with DisconnectLeg */
ENDNEWTYPE;
```



Process Type TerminatingBCSM

2(12)

```

;FPAR
StartState BCSMStateType, /* The start state of the BCSM. */
LegID LegType; /* The LegID as assigned by the CS. */
    
```

**** VARIABLE AND TIMER DECLARATIONS ****

```

TIMER
NoAnswerT := 15000,
SuspendT := 3600000;

DCL
/* Pointer to the call segment. */
CS PId,

/* Address of remote O-BCSM. */
remLegID LegType,
remCSAID CSAID,

calledPartyNumber CalledPartyNumber,

/* Used to indicate the status of the call processing. */
setupSent boolean := false, /* Indicates if SetupReq is sent to the called party */

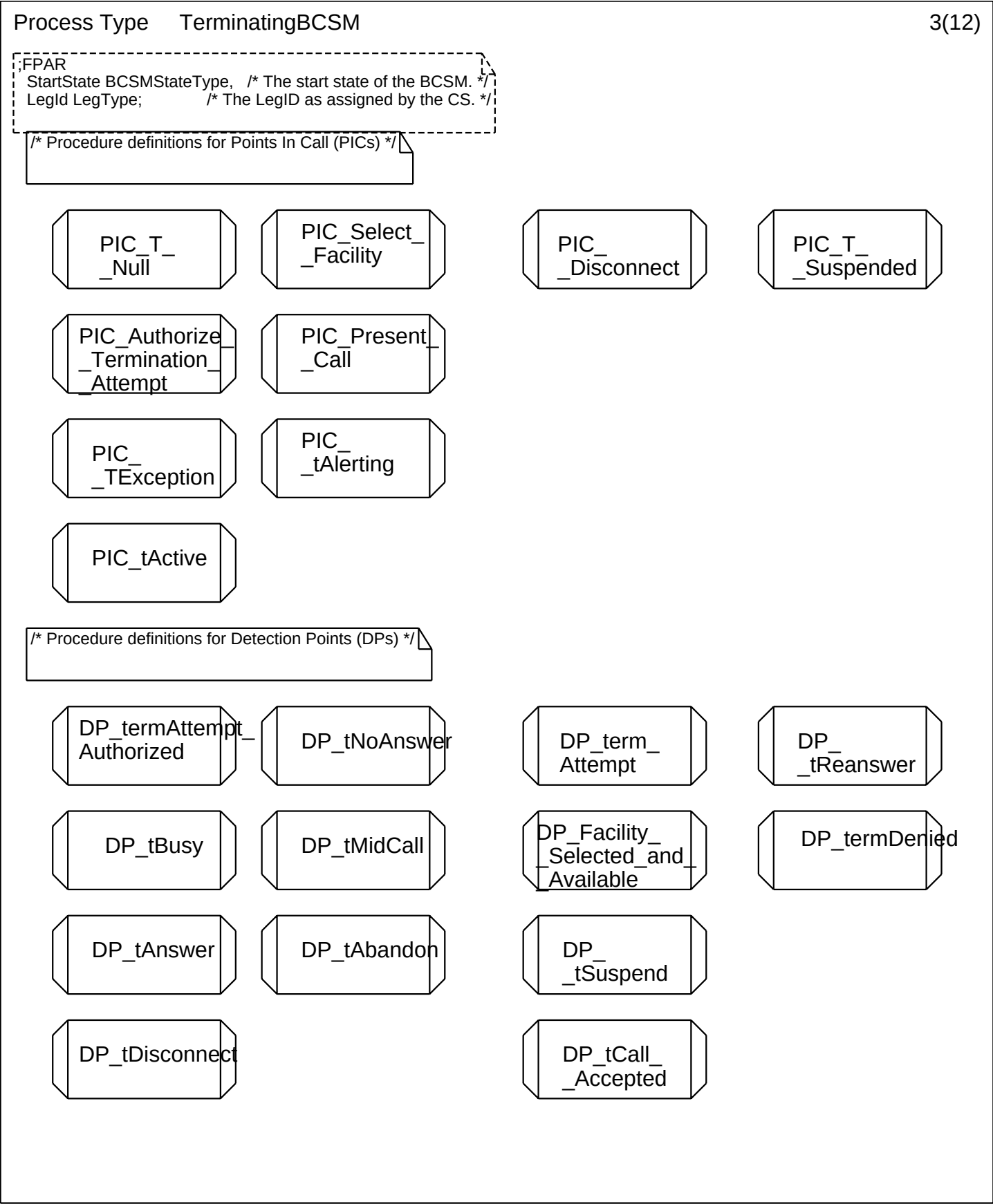
/* Other variables. */
PICResult PICResultType,
DPResult DPResultType,
DPCause Cause := noIndication,
PIC PICArg,
DP DPArg;
    
```

```

DCL
/* SigCon primitive parameters. */
aeArg AddressEndType,
cpArg CallProgressType,
farg FailureType,
rarg ReleaseType,
sftArg ServiceFeatureType,
sirArg SetupIRType,
scrArg SetupCRType,
saArg SubsequentAddressType;
    
```

```

DCL
currentLegID, newLegID LegType,
udArg UserDataTypes,
dpUTSIArg DPUTSIArg,
nsrArg NetworkSRTType;
    
```

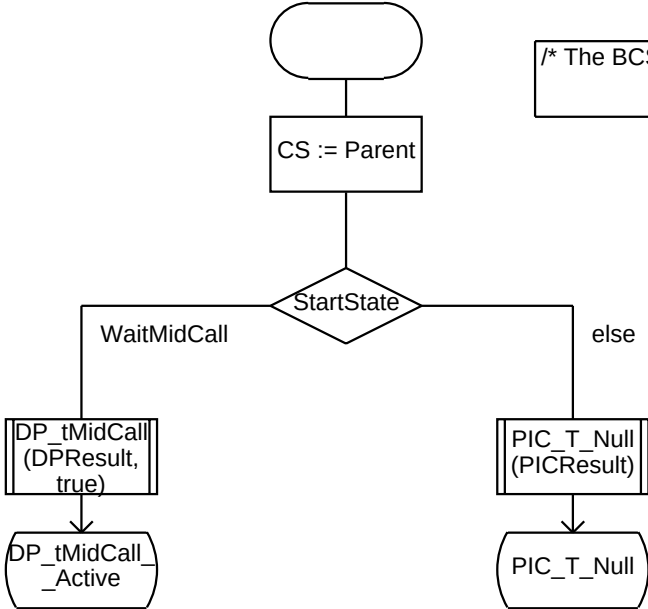


Process TypeTerminatingBCSM4(12)

;FPAR

StartState BCSMStateType, /* The start state of the BCSM. */

LegId LegType; /* The LegID as assigned by the CS. */

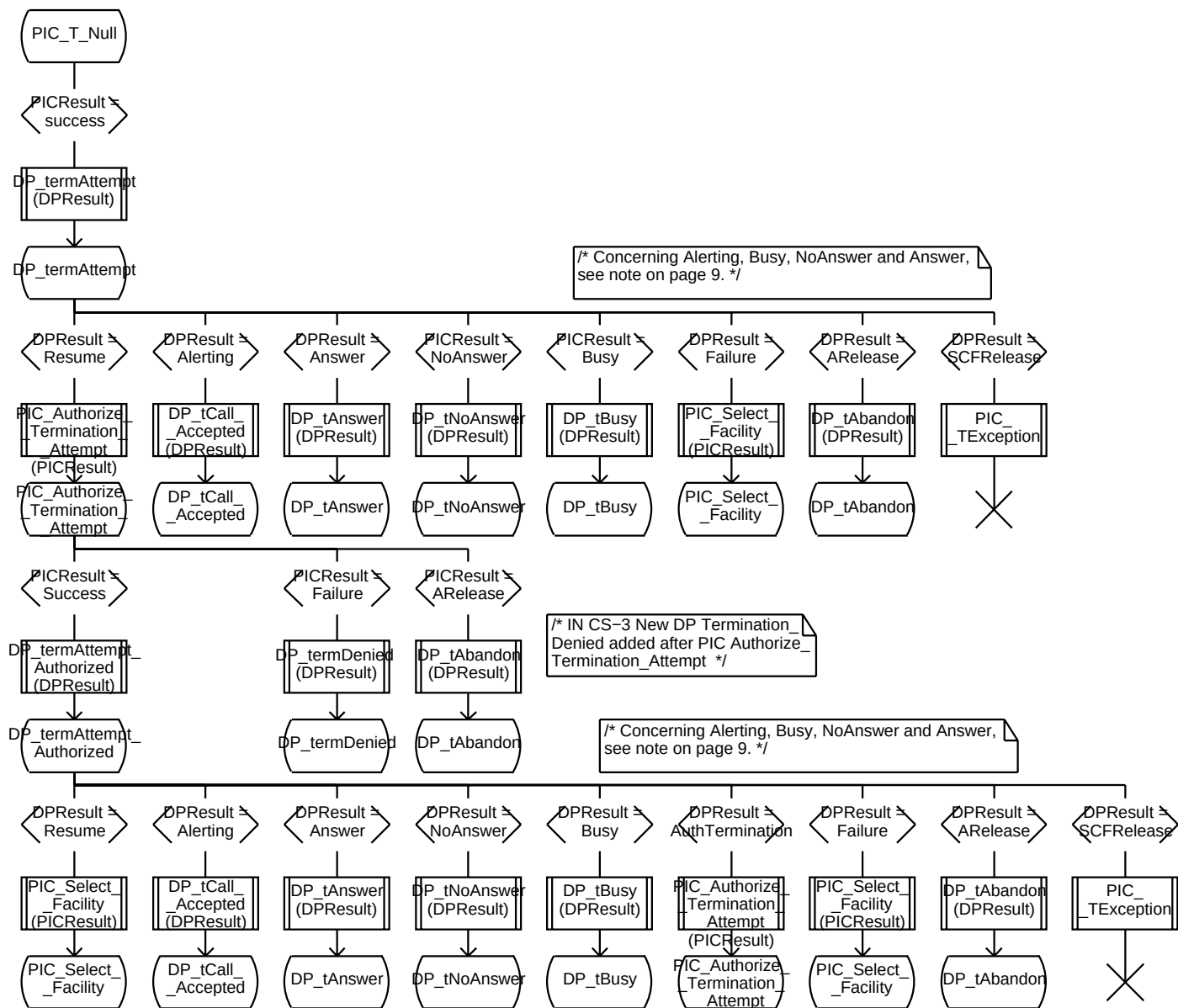


/* The BCSM may be started from different states. */

Process Type TerminatingBCSM

5(12)

FPAR
StartState BCSMStateType, /* The start state of the BCSM. */
LegID LegType; /* The LegID as assigned by the CS. */



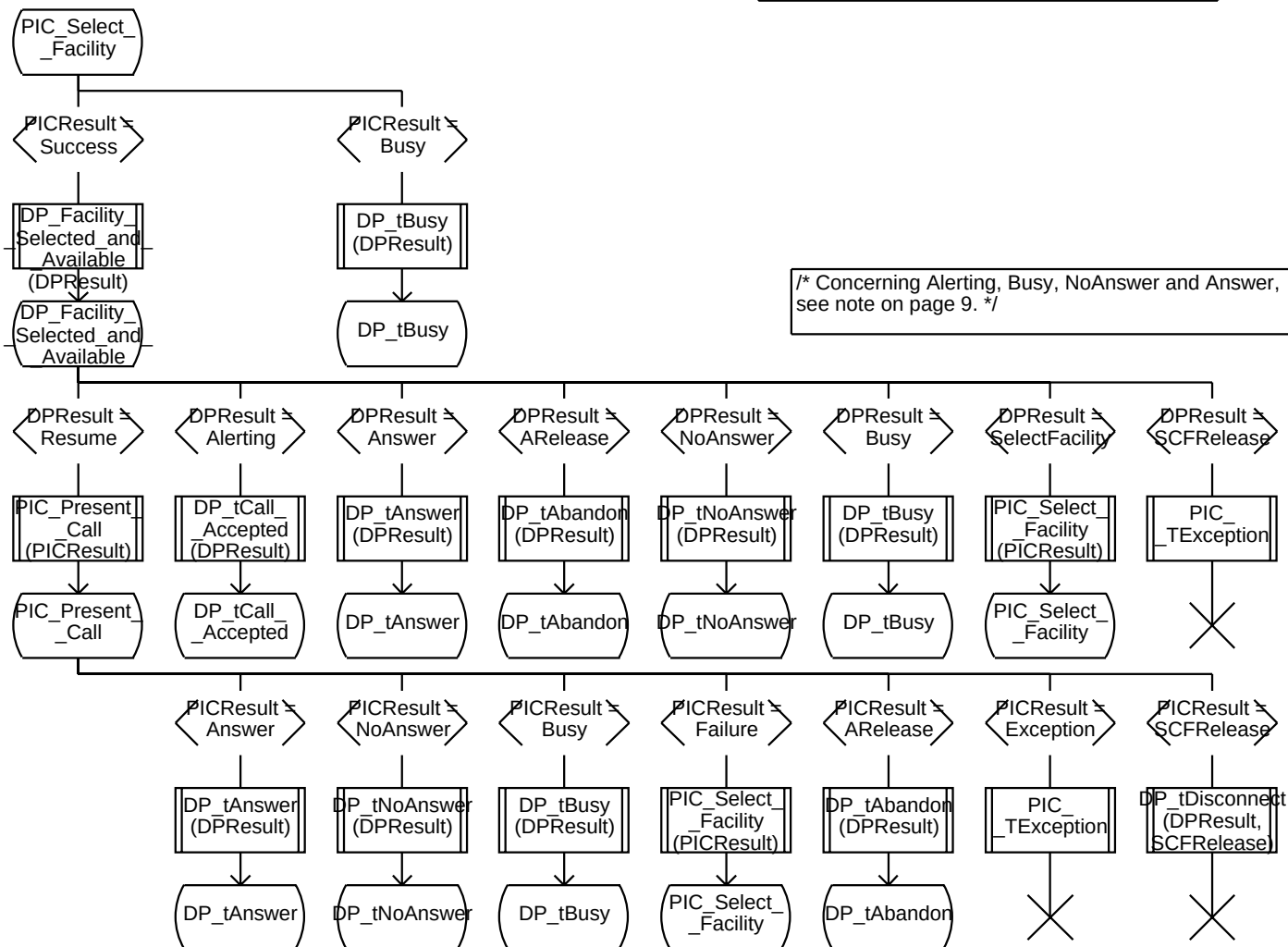
Process Type TerminatingBCSM

6(12)

FPAR
StartState BCSMStateType, /* The start state of the BCSM. */
LegID LegType; /* The LegID as assigned by the CS. */

/* The SCF may apply the DisconnectLeg operation. In case of DL on the controlling leg (DL_B) the BCSM must transit to the wait state of the MidCall DP. In case of DL on the passive leg (DL_A) the BCSM is released. */

/* Concerning Alerting, Busy, NoAnswer and Answer, see note on page 9. */

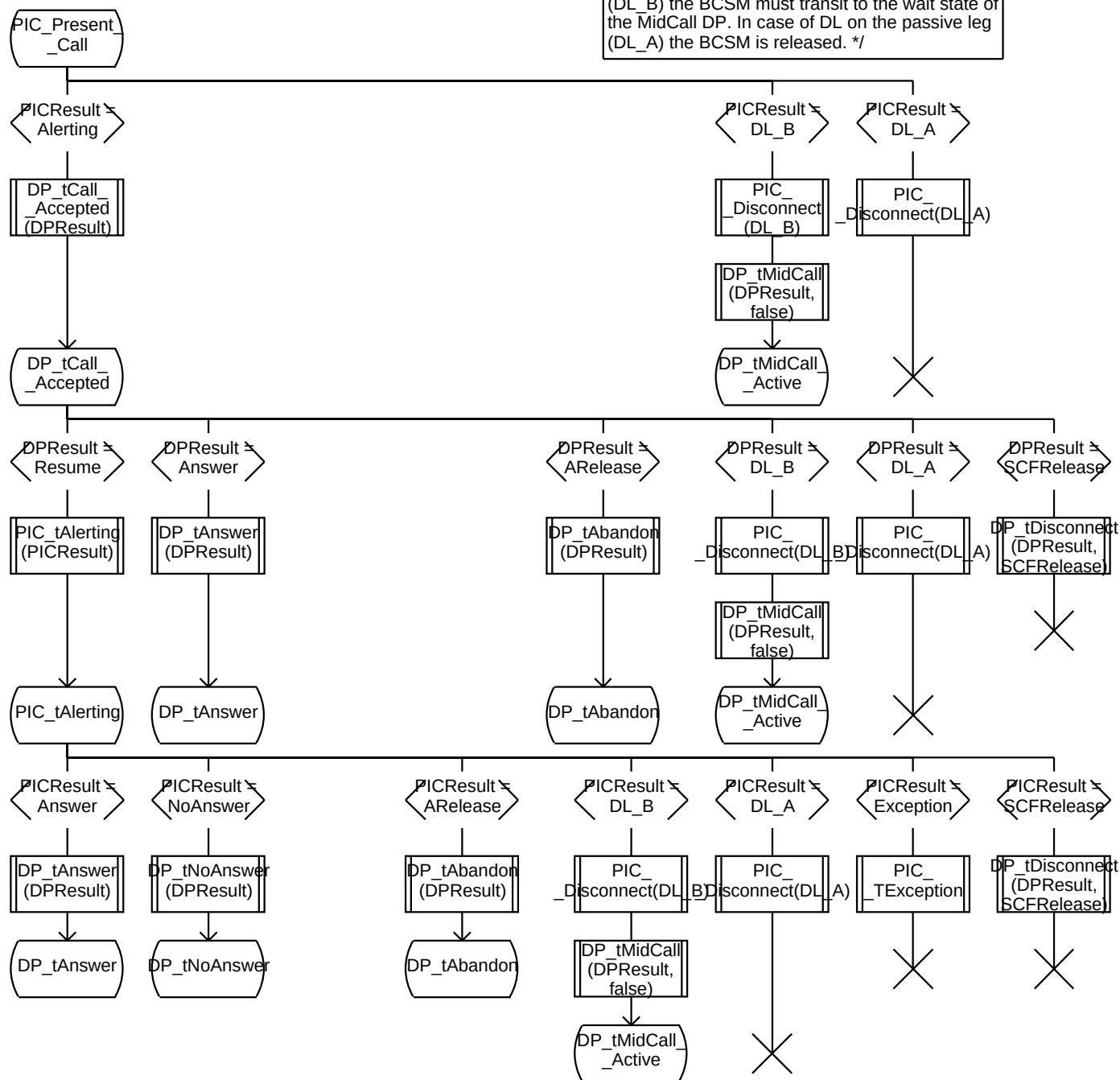


Process Type TerminatingBCSM

7(12)

FPAR
StartState BCSMStateType, /* The start state of the BCSM. */
LegID LegType; /* The LegID as assigned by the CS. */

/* The SCF may apply the DisconnectLeg operation. In case of DL on the controlling leg (DL_B) the BCSM must transit to the wait state of the MidCall DP. In case of DL on the passive leg (DL_A) the BCSM is released. */



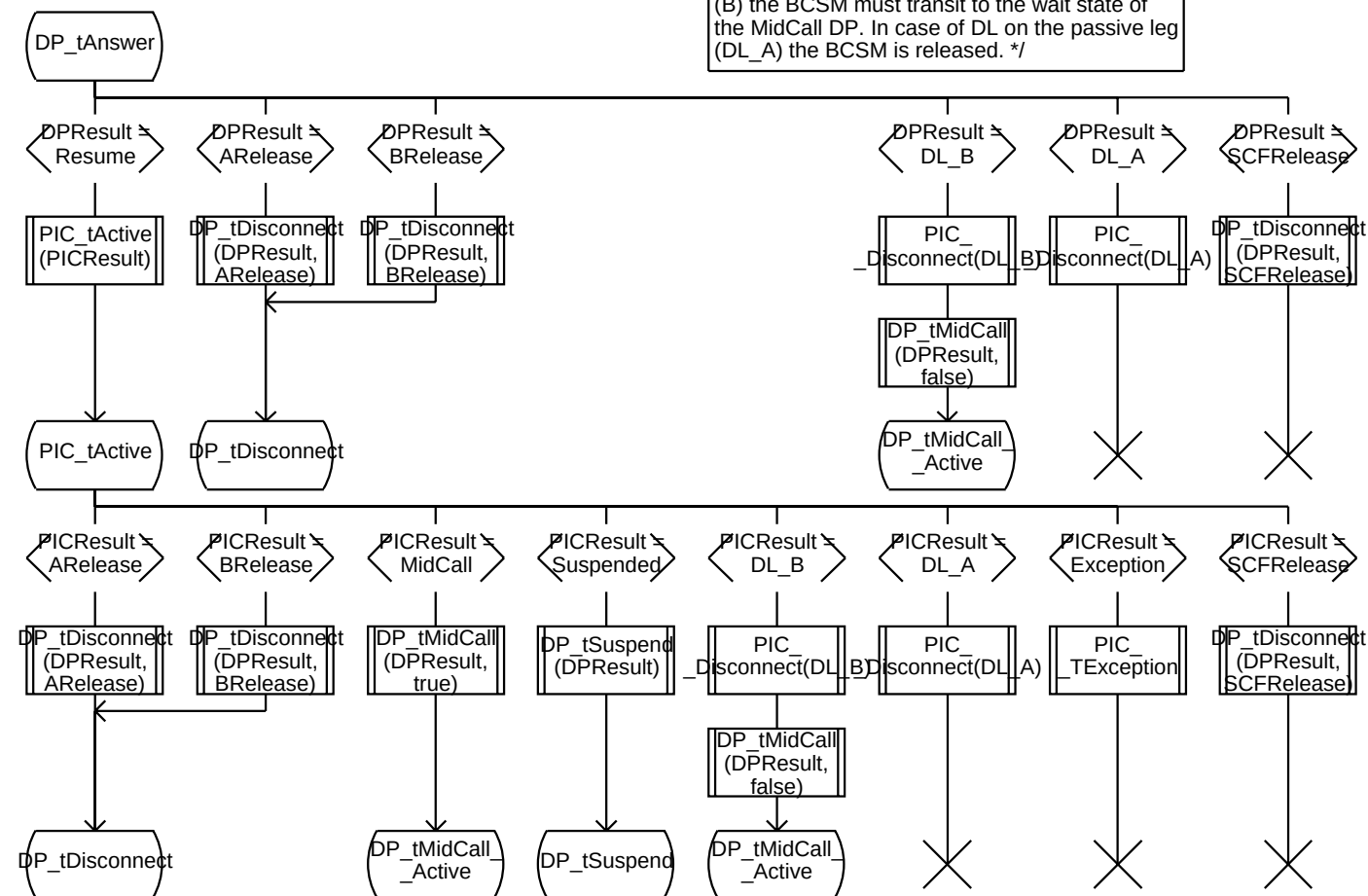
Process Type TerminatingBCSM

8(12)

```

;FPAR
StartState BCSMStateType, /* The start state of the BCSM. */
LegID LegType; /* The LegID as assigned by the CS. */
    
```

/* The SCF may apply the DisconnectLeg operation. In case of DL on the controlling leg (B) the BCSM must transit to the wait state of the MidCall DP. In case of DL on the passive leg (DL_A) the BCSM is released. */



Process Type TerminatingBCSM

9(12)

```

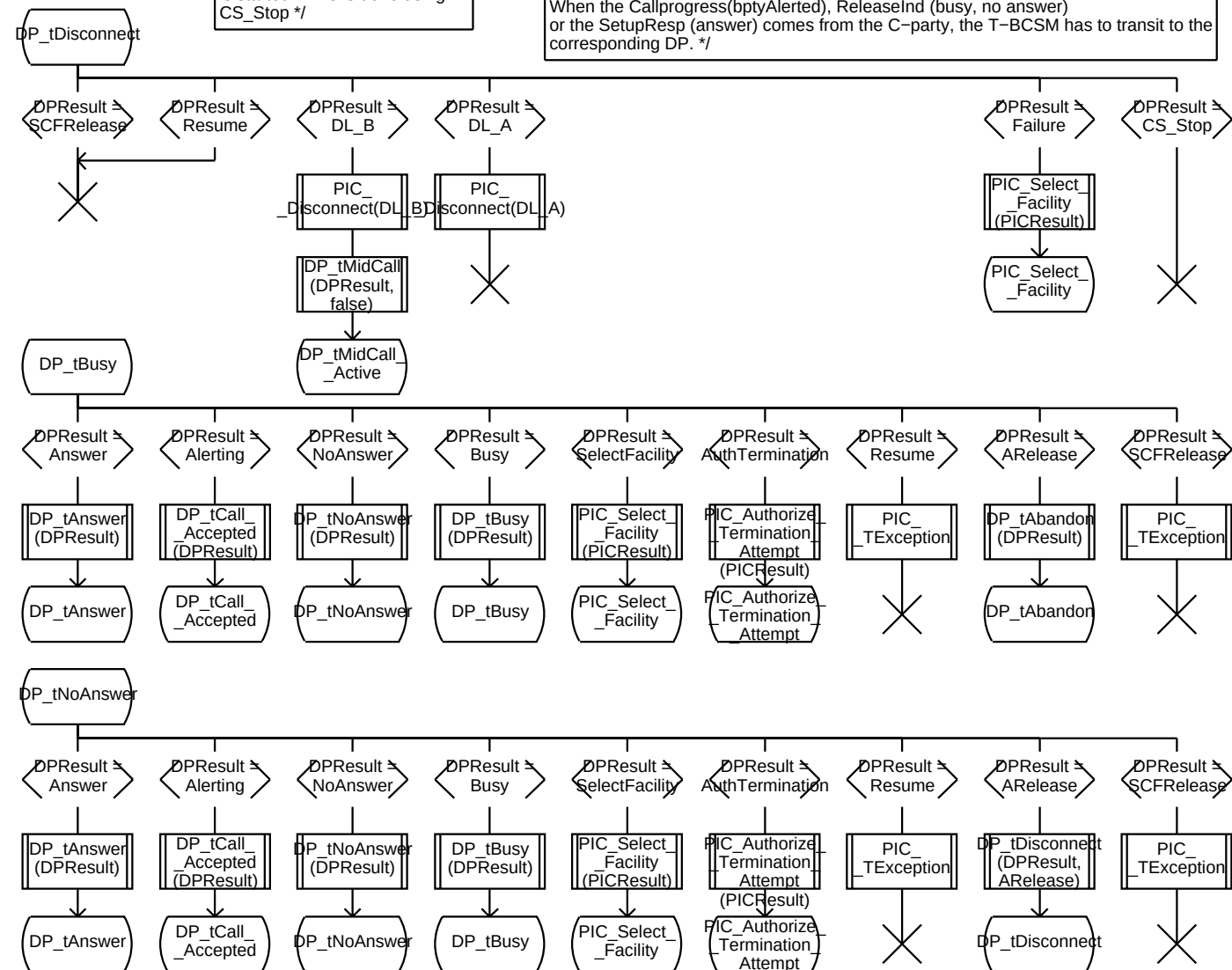
;FPAR
StartState BCSMStateType; /* The start state of the BCSM. */
LegID LegType; /* The LegID as assigned by the CS. */

```

/* The SCF may perform a Connect in the CS state Terminating-Setup to forward the call. When in CS state Terminating-Setup, the T-BCSM is suspended at one of the following DPs: termAttempt, termAttemptAuthorized, facilitySelectedAndAvailable, tNoAnswer, tBusy

The T_BCSM should transit to PIC_Present_Call (instructed by SSF_FSM), but in this model the SSF_FSM does not know the Connect operation is for call forwarding in Terminating Setup, so the SSF_FSM doesn't instruct the BCSM in this case. The behaviour of PIC_Present_Call on call forwarding is duplicated at the wait states of the above DP procedures. When the Callprogress(bptyAlerted), ReleaseInd (busy, no answer) or the SetupResp (answer) comes from the C-party, the T-BCSM has to transit to the corresponding DP. */

/* When Connect Operation is applied in Forward, BCSM is terminated and new O_BCSM is started. This is done using CS_Stop */

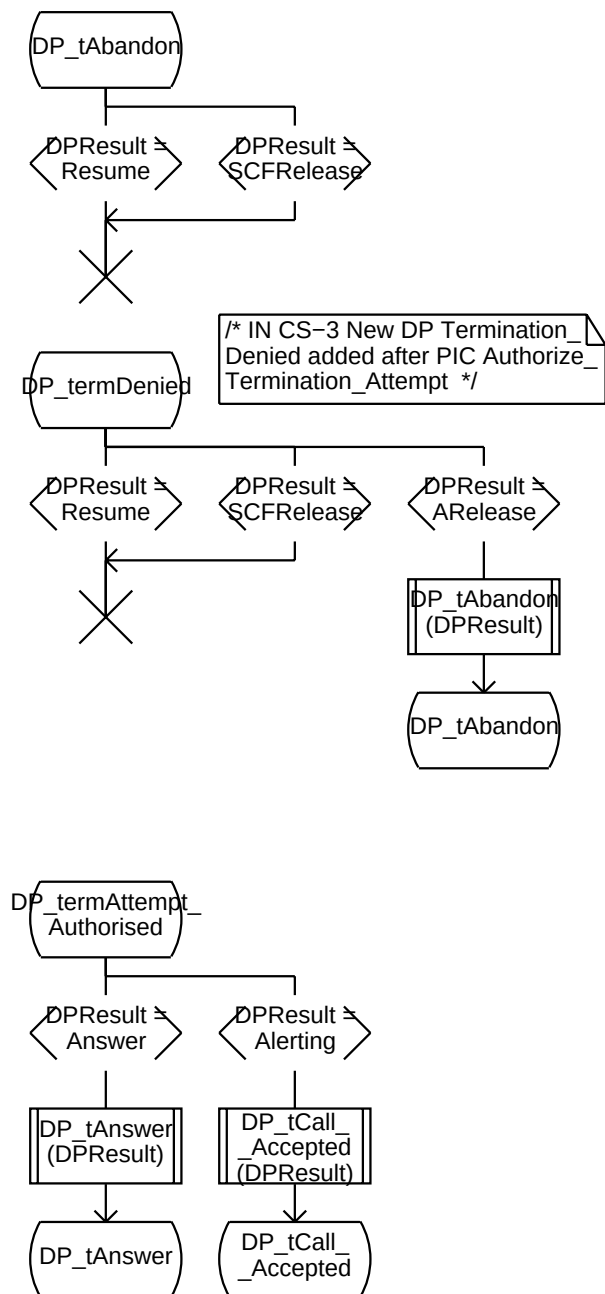


Process Type TerminatingBCSM

10(12)

```

;FPAR
StartState BCSMStateType, /* The start state of the BCSM. */
LegID LegType; /* The LegID as assigned by the CS. */
    
```

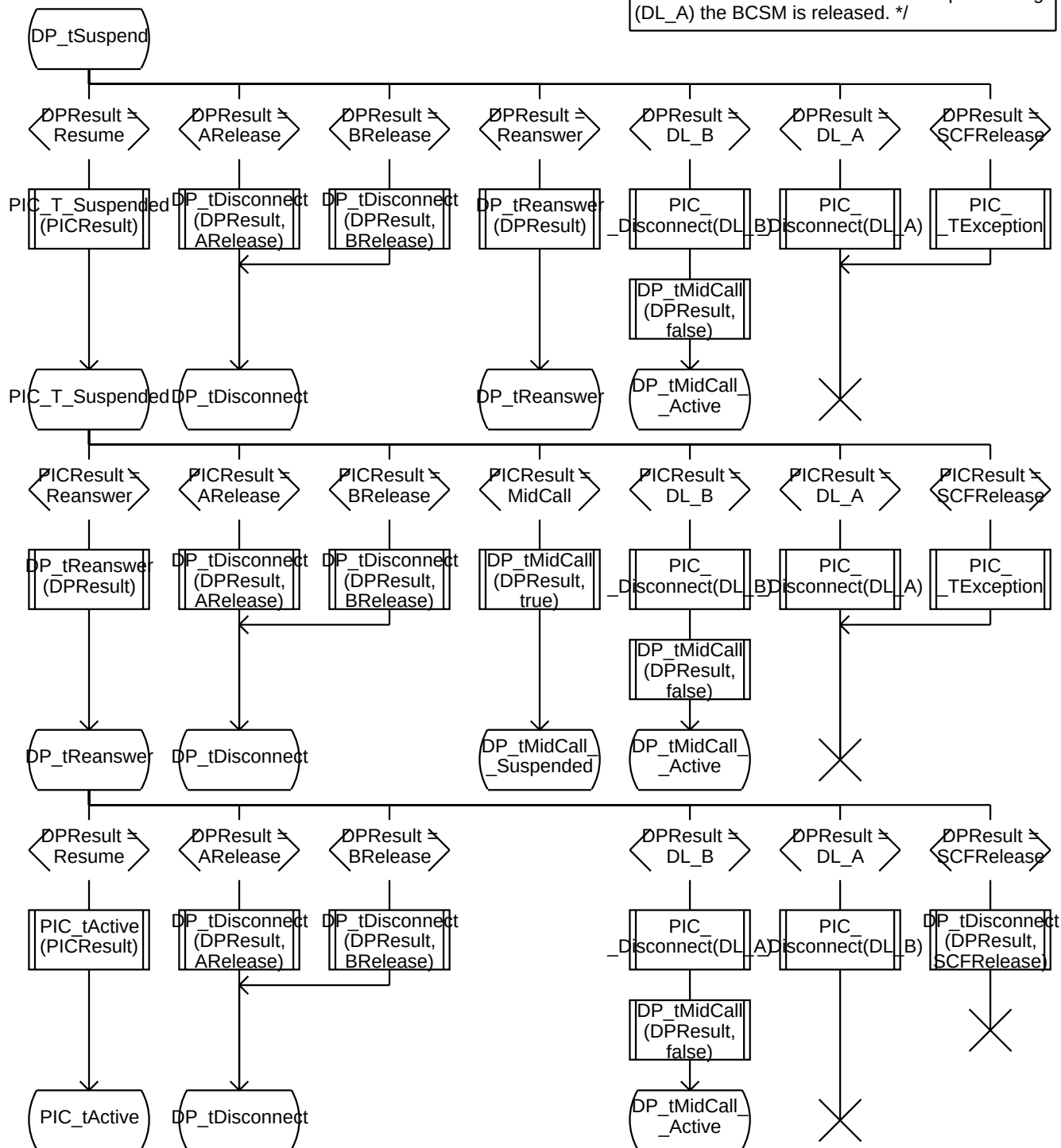


Process Type TerminatingBCSM

11(12)

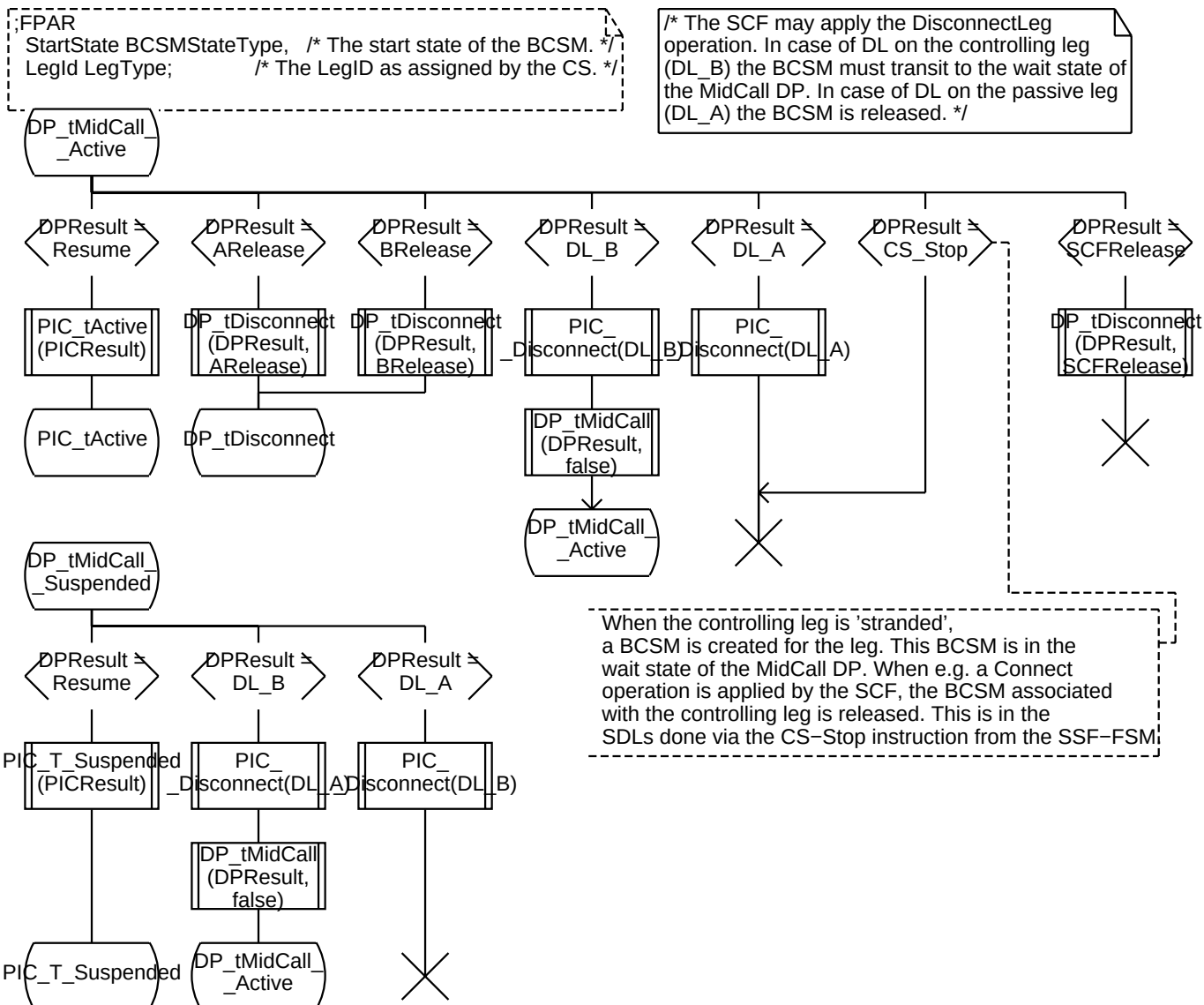
FPAR
StartState BCSMStateType, /* The start state of the BCSM. */
LegID LegType; /* The LegID as assigned by the CS. */

/* The SCF may apply the DisconnectLeg operation. In case of DL on the controlling leg (DL_B) the BCSM must transit to the wait state of the MidCall DP. In case of DL on the passive leg (DL_A) the BCSM is released. */



Process Type TerminatingBCSM

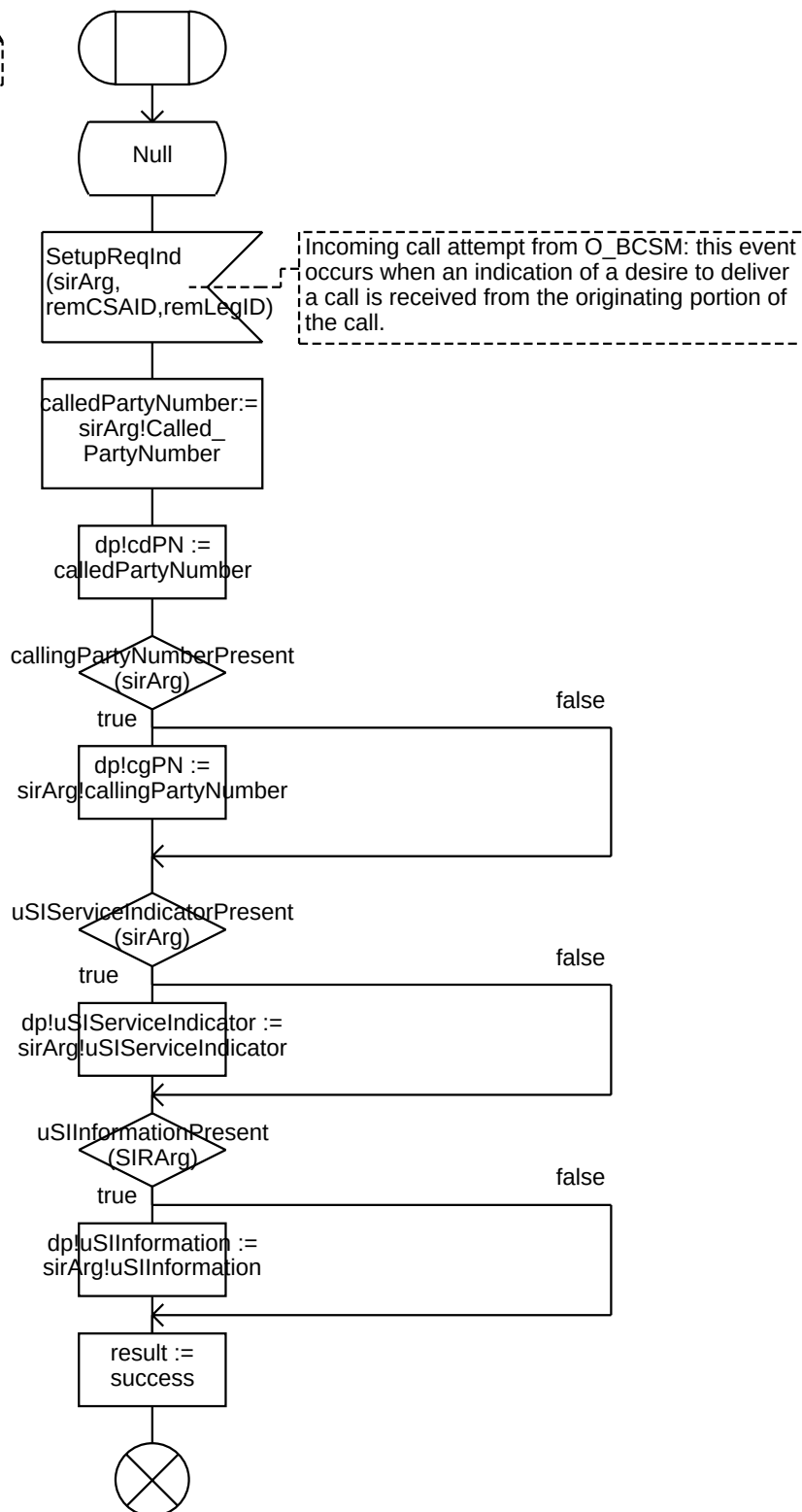
12(12)



Procedure PIC_T_Null

1(1)

FPAR
IN/OUT Result PICResultType;



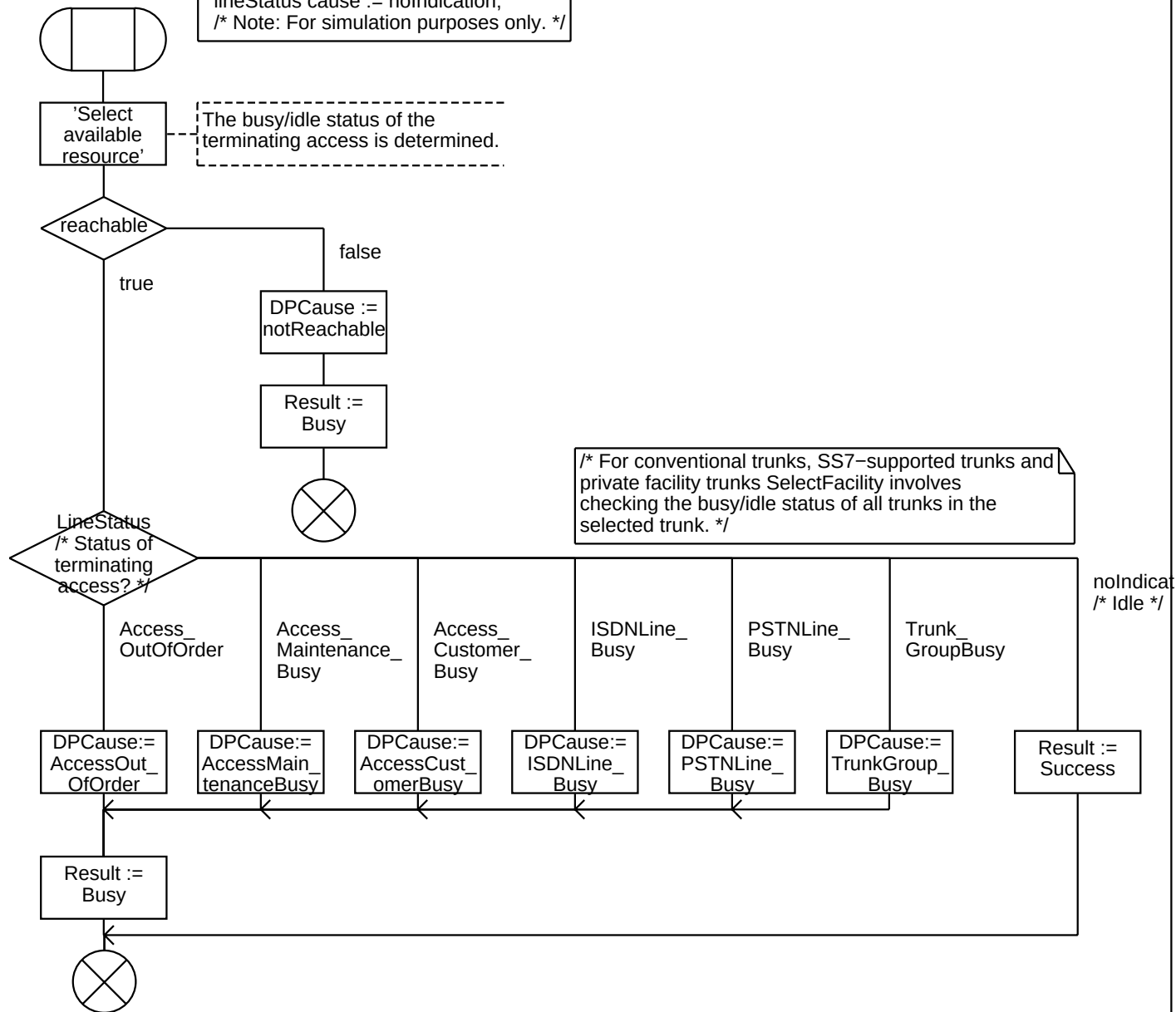
Procedure PIC_Select_Facility

1(1)

```

;FPAR
IN/OUT Result PICResultType;
    
