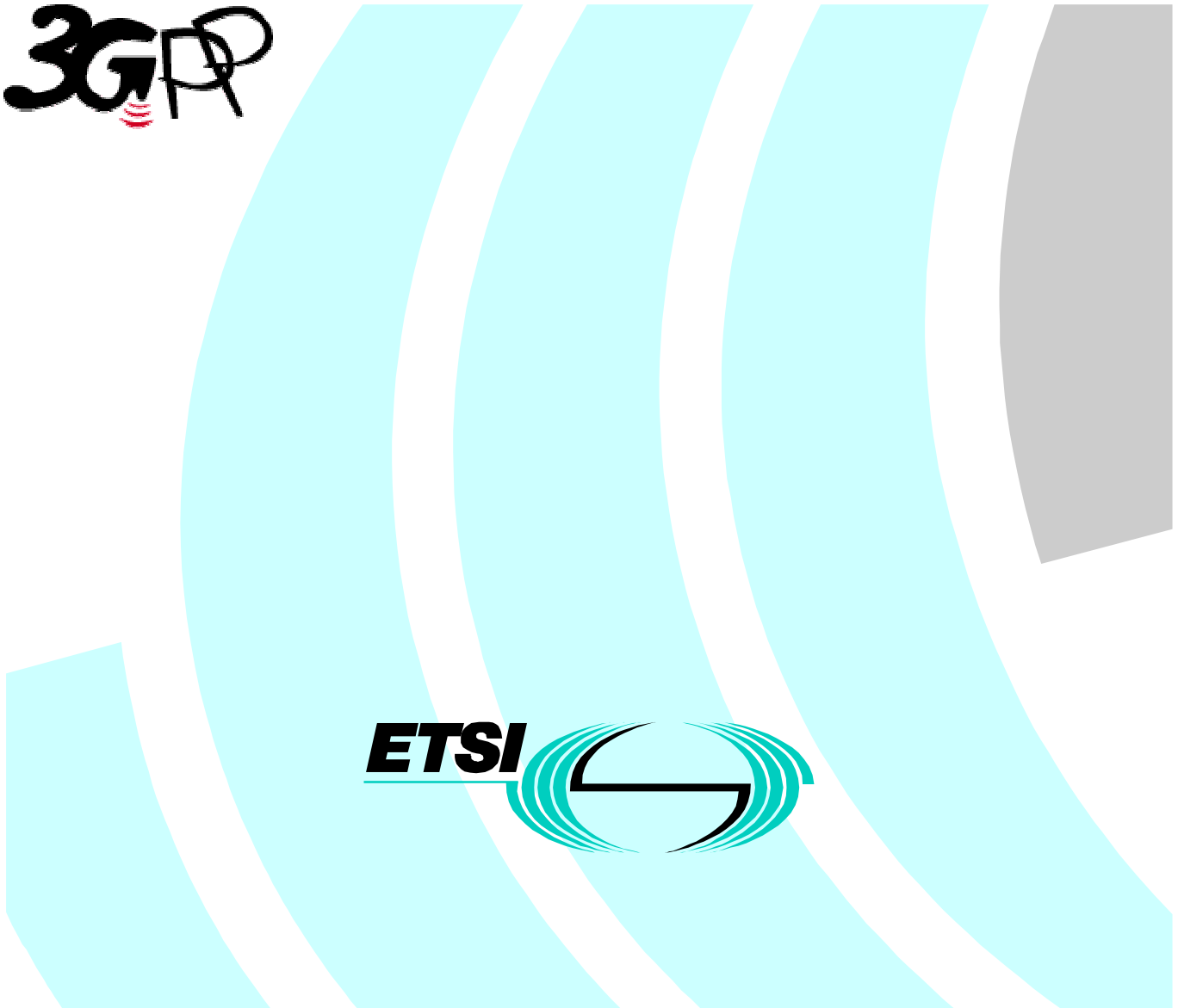
ETSI TR 129 998 V3.1.0 (2000-09)

Technical Report

Universal Mobile Telecommunications System (UMTS);
Open Services Architecture Application Programming Interface - Part 2
(3GPP TR 29.998 version 3.1.0 Release 1999)



Reference
RTR/TSGN-0529998UR1

Keywords
UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

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

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

Foreword

This Technical Report (TR) has been produced by the ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under www.etsi.org/key .