```

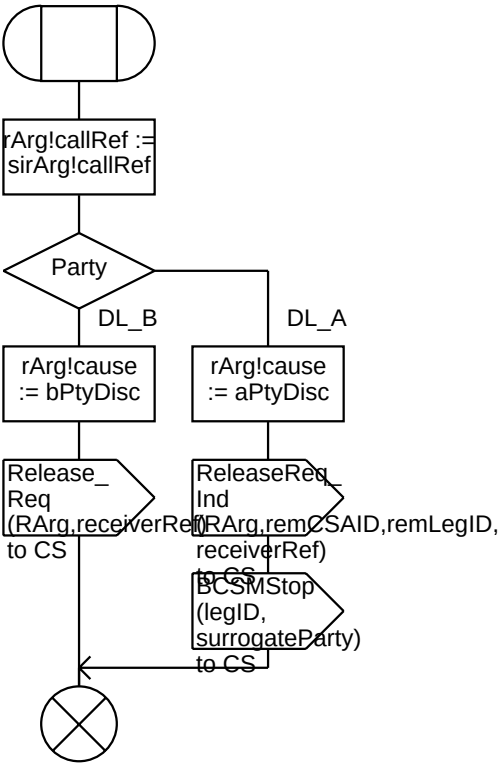
DCL
reachable Boolean := true,
lineStatus cause := noIndication;
/* Note: For simulation purposes only. */



Procedure <<Process Type TerminatingBCSM>> PIC_Disconnect

1(1)

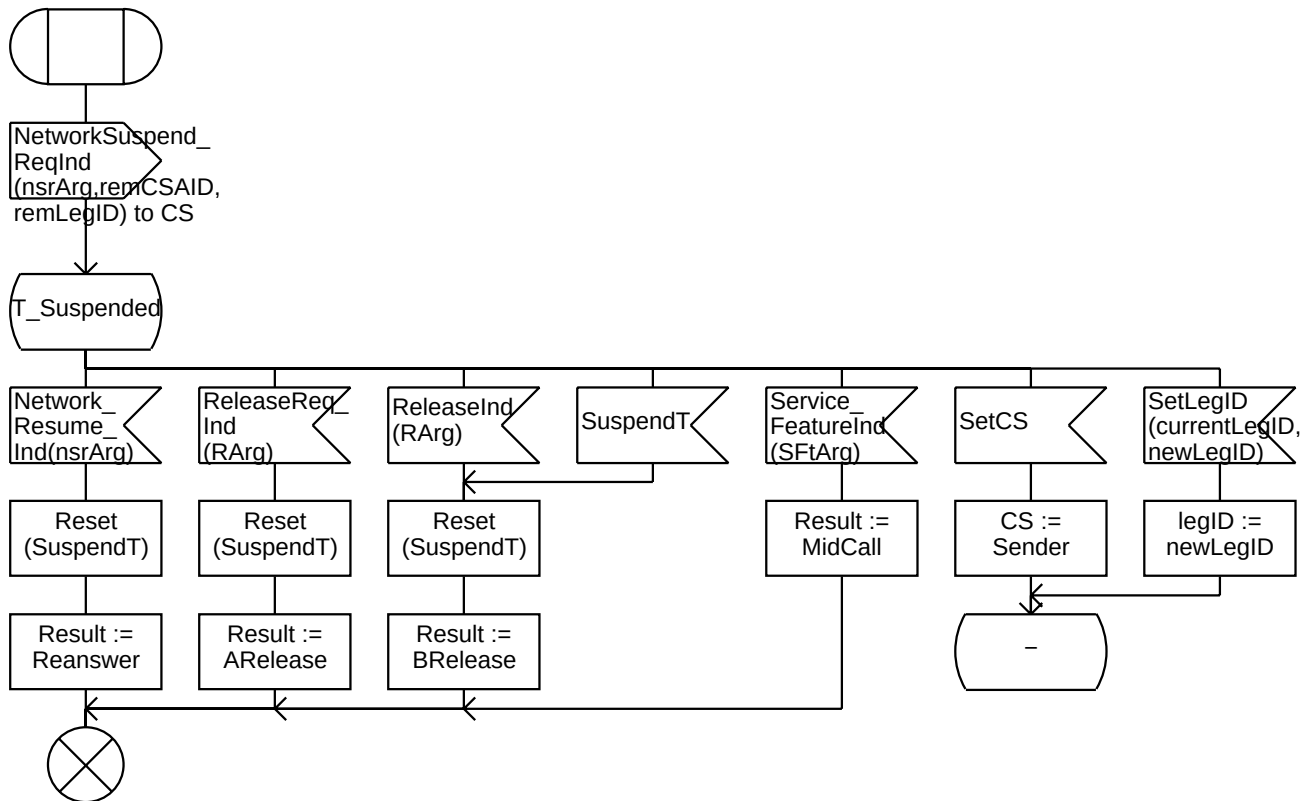
FPAR
IN Party DResultType;



Procedure PIC_T_Suspended

1(3)

FPAR
 IN/OUT Result PICResultType;

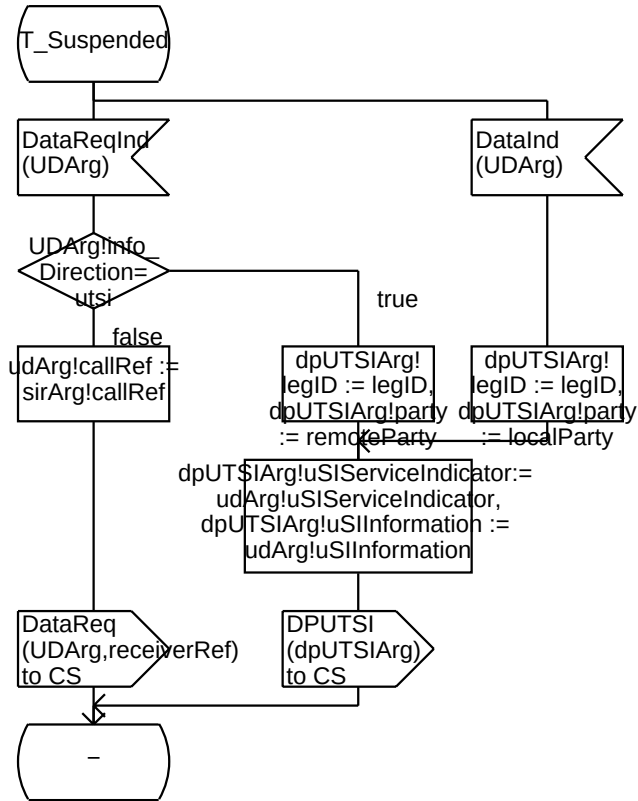


Procedure PIC_T_Suspended

2(3)

```

;FPAR
IN/OUT Result PICResultType;
    
```

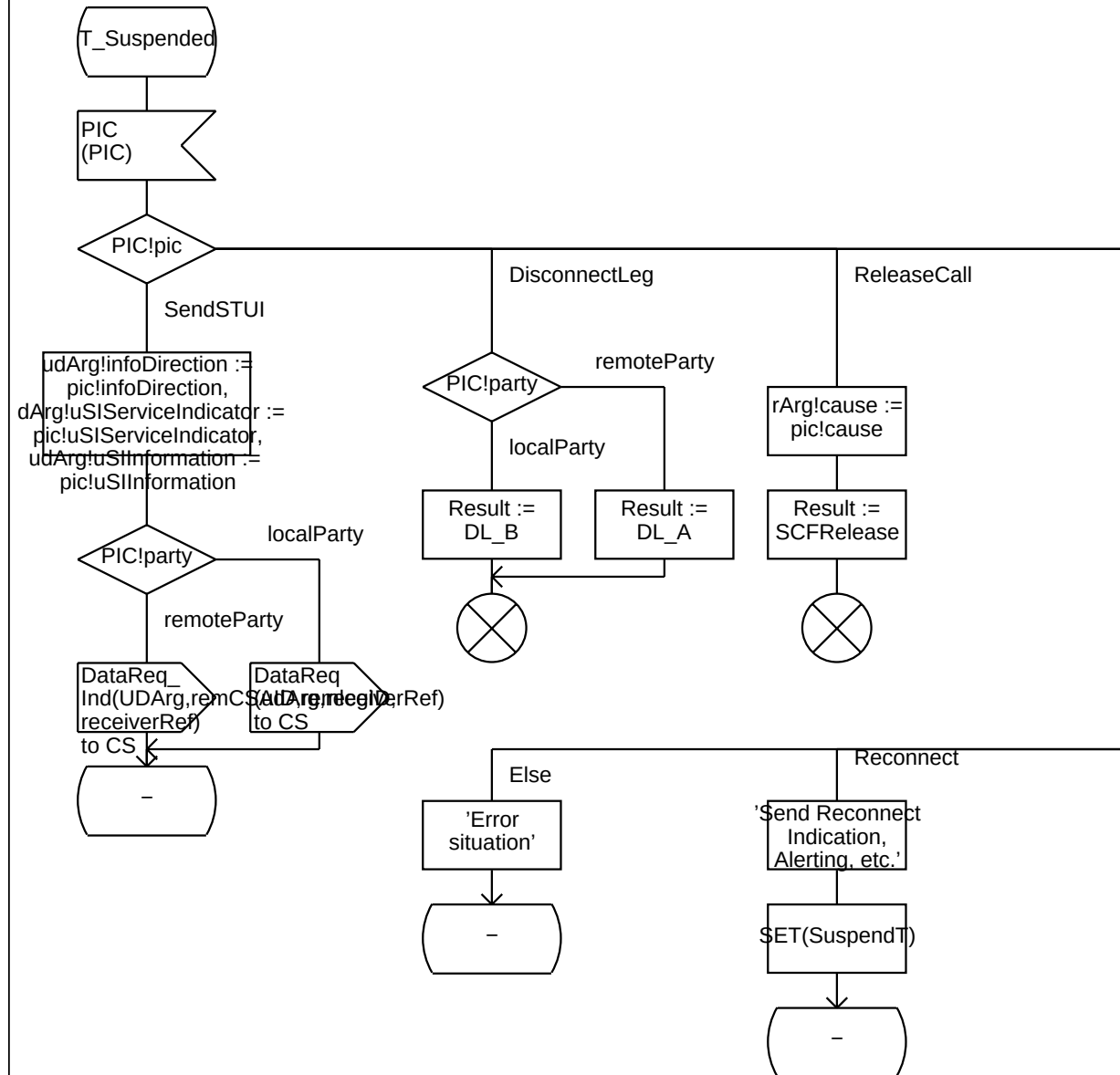


Procedure PIC_T_Suspended

3(3)

```

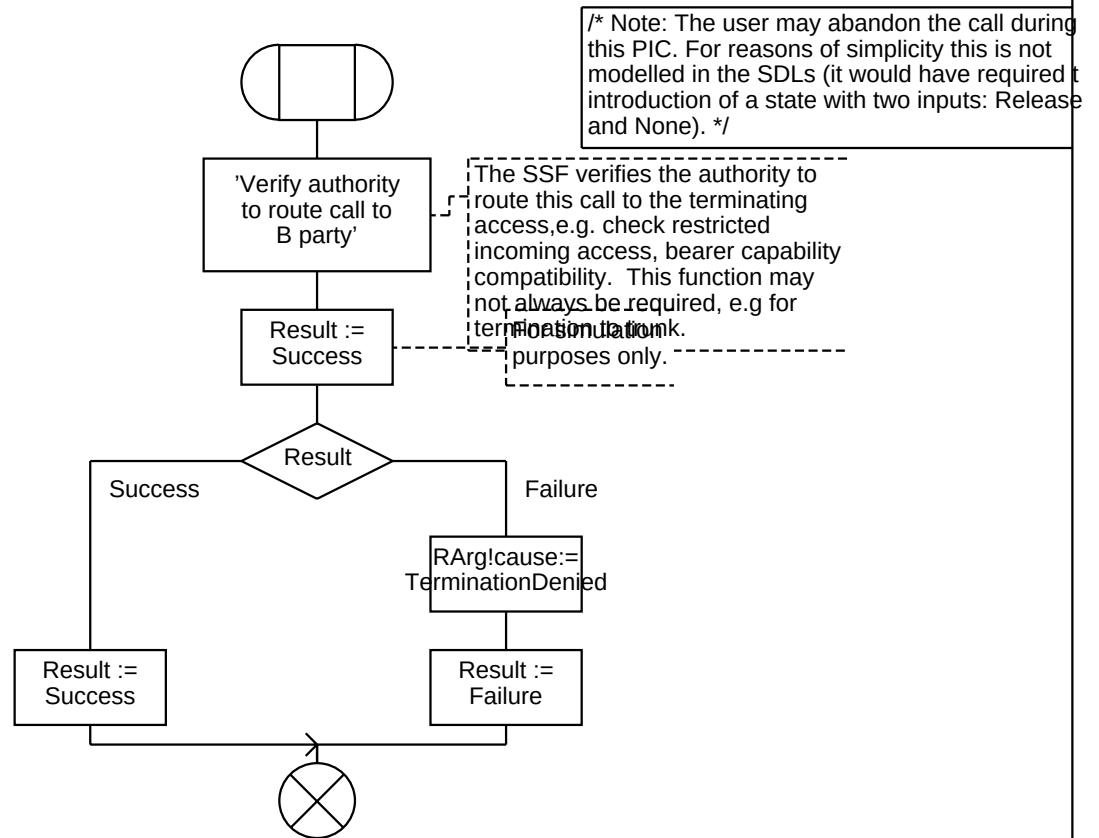
;FPAR
IN/OUT Result PICResultType;
    
```



Procedure PIC_Authorize_Termination_Attempt

1(1)

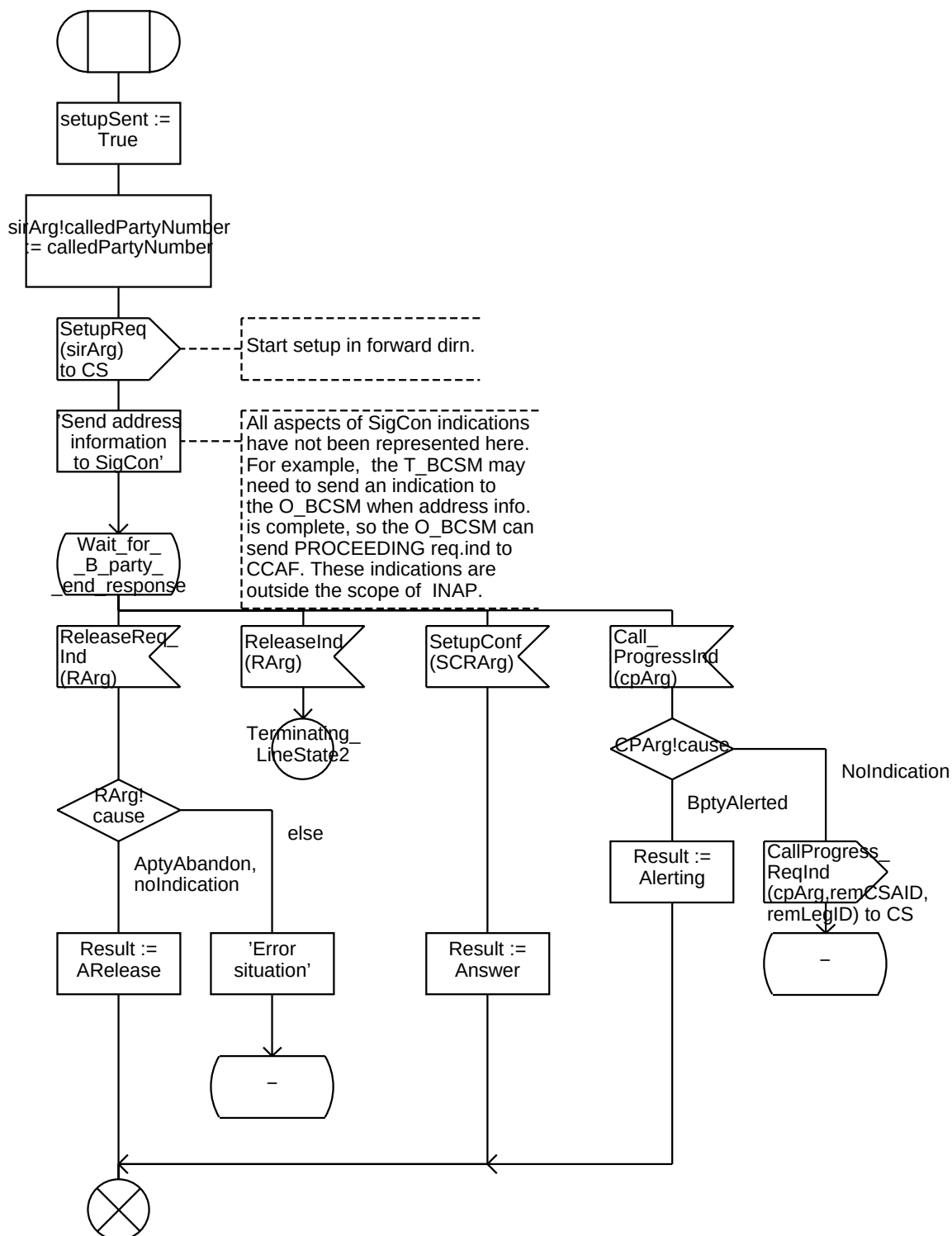
FPAR
IN/OUT Result PICResultType;



Procedure PIC_Present_Call

1(3)

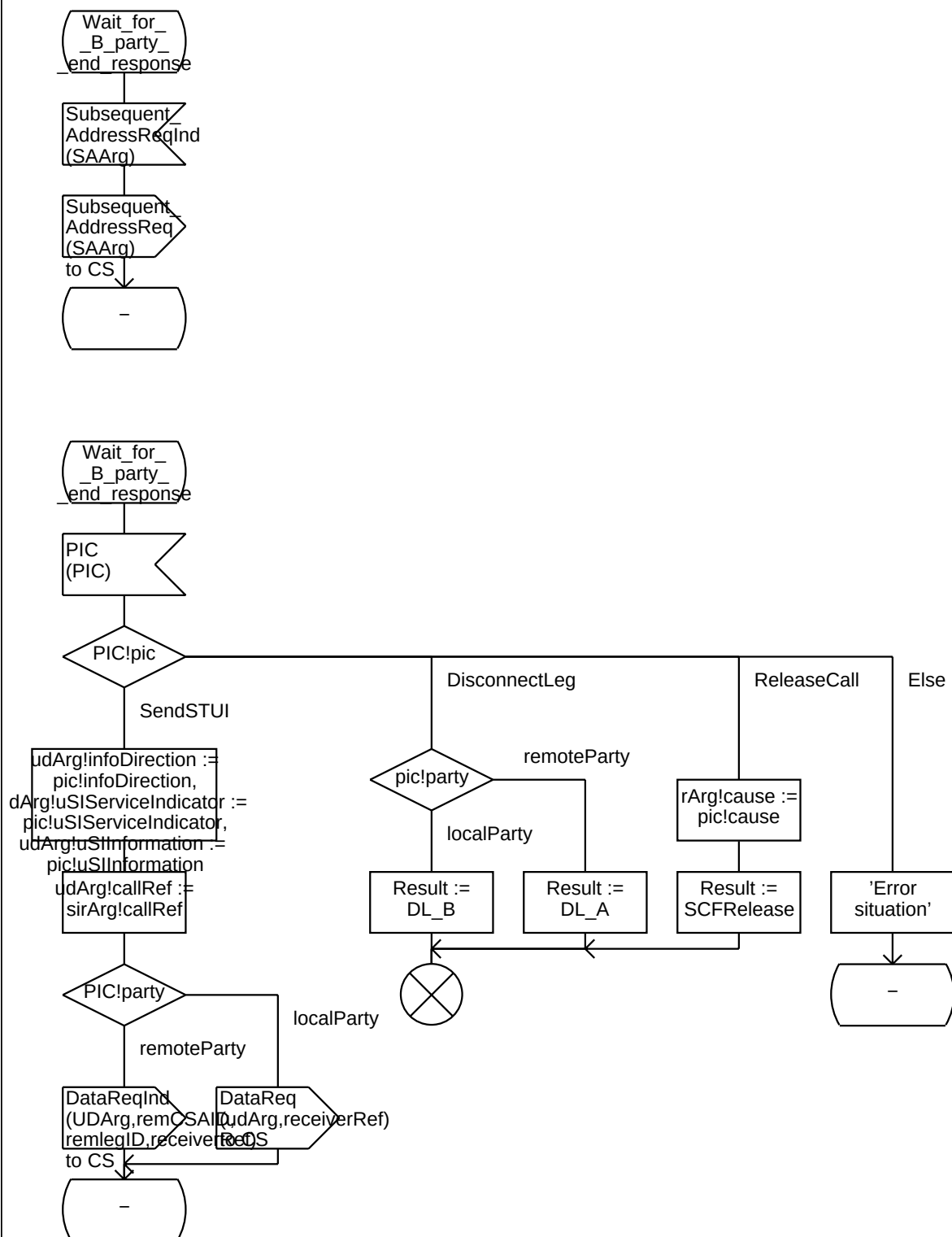
FPAR
IN/OUT Result PICResultType;



Procedure PIC_Present_Call

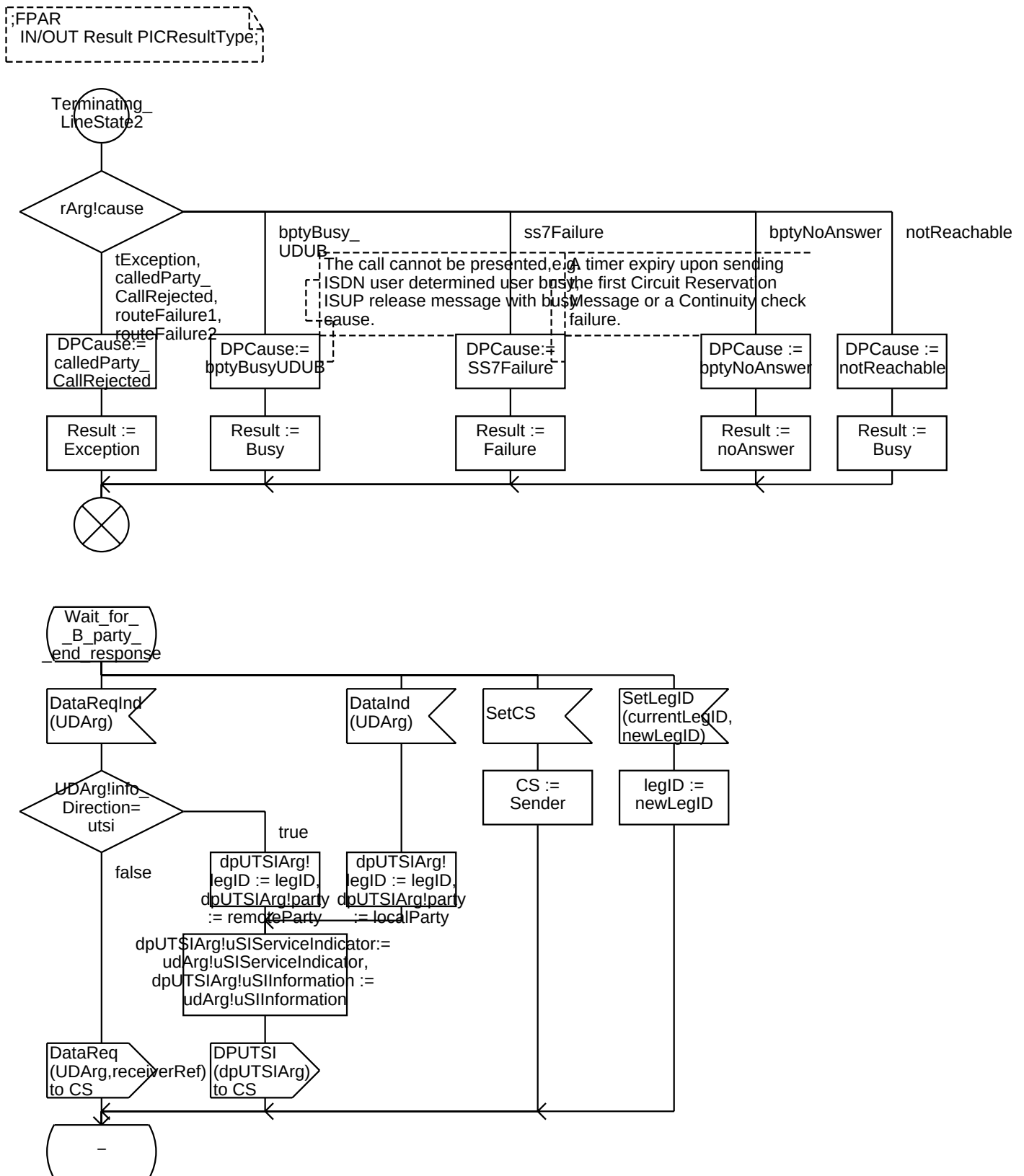
2(3)

FPAR
IN/OUT Result PICResultType;



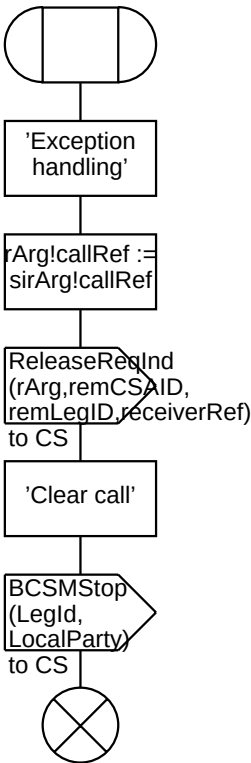
Procedure PIC_Present_Call

3(3)



Procedure PIC_TException

1(1)



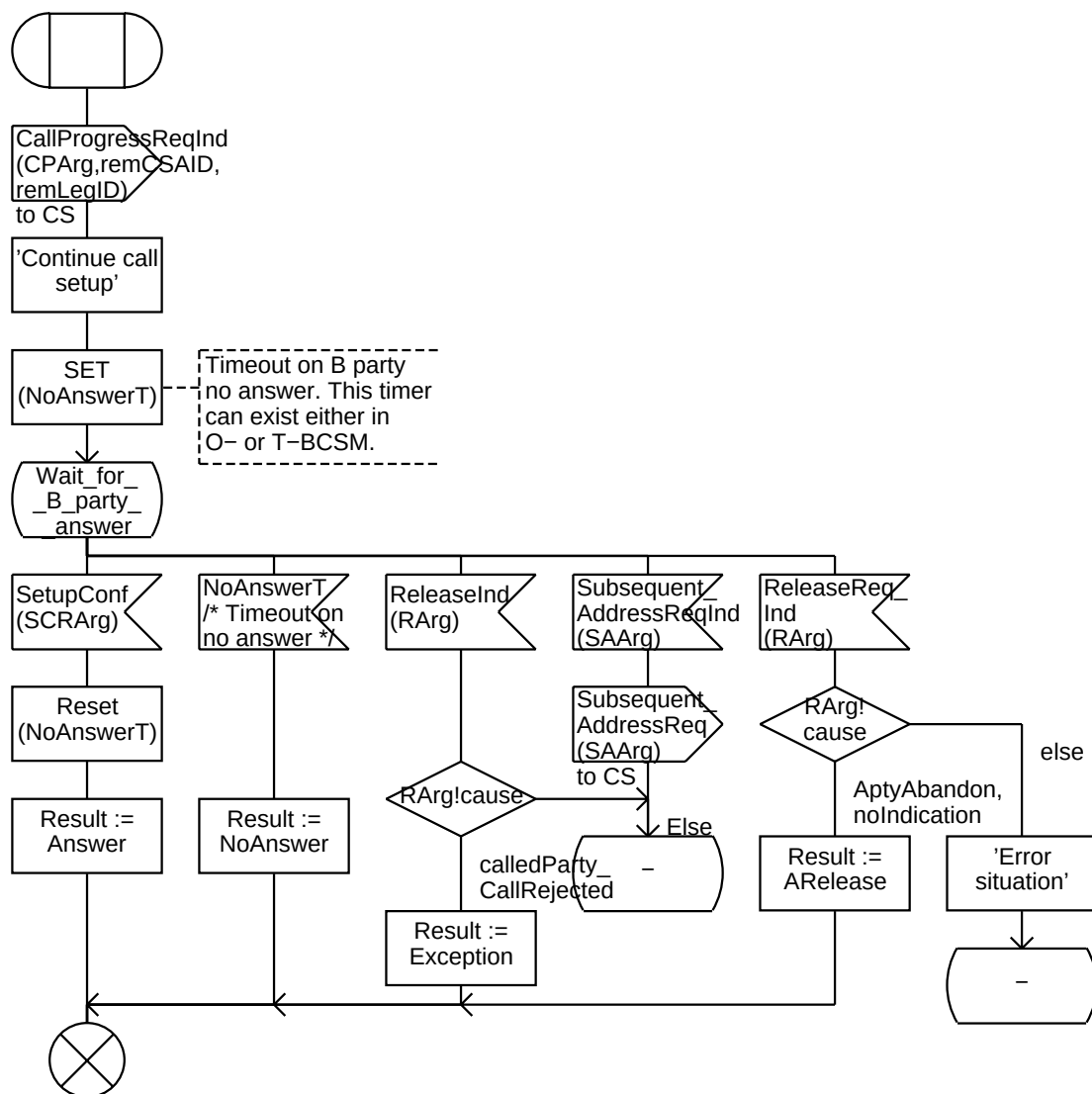
Procedure PIC_tAlerting

1(3)

```

;FPAR
IN/OUT Result PICResultType;

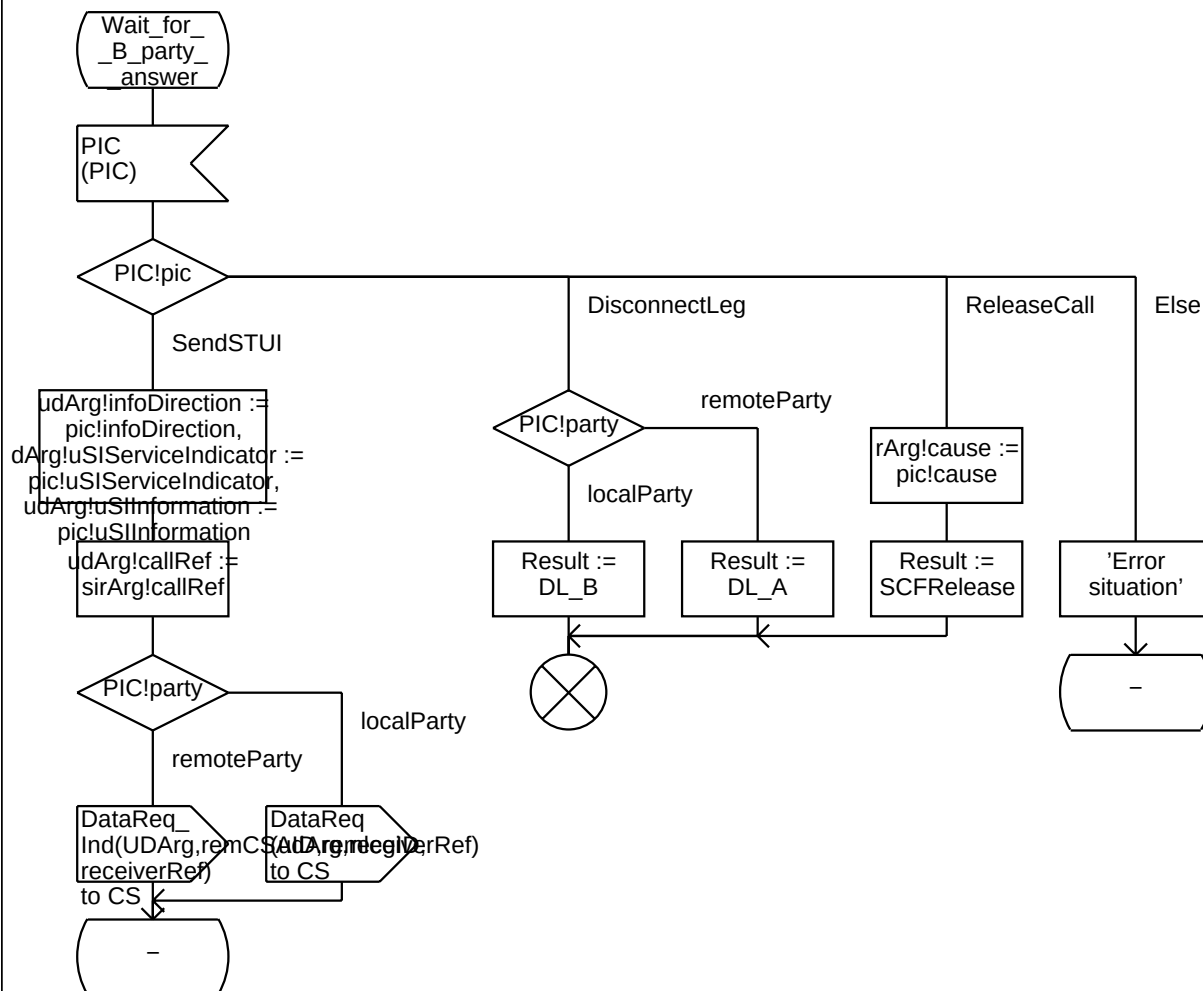
```



Procedure PIC_tAlerting

2(3)

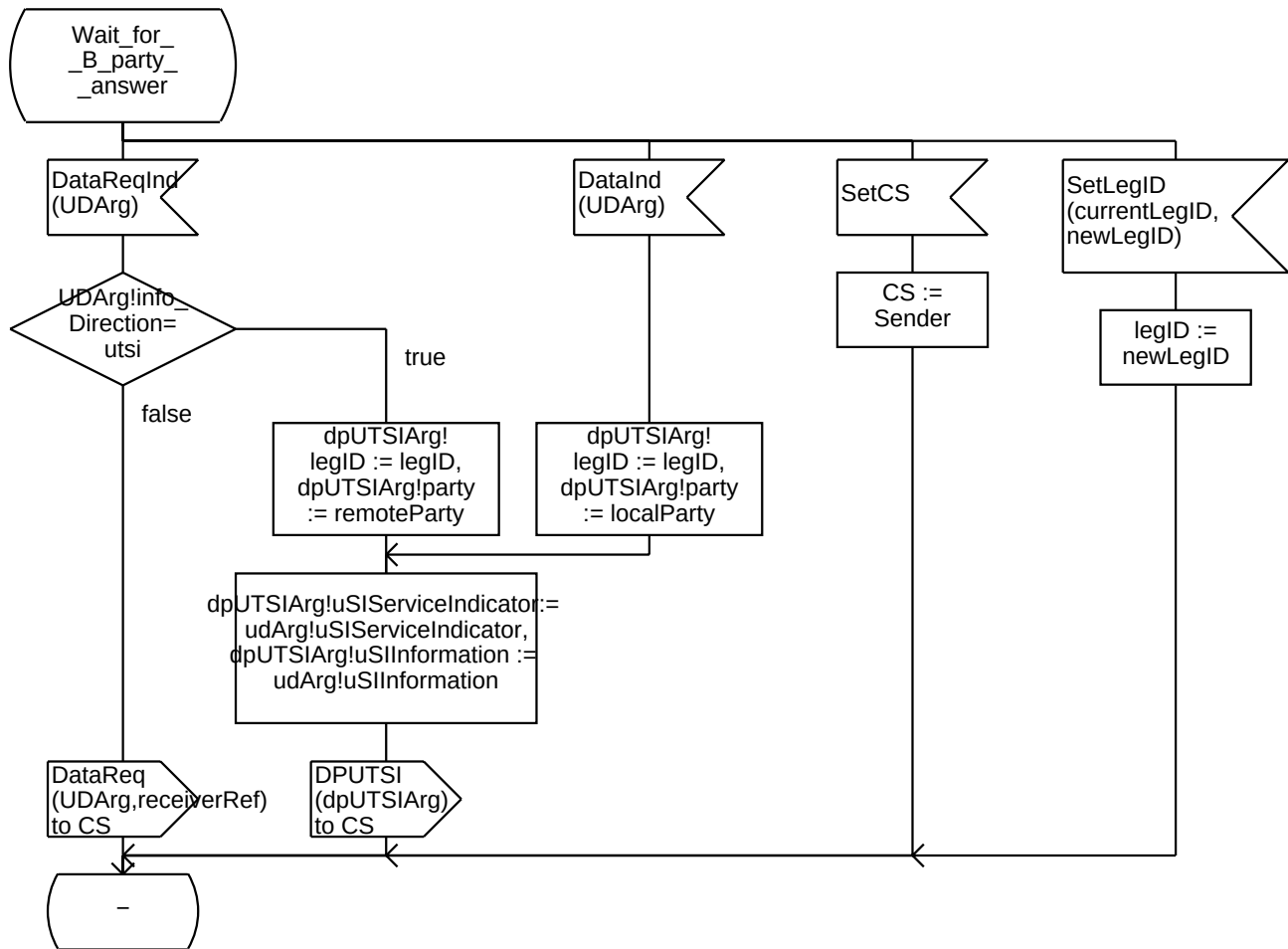
FPAR
IN/OUT Result PICResultType;



Procedure PIC_tAlerting

3(3)

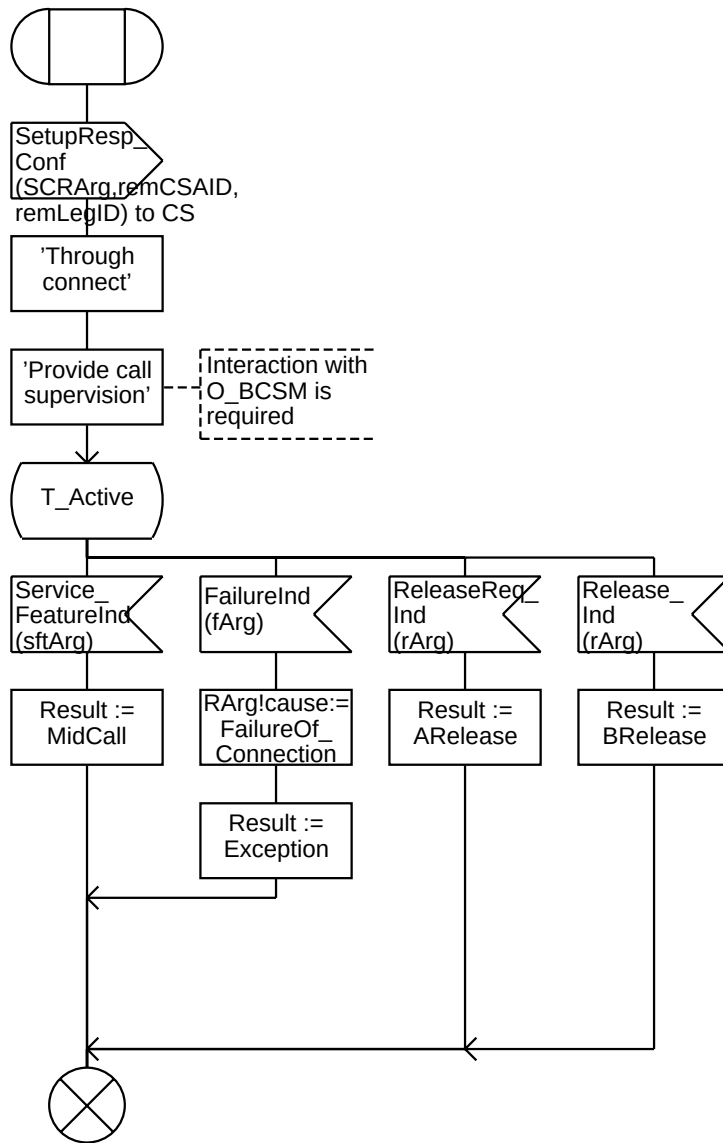
FPAR
IN/OUT Result PICResultType;



Procedure PIC_tActive

1(3)

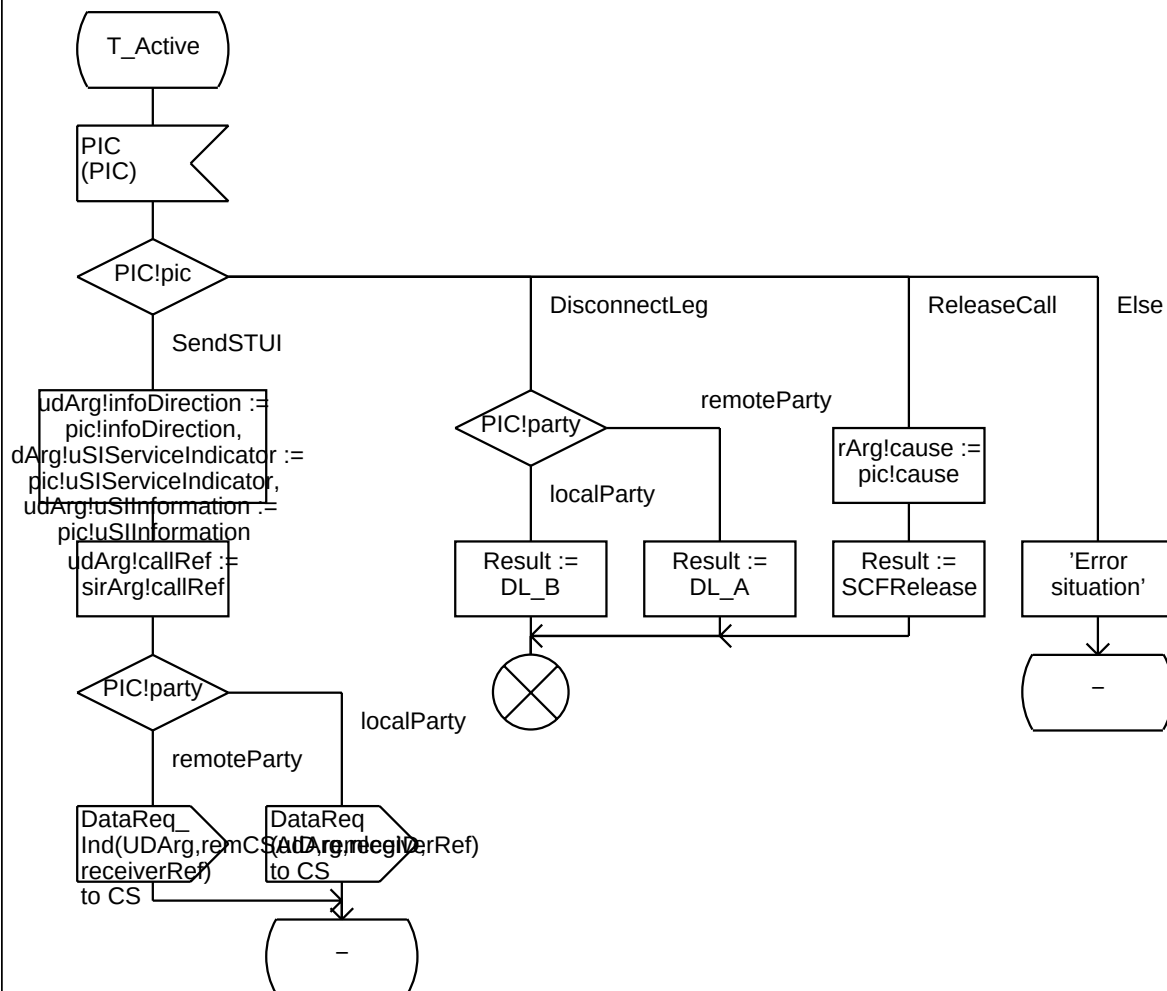
FPAR
IN/OUT Result PICResultType;



Procedure PIC_tActive

2(3)

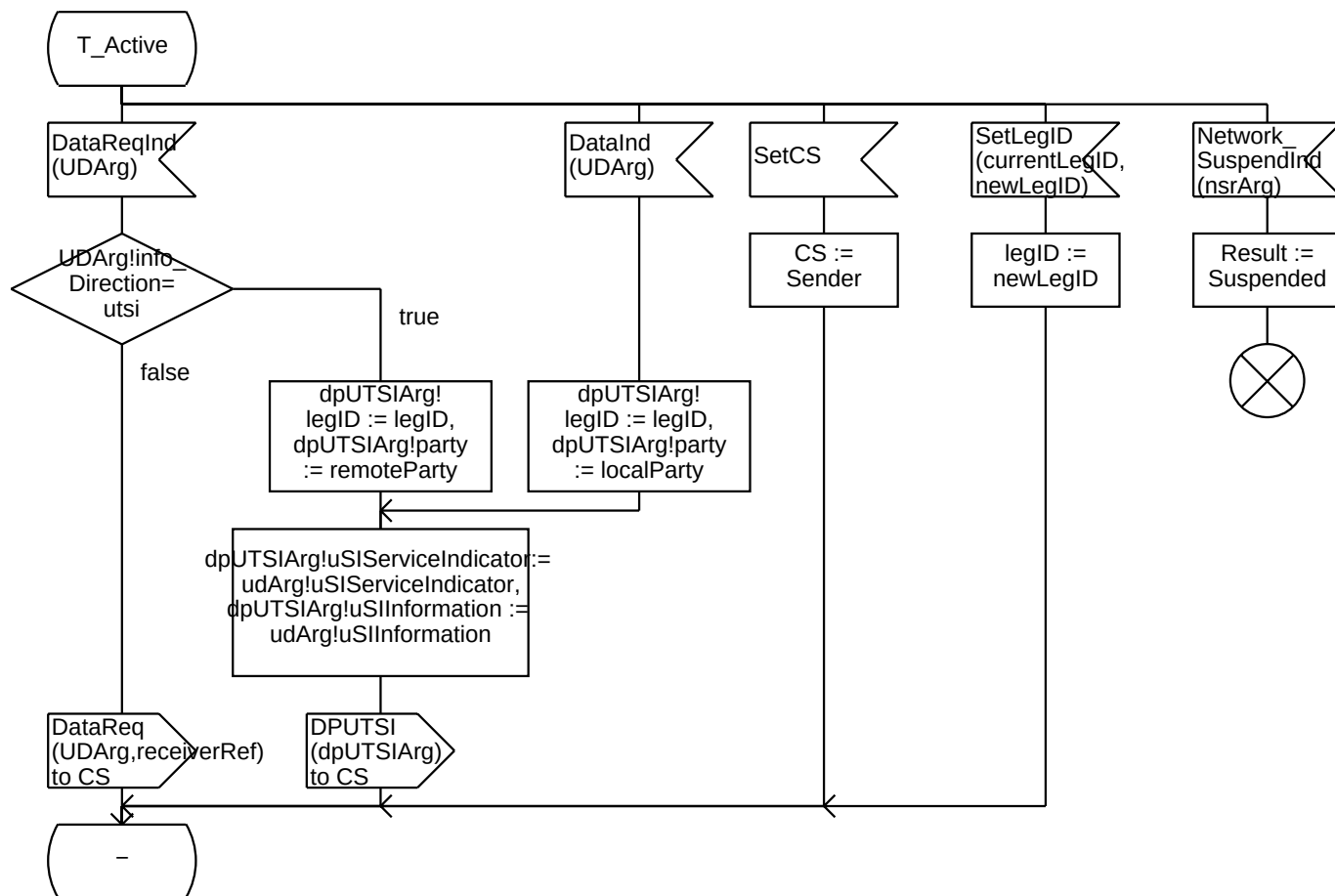
FPAR
IN/OUT Result PICResultType;



Procedure PIC_tActive

3(3)

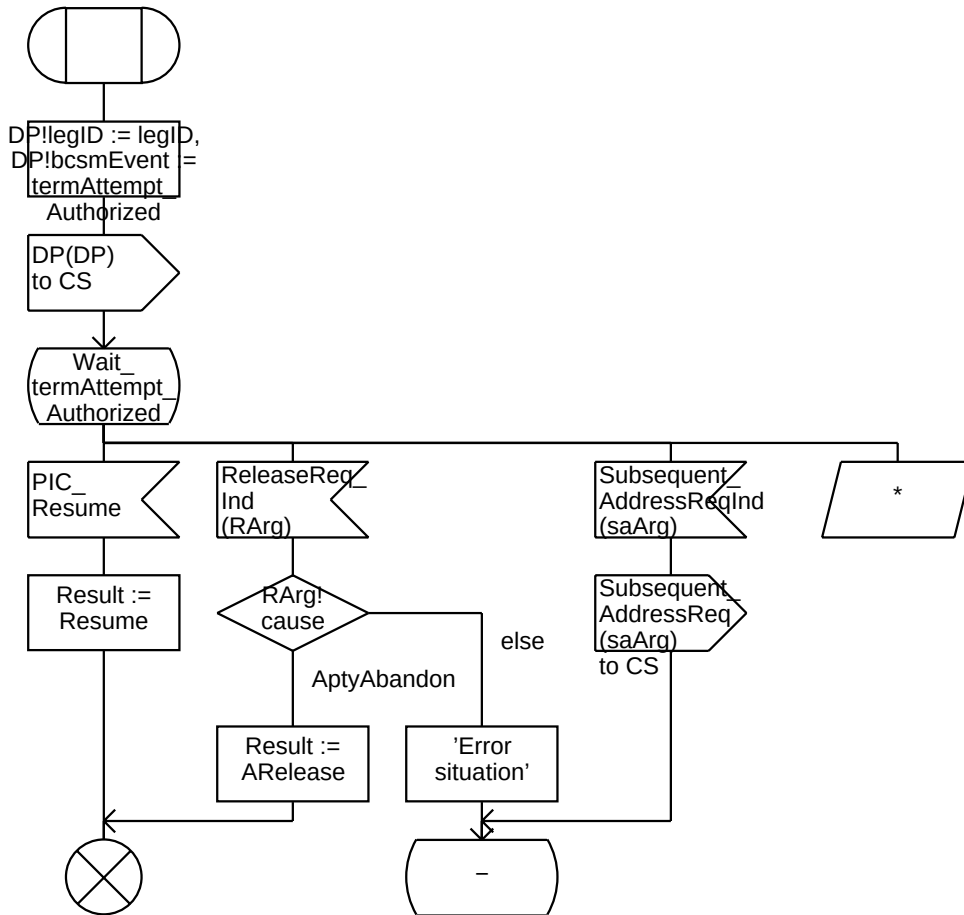
FPAR
IN/OUT Result PICResultType;



Procedure DP_termAttemptAuthorized

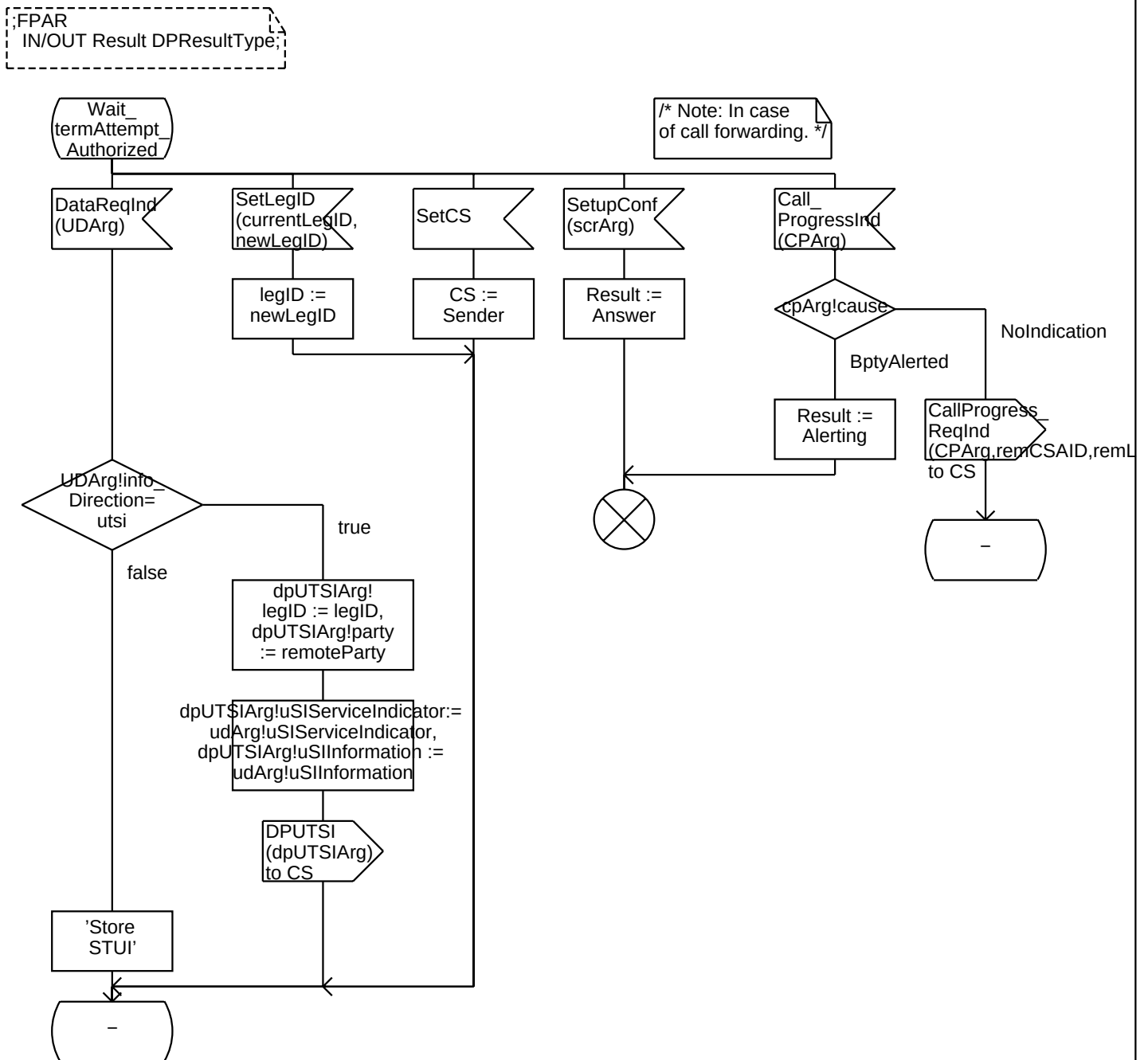
1(4)

FPAR
IN/OUT Result DPResultType;



Procedure DP_termAttemptAuthorized

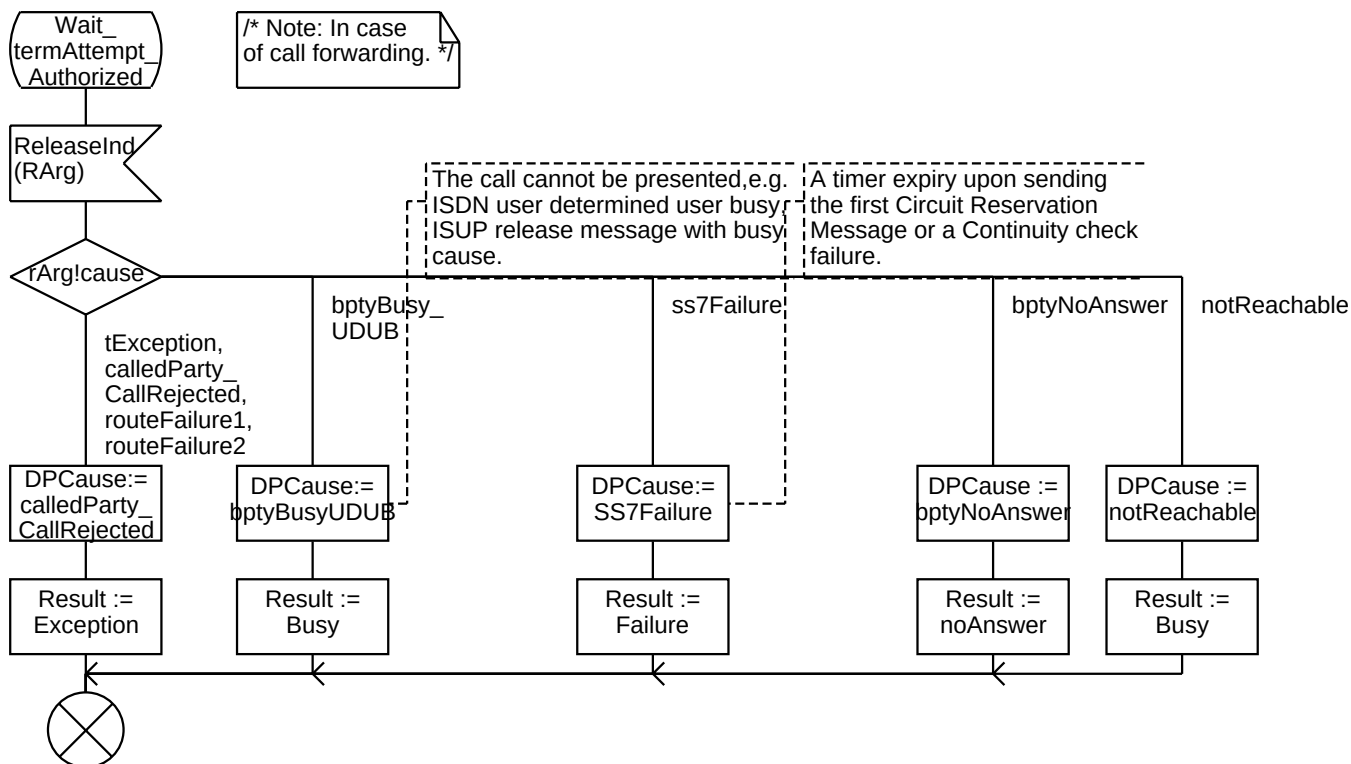
2(4)



Procedure DP_termAttemptAuthorized

3(4)

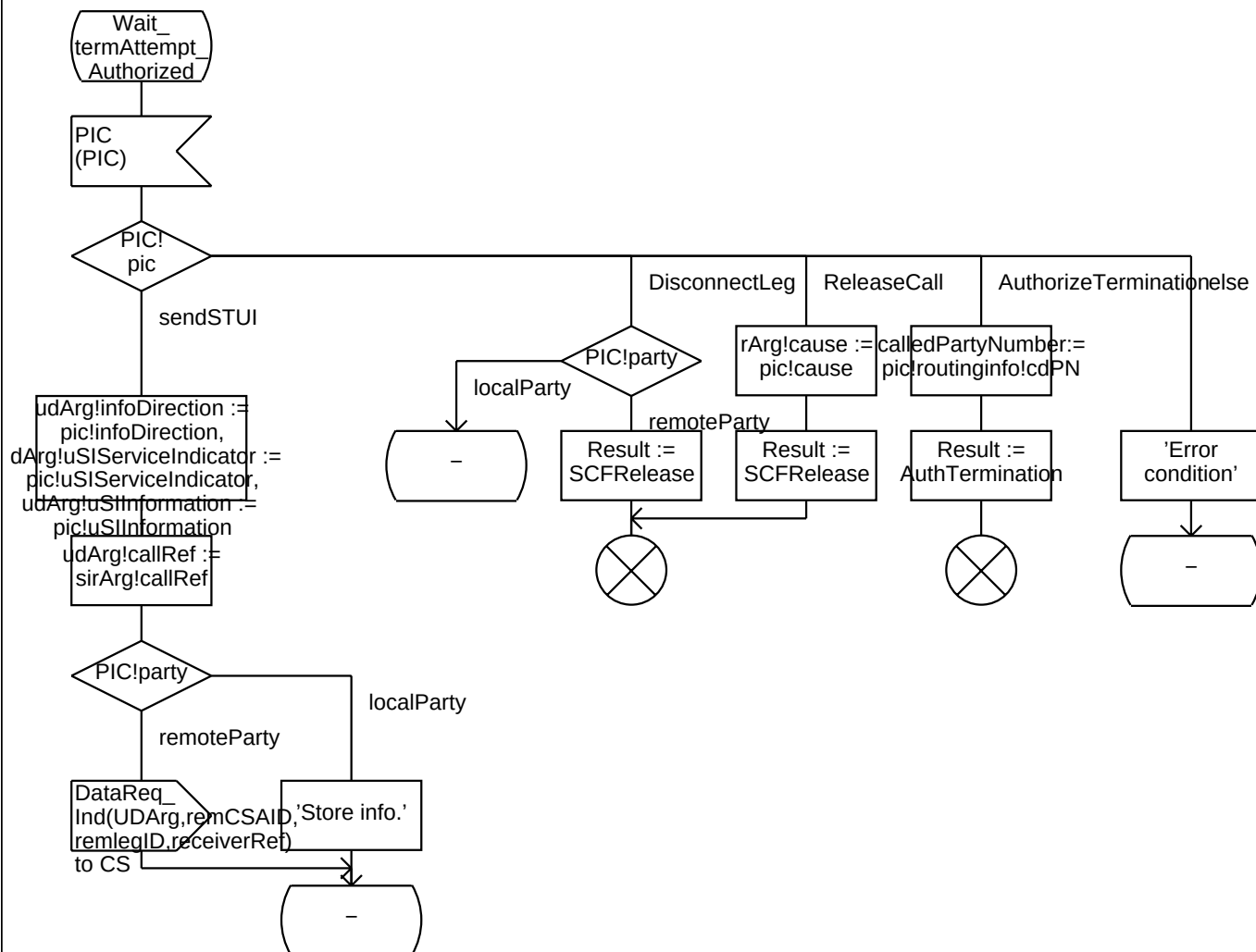
FPAR
IN/OUT Result DPResultType;



Procedure DP_termAttemptAuthorized

4(4)

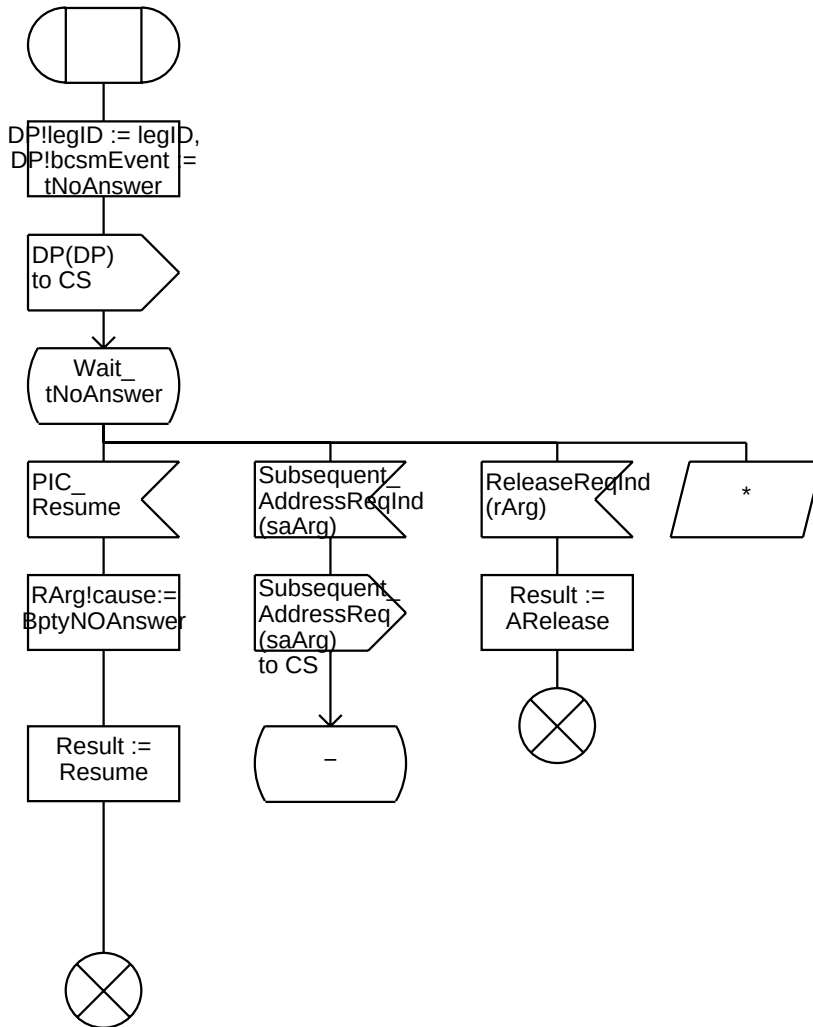
FPAR
IN/OUT Result DPResultType;



Procedure DP_tNoAnswer

1(4)

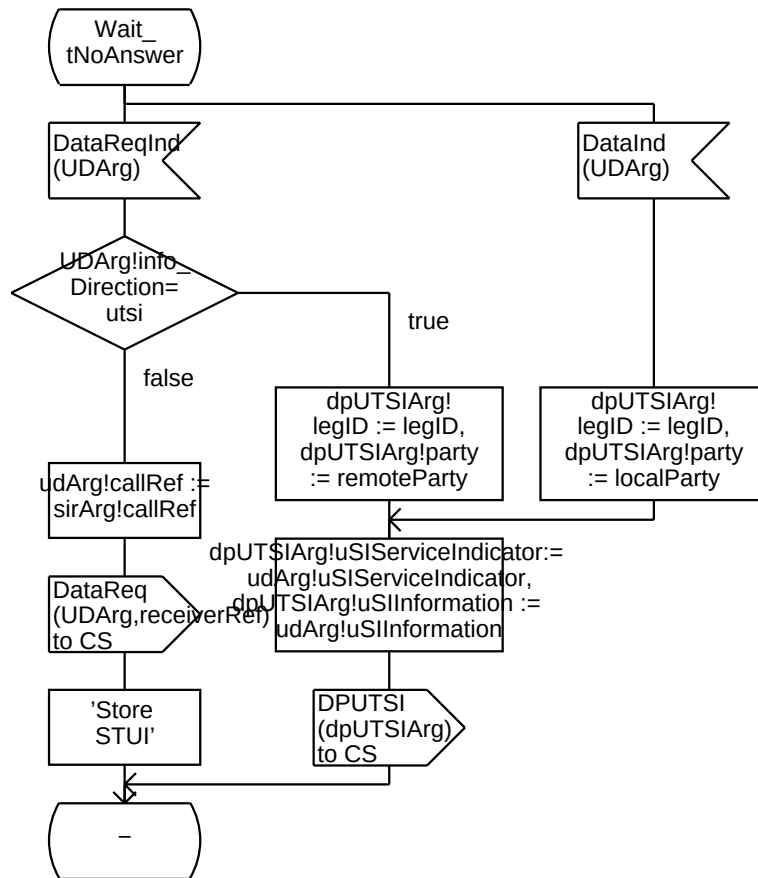
FPAR
 IN/OUT Result DPResultType;



Procedure DP_tNoAnswer

2(4)

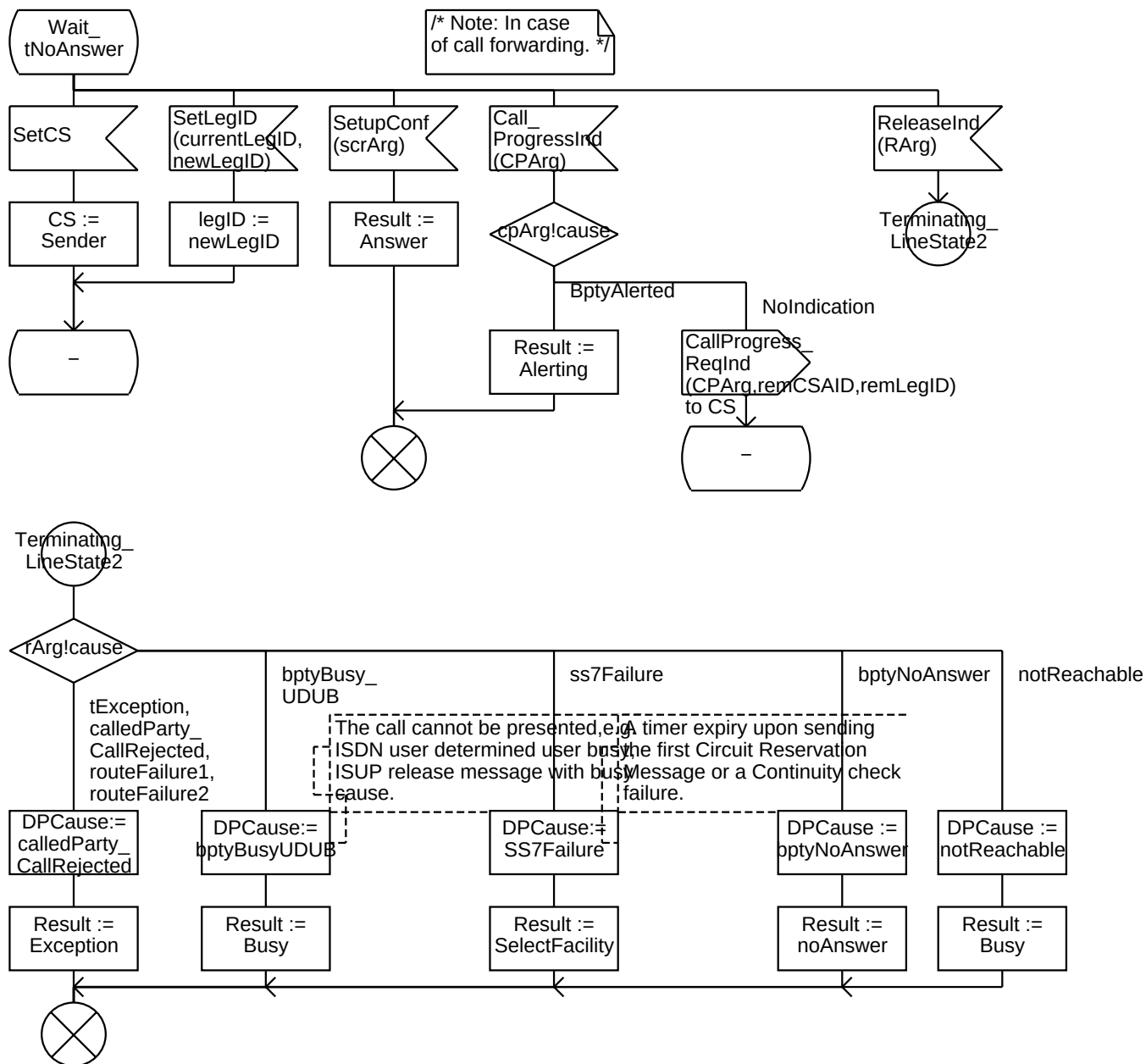
FPAR
 IN/OUT Result DPResultType;



Procedure DP_tNoAnswer

3(4)

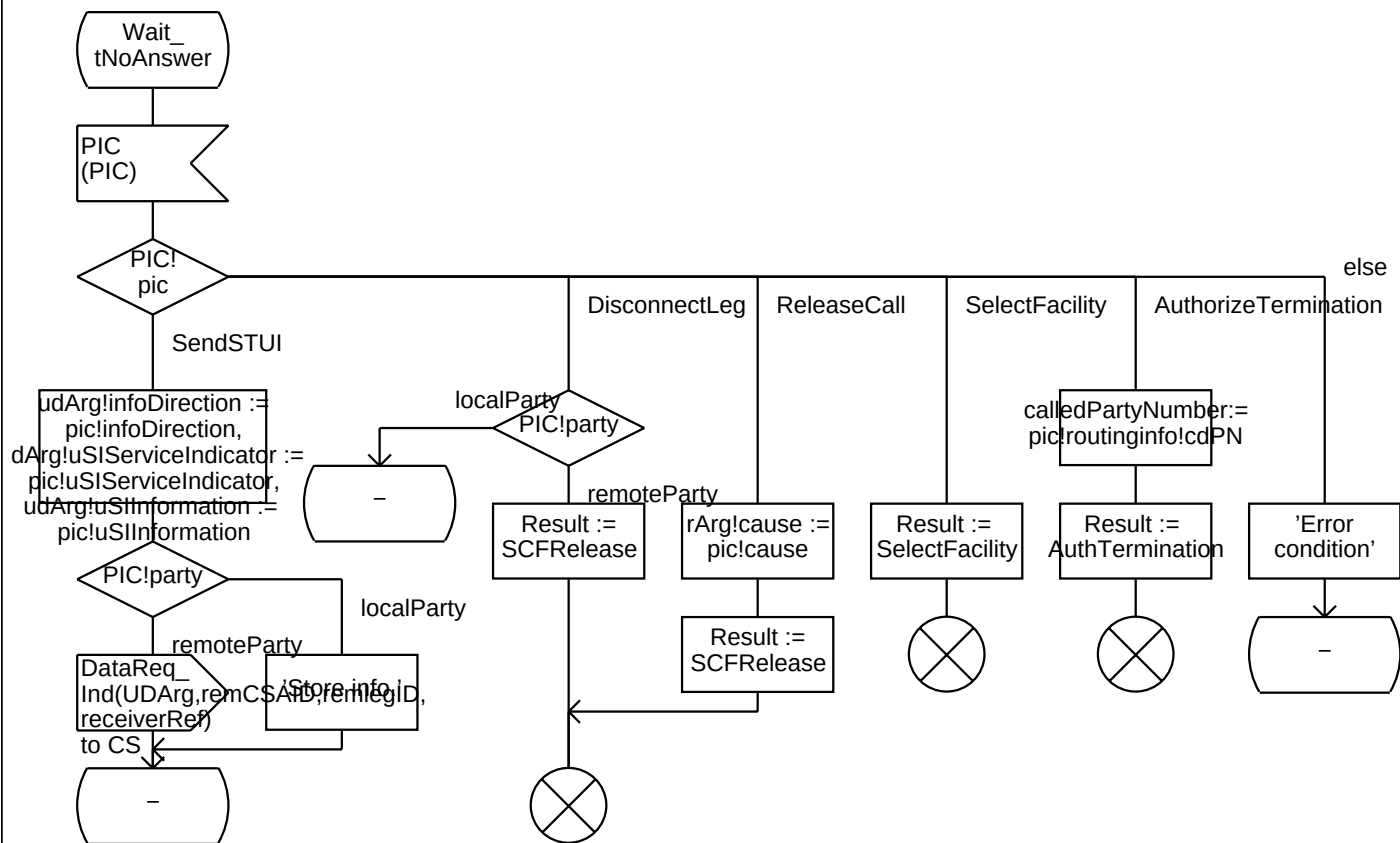
FPAR
IN/OUT Result DPResultType;



Procedure DP_tNoAnswer

4(4)

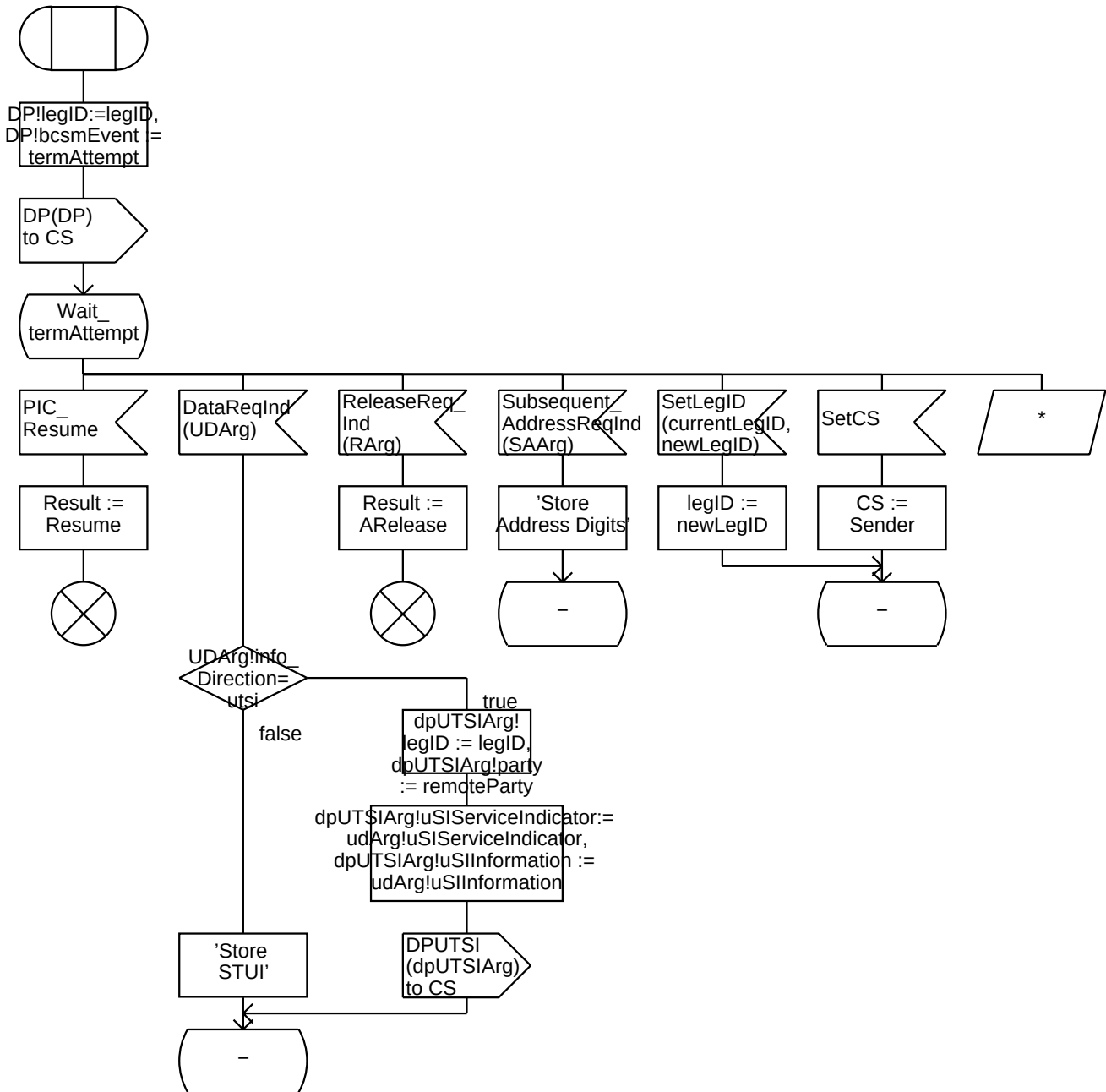
FPAR
IN/OUT Result DPResultType;



Procedure DP_termAttempt

1(3)

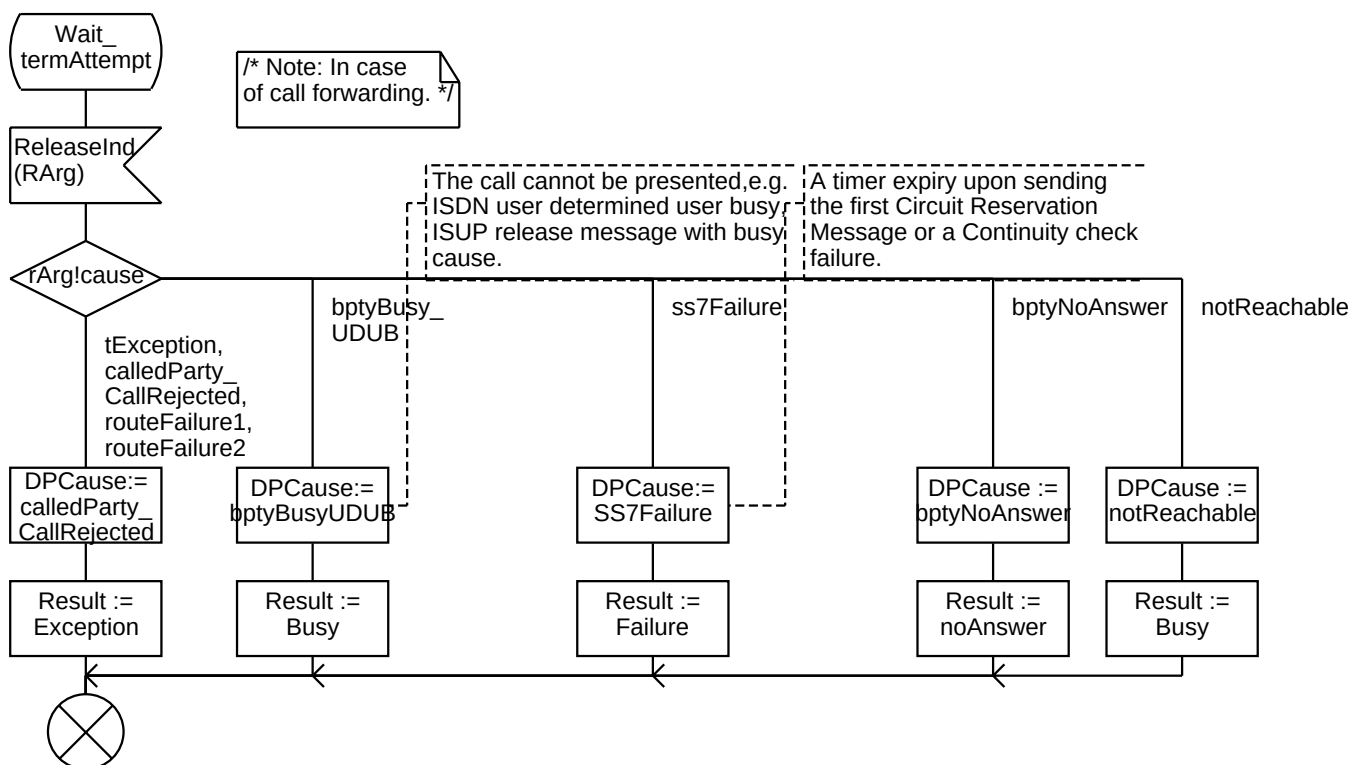
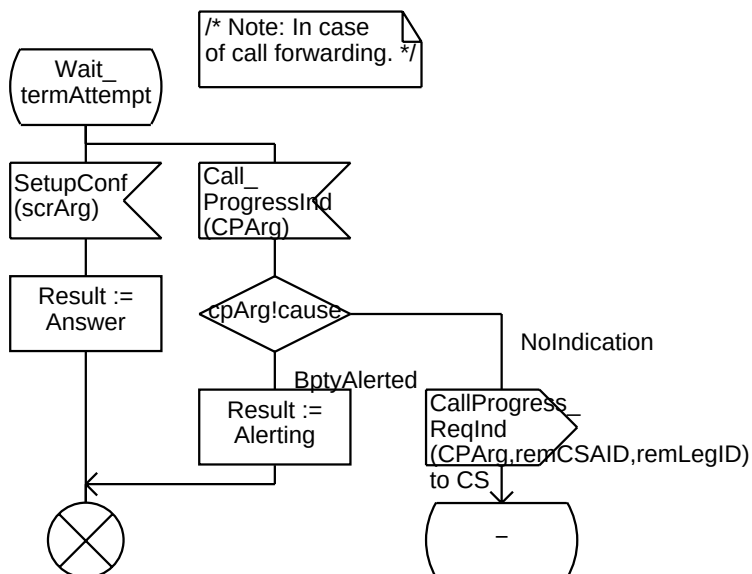
FPAR
IN/OUT Result DPResultType;



Procedure DP_termAttempt

2(3)

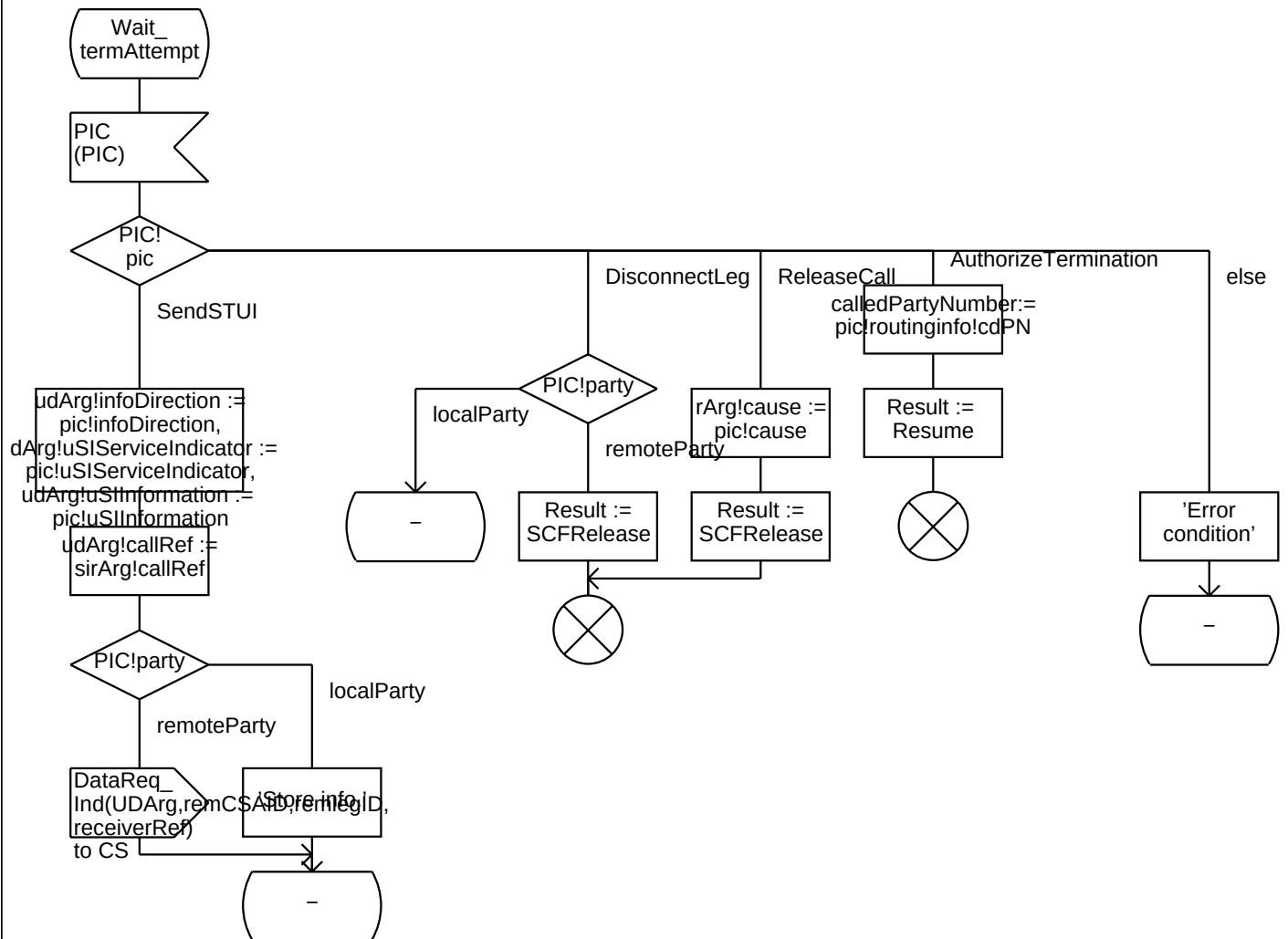
FPAR
IN/OUT Result DPResultType;



Procedure DP_termAttempt

3(3)

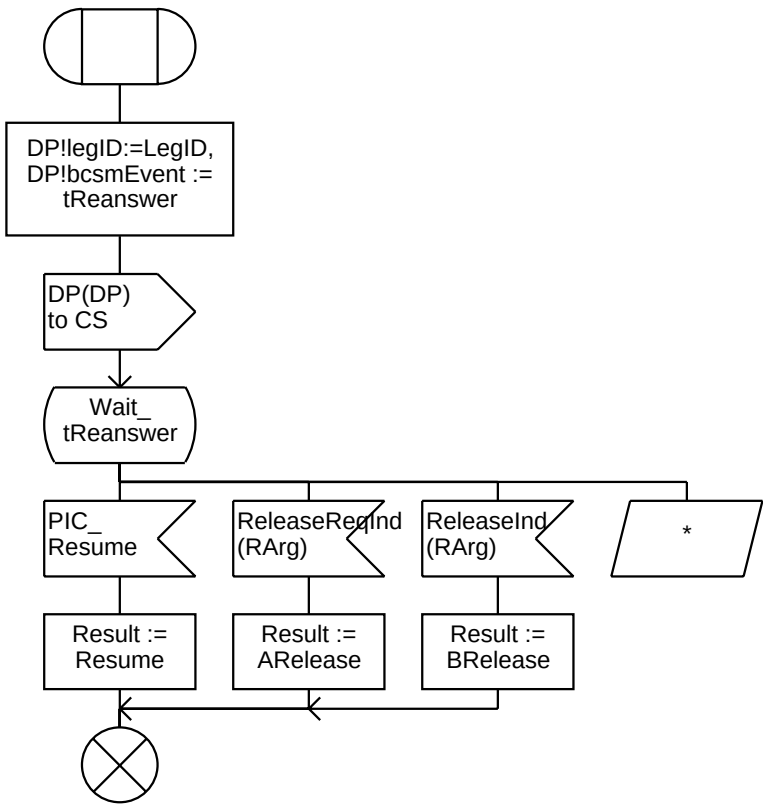
FPAR
IN/OUT Result DPResultType;



Procedure DP_tReanswer

1(3)

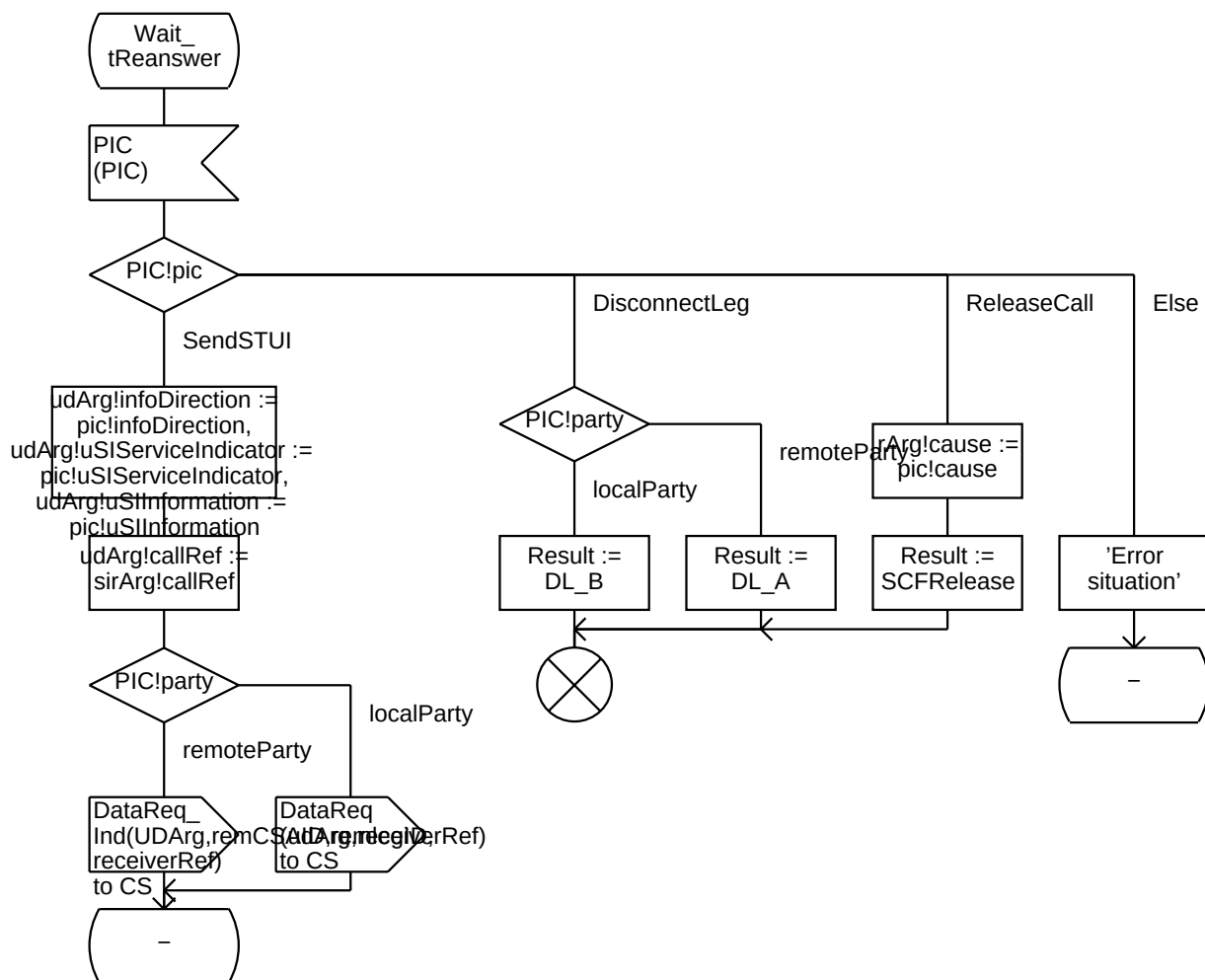
;FPAR
IN/OUT Result DPResultType;



Procedure DP_tReanswer

2(3)

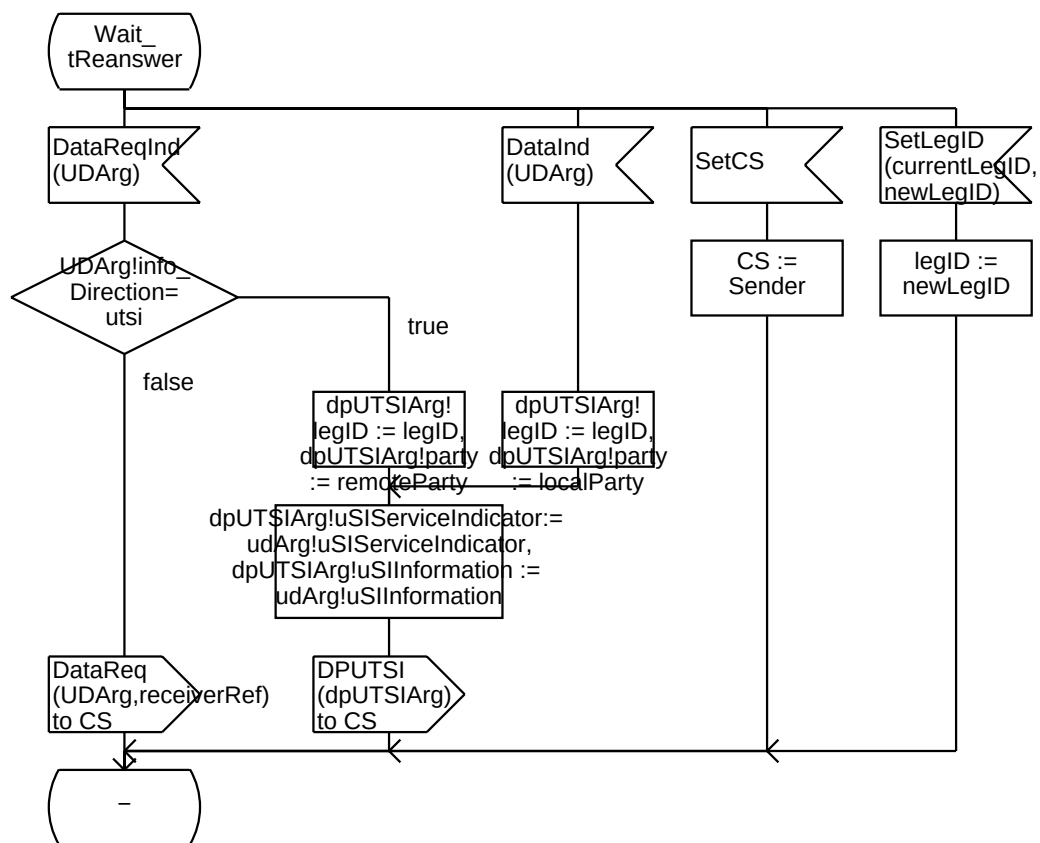
FPAR
IN/OUT Result DPResultType;



Procedure DP_tReanswer

3(3)

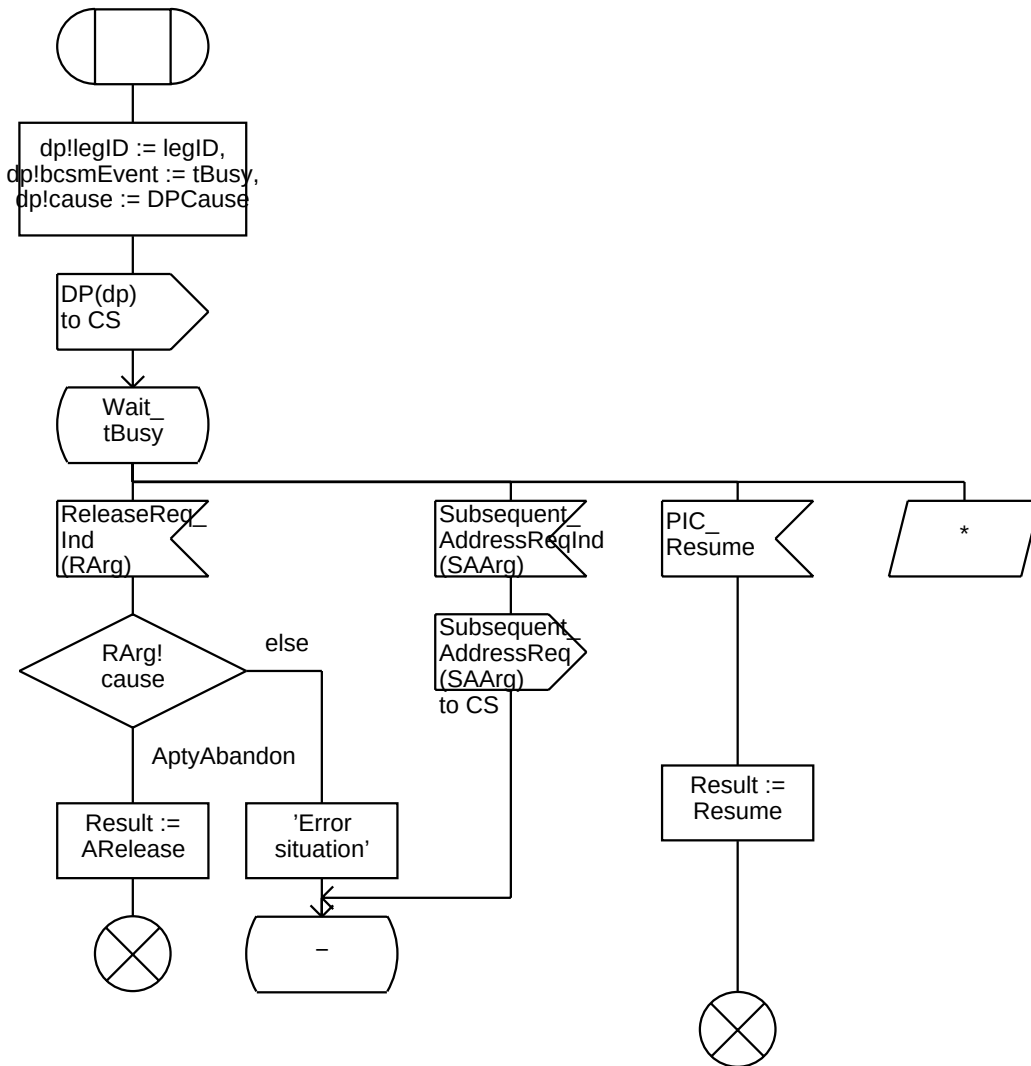
FPAR
IN/OUT Result DPResultType;



Procedure DP_tBusy

1(4)

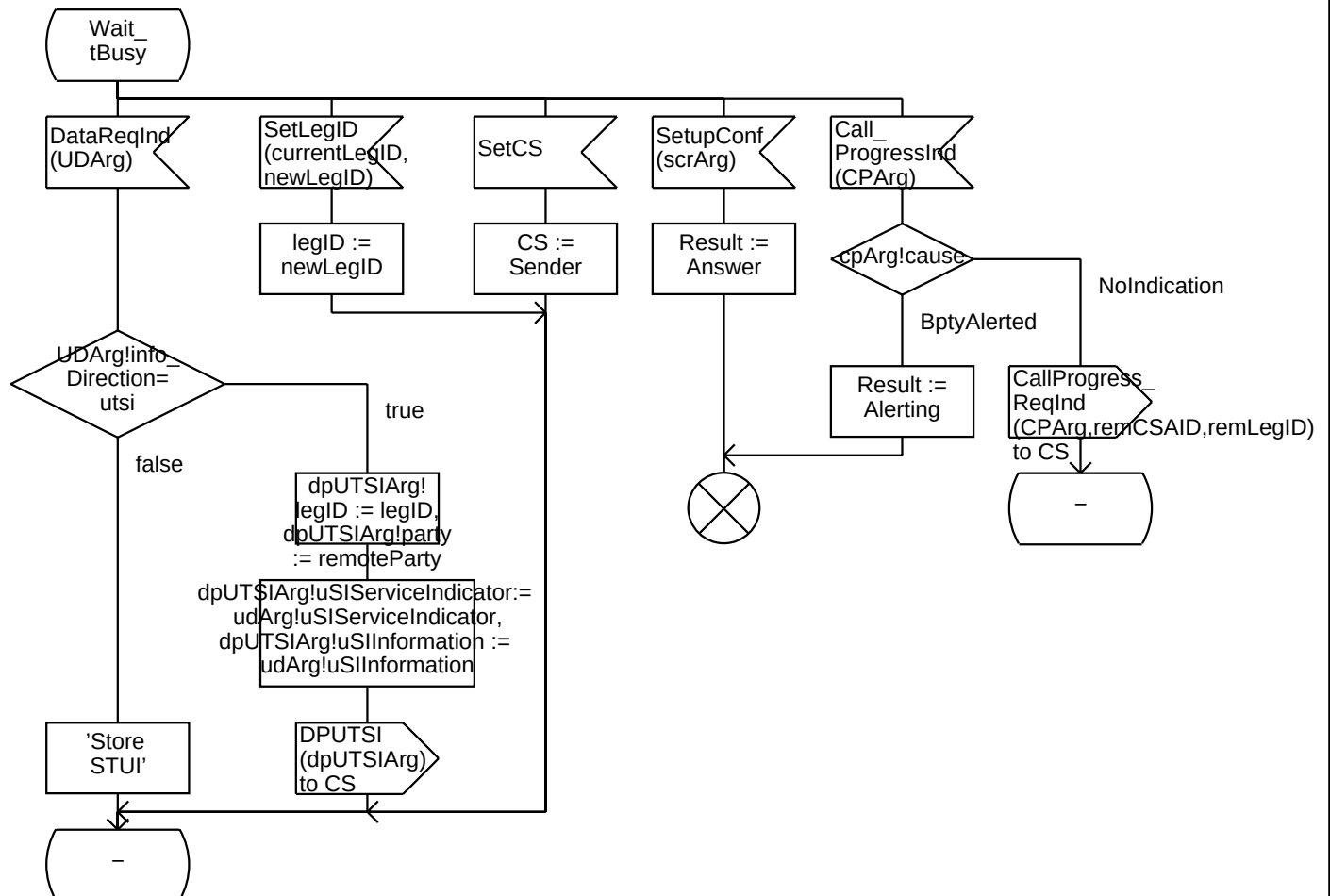
FPAR
IN/OUT Result DPResultType;



Procedure DP_tBusy

2(4)

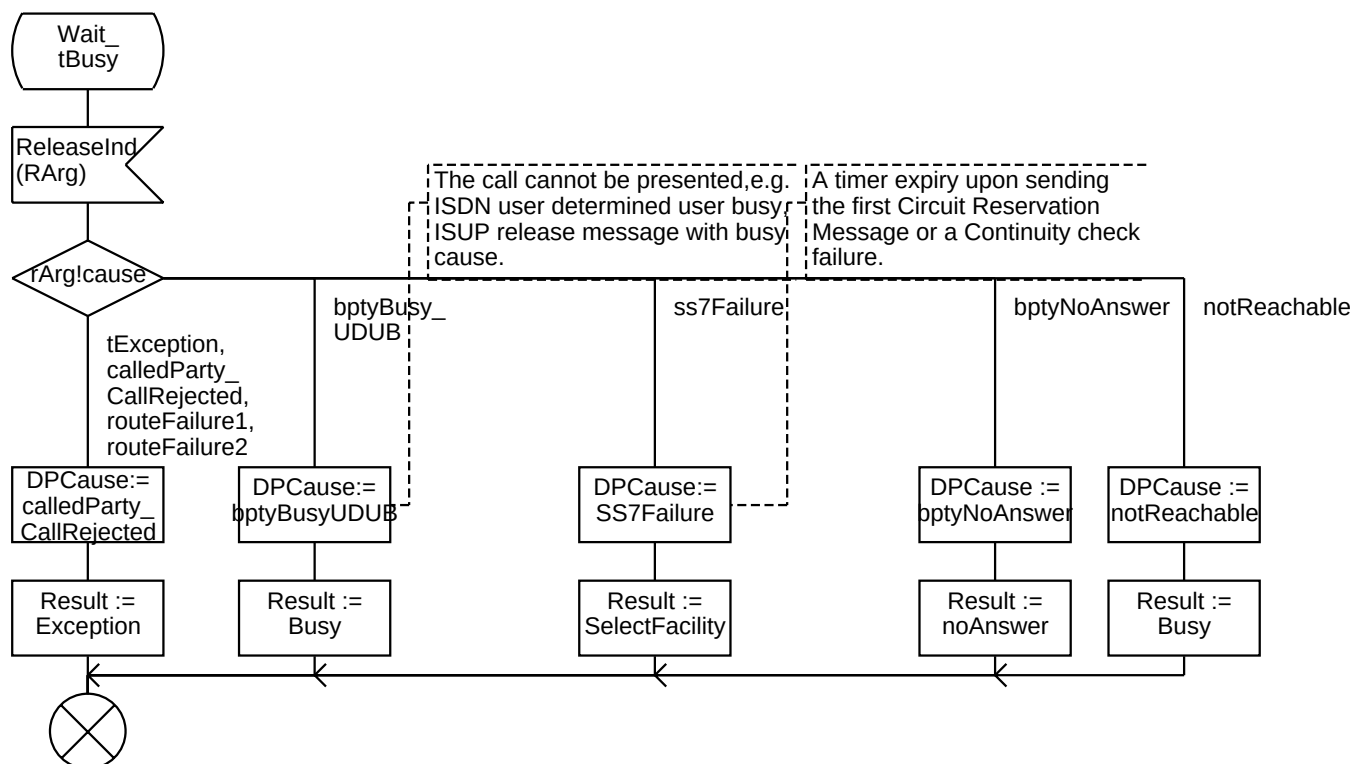
FPAR
IN/OUT Result DPResultType;



Procedure DP_tBusy

3(4)

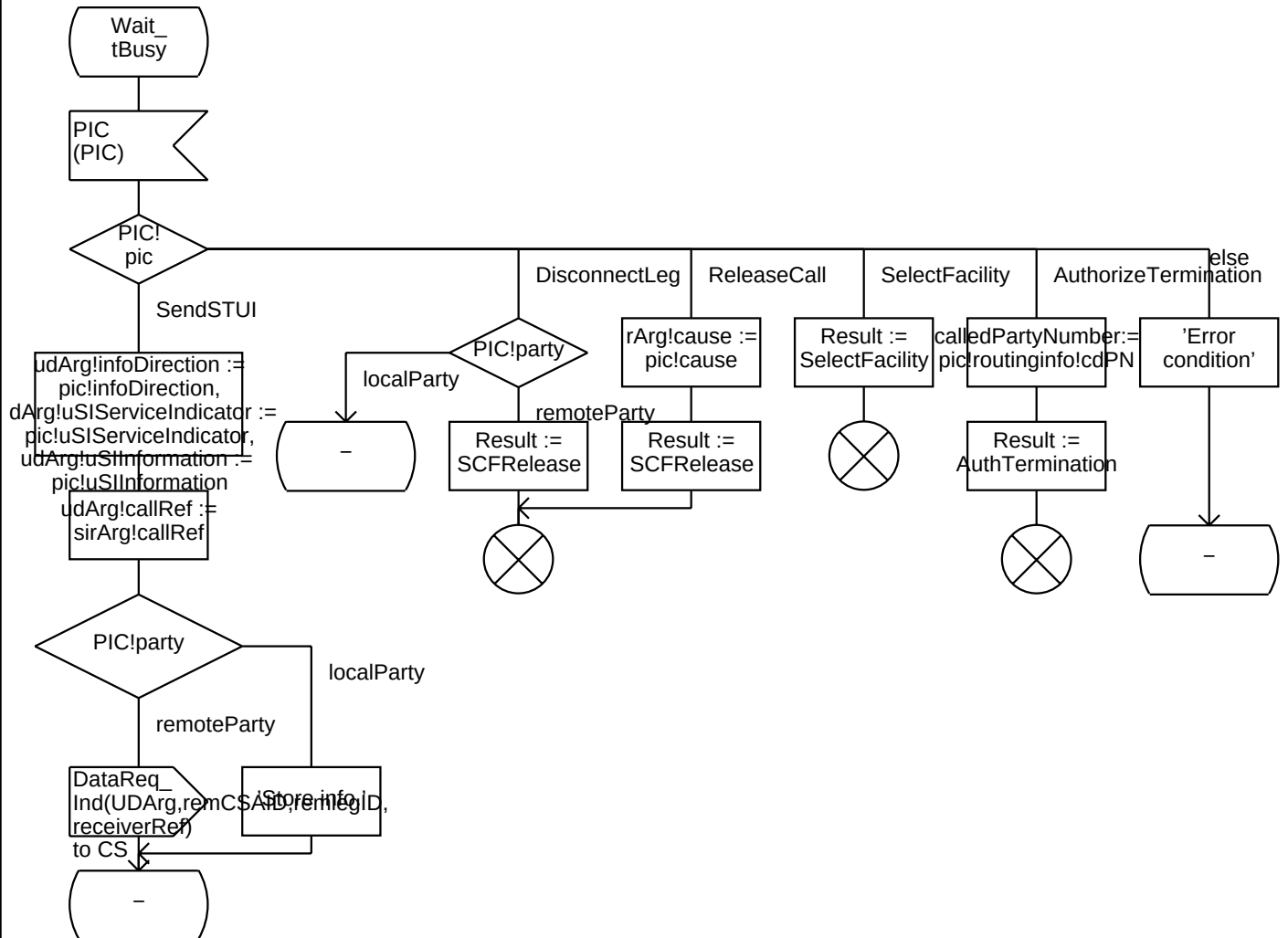
FPAR
IN/OUT Result DPResultType;



Procedure DP_tBusy

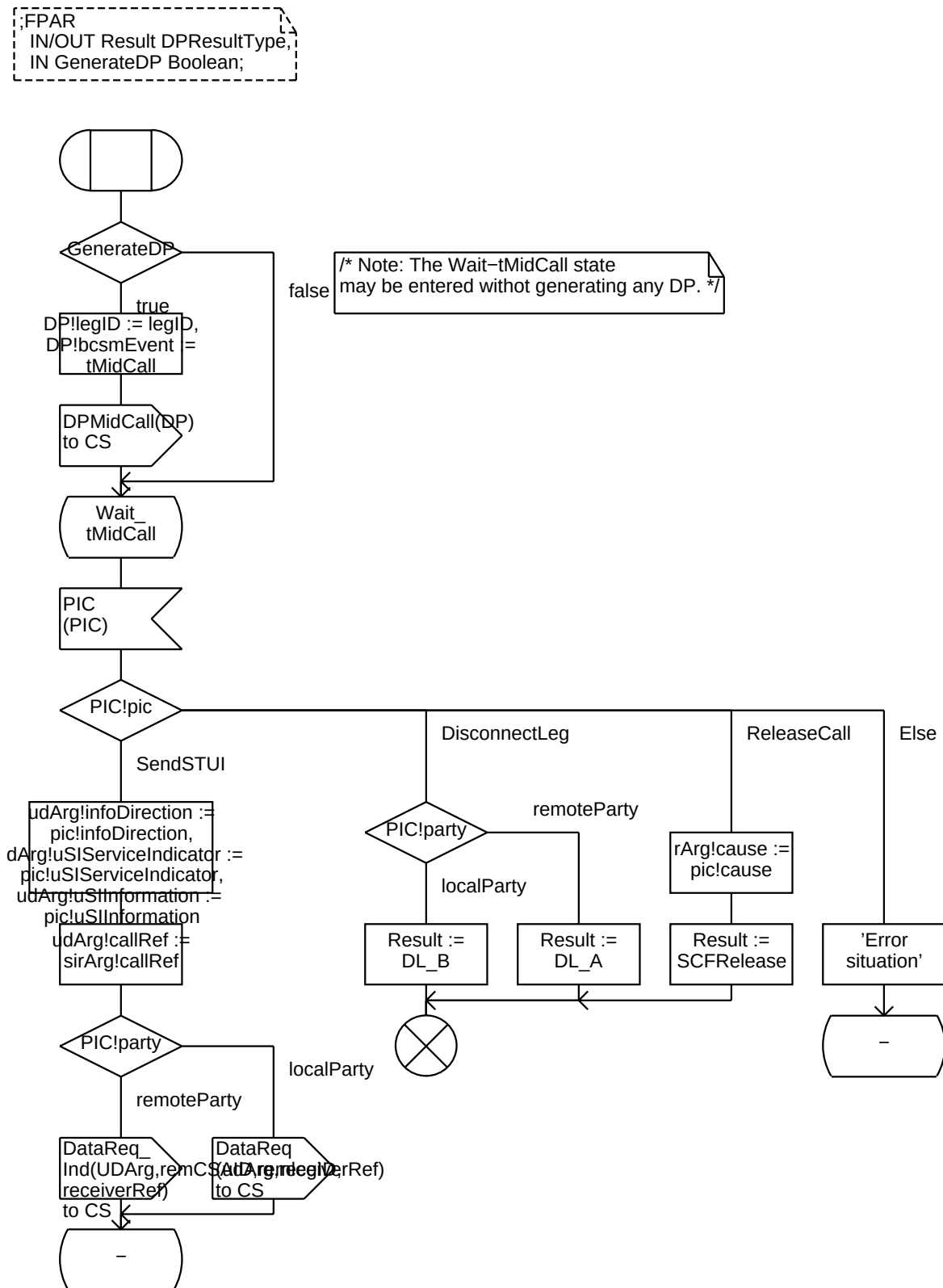
4(4)

FPAR
IN/OUT Result DPResultType;



Procedure DP_tMidCall

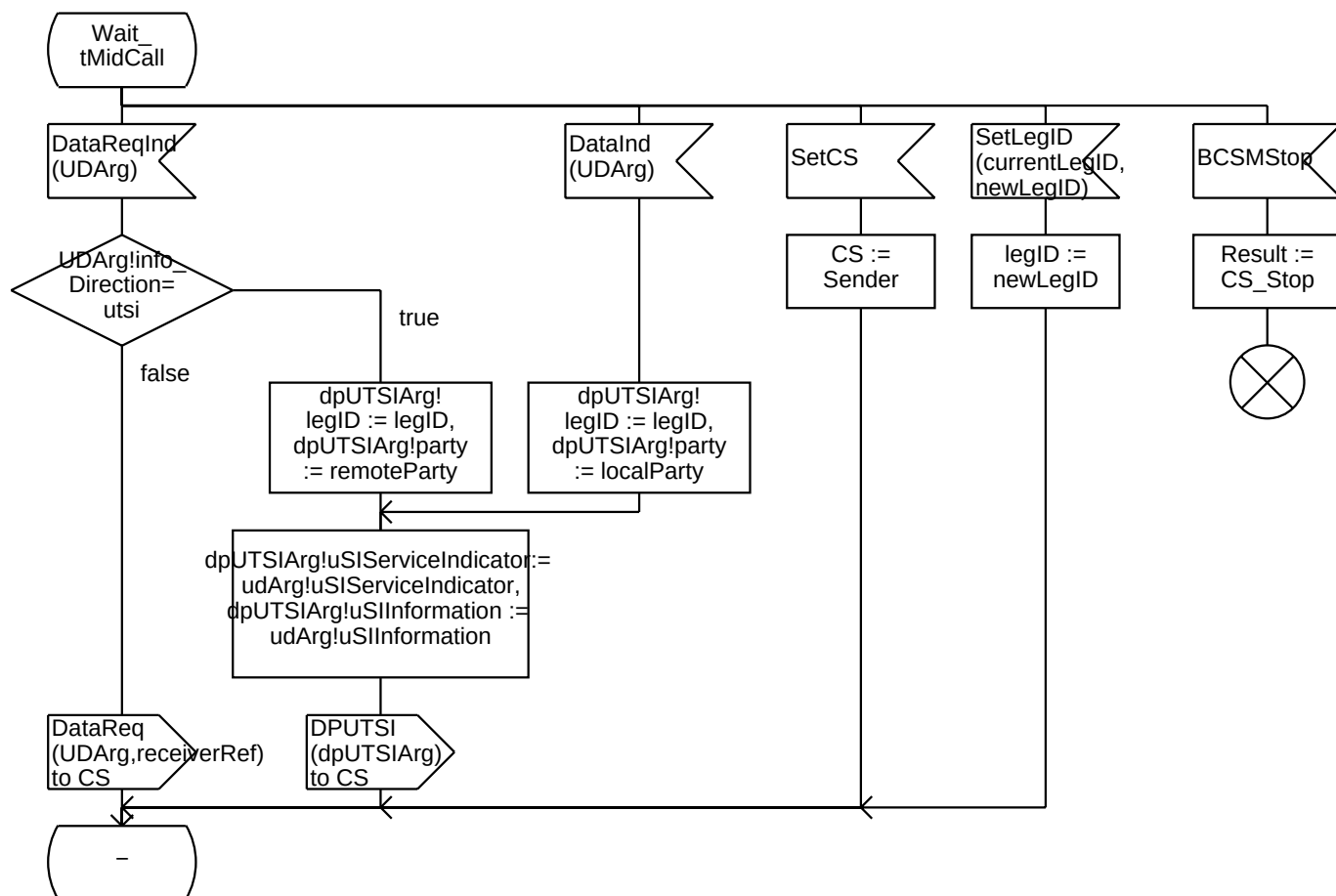
1(2)



Procedure DP_tMidCall

2(2)

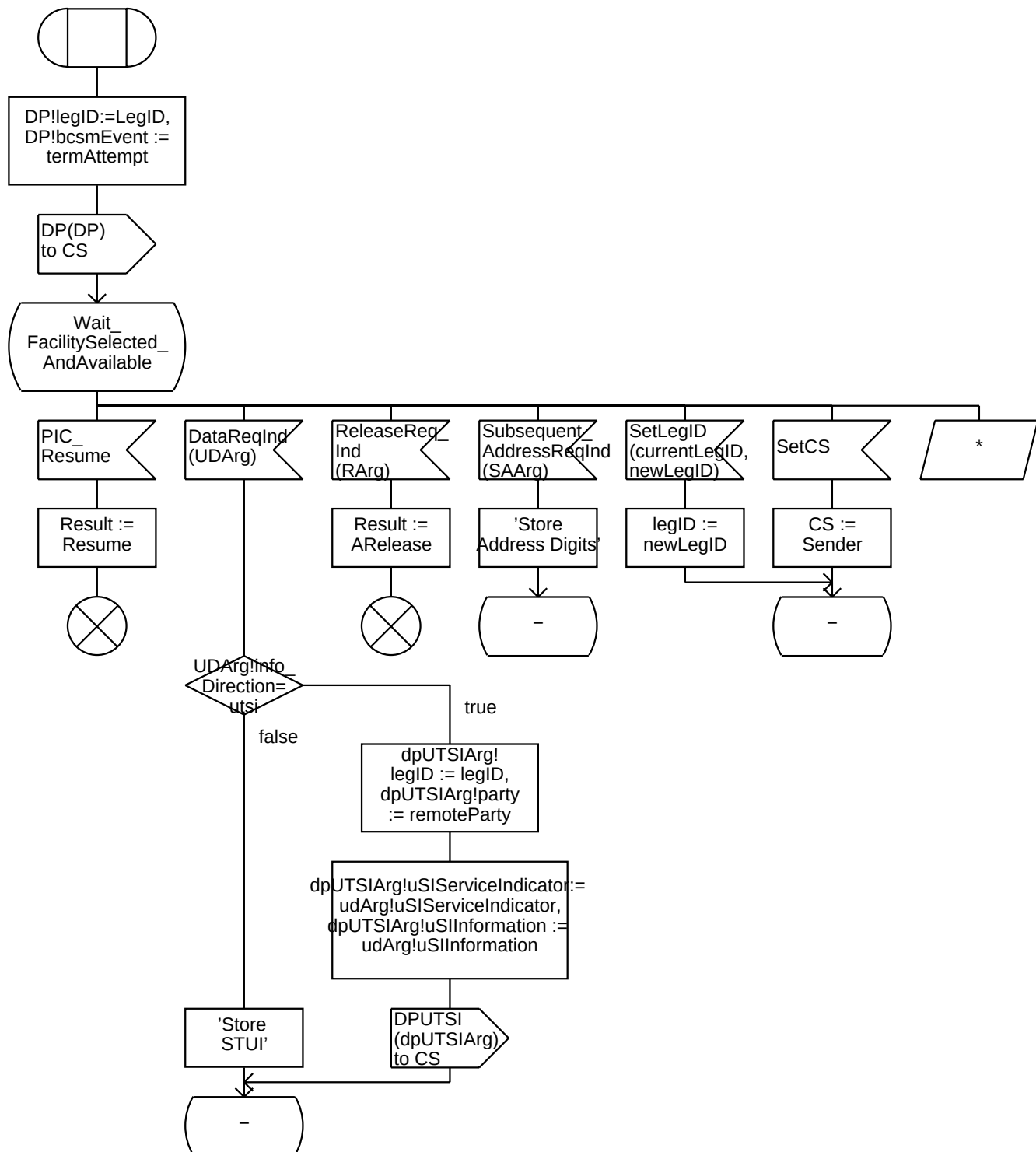
FPAR
IN/OUT Result DPResultType,
IN GenerateDP Boolean;



Procedure DP_Facility_Selected_and_Available

1(3)

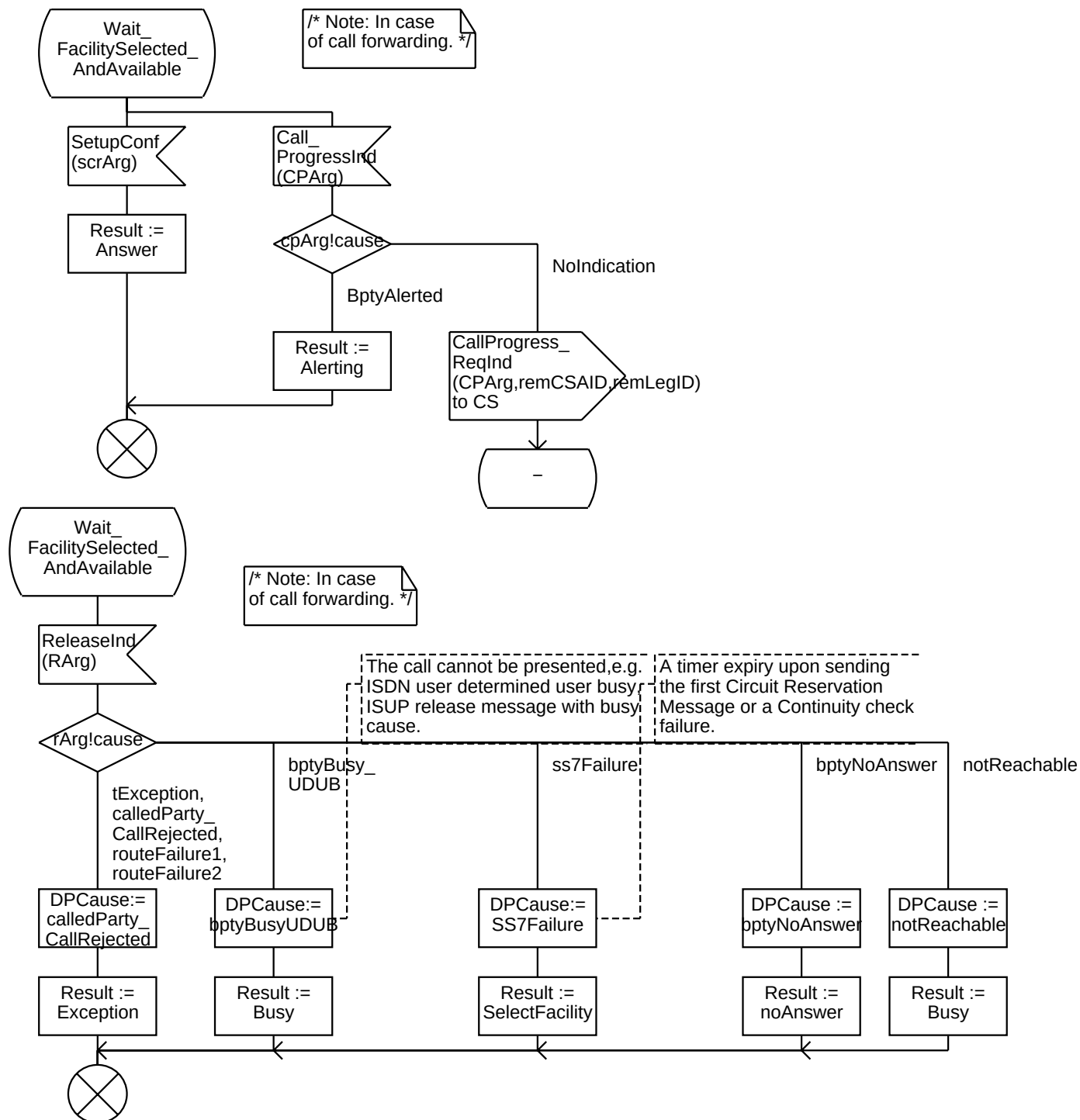
FPAR
IN/OUT Result DPResultType;



Procedure DP_Facility_Selected_and_Available

2(3)

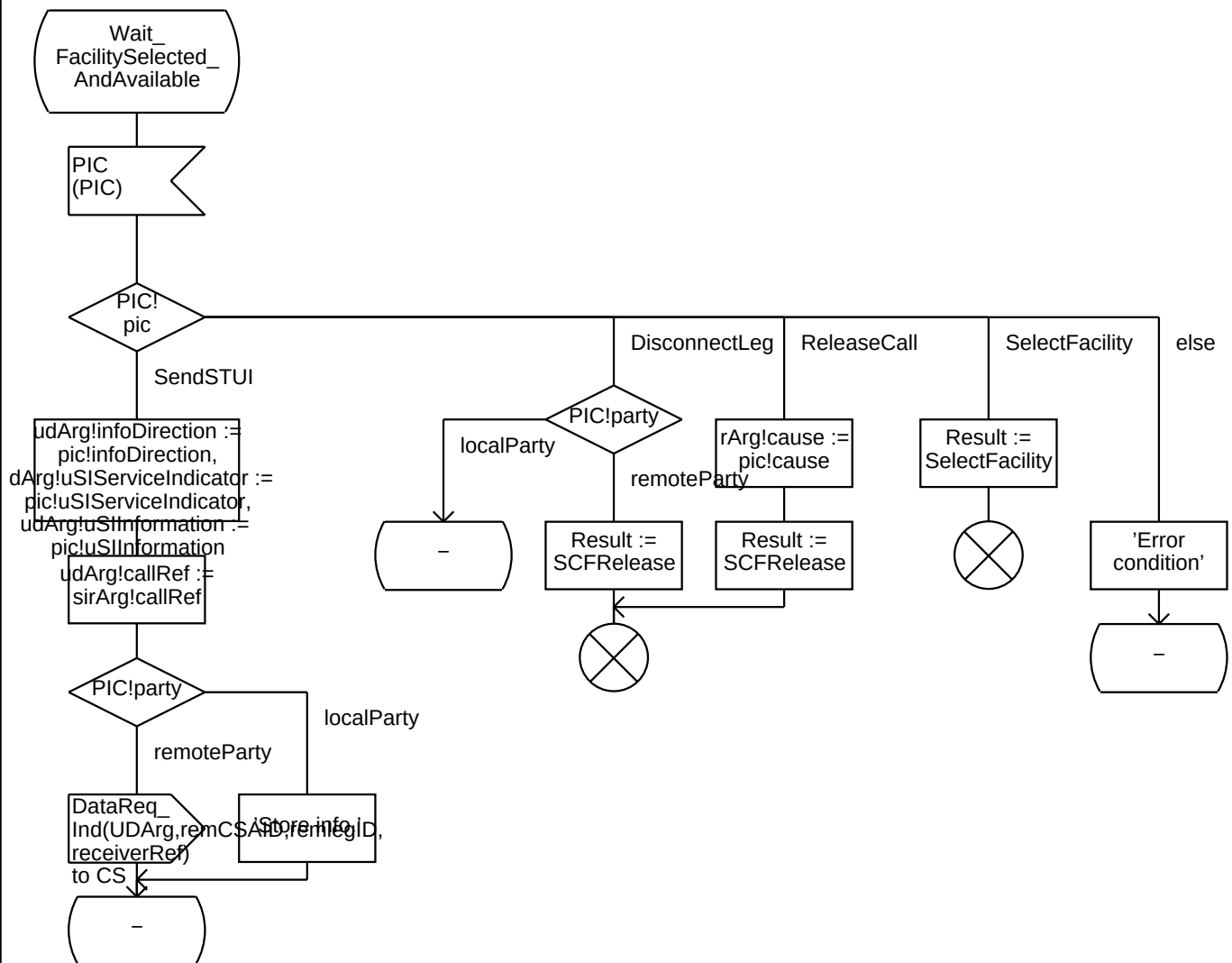
FPAR
IN/OUT Result DPResultType;



Procedure DP_Facility_Selected_and_Available

3(3)

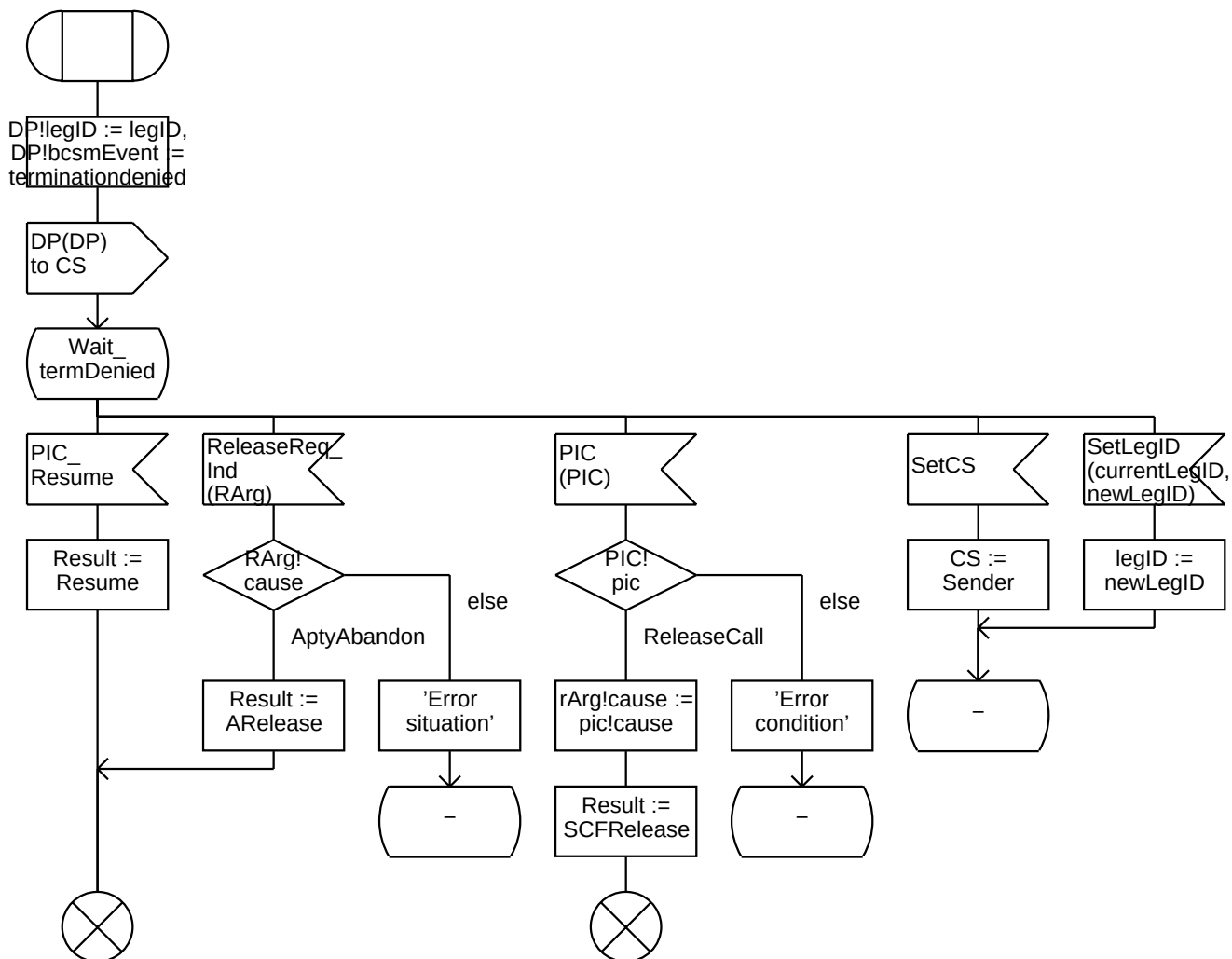
FPAR
IN/OUT Result DPResultType;



Procedure DP_termDenied

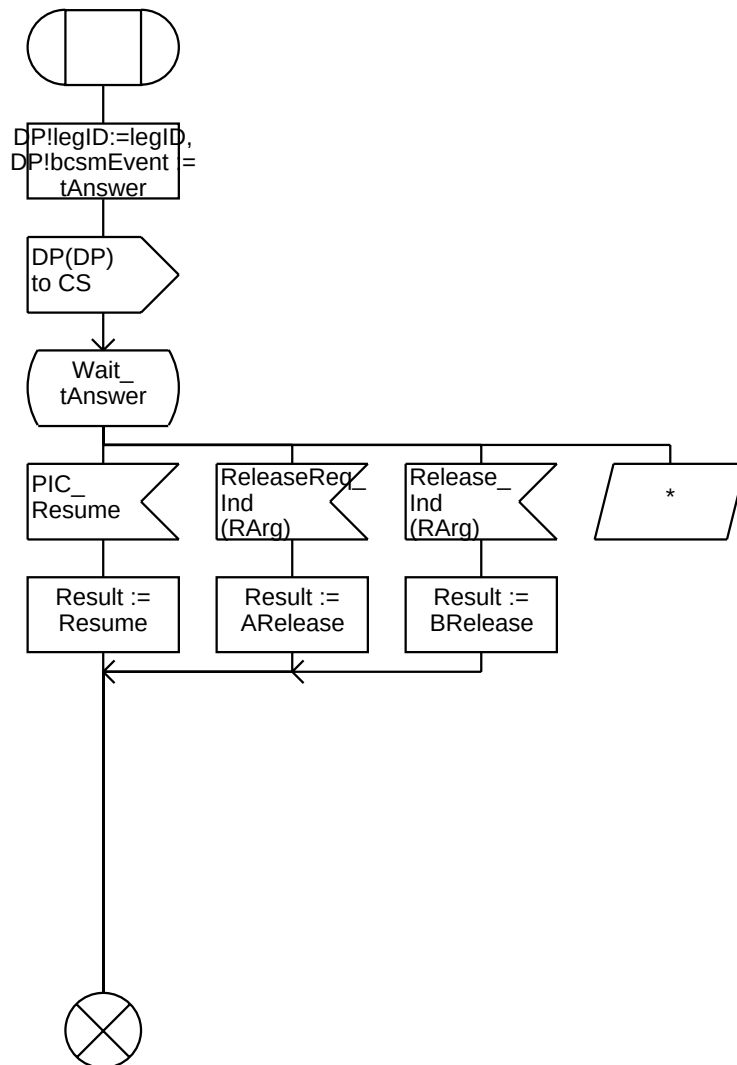
1(1)

FPAR
IN/OUT Result DPResultType;



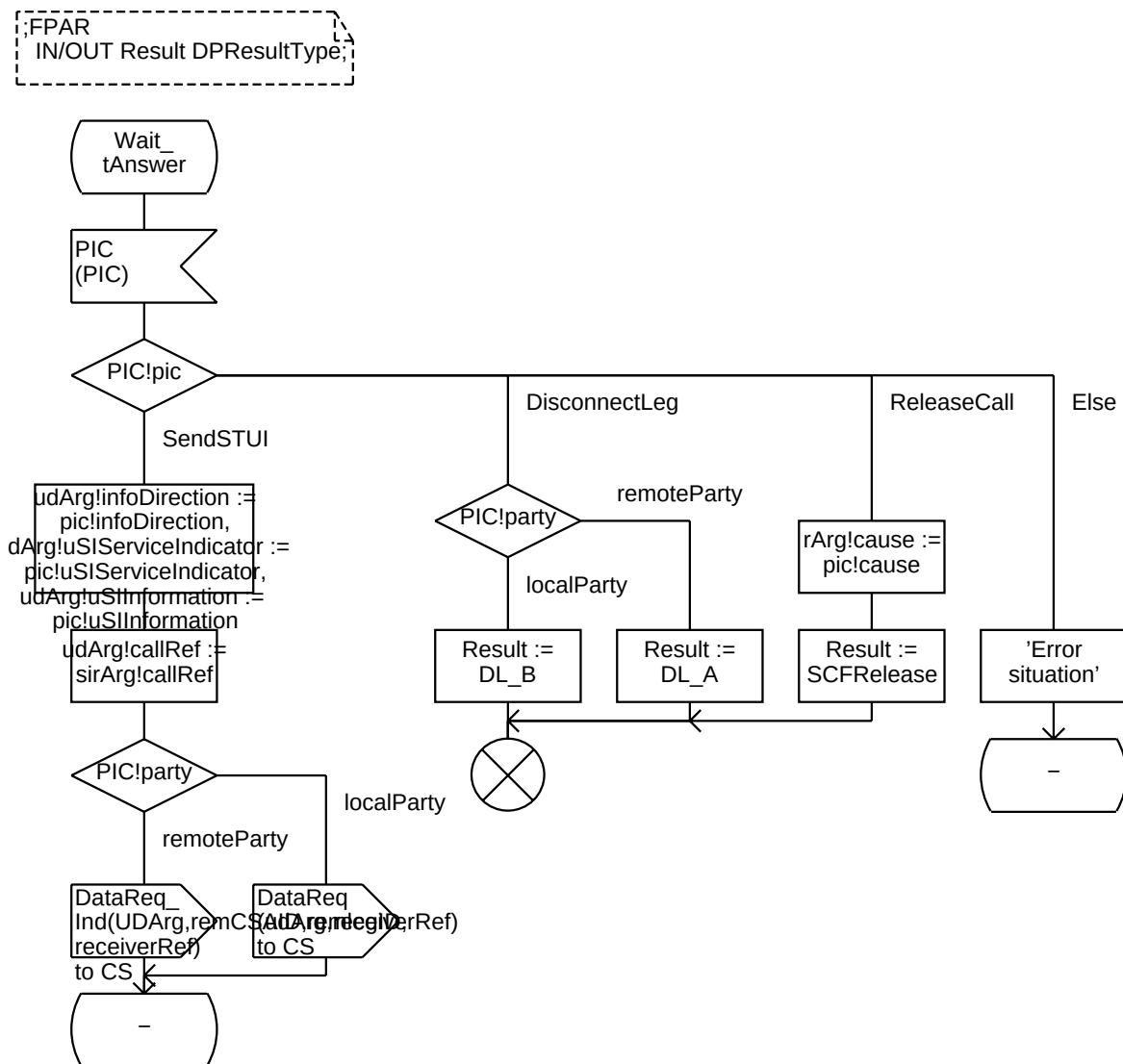
1(3)