Contents

Foreword

This Technical Report (TR) has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document investigates how the OSA Interface Class methods can be mapped onto CAMEL Application Part operations and MAP Application Part operations. The mapping of the OSA API to the CAP and relevant MAP operations is considered informative, and not normative.

The Open Service Architecture (OSA) defines an architecture that enables operator and third party applications to make use of network functionality through an open standardized interface (the OSA Interface). OSA provides the glue between applications and service capabilities provided by the network. In this way applications become independent from the underlying network technology. The applications constitute the top level of the Open Service Architecture (OSA). This level is connected to the Service Capability Servers (SCSs) via the OSA interface. The SCSs map the OSA interface onto the underlying telecommunications specific protocols (e.g. MAP, CAP, etc.) and are therefore hiding the network complexity from the applications.

The specific Service Capability Server under consideration in this technical report is the CSE. In this case, the OSA API provides the operator or third party applications access to the CAMEL Application Part protocol operations, via the OSA Interface Class methods. On the gsmSCF, the OSA Interface Class methods need to be mapped, or translated, onto the relevant CAP and/or MAP operations. Only the non-framework Service Capability Features will be taken into account for the mapping. This document is not exhaustive in covering all the mappings that can be expected. It provides several examples, but it should be noted that several other possibilities exist. In particular, only general cases of normal operations are covered and exception scenarios are not within the scope of the document.

The OSA API to CAP and MAP mapping is part of Release99.

2 References

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

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies.

[1] 3GPP TR 22.905: "3GPP Vocabulary".

[2] 3GPP TS 29.198: "Open Service Architecture; Application Programming Interface - Part 1".

[3] 3GPP TS 29.002: "Mobile Application Part (MAP) specification".

[4] 3GPP TS 29.078: "CAMEL Application Part (CAP) specification – Phase 3".

[5] 3GPP TS 23.127: "Virtual Home Environment / Open Service Architecture".

[6] 3GPP TS 22.101: "Universal Mobile Telecommunications System (UMTS): Service Aspects; Service Principles".

[7] ITU-T Q.850: “Usage of cause and location in the Digital Subscriber Signalling System No. 1 and the Signalling System No. 7 ISDN User Part.”

3 Definitions and abbreviations

3.1 Definitions

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

Service Capabilities: Bearers defined by parameters, and/or mechanisms needed to realize services. These are within networks and under network control.

Service Capability Feature: Functionality offered by service capabilities that are accessible via the standardized OSA interface

Service Capability Server: Functional Entity providing OSA interfaces towards an application

Services: Services are made up of different service capability features.

Applications: Services, which are designed using service capability features.

OSA Interface: Standardized Interface used by application to access service capability features.

Virtual Home Environment: A concept for personal service environment portability across network boundaries and between terminals.

Further UMTS related definitions are given in 3GPP TS 22.101.

3.2 Abbreviations

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

API	Application Programming Interface
CAMEL	Customised Application for Mobile network Enhanced Logic
CAP	CAMEL Application Part
CSE	Camel Service Environment
HE	Home Environment
HE-VASP	Home Environment Value Added Service Provider
HLR	Home Location Register
IDL	Interface Description Language
MAP	Mobile Application Part
ME	Mobile Equipment
MExE	Mobile Station (Application) Execution Environment
MS	Mobile Station
MSC	Mobile Switching Centre
OSA	Open Service Architecture

PLMN	Public Land Mobile Network
PSE	Personal Service Environment
SAT	SIM Application Tool-Kit
SCP	Service Control Point
SRF	Specialised Resource Function
SIM	Subscriber Identity Module
SMS	Short Message Service
USIM	User Service Identity Module
VASP	Value Added Service Provider
VHE	Virtual Home Environment
WAP	Wireless Application Protocol
WGP	WAP Gateway Proxy
WPP	WAP Push Proxy

Further GSM related abbreviations are given in GSM 01.04. Further UMTS related abbreviations are given in 3GPP TR 22.905.

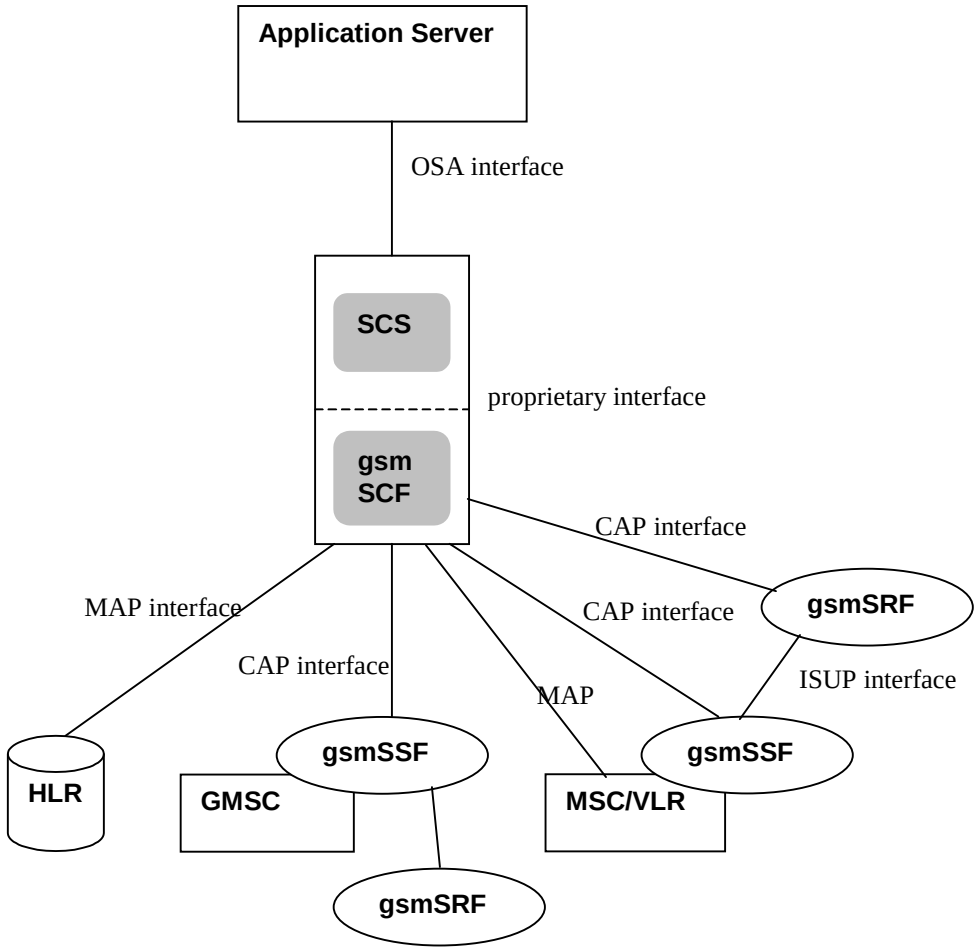
4 Virtual Home Environment and Open Service Architecture

The Open Service Architecture (OSA) is the architecture enabling applications to make use of network capabilities. The applications will access the network through the OSA interface that is specified in 3GPP TS 23.127.

The access to network functionality is offered by different Service Capability Servers (SCSs) and appear as service capability features in the OSA interface. These are the capabilities that the application developers have at their hands when designing new applications (or enhancements/variants of already existing ones). The different features of the different SCSs can be combined as appropriate. The service logic executes toward the OSA interfaces, while the underlying core network functions use their specific protocols. This technical report specifically considers the CSE SCS and the CAMEL Phase3 capabilities. An informative mapping of OSA API methods onto CAP and relevant MAP operations is provided.

4.1 The Interface

The OSA API interface and the protocol onto which the Interface Class methods are mapped, are depicted in Figure 4-1. The applications are executed on an Application Server. The OSA API interface allows the application access to the functionality provided by the Service Capability Server. The OSA interface resides between the Application Server and the SCS, while the CAP and MAP interfaces reside in the network domain as illustrated in Figure 4-1 below.