```
IN/OUT Result DPRResultType;
```



Procedure DP_tAnswer

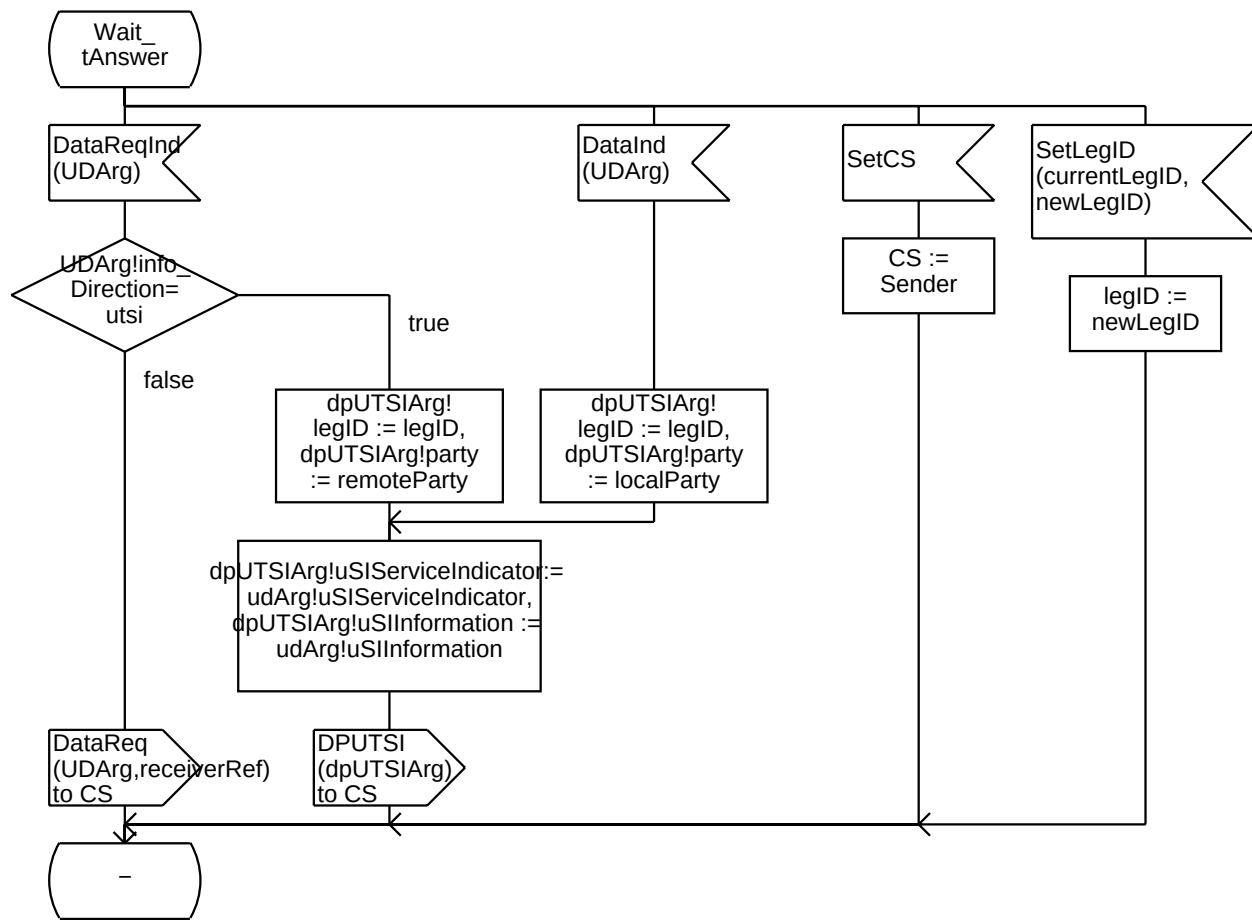
2(3)



Procedure DP_tAnswer

3(3)

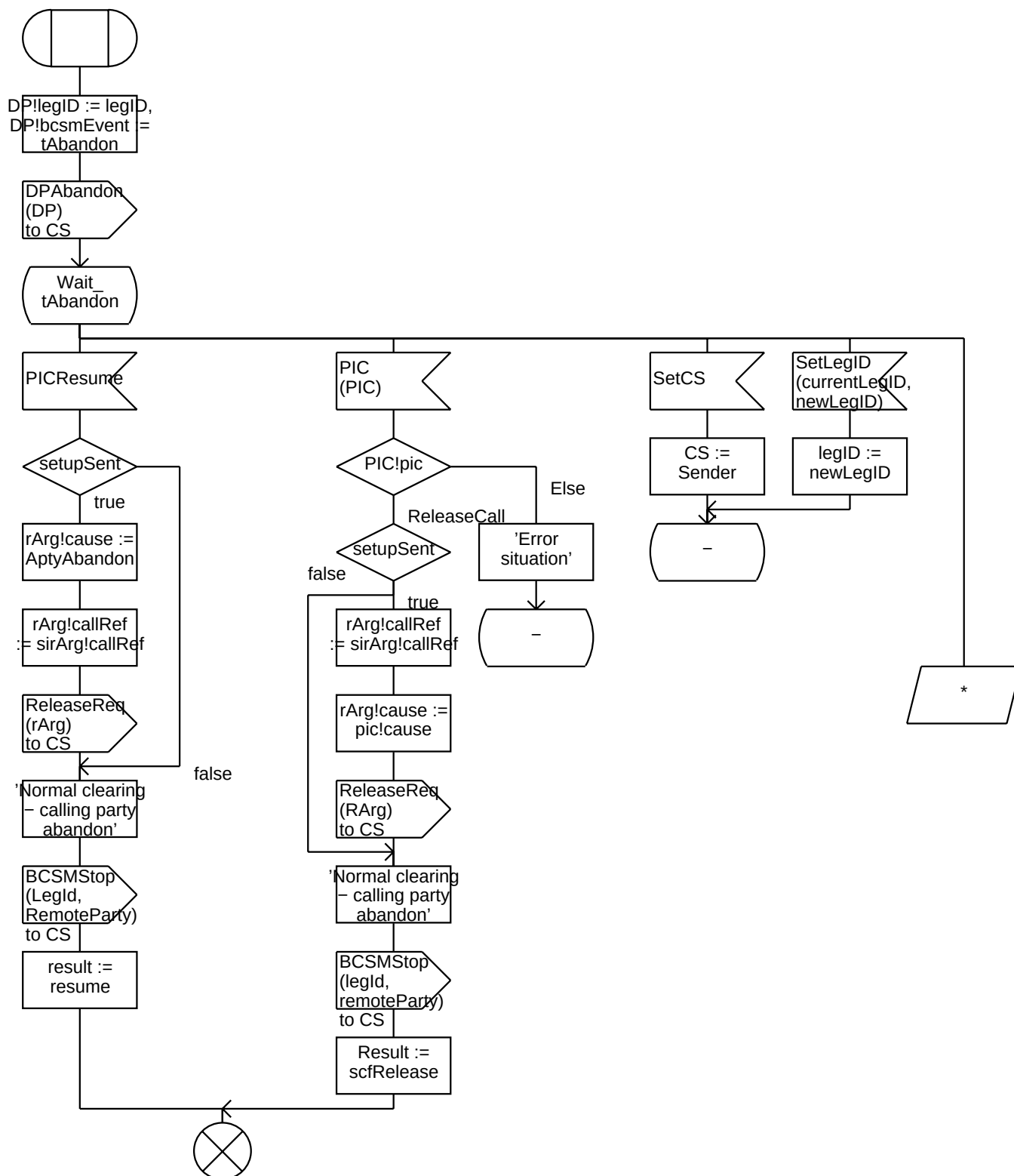
FPAR
IN/OUT Result DPResultType;



Procedure DP_tAbandon

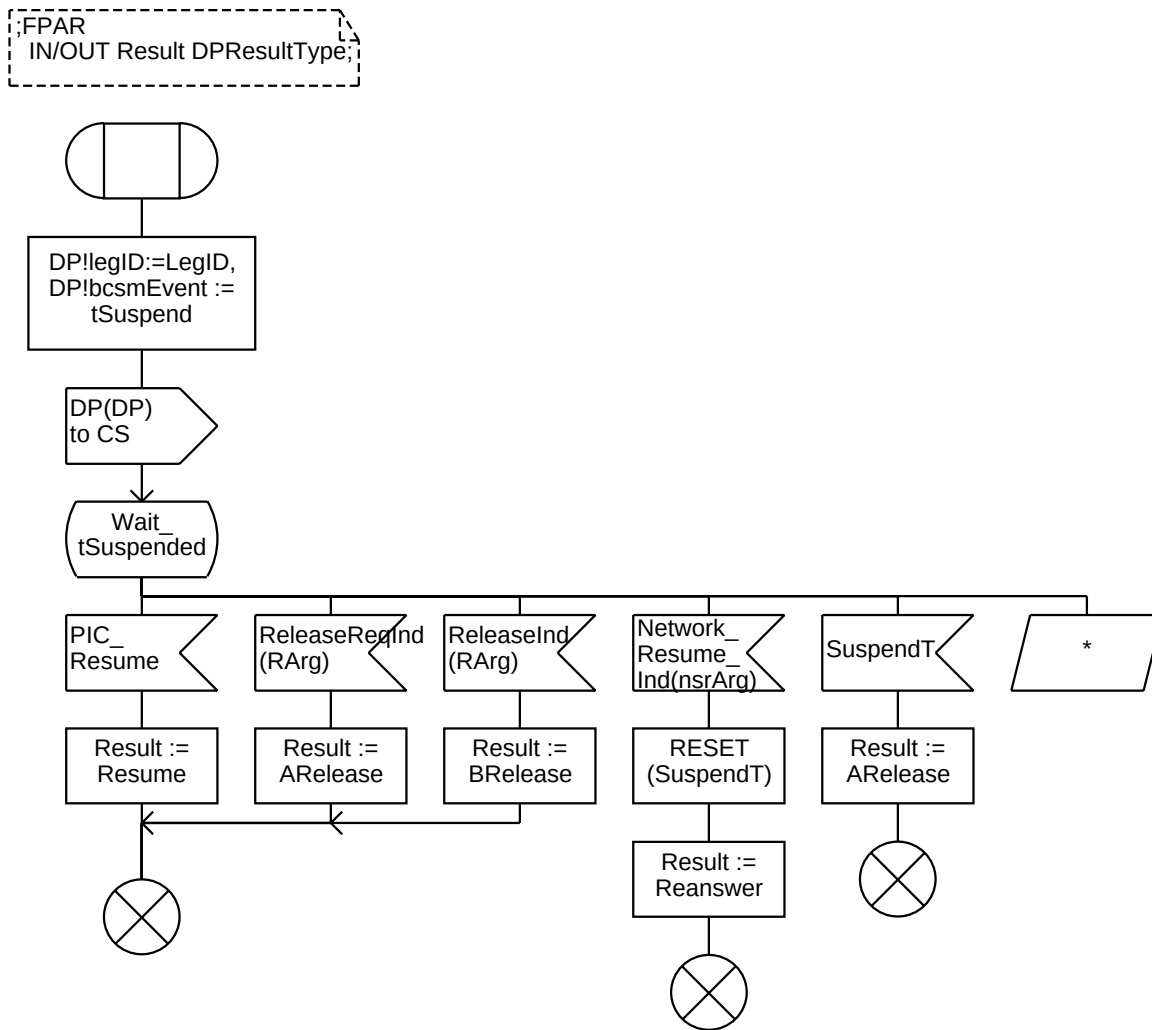
1(1)

FPAR
IN/OUT Result DPResultType;



Procedure DP_tSuspend

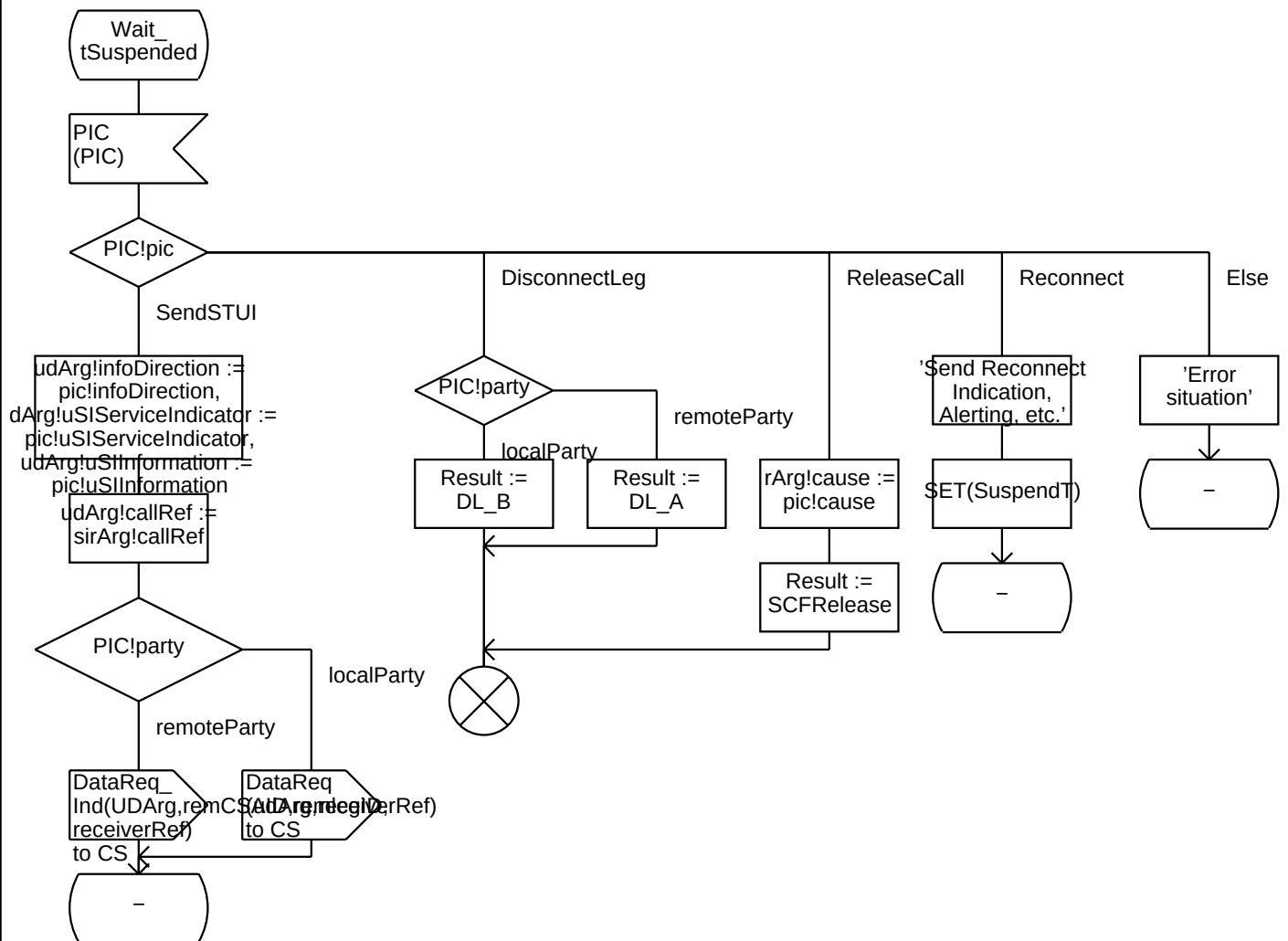
1(3)



Procedure DP_tSuspend

2(3)

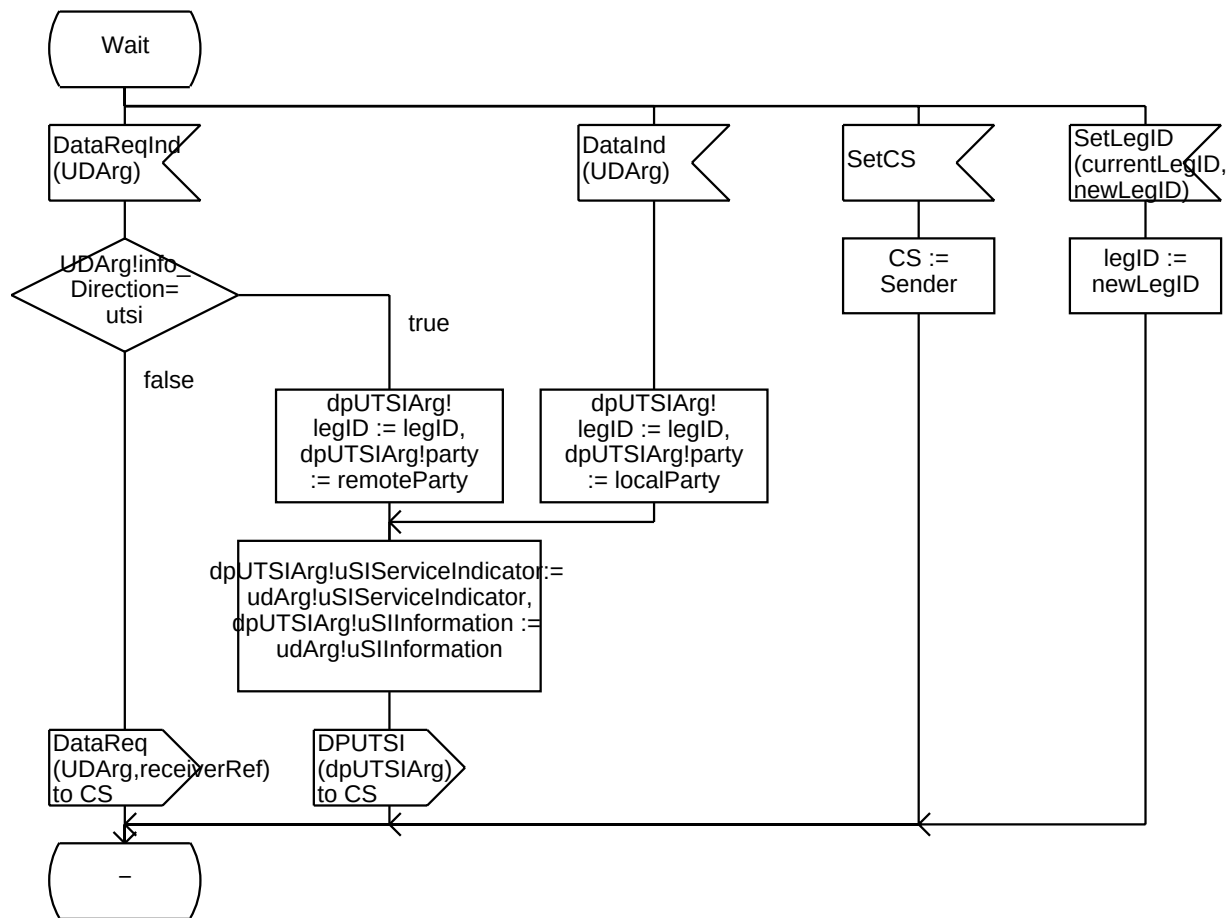
FPAR
IN/OUT Result DPResultType;



Procedure DP_tSuspend

3(3)

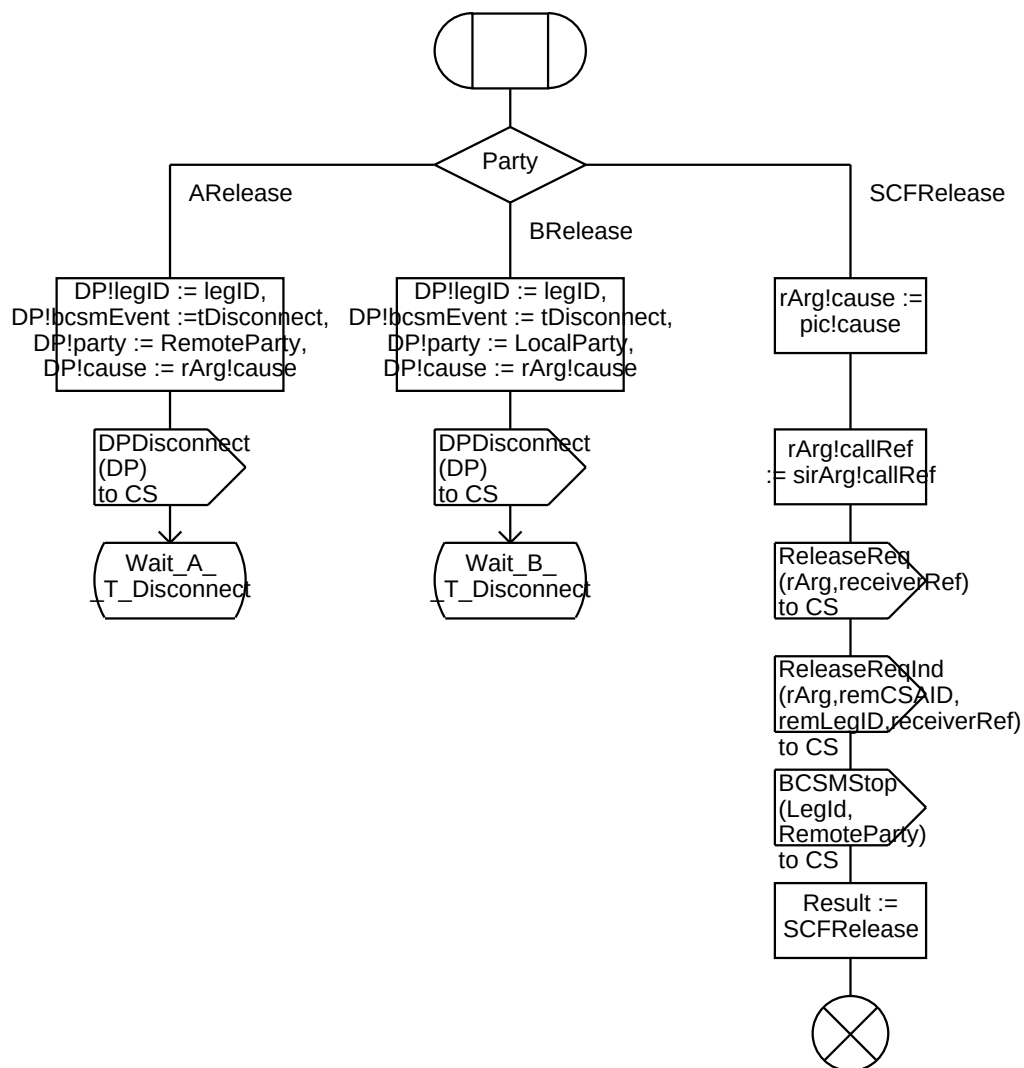
FPAR
IN/OUT Result DPResultType;



Procedure DP_tDisconnect

1(7)

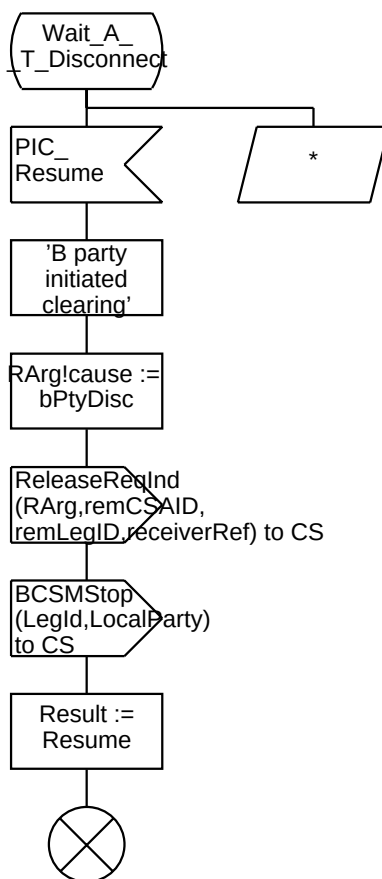
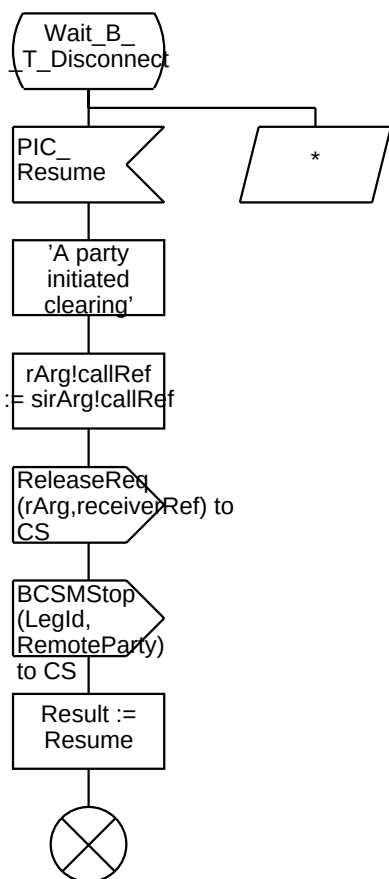
FPAR
IN/OUT Result DPResultType,
IN Party DPResultType;



Procedure DP_tDisconnect

2(7)

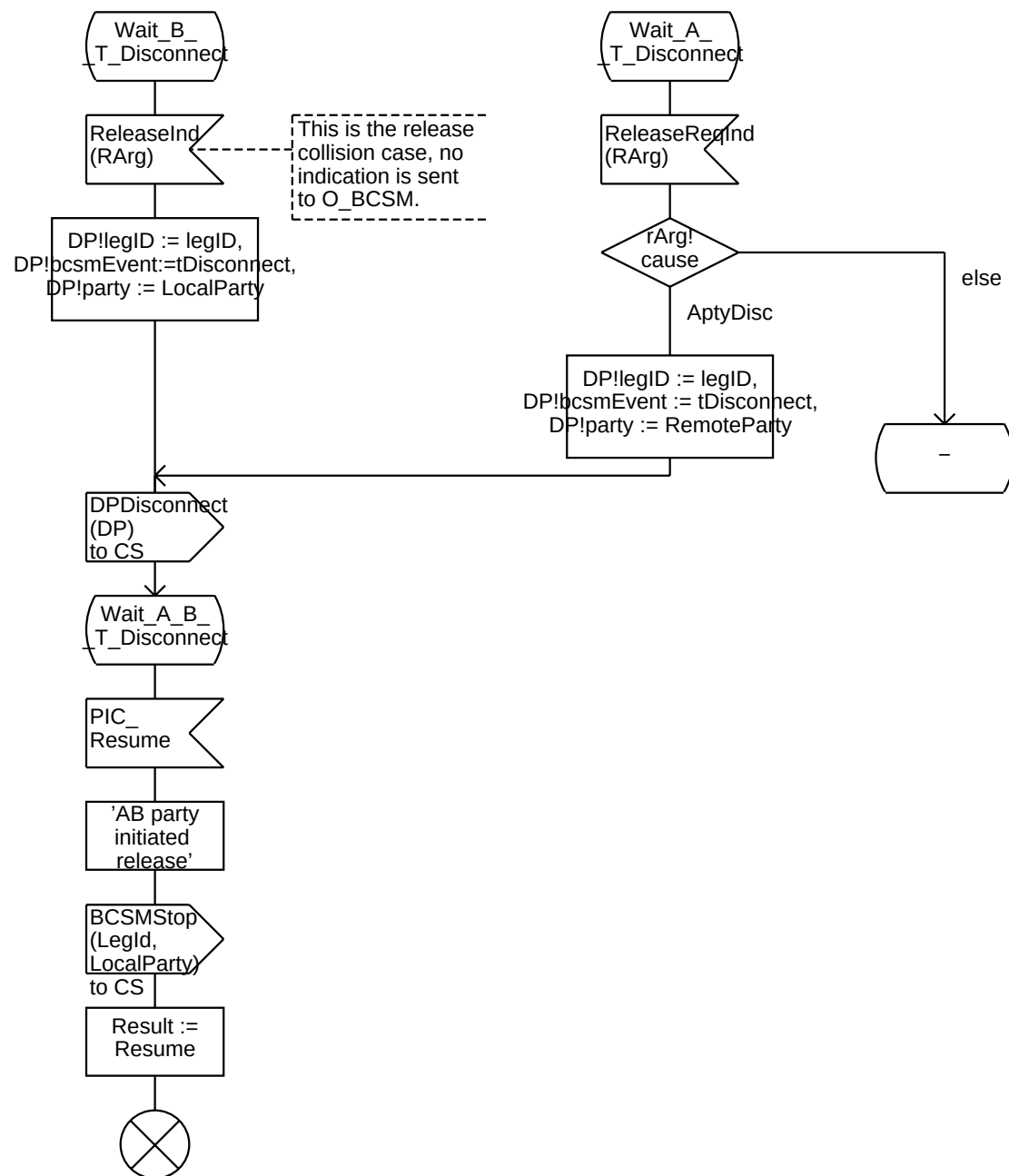
FPAR
IN/OUT Result DPResultType;
IN Party DPResultType;



Procedure DP_tDisconnect

3(7)

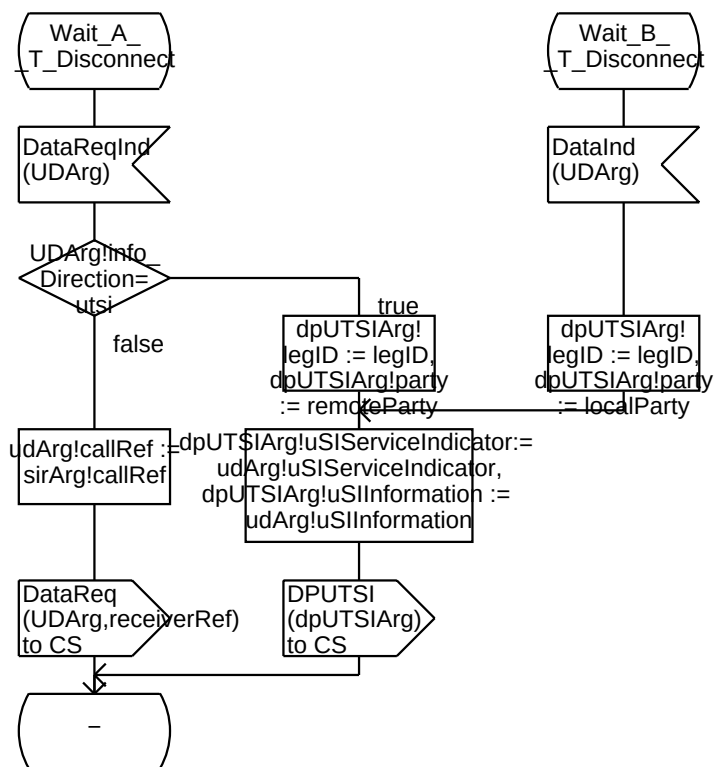
FPAR
IN/OUT Result DPResultType,
IN Party DPResultType;



Procedure DP_tDisconnect

4(7)

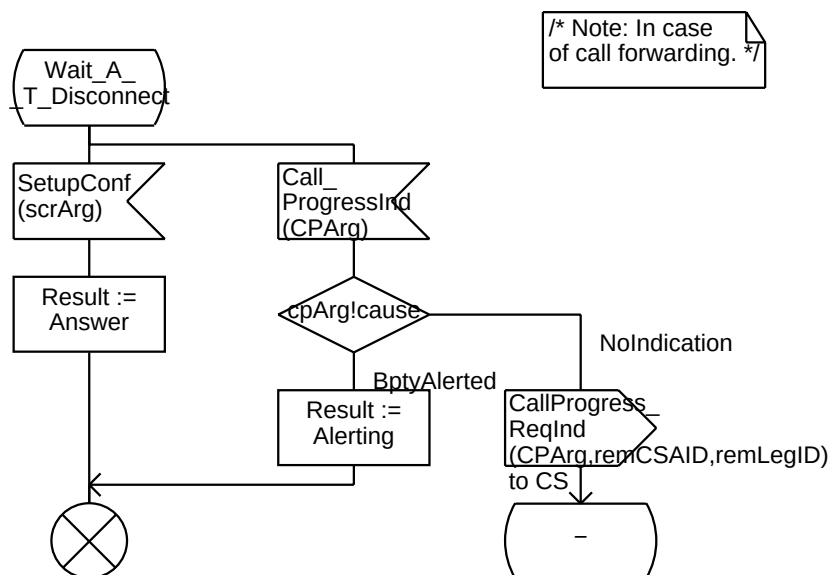
FPAR
IN/OUT Result DPResultType,
IN Party DPResultType;



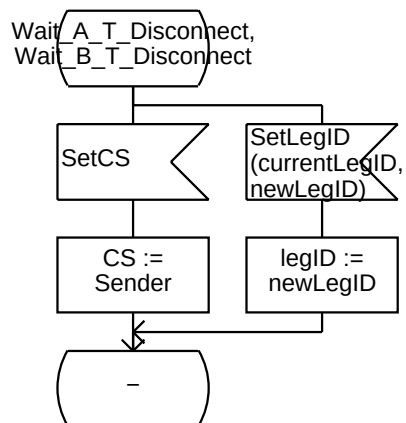
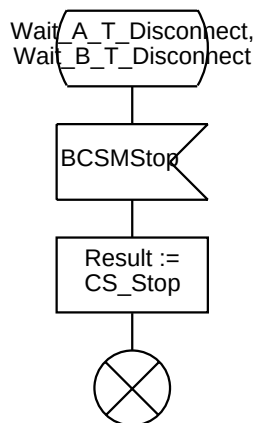
Procedure DP_tDisconnect

5(7)

FPAR
IN/OUT Result DPResultType,
IN Party DPResultType;



/* BCSMStop is sent by the CS/SSF-FSM to terminate the BCSM. */



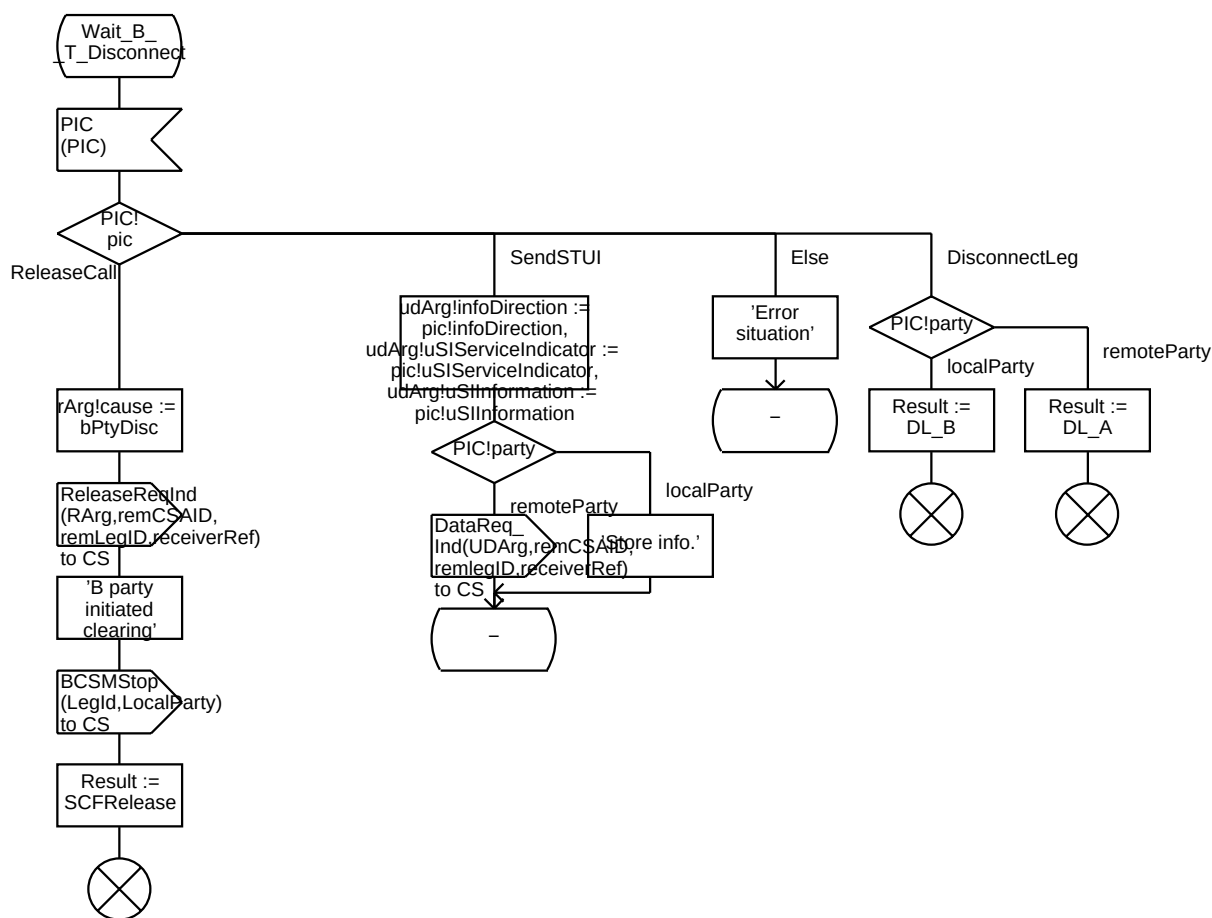
Procedure DP_tDisconnect

6(7)