Key

CAP	CAMEL Application Part
CSE	CAMEL Service Environment
GMSC	Gateway Mobile Switching Center
gsmSSF	GSM Service Switching Function
gsmSRF	GSM Specialised Resource Function
HLR	Home Location Register
OSA	Open Services Architecture
SCS	Service Capability Server

Figure 4-1: The Interface under consideration

The SCS uses network capabilities through an undefined proprietary interface. The actual implementation of the SCS is not defined. However, the mapping is independent of the implementation option for the SCS and the gsmSCF, i.e. independent of the fact whether SCS and gsmSCF are implemented in the same physical entity or separate physical entities. The network may include non-CAMEL capabilities to implement the API, but these capabilities are not shown in the figure and are without the scope of this technical report.

5 General Parameter Mapping Issues

5.1 API Parameters that do not require a mapping

A number of the API method parameters have significance only on the OSA interface and in the SCS. They are used to identify objects implementing parts of the interface for instance. No mapping is required for these parameters.

- appInterface - specifies a reference to the application object which implements the callback interface for a call
- assignmentID - specifies the assigned ID which is used to link associated requests and responses
- callReference - specifies the reference to the call object
- callSessionID - specifies the call session ID of the call object to which this method invocation applies

5.2 Protocol Operation Parameters that do not require a mapping

A number of the CAP and MAP protocol operation parameters deal with the specifics of the underlying core network. these are typically those details that the OSA API was designed to abstract from and therefore do not require a mapping. Examples include:

CAP InitialDP:

- gsmSCFAddress
- MSCAddress
- GMSCAddress
- IPSSPCapabilities

MAP AnyTimeModification

- gsmSCFAddress

6 Generic Call Control Service CAMEL Call Flows

6.1 Call Manager

The generic call manager interface class provides the management functions to the generic call Service Capability Features. The application programmer can use this interface to create call objects and to enable or disable call-related event notifications.

6.1.1 enableCallNotification

enableCallNotification is used to enable call notifications to be sent to the application.

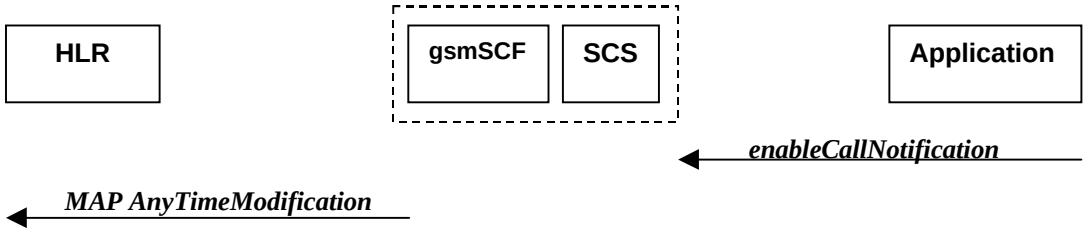


Figure 6-1: Call Flow for enableCallNotification

Normal Operation

Two alternatives have been identified.

1 The application requests notifications to be enabled.

Pre-conditions	An agreement is established between the network operator and the service provider for the event

	notification to be enabled
1	The application invokes the ***enableCallNotification*** method
2	The gsmSCF sends a MAP ***AnyTimeModification*** to the HLR in order to Activate the necessary CAMEL Subscription Information (O-CSI, D-CSI, T-CSI, VT-CSI) Note : CAMEL phase 3 only allows for activation/deactivation of the CSI and not modification of the contents of the CSIs. The O-CSI and D-CSI will be activated if the originating address is present and the T-CSI and VT-CSI will be activated if the destination address is present

Error condition

2 HLR rejects CSI updates

Pre-conditions	gsmSCF had previously sent a MAP ***AnyTimeModification*** message to the HLR as a result of an ***enableCallNotification*** request from the application
1	HLR rejects the request to update the CSI
2	The gsmSCF sends an internal message to the SCS to indicate the up date failure
3	The SCS invokes the exception on ***enableCallNotification***