```

;FPAR
IN/OUT Result DPResultType;
IN Party DPResultType;

```

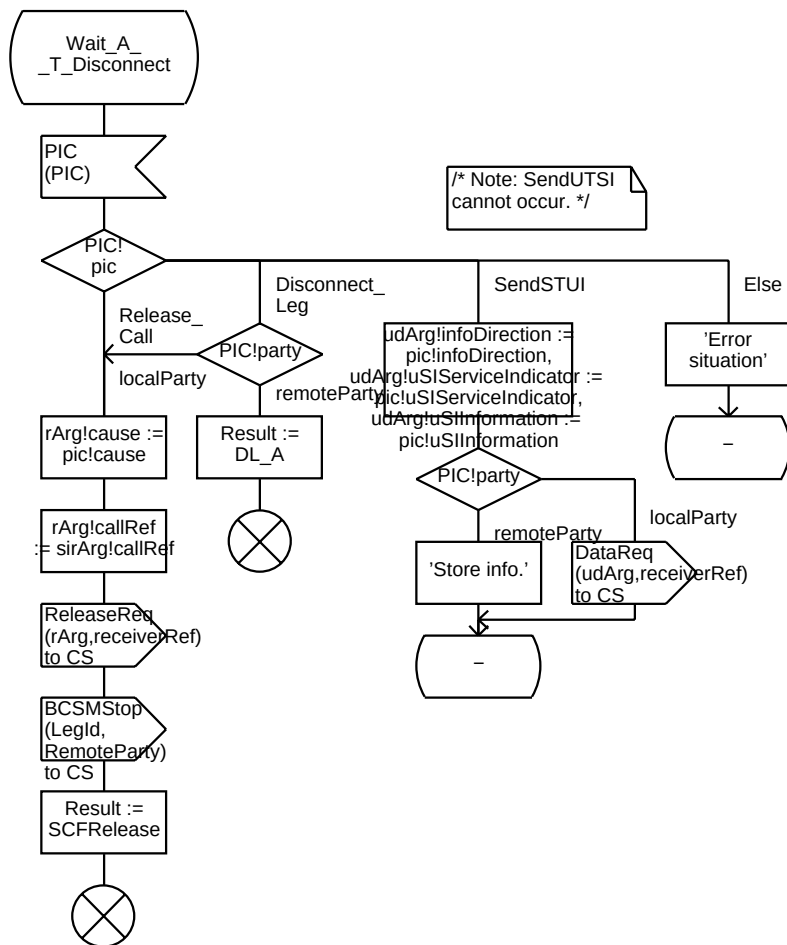


Procedure DP_tDisconnect

7(7)

```

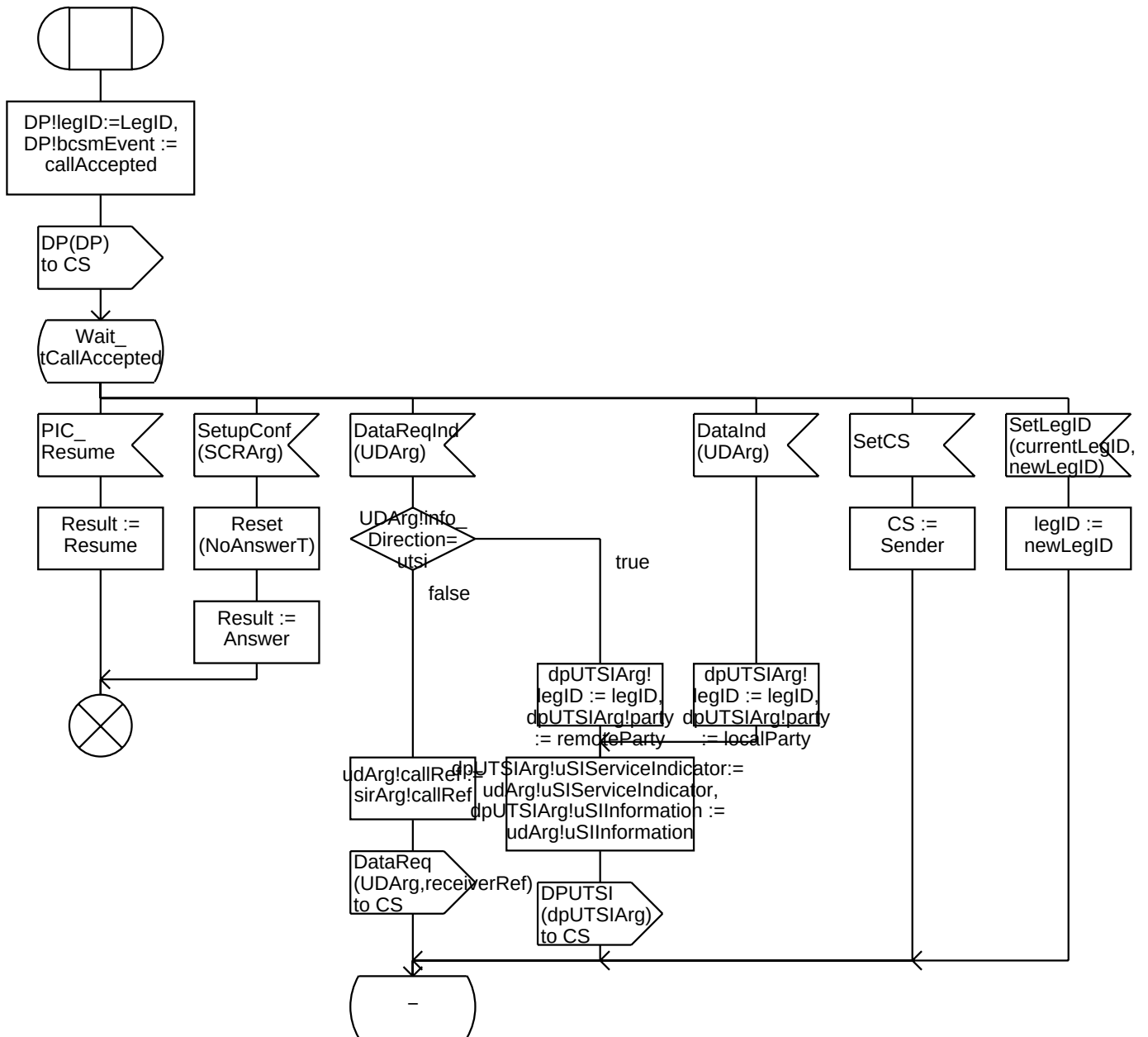
;FPAR
IN/OUT Result DPResultType;
IN Party DPResultType;
    