Parameter Mapping

From: ***enableCallNotification***	To: MAP ***AnyTimeModification***
appInterface	
eventCriteria (TpCallEventCriteria) :	
DestinationAddress	**subscriberIdentity**[1] **modificationRequestFor-CSI**
OriginationAddress	**subscriberIdentity**[2] **modificationRequestFor-CSI**
CallEventName (TpCallEventName : section 0)	**CAMEL Subscription Information** - **T-CSI** - **VT-CSI** - **O-CSI** - **D-CSI**
CallNotificationType	
assignmentID	
	modificationRequestFor-SS-Info
	gsmSCF address

1 in case an address range is used, a separate MAP AnyTimeModificationRequest must be sent for every address in the range

2 in case an address range is used, a separate MAP AnyTimeModificationRequest must be sent for every address in the range

6.1.2 disableCallNotification

disableCallNotification is used by the application to disable call notifications.

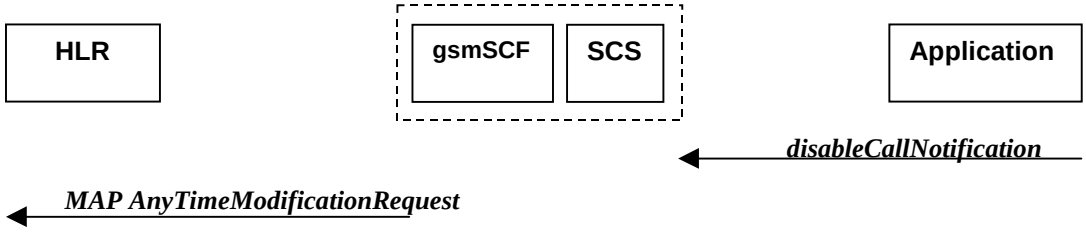


Figure 6-2: Call Flow for disableCallNotification

Normal Operation

Pre-conditions	An agreement is established between the network operator and the service provider for the event notification to be disabled
1	The application invokes the ***disableCallNotification*** method
2	The gsmSCF sends a MAP ***AnyTimeModification*** to the HLR in order to de-activate the CAMEL subscription Information (O-CSI, D-CSI, T-CSI, VT-CSI). Note that CAMEL Phase 3 only allows the capability to activate/deactivate CSI and not to modify the triggering information. The O-CSI and D-CSI will be deactivated if the originating address is present and the T-CSI and VT-CSI will be deactivated if the destination address is present

Parameter Mapping

From: ***disableCallNotification***	To: MAP ***AnyTimeModification***
assignmentID	
	gsmSCFAddress

6.1.3 changeCallNotification

changeCallNotification is used by the application to change the call notifications previously set by *enableCallNotification()*.

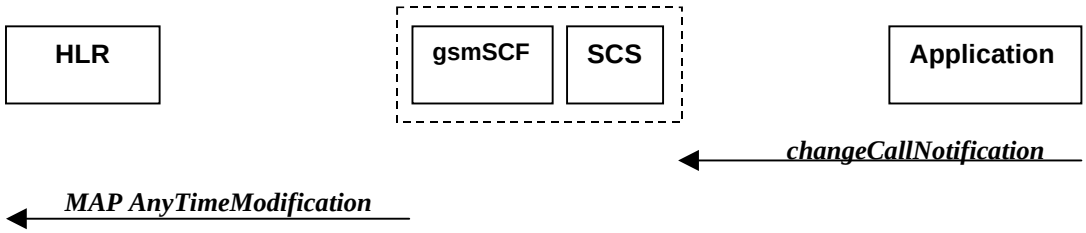


Figure 6-3: Call Flow for changeCallNotification

Normal Operation

Pre-conditions	Notifications have been enabled by the application.
1	The application invokes the ***changeCallNotification*** method

2	The gsmSCF sends a MAP ***AnyTimeModification*** to the HLR in order to active and de-activate the CAMEL subscription Information (O-CSI, T-CSI, VT-CSI). The SCS and gsmSCF will have to determine which CSI's to active and which to de-activate in order to reflect the changed set of notifications. The O-CSI and D-CSI will be modified if the originating address is present and the T-CSI and VT-CSI will be modified if the destination address is present