```



Procedure DP_tCall_Accepted

1(2)

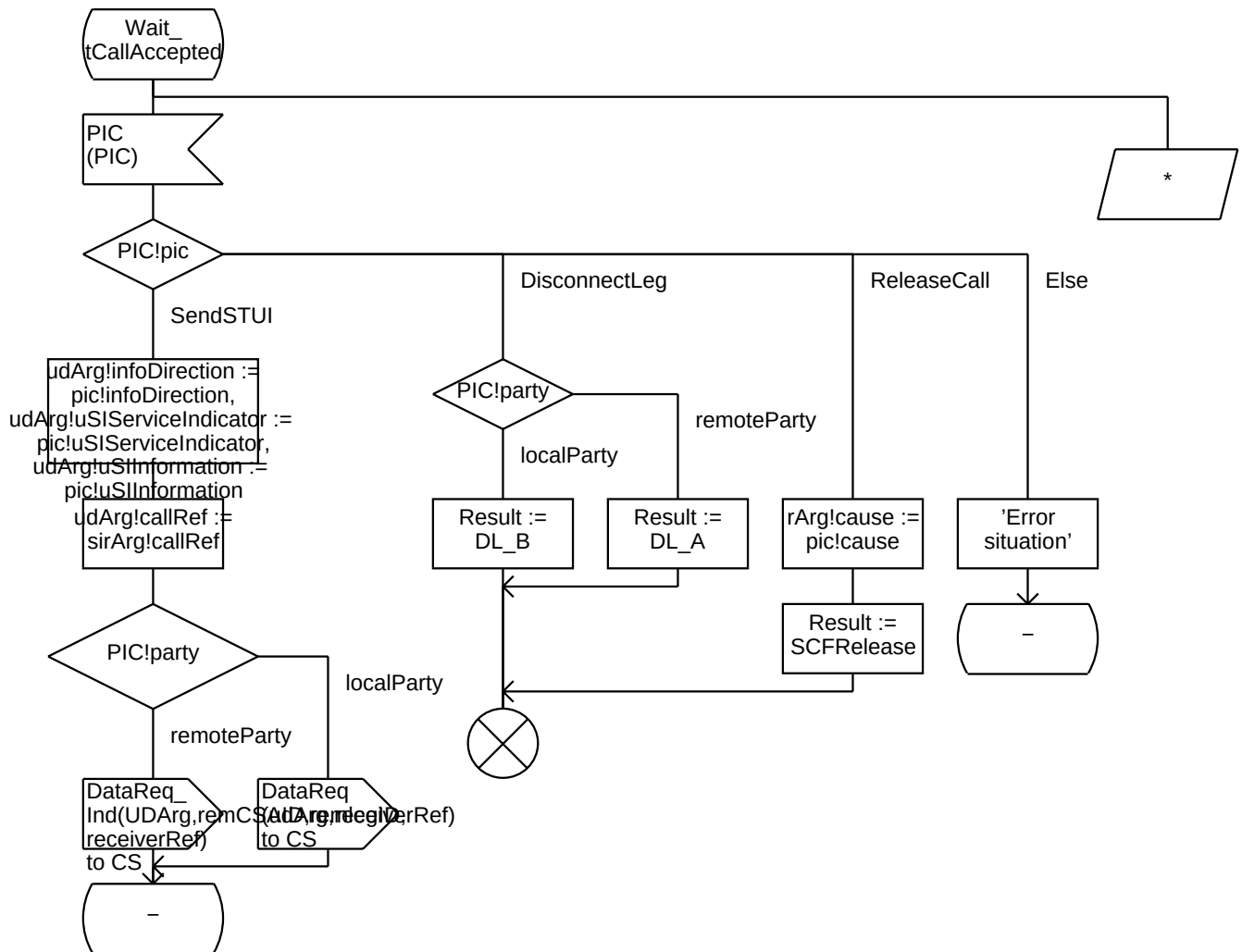
FPAR
IN/Out Result DPResultType;

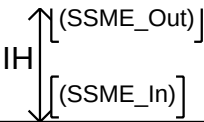


Procedure DP_tCall_Accepted

2(2)

FPAR
IN/Out Result DPResultType;





Process Type SSME_FSM

1(5)



DCL
/* Declaration of operation arguments. */
mtdArg ManageTriggerDataArg;

/* Procedures for processing of IN operations. */

Process_
ActivateService_
Filtering

Process_
CallGap

/* Procedures accessed by the SSF-FSM for
TDP, Service Filtering and Call Gap management. */

Arm_
TDPs

Initialise the TDPs
(SMF Service Feature
Provisioning). The procedure
is called by the SSF-FSM.

Matching
Service_
Filtering
Criteria

Determines if a service filtering
criteria is met.

CheckACG

Determines if call gapping
should be applied.

CallFiltered

Determines if a call
is filtered.

/* VARIABLE DECLARATIONS. */

DCL
/* Operation arguments. */
asfArg ActivateServiceFilteringArg,
cgArg CallGapArg,
sfrArg ServiceFilteringResponseArg,
invokeID Integer,

mgt_STTArg MGTSetTriggerTableArg,

/* Other variables. */
tdpTable EventType, /* The TDP table */
csaID CSAID := 1,
csID CallSegmentID := 1,
dID DialogIDtype;

/* Operations on the TDP table. */

Initialise_
TDPTable

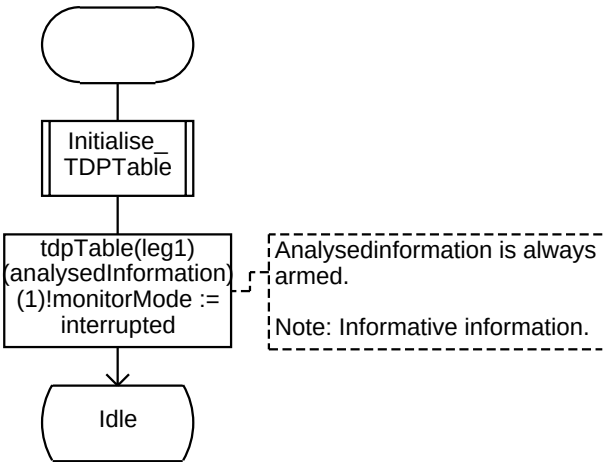
Mgt_Set_
TriggerTable

Process
ManageTriggerData

Processing of the ManageTriggerData
operation.

Process Type SSME_FSM

2(5)

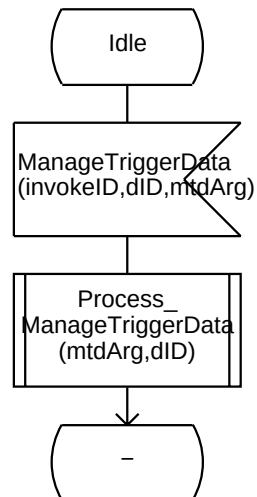
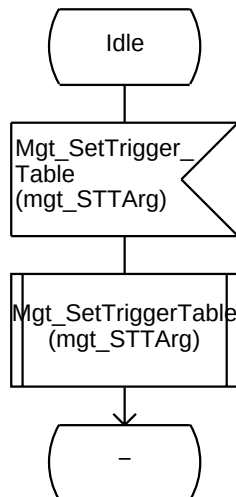
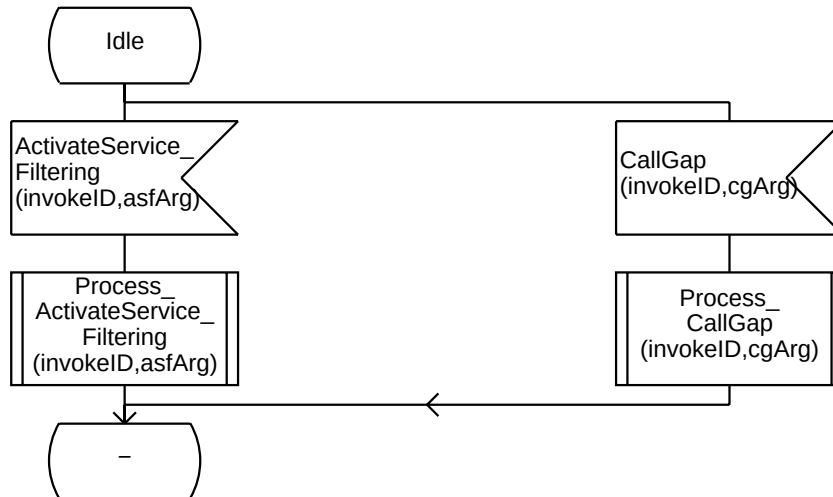


Process Type SSME_FSM

3(5)

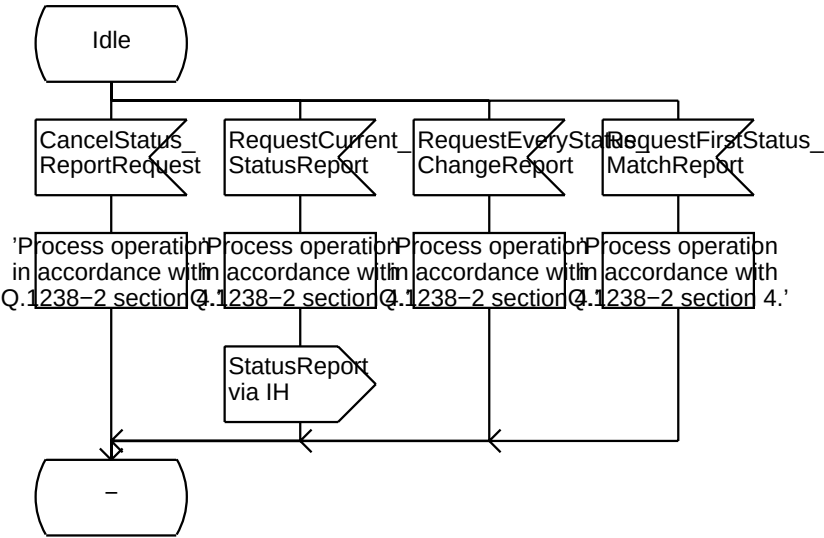


/* PROCESSING OF OPERATIONS. */



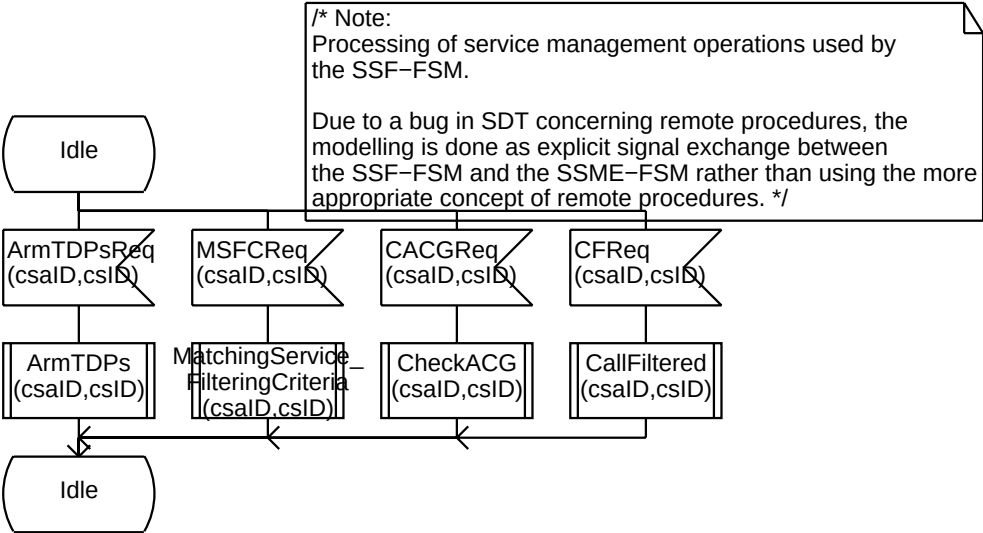
Process Type SSME_FSM

4(5)



Process Type SSME_FSM

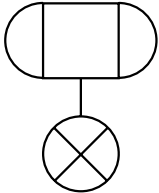
5(5)



Procedure ProcessActivateServiceFiltering

1(1)

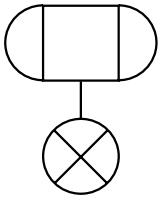
;FPAR
IN invokeID Integer,
IN asfArg ActivateServiceFilteringArg;
;



Procedure ProcessCallGap

1(1)

;FPAR
IN invokeID Integer,
IN cgArg CallGapArg;



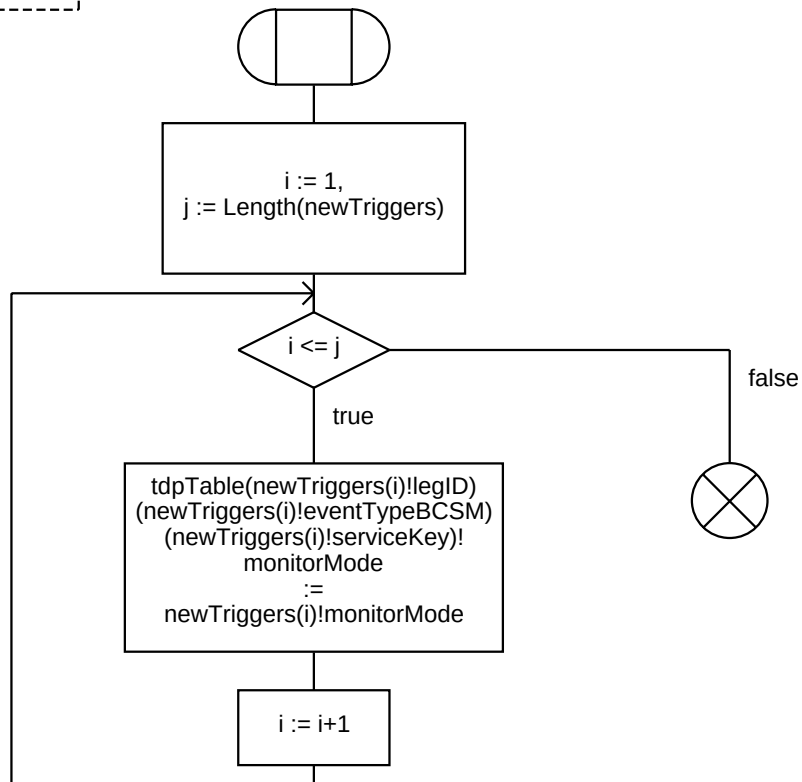


Procedure Mgt_SetTriggerTable

1(1)

FPAR
IN newTriggers MGTSetTriggerTableArg;

DCL
i,j Integer;

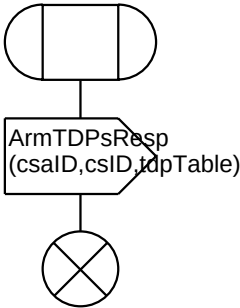


Procedure <<Process Type SSME_FSM>> ArmTDPs

1(1)

;fpar
in csaID CSAID,
in csID CallSegmentID;

/* This procedure is called
by the SSF-FSM to arm the TDPs for a
given call. */



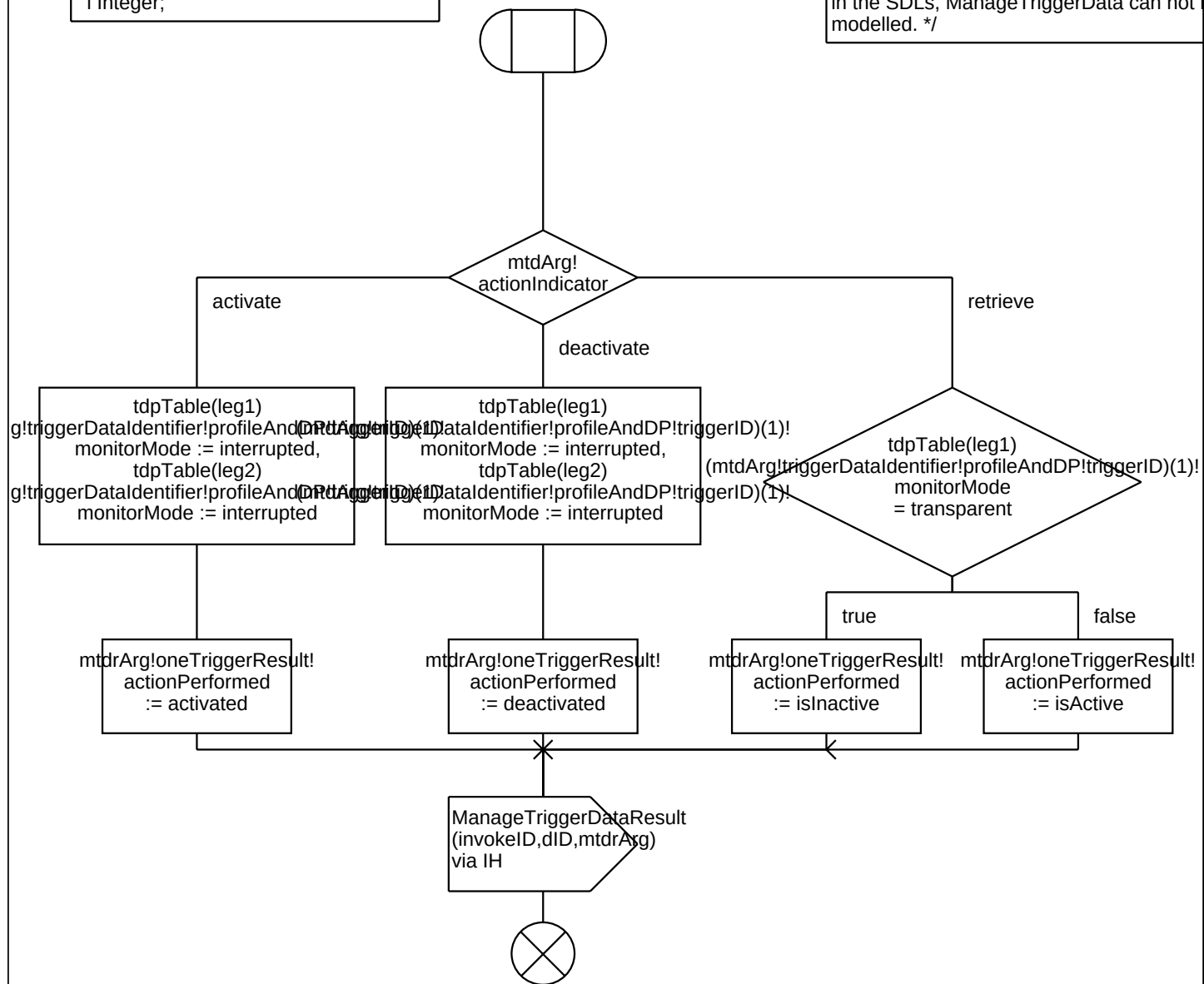
Procedure ProcessManageTriggerData

1(2)

FPAR
IN mtdArg ManageTriggerDataArg,
IN dID DialogIDtype;

DCL
mtdrArg ManageTriggerDataResultArg,
i Integer;

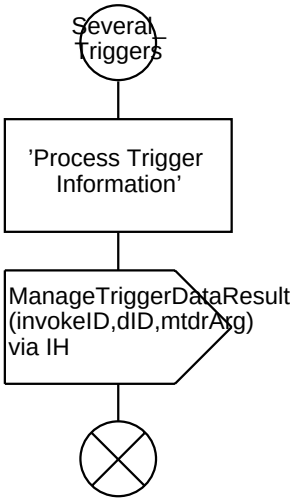
/* Note:
Due to the simplified modelling of the trigger
in the SDLs, ManageTriggerData can not be
modelled. */



Procedure ProcessManageTriggerData

2(2)

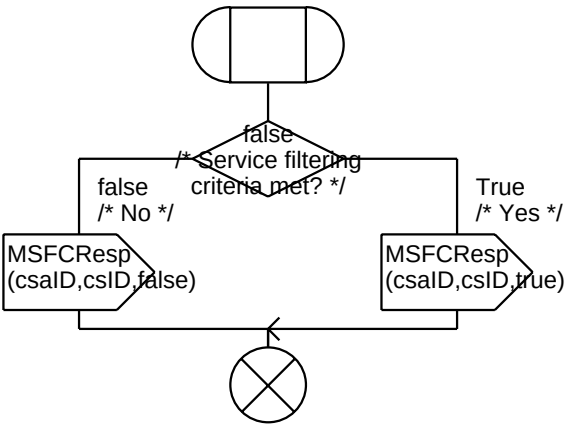
;FPAR
IN mtdArg ManageTriggerDataArg,
IN dID DialogIDtype;



Procedure <<Process Type SSME_FSM>> MatchingServiceFilteringCriteria

1(1)

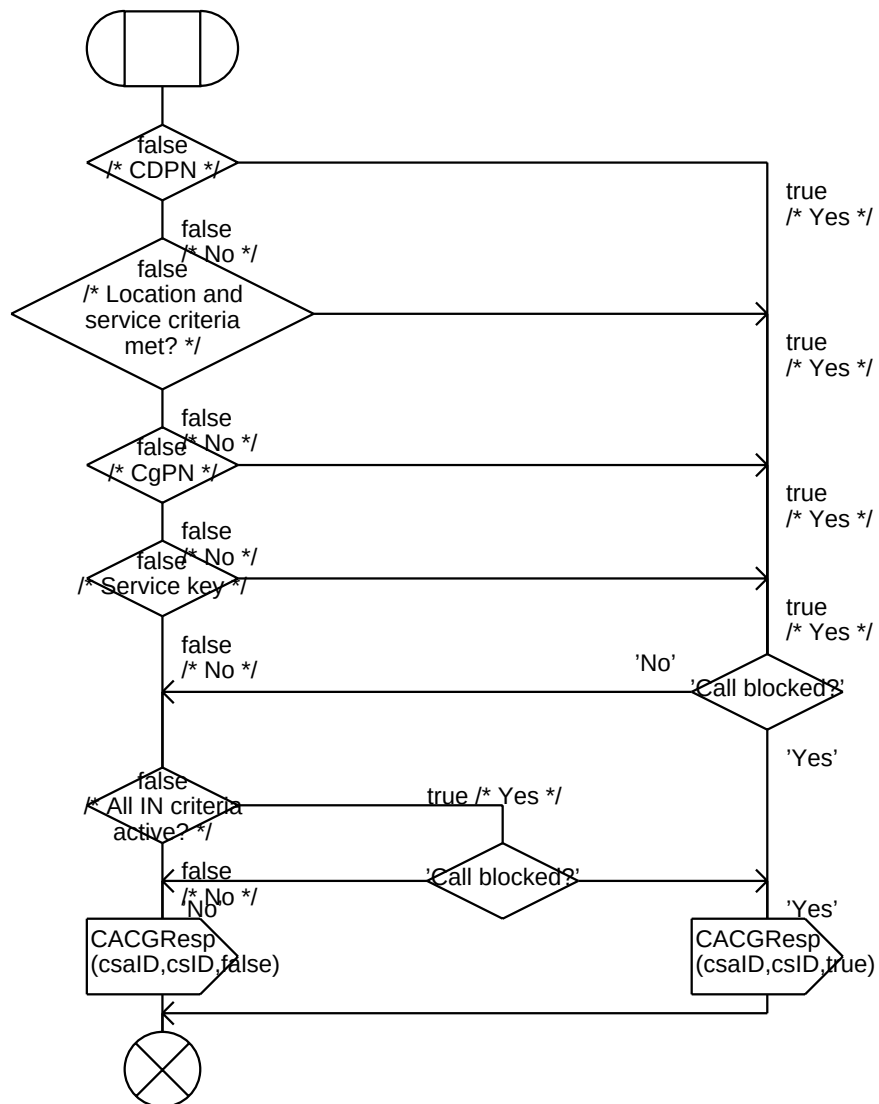
;par
in csalD CSAID,
in csID CallSegmentID;



Procedure <<Process Type SSME_FSM>> CheckACG

1(1)

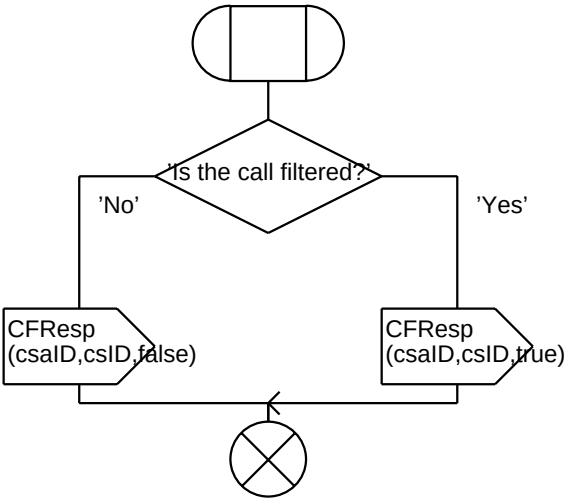
par
in csalD CSAID,
in csID CallSegmentID;



Procedure <<Process Type SSME_FSM>> CallFiltered

1(1)

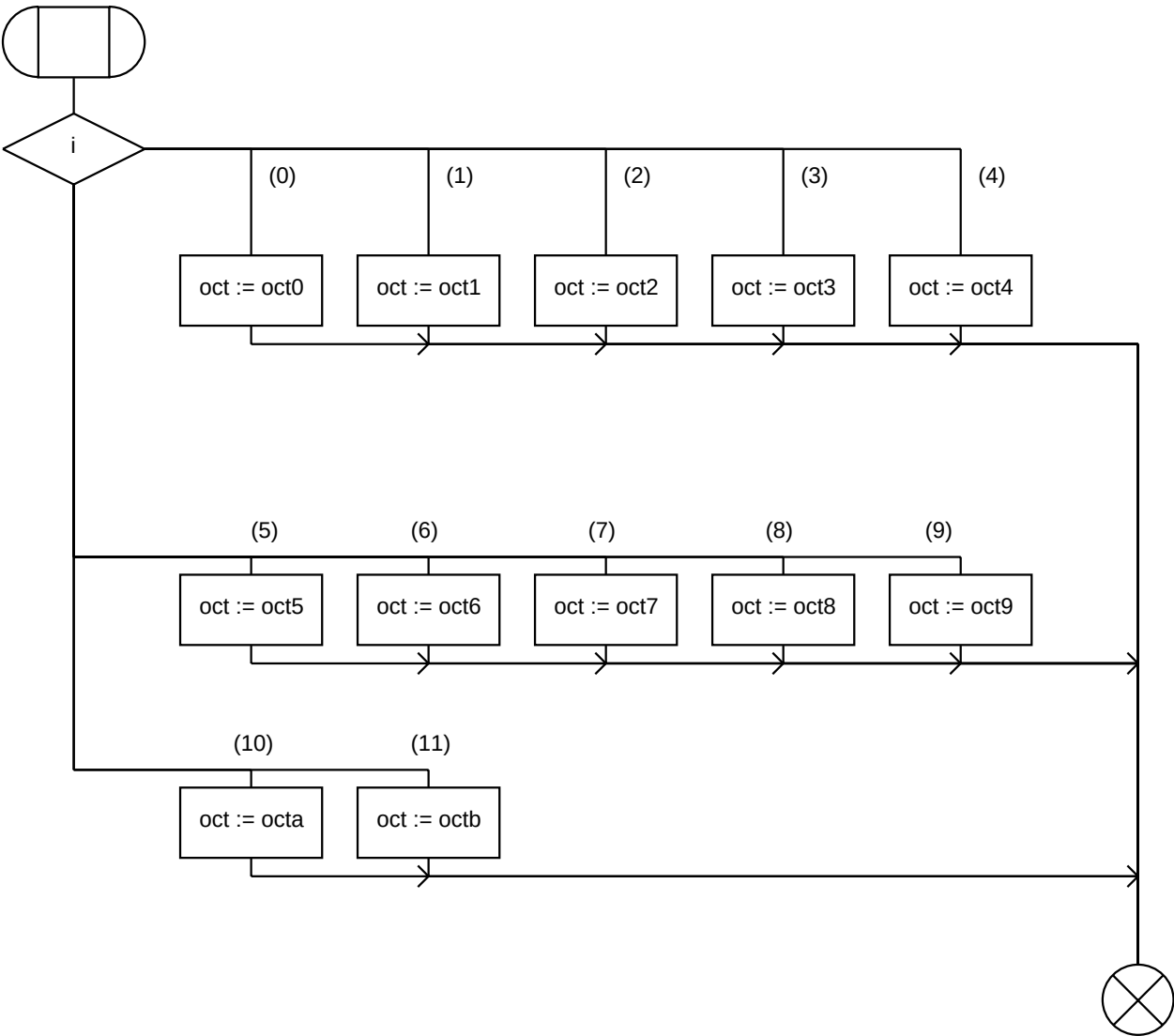
;fpar
in csalD CSAID,
in csID CallSegmentID;



Operator Int2Oct

1(1)

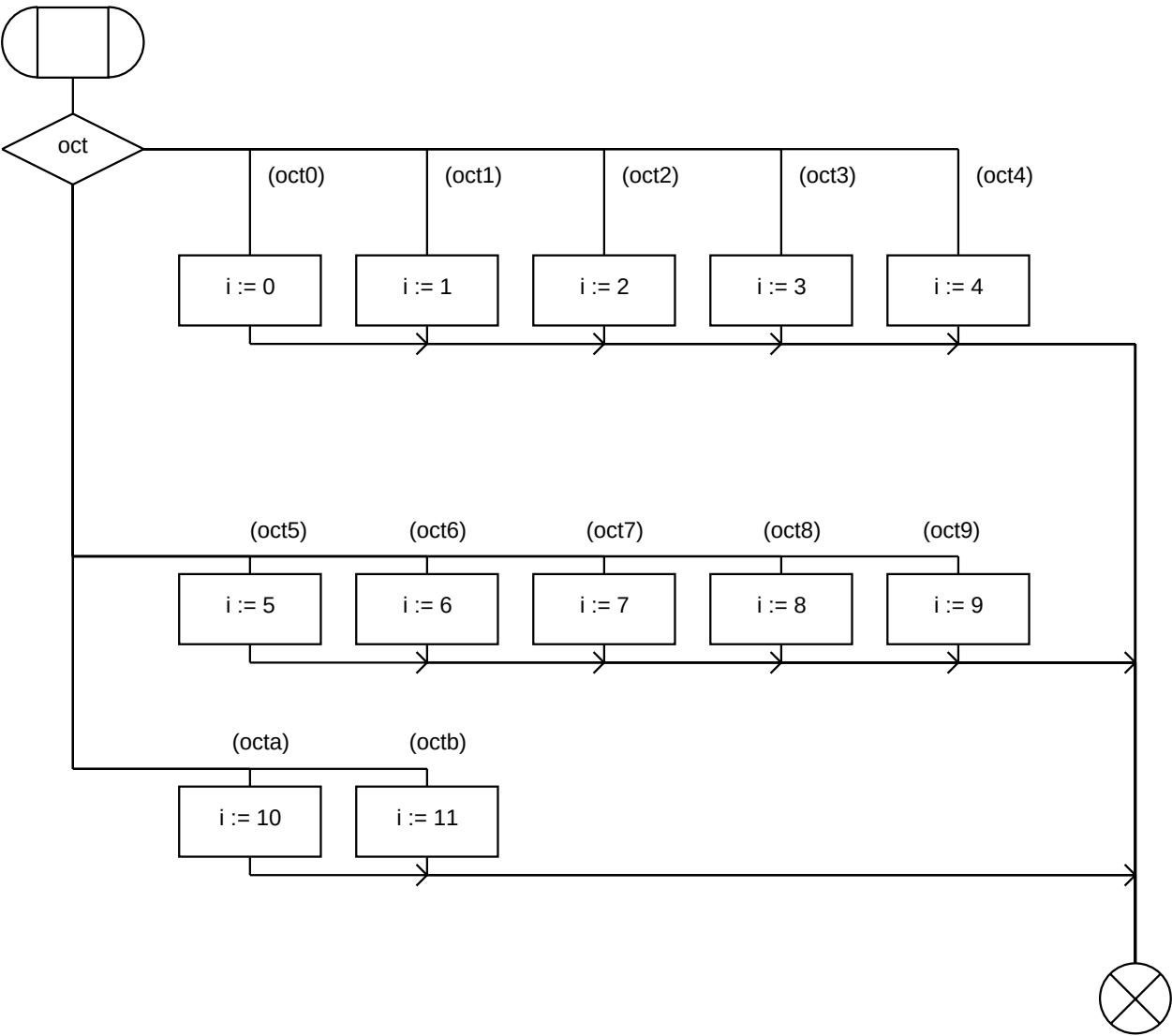
;FPAR i Integer;
RETURNS oct Octet_String;

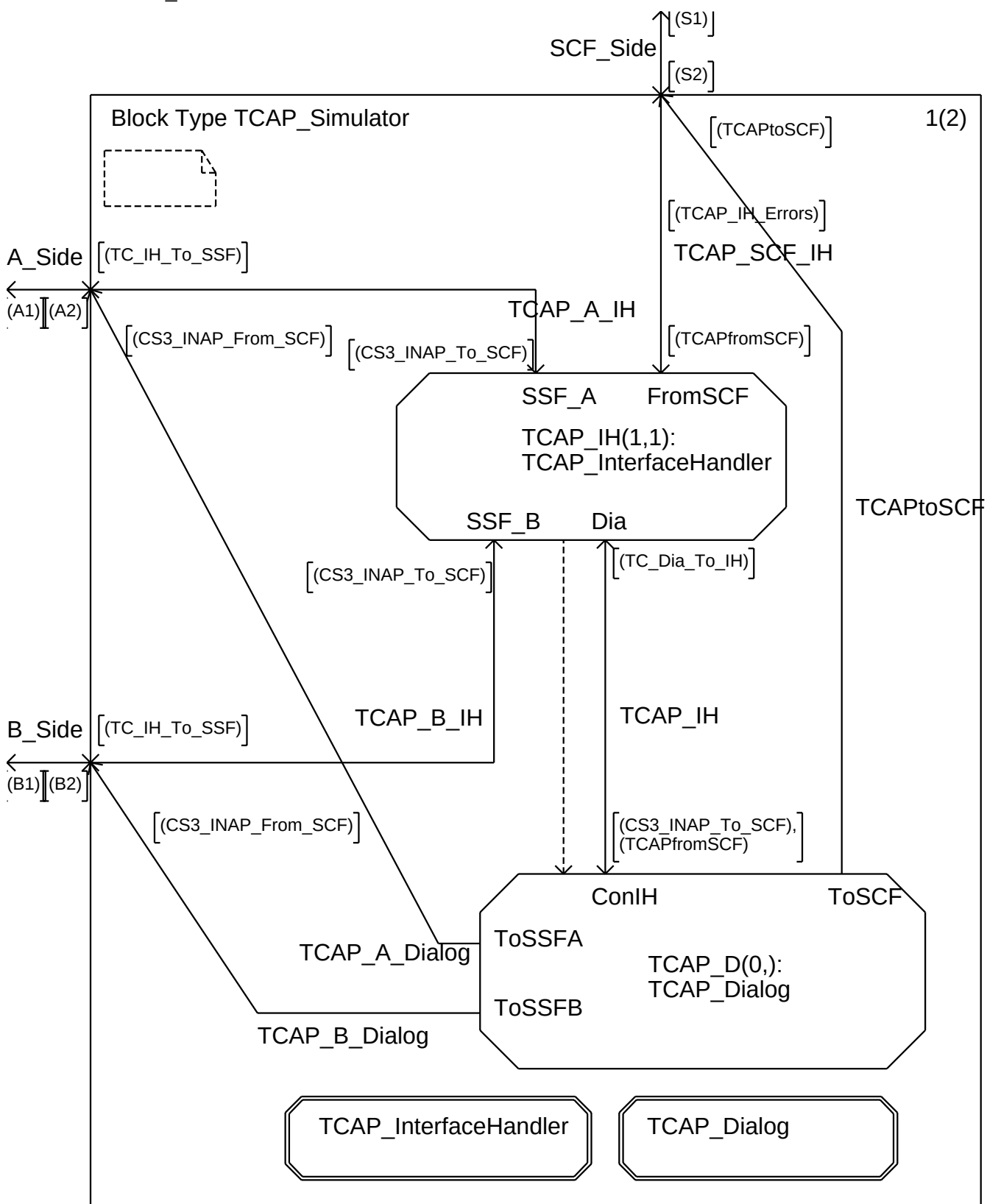


Operator Oct2Int

1(1)

;FPAR oct Octet_String;
RETURNS i INTEGER;





Block Type TCAP_Simulator

2(2)



/* Datatypedefinitions used locally in this block type */

/* Used to describe the status of a dialog */

```
NEWTYPE DialogStatusType  
  LITERALS free, busy, csaPending, used;  
ENDNEWTYPE;
```

/* Signal- and Signallistdefinitions used locally in this block type */

```
/* A Dummy Signal */  
SIGNAL  
  ReleaseDialog(DialogIDtype);
```

```
/* TCAP dialogs to TCAP interface */  
SIGNALLIST TC_Dia_To_IH =  
  ReleaseDialog;
```

Process Type TCAP_InterfaceHandler

1(15)



```
NEWTYPE DialogType
STRUCT
  status DialogStatusType;
  csa CSAID;
  dialogHandler PId;
OPERATORS newDialog: DialogStatusType, CSAID, PId -> DialogType;
OPERATOR newDialog REFERENCED;
ENDNEWTYPE;

/* Remark: Type CSAID (= Integer) is defined in ASN.1 file 'CS3i.asn' */

NEWTYPE AllDialogsType
ARRAY (DialogIDtype, DialogType); /* DialogID is used as index in the array */
OPERATORS
  initDialogs: Boolean -> allDialogsType;
  getDialogID: CSAID, AllDialogsType -> DialogIDtype;
  nextFreeDialogID: AllDialogsType -> DialogIDtype;
OPERATOR initDialogs REFERENCED;
OPERATOR getDialogID REFERENCED;
OPERATOR nextFreeDialogID REFERENCED;
ENDNEWTYPE;
```

operator

newDialog

operator

initDialogs

operator

getDialogID

operator

nextFreeDialogID

```
NEWTYPE IHresourceType
ARRAY (IHroleType,PId); /* IHroleType (= A_Side or B_Side) is defined on system level */
ENDNEWTYPE;
```

Process Type TCAP_InterfaceHandler

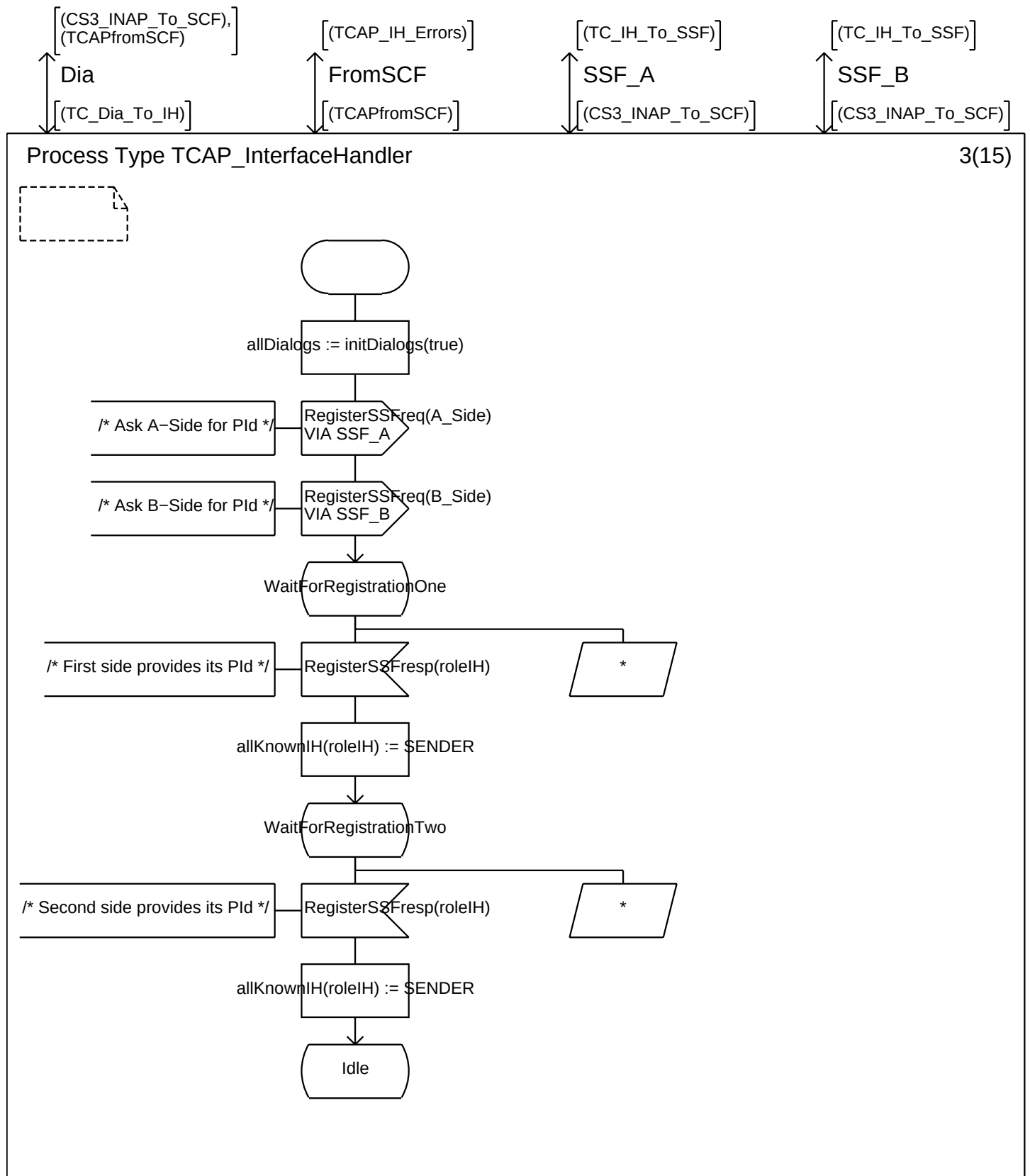
2(15)



```
DCL allKnownIH IHresourceType;  
DCL roleIH IHroleType;  
  
DCL allDialogs AllDialogsType;
```

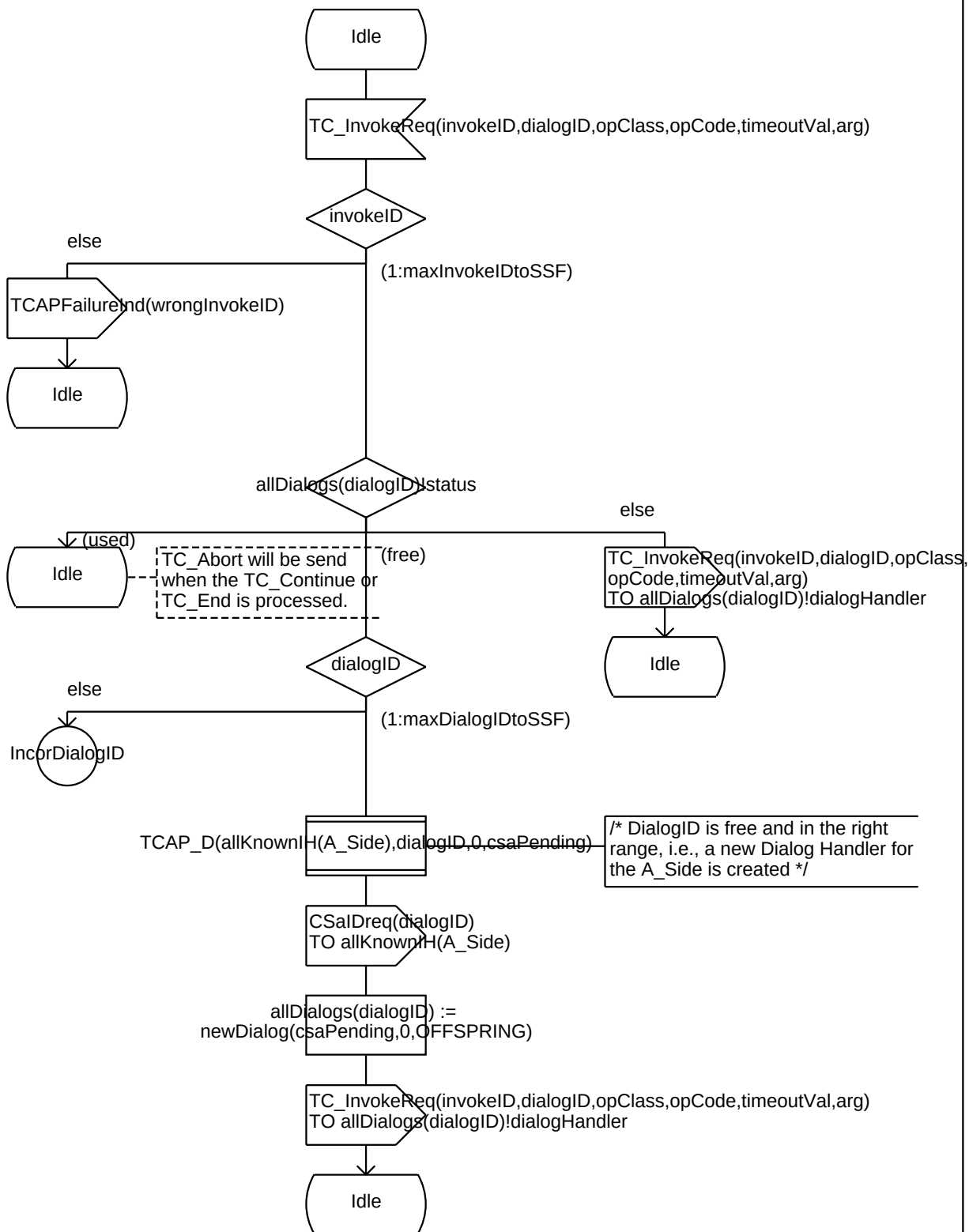
```
DCL invokeID InvokeIDtype;  
DCL dialogID DialogIDtype;  
DCL opClass OpClassType;  
DCL opCode OpCodeType;  
DCL timeoutVal TimeoutValType;
```