Parameter Mapping

From: ***changeCallNotification***	To: MAP ***AnyTimeModification***
assignmentID	
eventCriteria (TpCallEventCriteria) :	
DestinationAddress	**subscriberIdentity**[3] **modificationRequestFor-CSI**
OriginationAddress	**subscriberIdentity**[4] **modificationRequestFor-CSI**
CallEventName (TpCallEventName : see section 14)	**CAMEL Subscription Information** - **T-CSI** - **VT-CSI** - **O-CSI** - **D-CSI**
CallNotificationType	
	modificationRequestFor-SS-Info
	gsmSCFAddress

6.1.5 getCriteria

getCriteria is used by the application to query the event criteria set with ***enableCallNotification***.

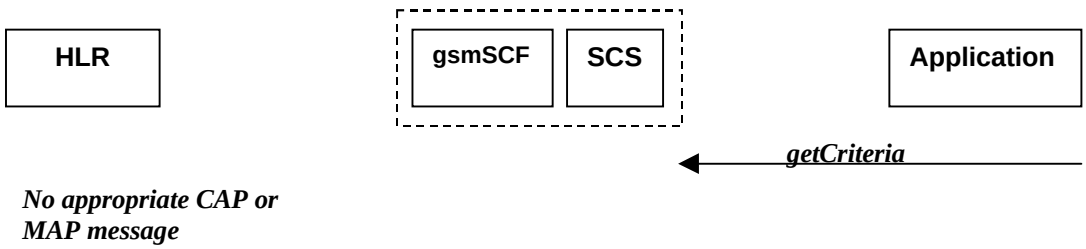


Figure 6-4: Call Flow for getCriteria

[3] in case an address range is used, a separate MAP AnyTimeModificationRequest must be sent for every address in the range

[4] in case an address range is used, a separate MAP AnyTimeModificationRequest must be sent for every address in the range

Normal Operation

Pre-conditions	Notifications have been enabled by the application.
1	The application invokes the ***getCriteria*** method
2	The SCS returns the criteria

Parameter Mapping

None.

6.1.7 callNotificationInterrupted

callNotificationInterrupted indicates to the application that all event notifications have been interrupted, for example due to faults detected.

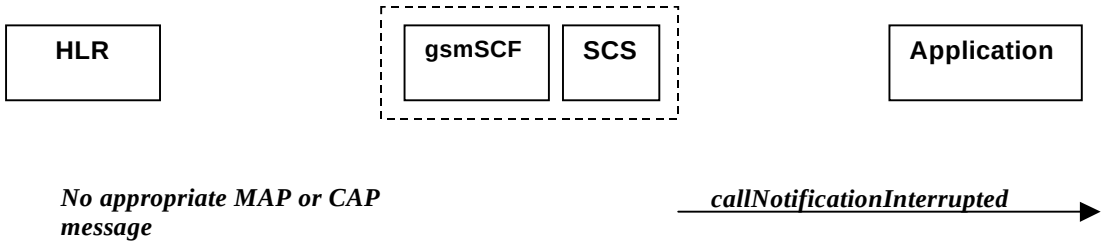


Figure 6-5: Call Flow for callNotificationInterrupted

Normal Operation

Pre-conditions	Call notifications have been enabled using the ***enableNotification*** method on the Call Manager interface
1	The SCS has detected, or has been informed of, a fault which prevents further events from being notified
2	The SCS invokes the ***callNotificationInterrupted*** method

Parameter Mapping

None.

6.1.8 callNotificationContinued

callNotificationContinued indicates to the application that all event notifications have been previously interrupted, have now started again.