```
DCL arg ArgType; /* refers to ASN.1 Definition in INCS3BundleArg */  
DCL errArg errorArg;  
DCL compEnd Boolean;
```



Process Type TCAP_InterfaceHandler

4(15)

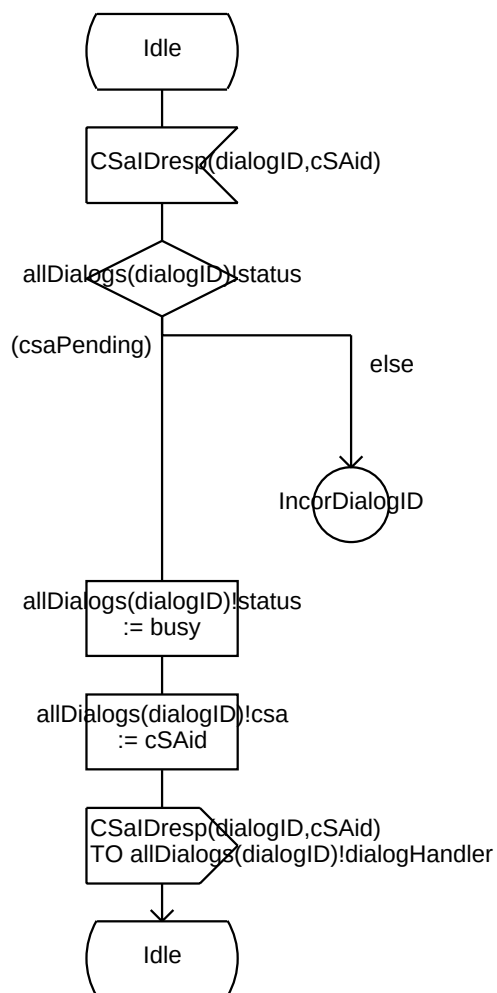
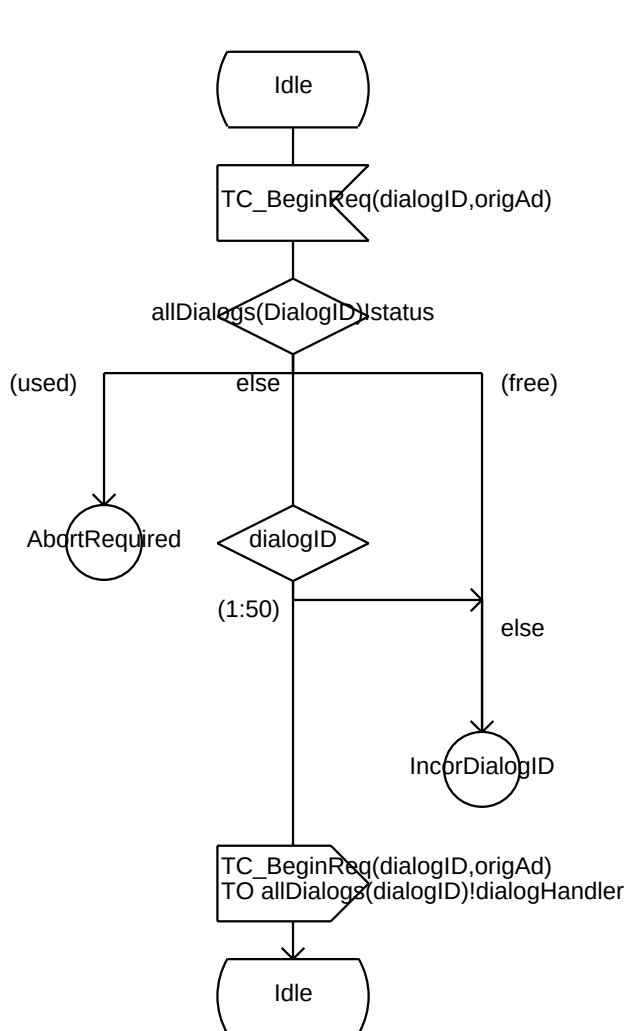


Process Type TCAP_InterfaceHandler

5(15)

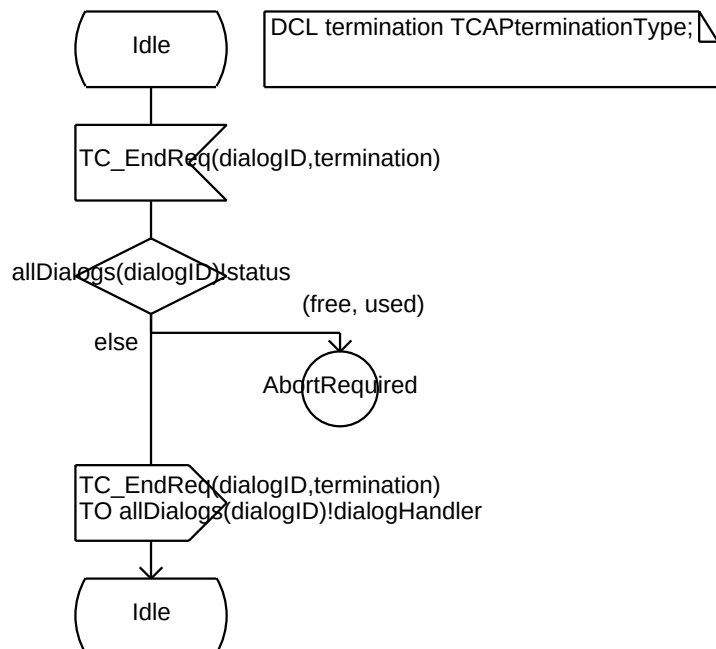
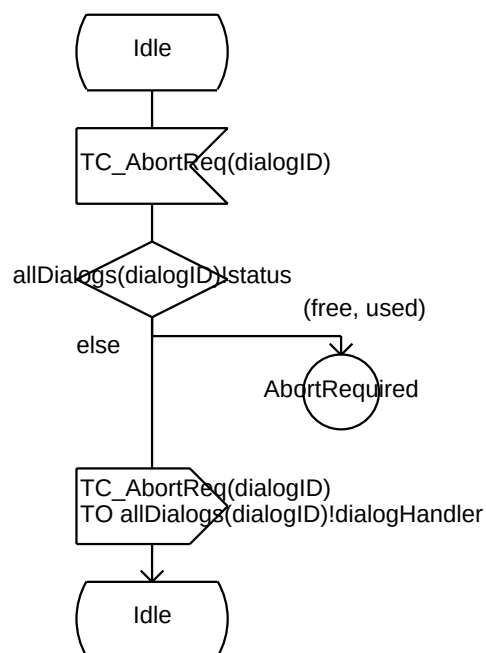
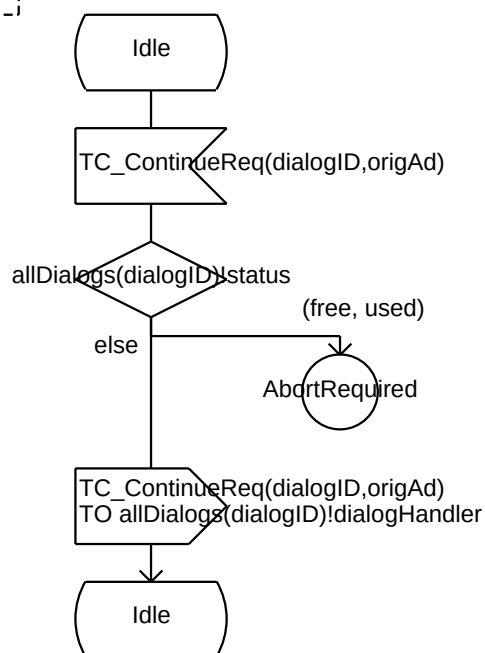


DCL origAd TCoriginType;
DCL cSAid CSAID;



Process Type TCAP_InterfaceHandler

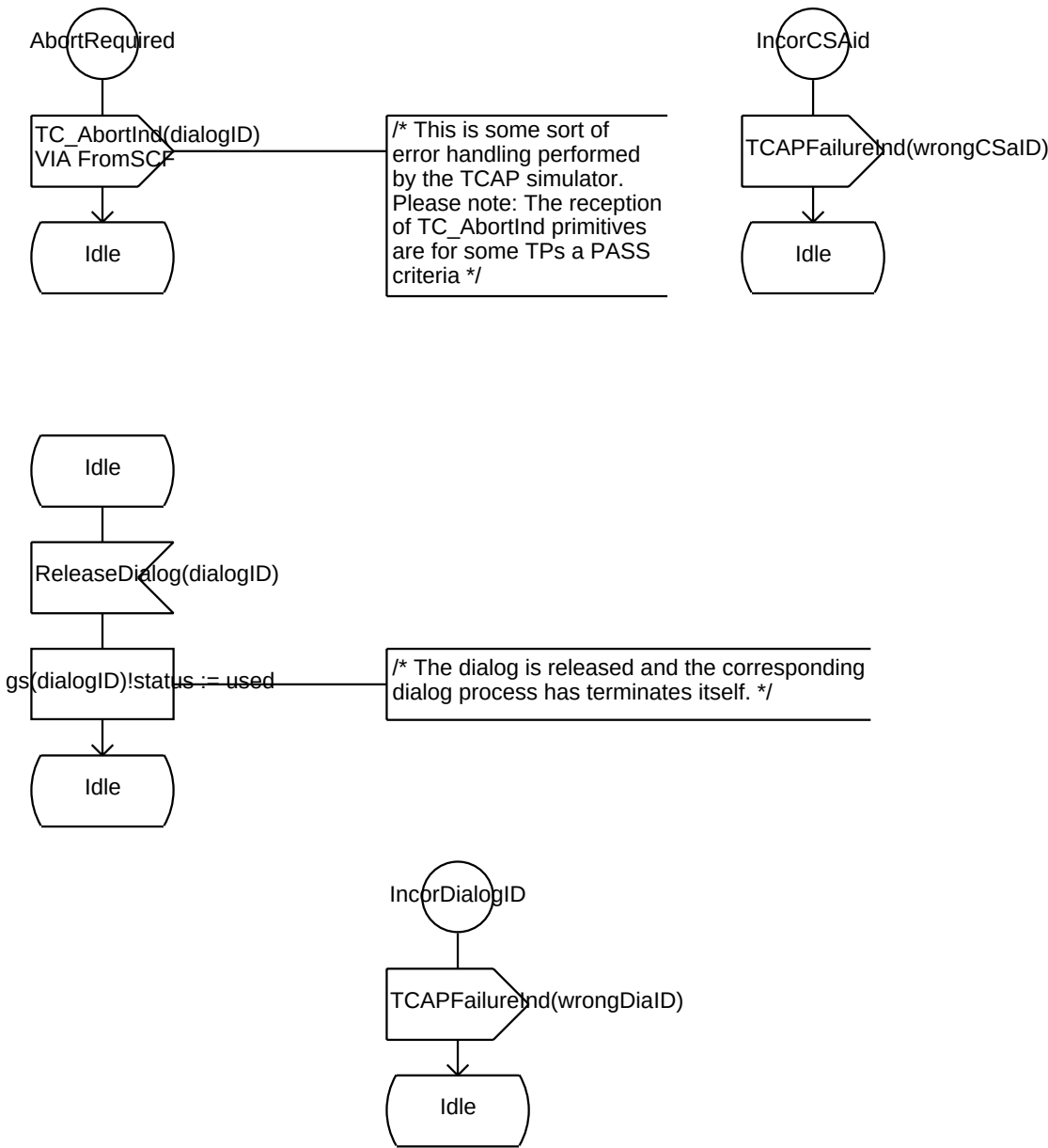
6(15)



DCL termination TCAPterminationType;

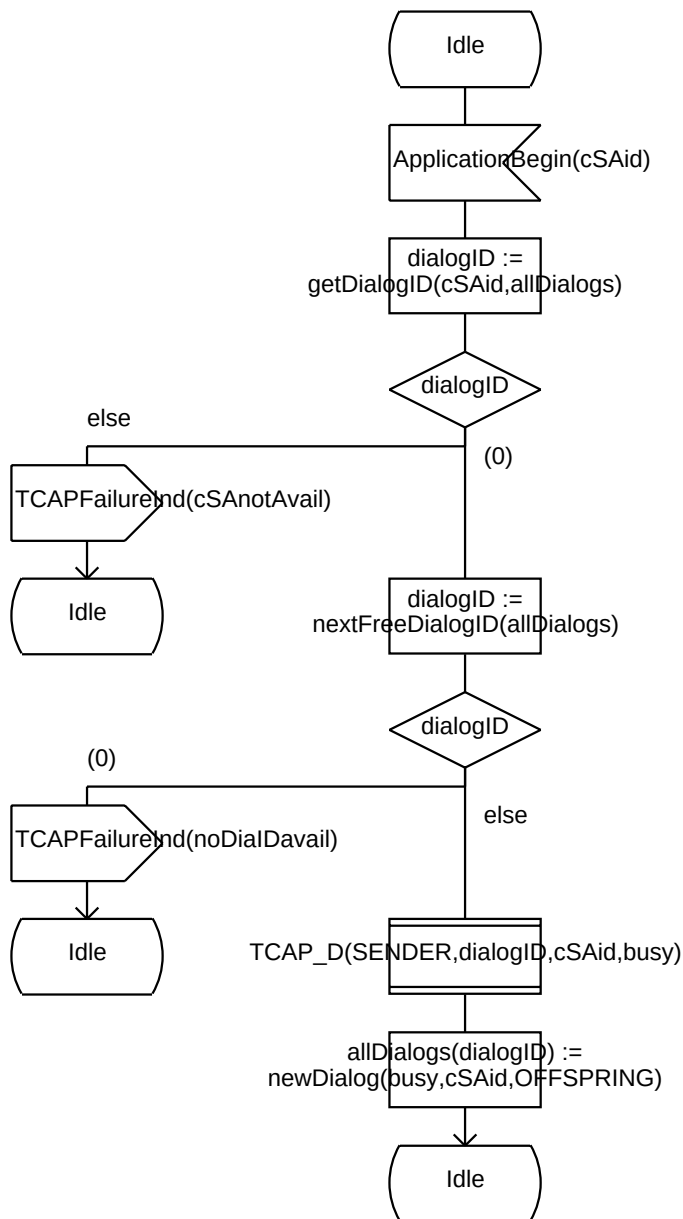
Process Type TCAP_InterfaceHandler

7(15)



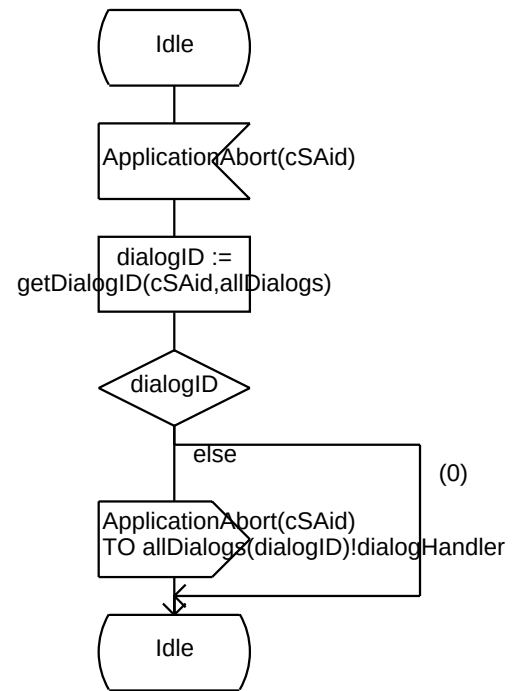
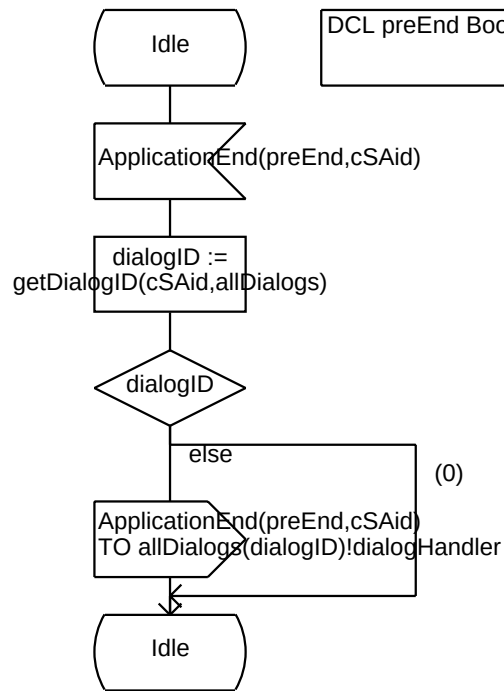
Process Type TCAP_InterfaceHandler

8(15)



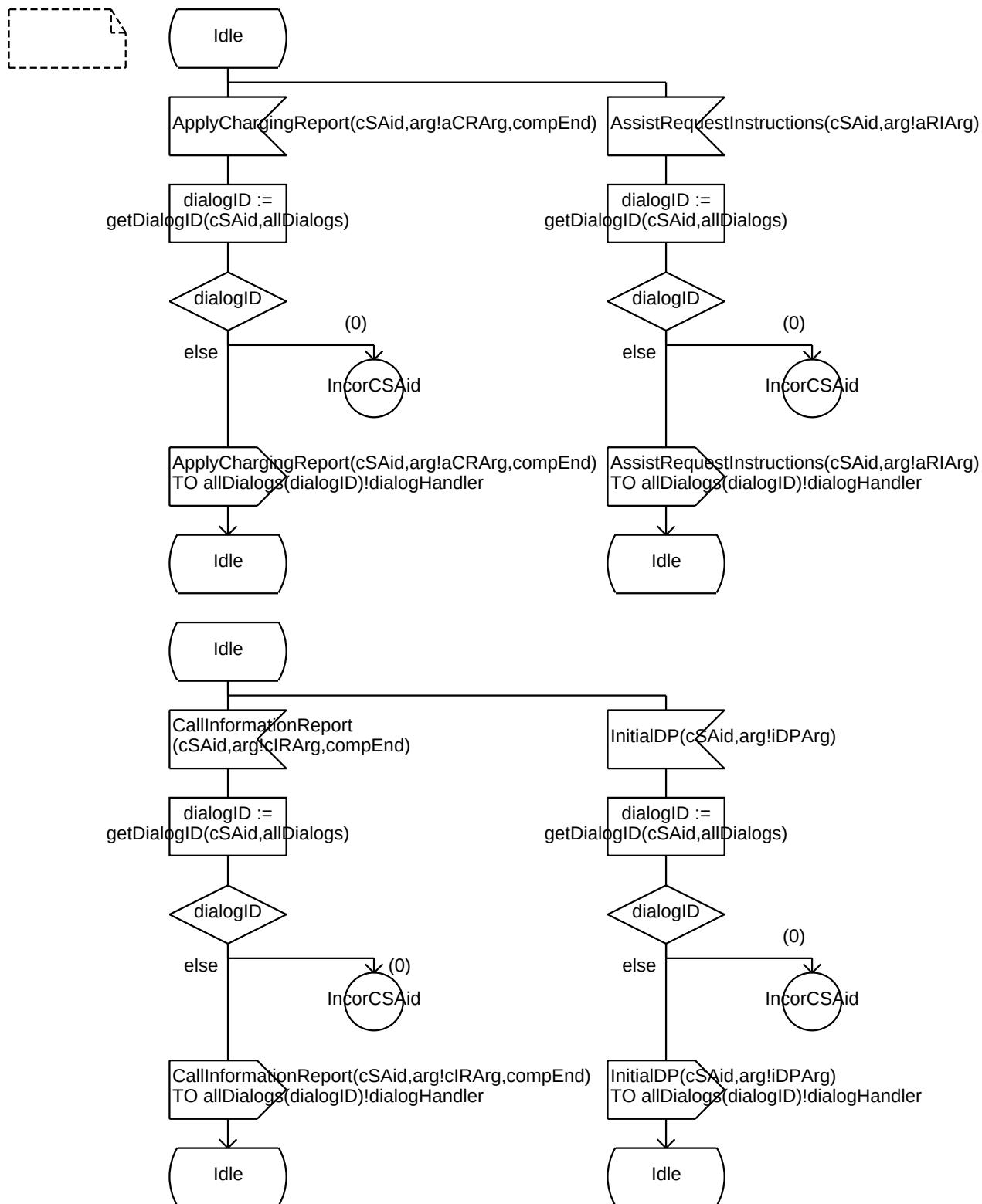
Process Type TCAP_InterfaceHandler

9(15)



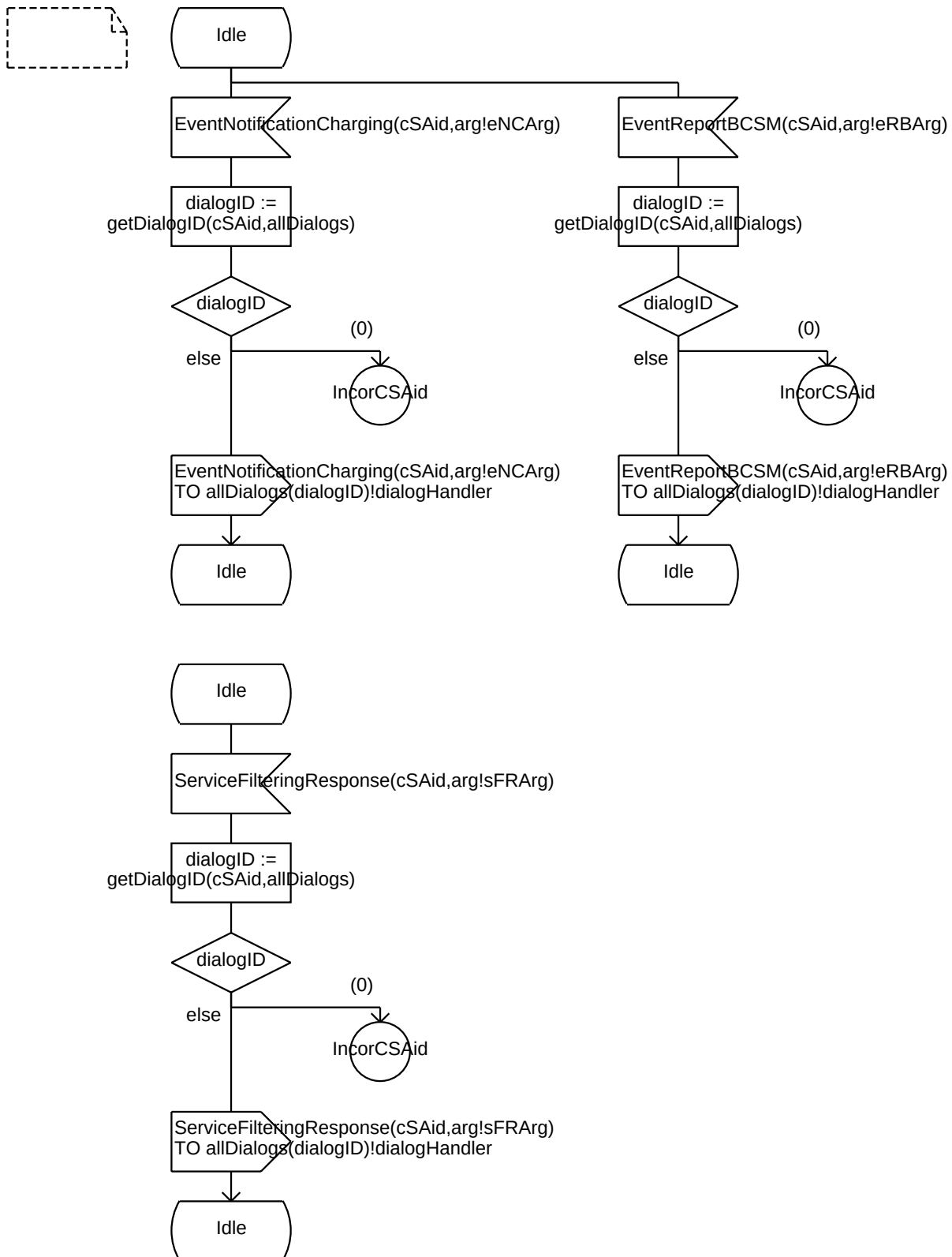
Process Type TCAP_InterfaceHandler

10(15)



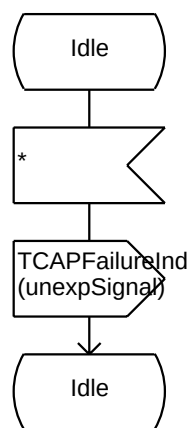
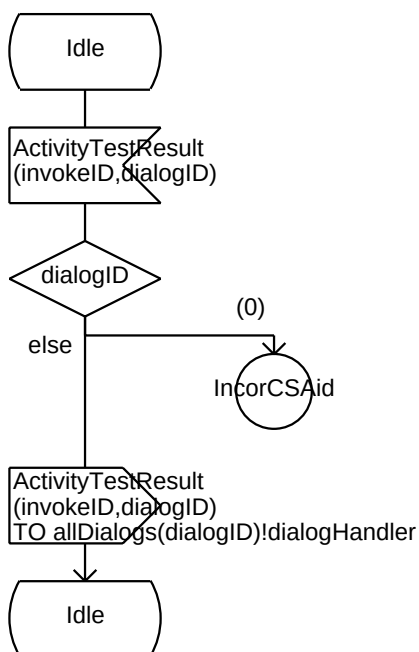
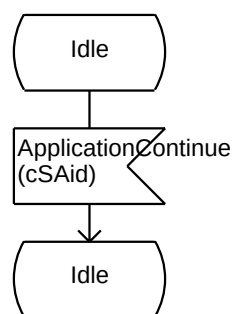
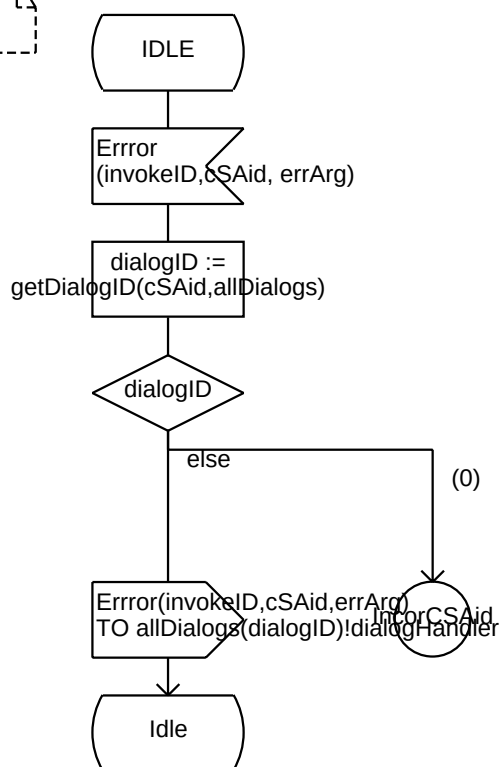
Process Type TCAP_InterfaceHandler

11(15)



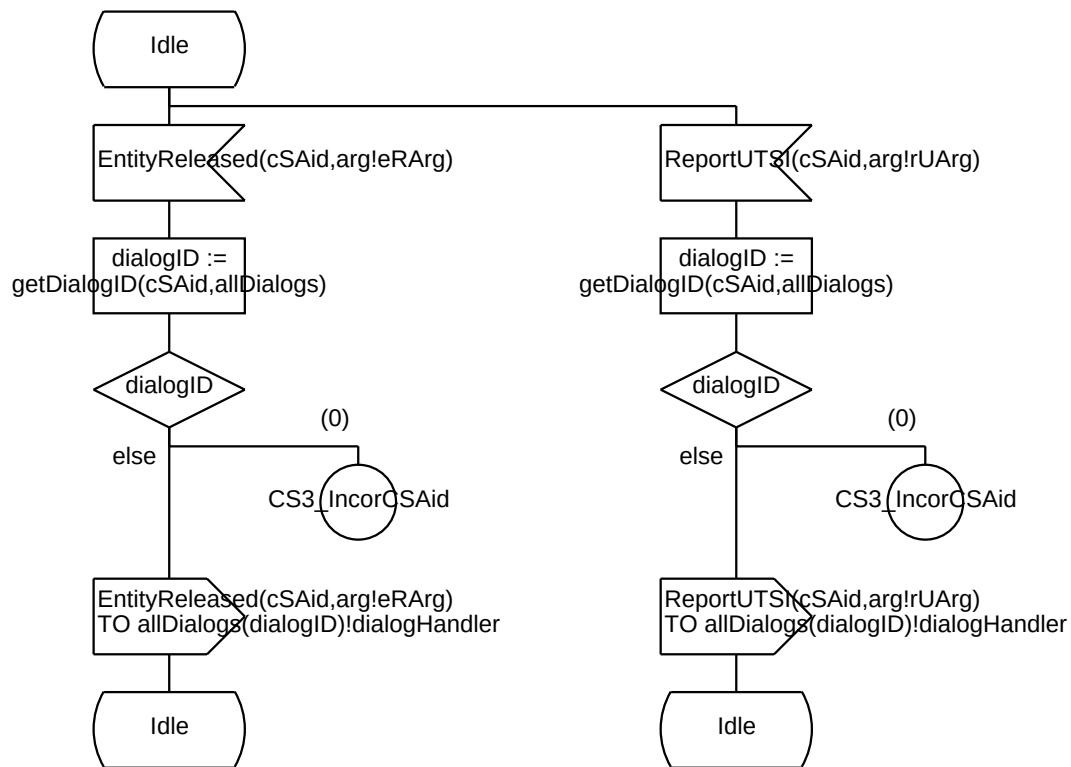
Process Type TCAP_InterfaceHandler

12(15)



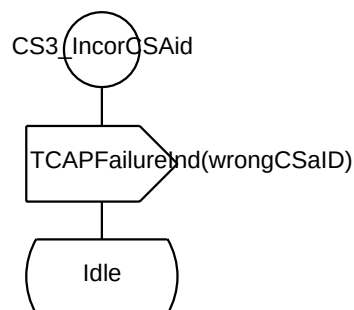
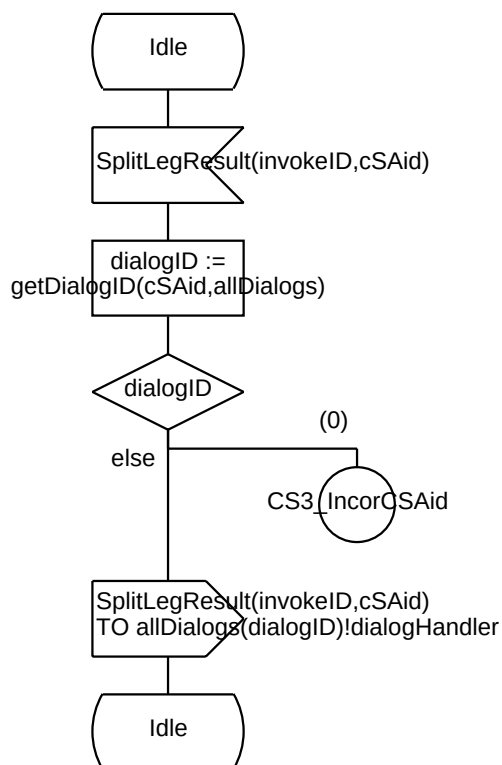
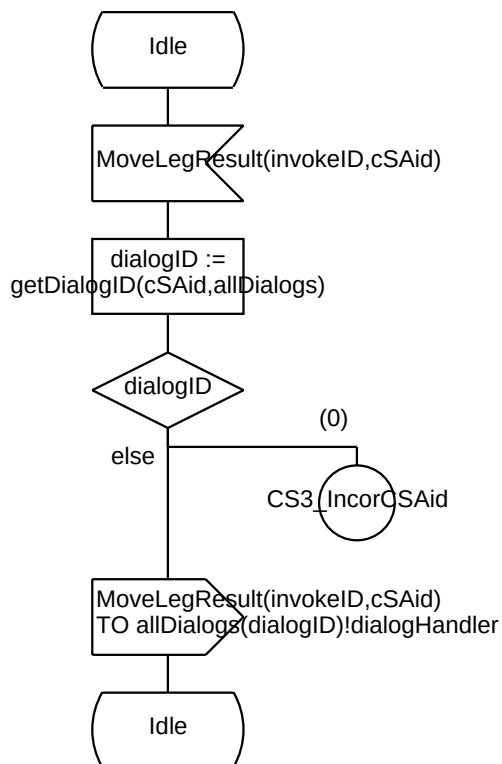
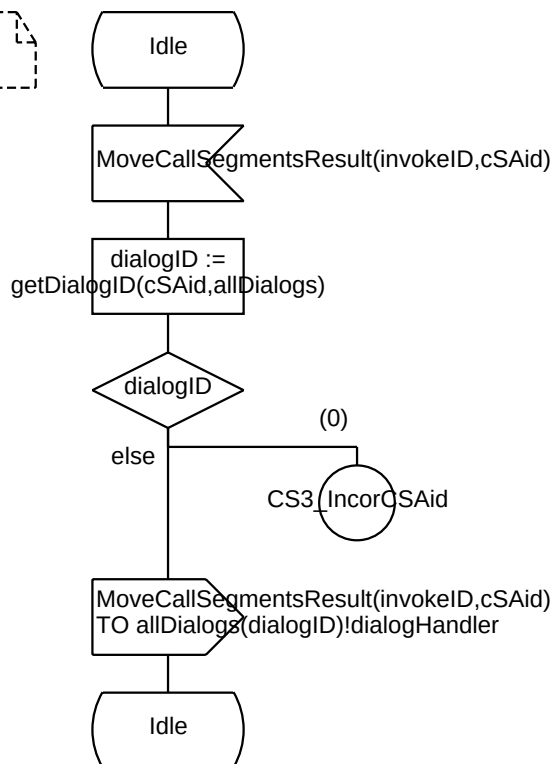
Process Type TCAP_InterfaceHandler

13(15)



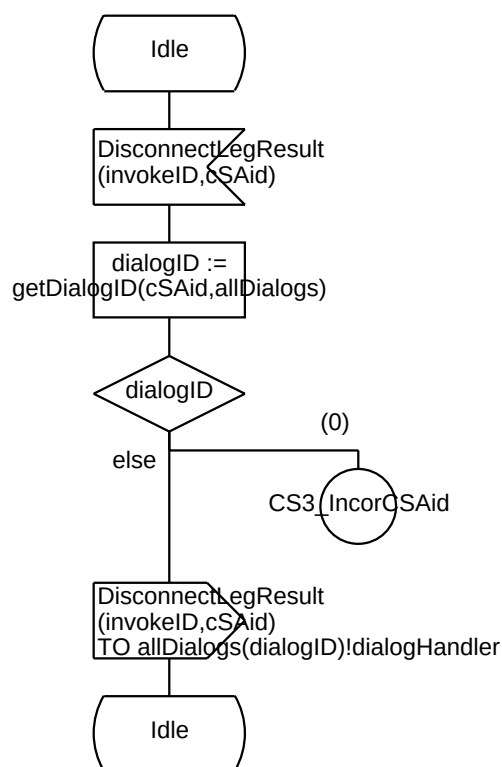
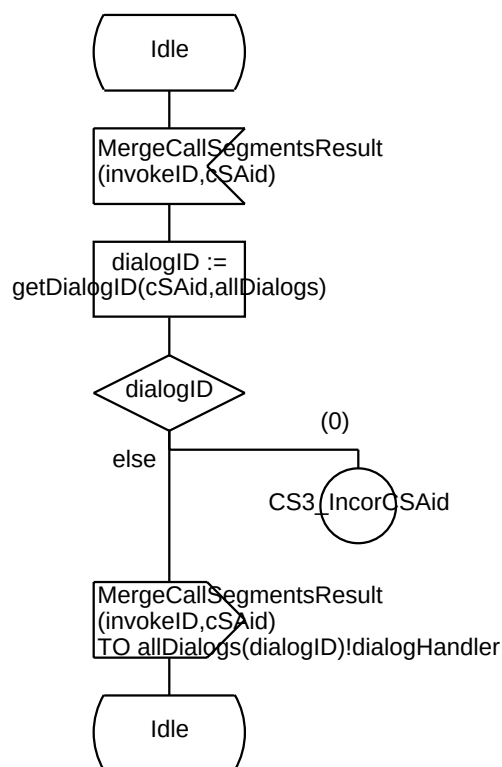
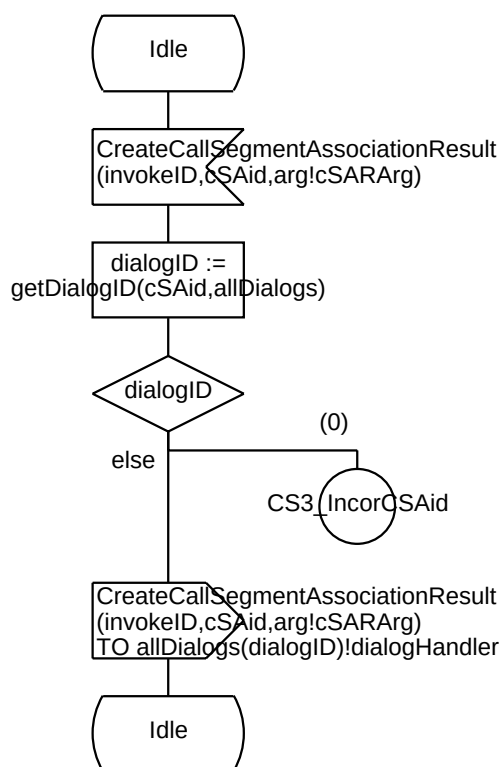
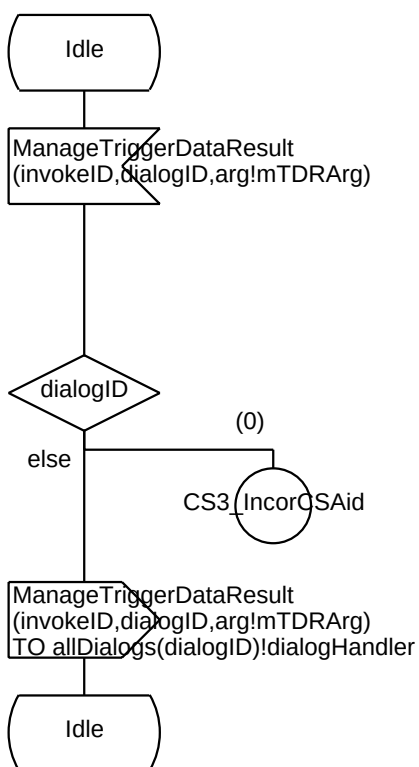
Process Type TCAP_InterfaceHandler

14(15)



Process Type TCAP_InterfaceHandler

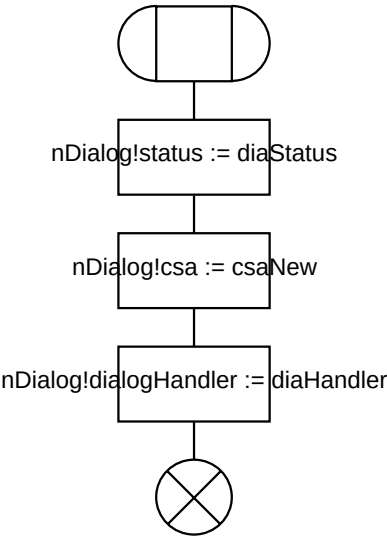
15(15)



Operator newDialog

1(1)

```
;FPAR diaStatus DialogStatusType, csaNew CSAID, diaHandler PId;  
RETURNS nDialog DialogType;
```

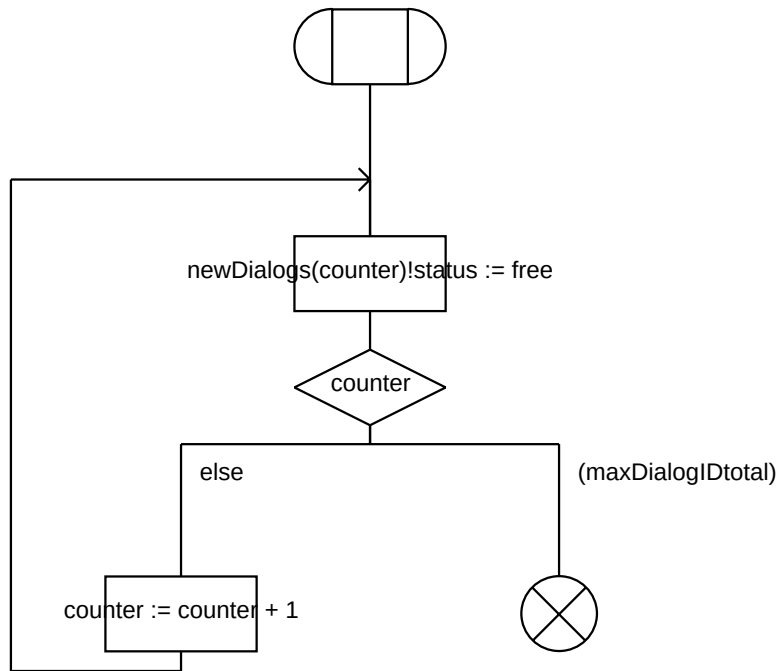


Operator initDialogs

1(1)

;FPAR dummy Boolean;
RETURNS newDialogs AllDialogsType;

DCL counter DialogIDtype := 1;

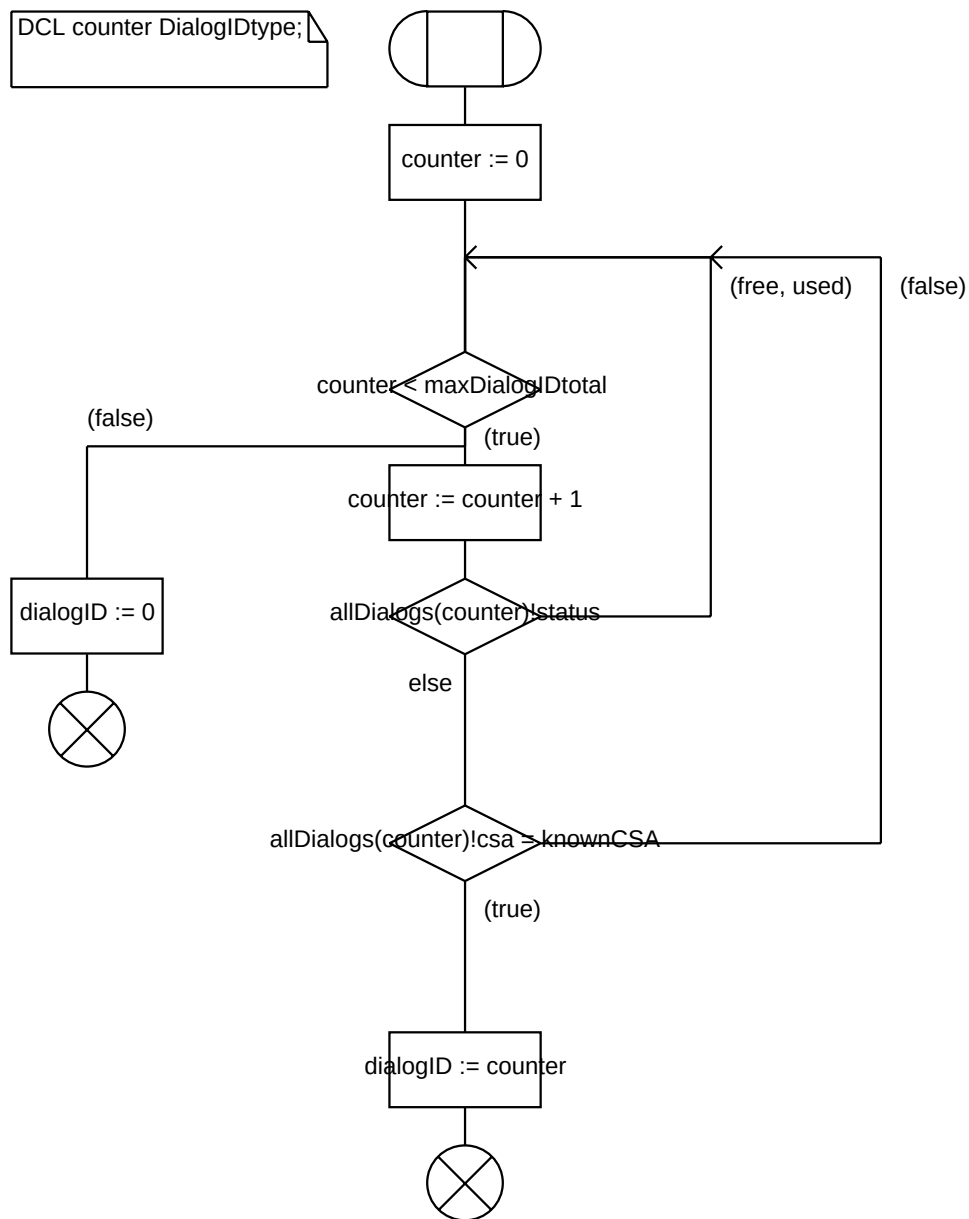


Operator getDialogID

1(1)

;FPAR knownCSA CSAID, allDialogs AllDialogsType;
RETURNS dialogID DialogIDtype;

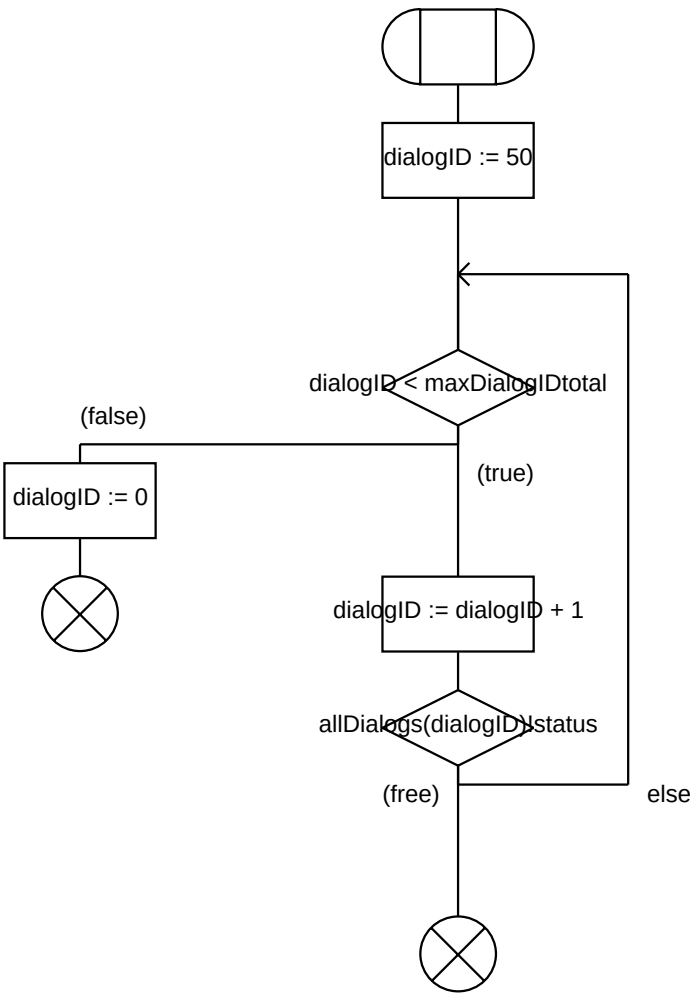
DCL counter DialogIDtype;

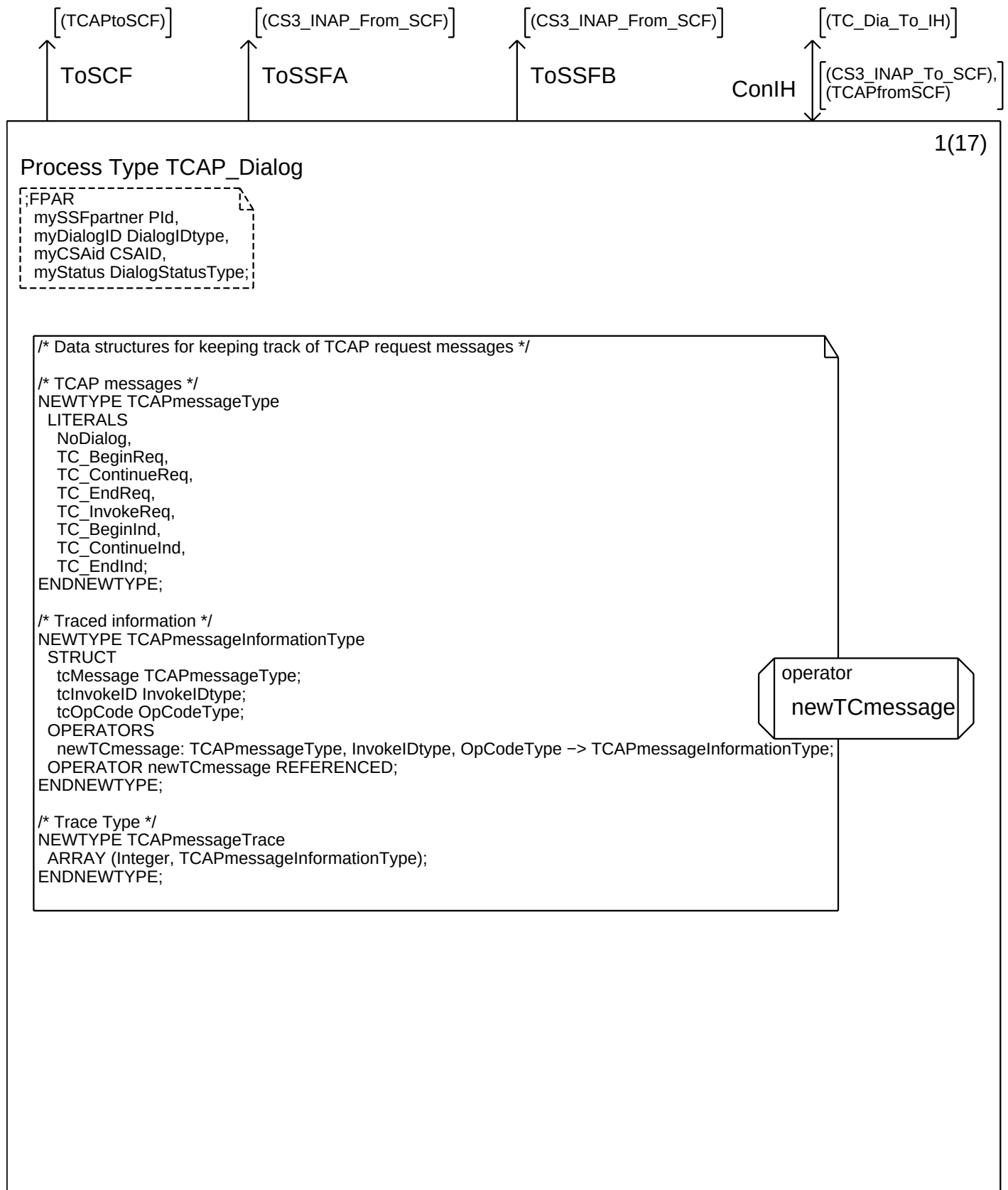


Operator nextFreeDialogID

1(1)

;FPAR allDialogs AllDialogsType;
RETURNS dialogID DialogIDtype;





Process Type TCAP_Dialog

```
;FPAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
```

```
/* State variables, to be updated during the entire lifetime of this process */
```

```
DCL traceSCFtoSSF TCAPmessageTrace;
DCL nextSCFtoSSFmessage Integer;
```

```
DCL lastTCdialog TCAPmessageType;
```

```
DCL nextInvokeIDtoSCF InvokeIDtype;
```

```
/* To be used to receive message parameters */
```

```
DCL dialogID DialogIDtype;
DCL invokeID InvokeIDtype;
DCL opClass OpClassType;
DCL opCode OpCodeType;
DCL timeoutVal TimeoutValType;
DCL origAd TCoriginType;
DCL termination TCAPterminationType;
DCL cSAid CSAID;
DCL compEnd Boolean := false;
```

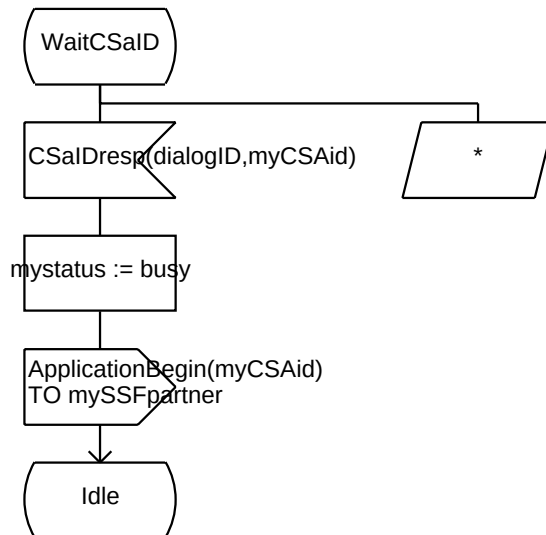
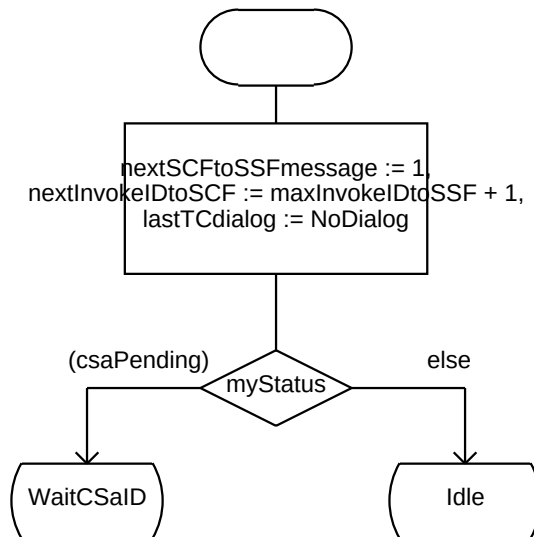
```
DCL arg ArgType; /* refers to ASN.1 definition INCS3BundleArg */
DCL errArg ErrorArg;
```

3(17)

Process Type TCAP_Dialog

```

;FAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```

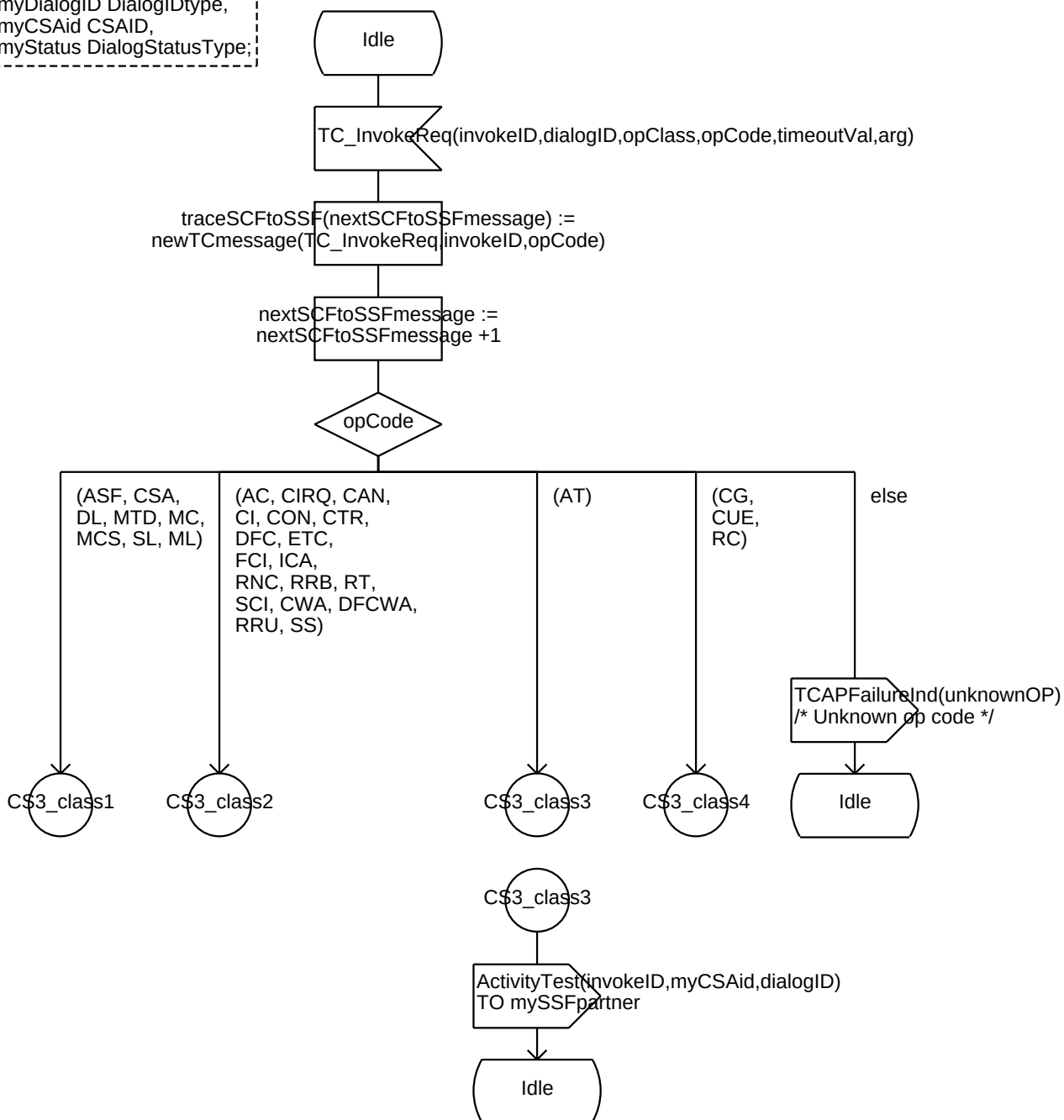


4(17)

Process Type TCAP_Dialog

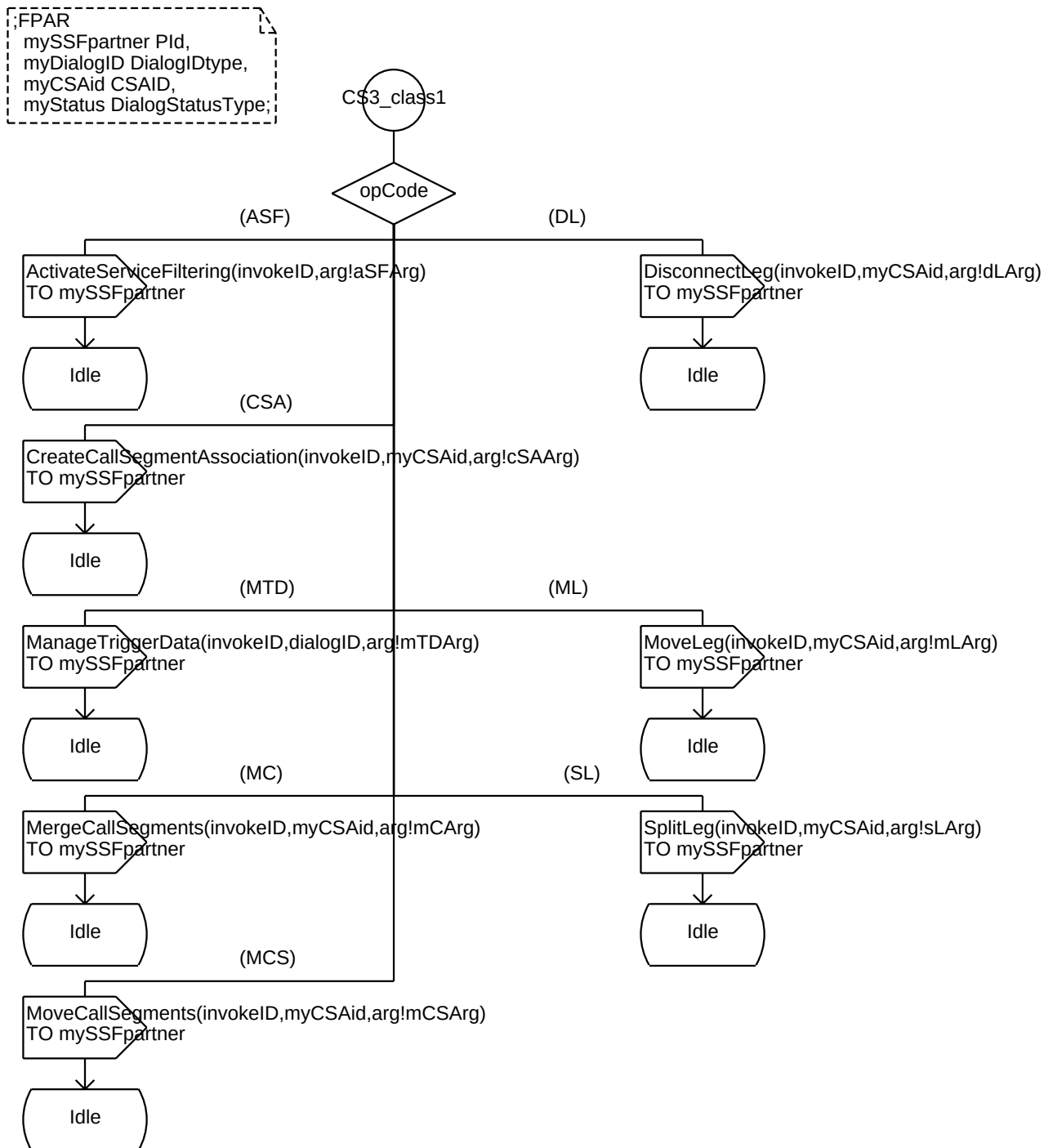
```

;FPAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```



5(17)

Process Type TCAP_Dialog



6(17)

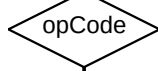
Process Type TCAP_Dialog

```

;FPAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
- (AC)

```

CS3_class2



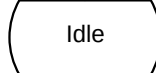
(CAN)

ApplyCharging(invokeID,myCSAid,arg!aCArg)
TO mySSFpartner



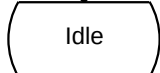
(CI)

Cancel(invokeID,myCSAid,arg!cANArg)
TO mySSFpartner



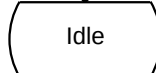
(CIRQ)

CollectInformation(invokeID,myCSAid,arg!cIArg)
TO mySSFpartner



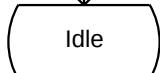
(CON)

CallInformationRequest(invokeID,myCSAid,arg!cIRQArg)
TO mySSFpartner



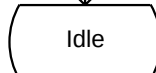
(CTR)

Connect(invokeID,myCSAid,arg!cONArg)
TO mySSFpartner



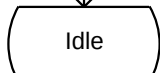
(DFC)

ConnectToResource(invokeID,myCSAid,arg!cTRArg)
TO mySSFpartner



(ETC)

DisconnectForwardConnection(invokeID,myCSAid)
TO mySSFpartner



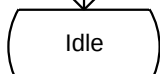
(FCI)

EstablishTemporaryConnection(invokeID,myCSAid,arg!eTCA)
TO mySSFpartner



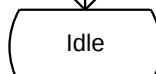
(ICA)

FurnishChargingInformation(invokeID,myCSAid,arg!fCIArg)
TO mySSFpartner



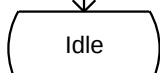
(RNC)

InitiateCallAttempt(invokeID,myCSAid,arg!iCAArg)
TO mySSFpartner

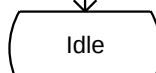


(RRB)

RequestNotificationChargingEvent(invokeID,myCSAid,arg!rNCArg)
TO mySSFpartner



RequestReportBCSMEvent(invokeID,myCSAid,arg!rRBAArg)
TO mySSFpartner



else

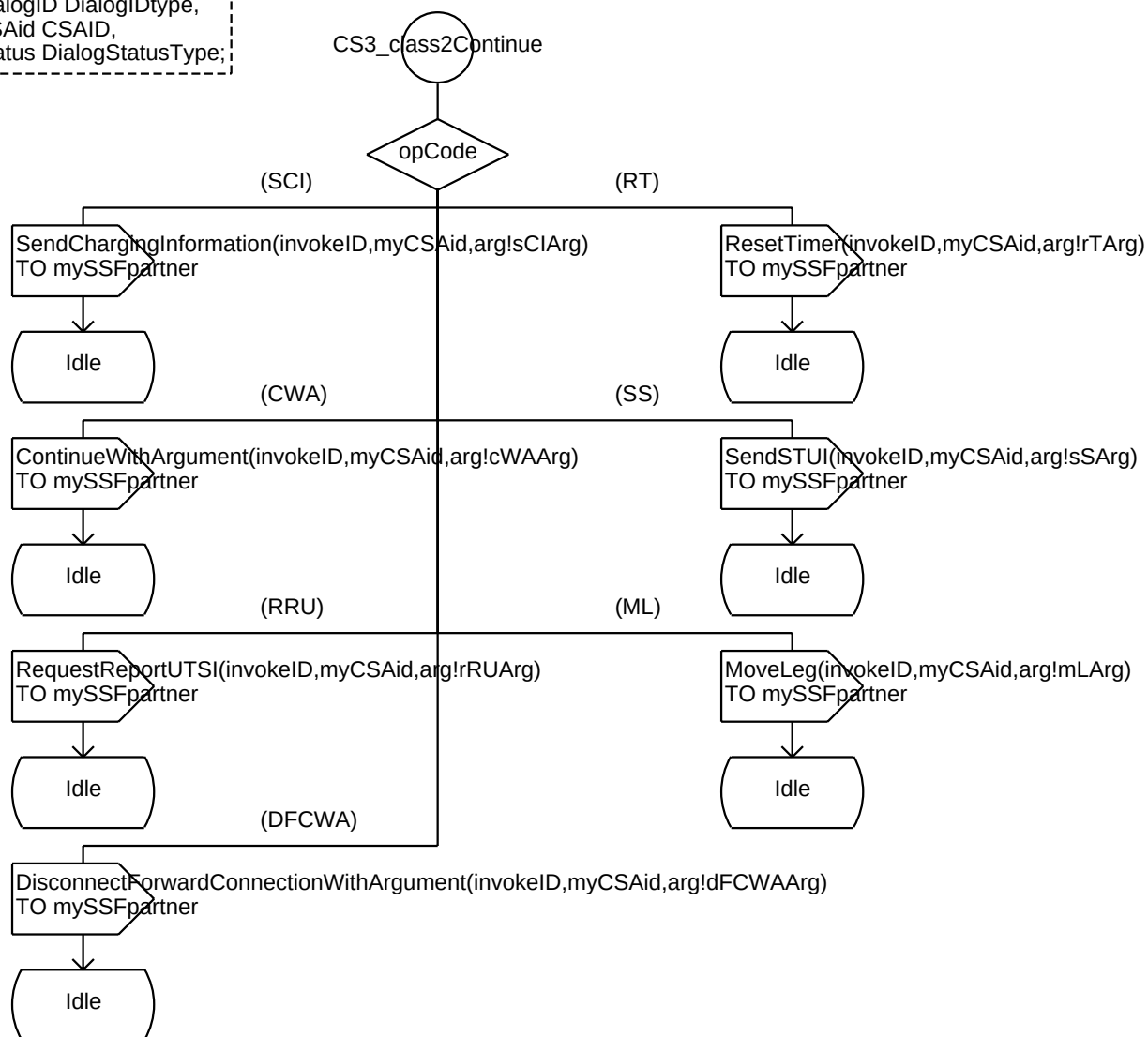
CS3_class2Continue

7(17)

Process Type TCAP_Dialog

```

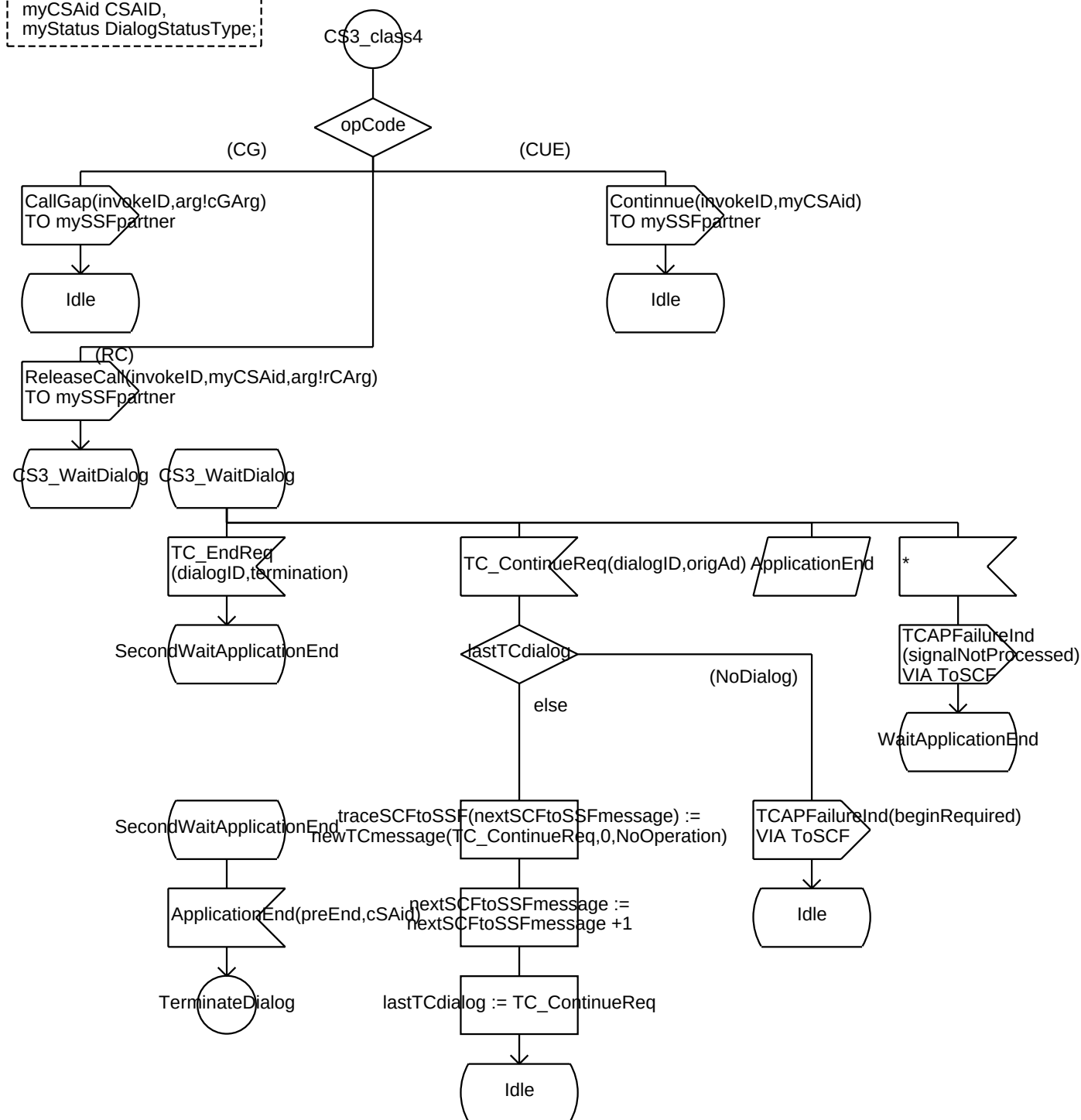
;FPAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```



8(17)

Process Type TCAP_Dialog

FPAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;

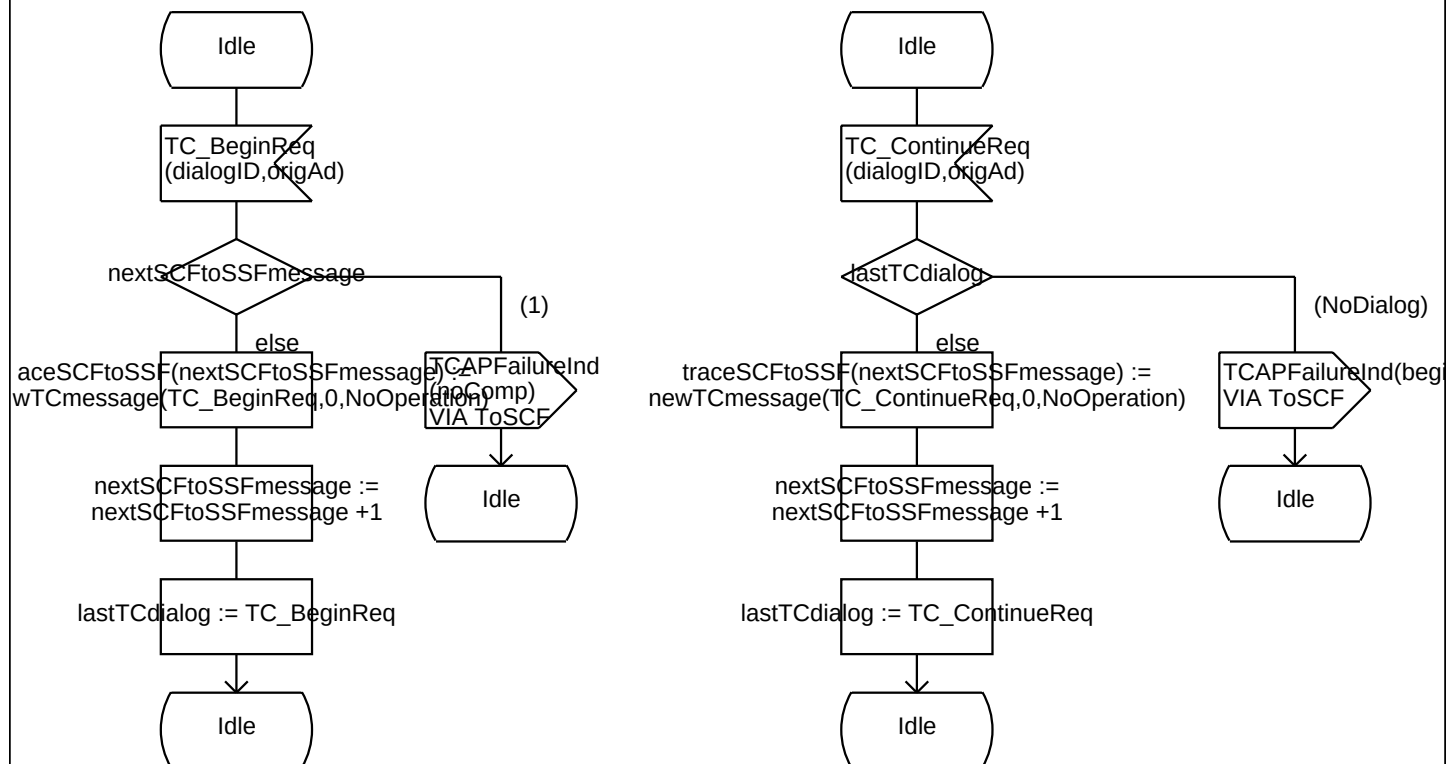


9(17)

Process Type TCAP_Dialog

```

;FPAK
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```

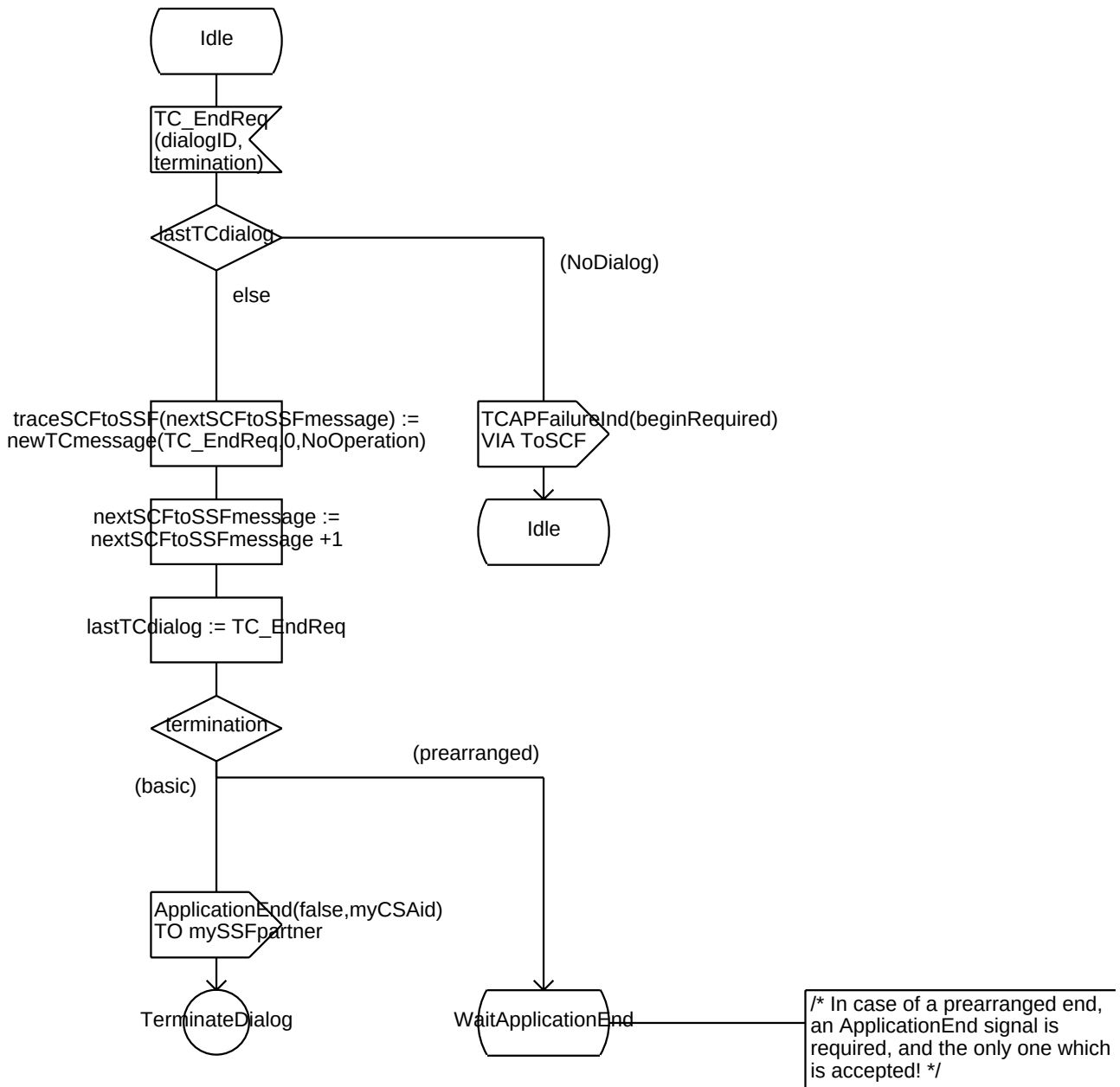


10(17)

Process Type TCAP_Dialog

```

;FAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```

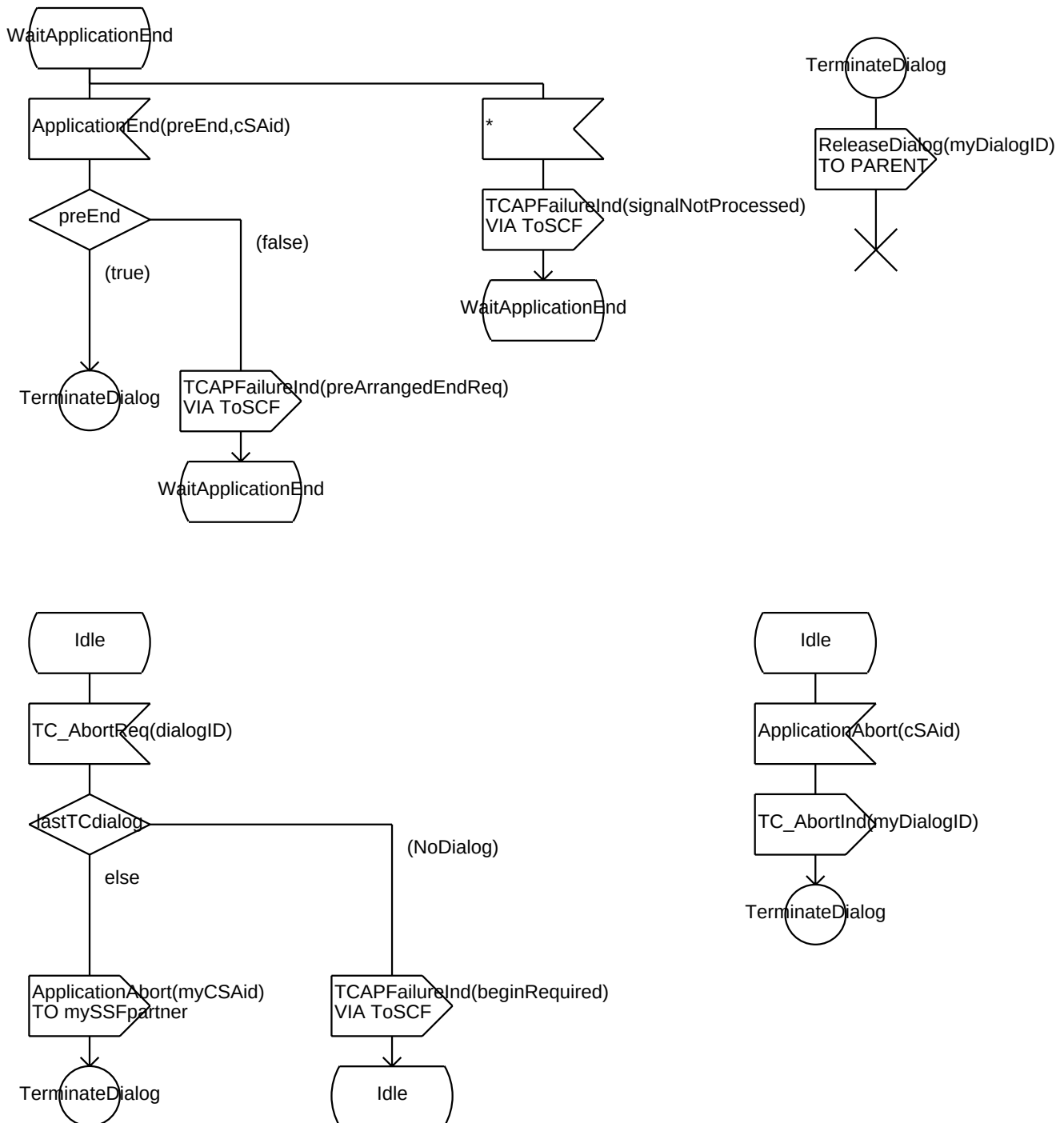


11(17)

Process Type TCAP_Dialog

```

;FAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```



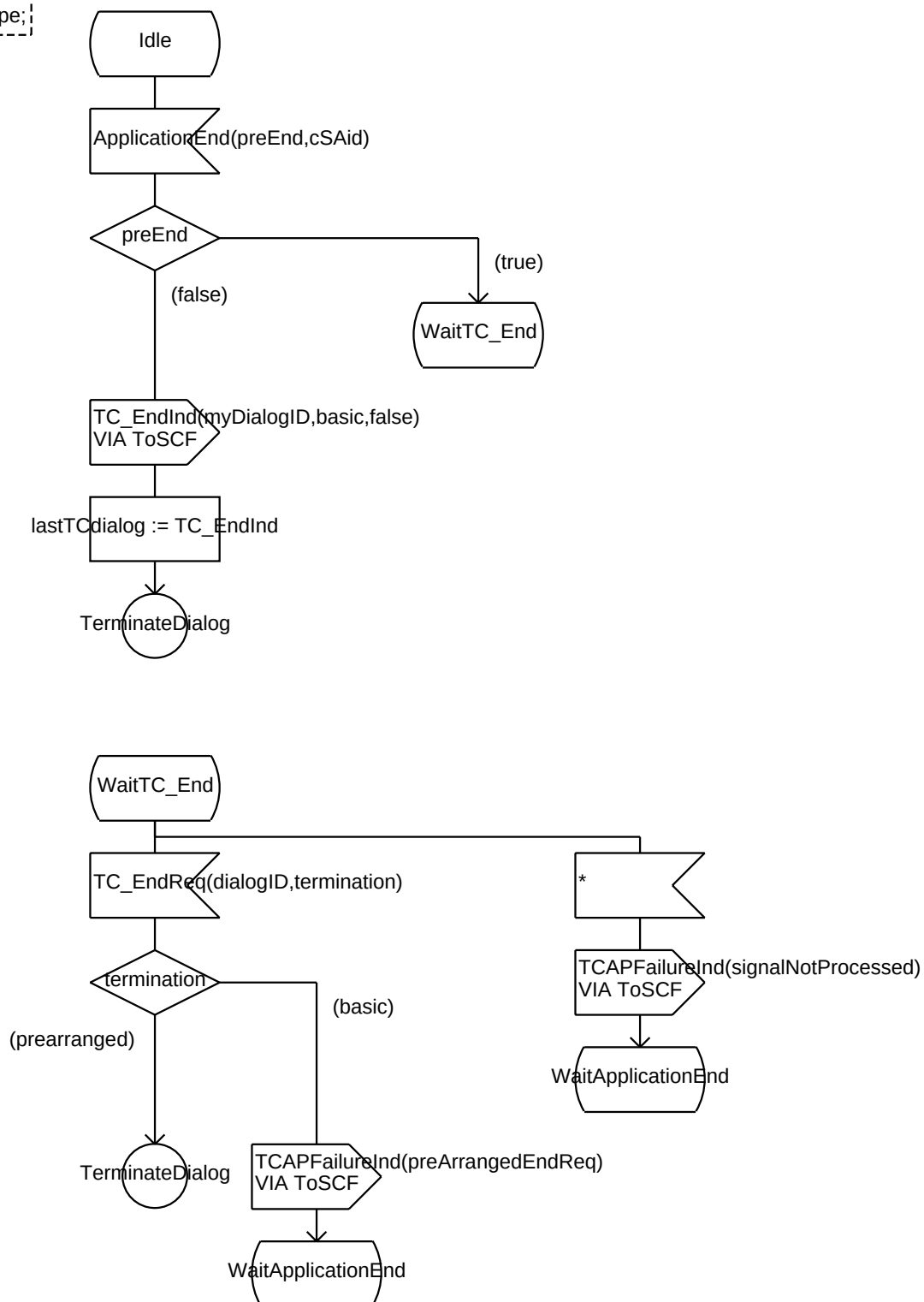
12(17)

Process Type TCAP_Dialog

```

;FPAK
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```

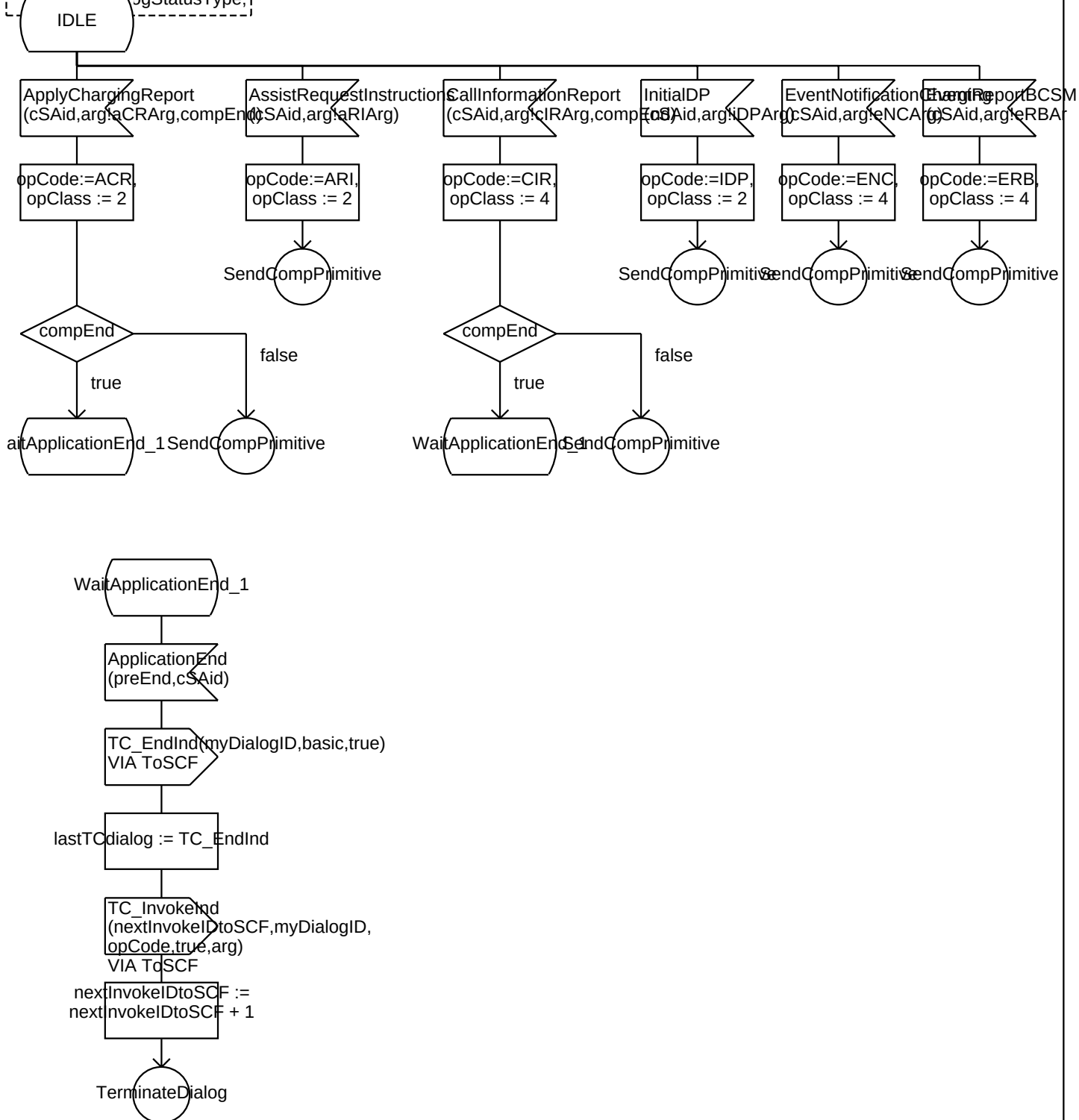
DCL preEnd Boolean;



13(17)

Process Type TCAP_Dialog

mySSFPartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
logStatusType;

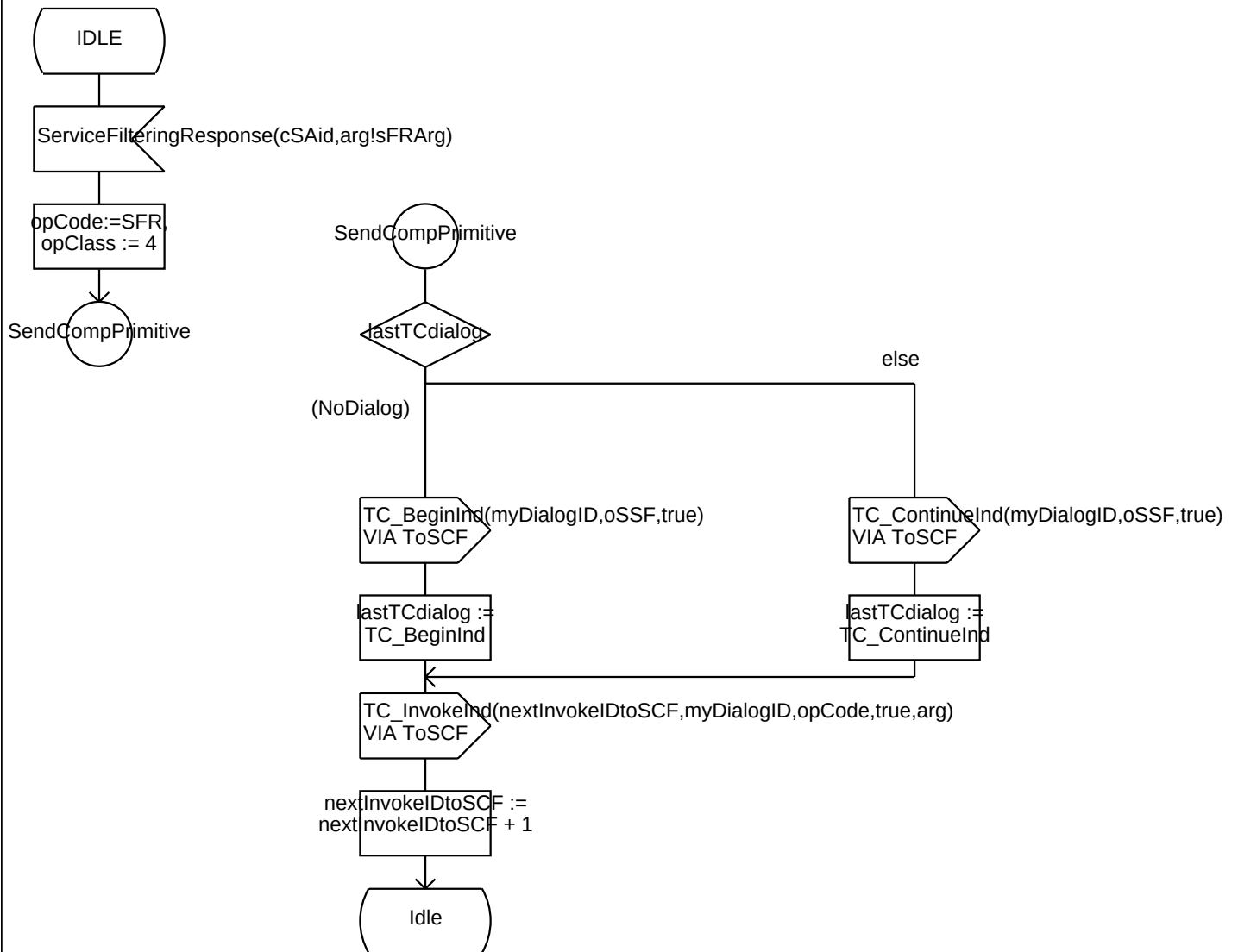


14(17)

Process Type TCAP_Dialog

```

;FAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```

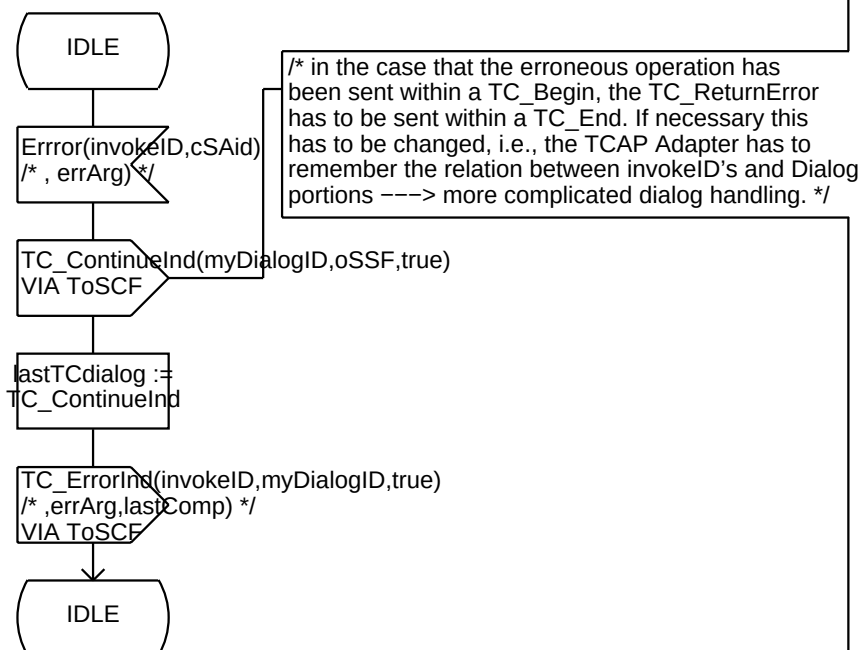
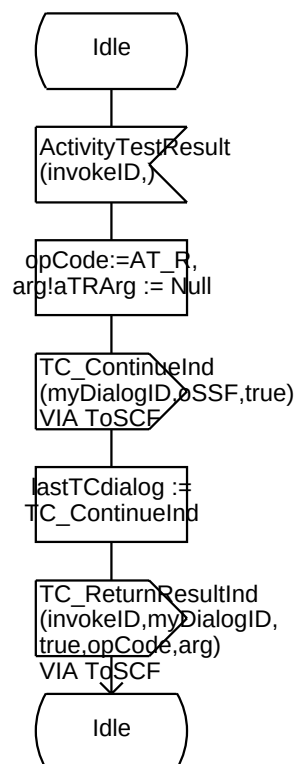


15(17)

Process Type TCAP_Dialog

```

;FPAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```

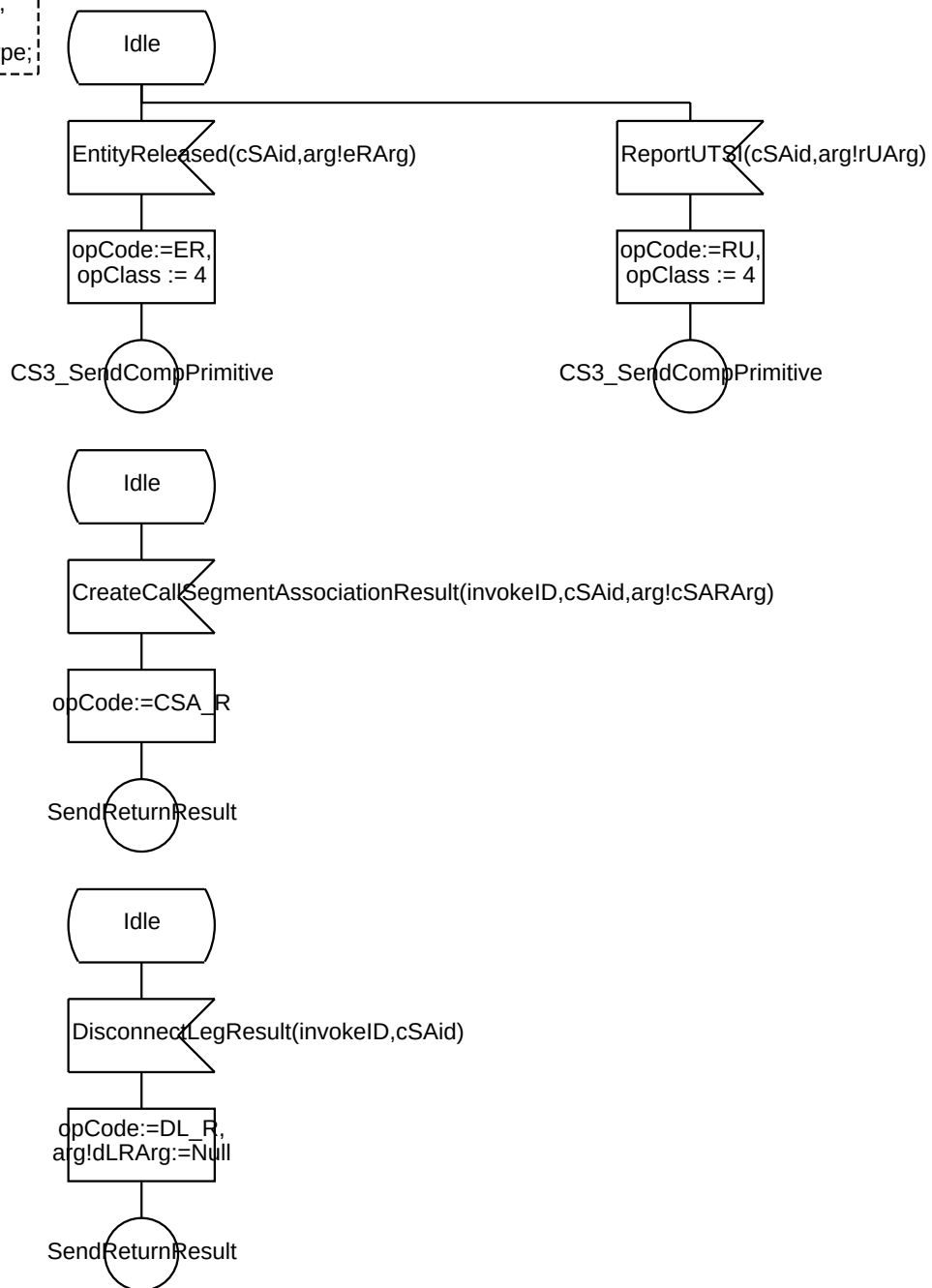


16(17)

Process Type TCAP_Dialog

```

;FPAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```

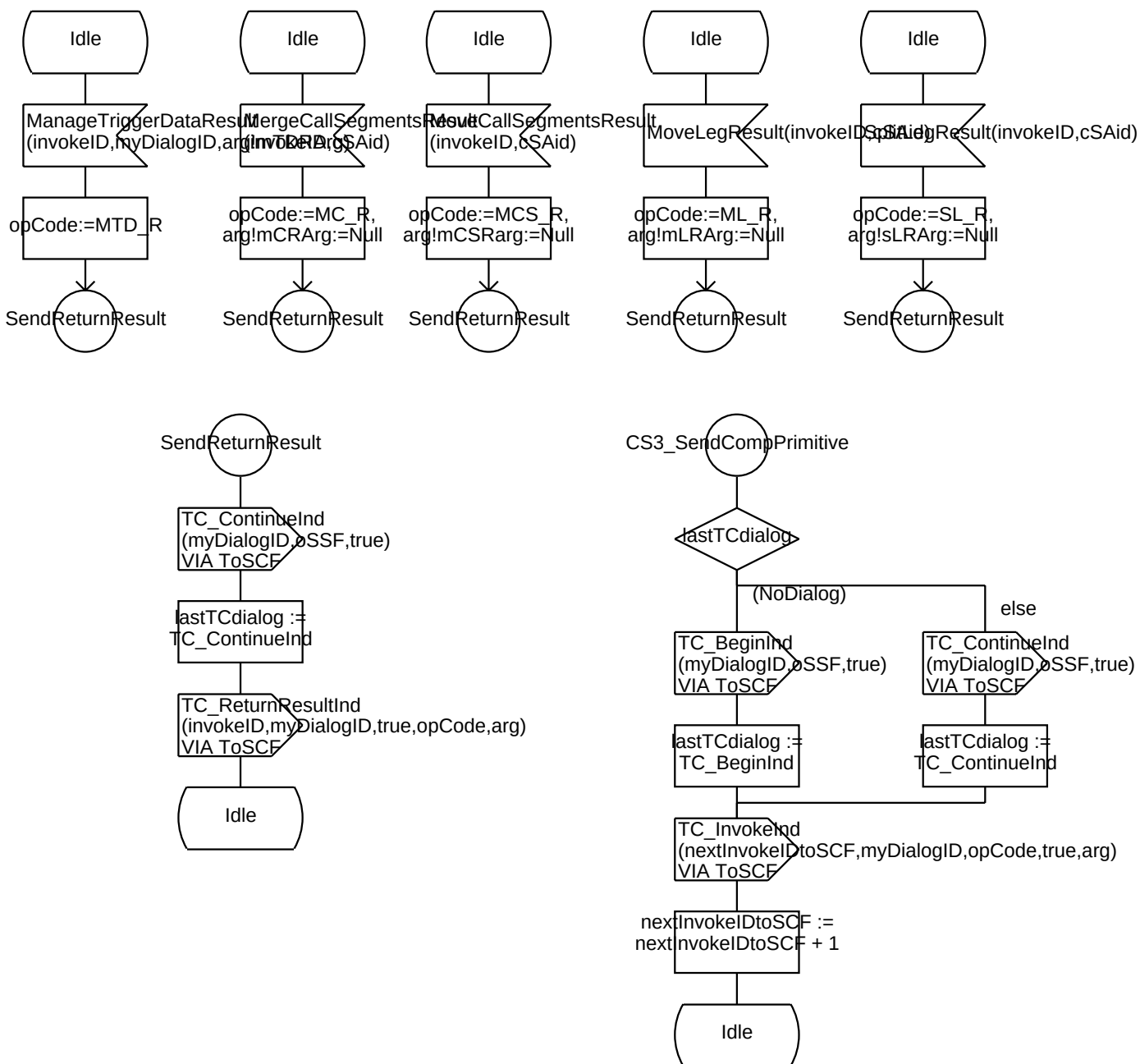


17(17)

Process Type TCAP_Dialog

```

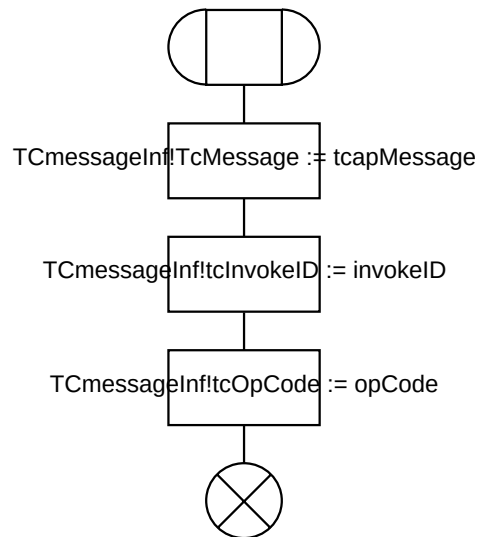
;FPAR
mySSFpartner PId,
myDialogID DialogIDtype,
myCSAid CSAID,
myStatus DialogStatusType;
    
```



Operator newTCmessage

1(1)

```
;FPAR tcapMessage TCAPMessageType, invokeID InvokeIDtype, opCode OpCodeType;  
RETURNS TCmessageInf TCAPmessageInformationType;
```



use CS3_INAP;

System CS3_INAP : CS3_INAP

1(1)