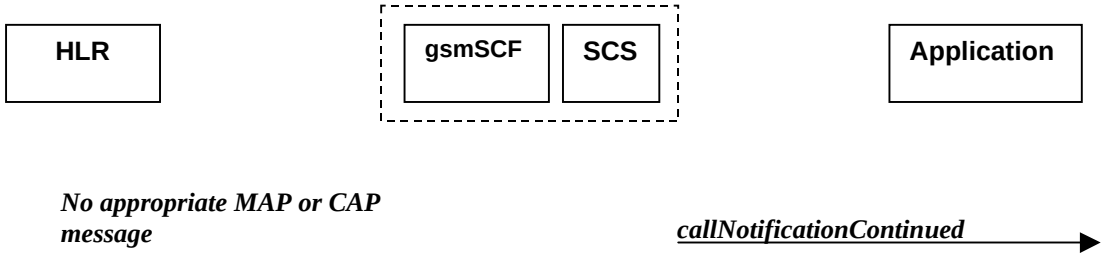


Figure 6-6: Call Flow for callNotificationContinued

Normal Operation

Pre-conditions	Call notifications have been interrupted and ***callNotificationInterrupted*** method has been invoked.
1	The SCS detects that call notifications are again possible.
2	The SCS invokes the ***callNotificationContinued*** method

Parameter Mapping

None.

6.1.9 callAborted

callAborted indicates to the application that the call object has aborted or terminated abnormally. No further communication will be possible between the call and the application.

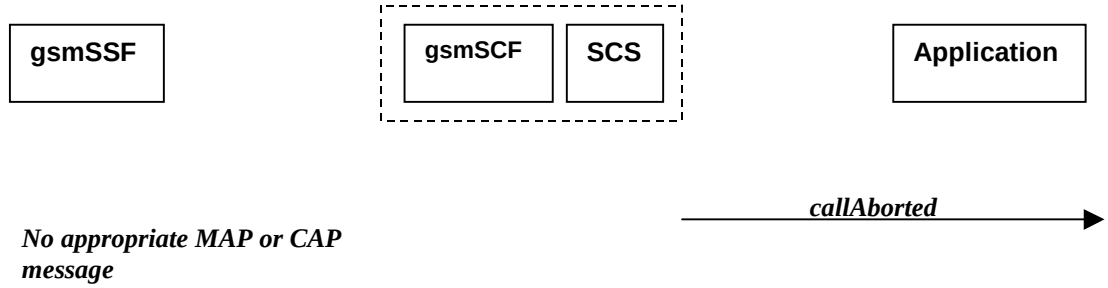


Figure 6-7: Call Flow for callAborted

Normal Operation

Pre-conditions	
1	The SCS detect a catastrophic failure in its communication with the gsmSCF
2	The SCS, invokes the ***callAborted*** method. The call running in the network may continue and will not have been affected by this failure between the gsmSCF and the SCS

Parameter Mapping

None.

6.1.10 callEventNotify

callEventNotify notifies the application of the arrival of a call-related event.

Figure 6-8: Call Flow for callEventNotify

Normal Operation

Pre-conditions	Call notifications have been enabled using the ***enableCallNotification*** method on the Call Manager interface
1	A call arrives at the gsmSSF causing initial triggering to the gsmSCF CAP ***InitialDP***
2	The gsmSCF recognizes the need for an API service and passes the triggering information to the SCS
3	The SCS identifies the application responsible for handling the call and invokes the ***callEventNotify*** method

Parameter Mapping

From: **CAP InitialDP**	To: ***callEventNotify***
	callReference
	eventInfo (TpCallEventInfo) :
calledPartyNumber **calledPartyBCDNumber BCD**	**destinationAddress**
calling Party Number	**originatingAddress**
originalCalledPartyID	**originalDestinationAddress**
redirectingPartyID	**redirectingAddress**
	callAppInfo (TpCallAppInfoSet) :
	CallAppAlertingMechanism
	CallAppNetworkAccessType
	CallAppInterworkingIndicators
ext-BasicServiceCode (1st priority)	**CallAppBearerService** **CallAppTeleService**
highLayerCompatibility (2nd priority)	**CallAppTeleService**
bearerCapability (2nd priority)	**CallAppBearerService**
callingPartysCategory	**CallAppPartyCategory**
	CallAppPresentationAddress
	CallAppGenericInfo
additionalCallingPartyNumber	**CallAppAdditionalAddress**
eventTypeBCSM	**callEventName (Table 1)**
	callNotificationType
	assignmentID
	appInterface
serviceKey	<Note: mapped to the method invocation>
cGEncountered	

iPSSPCapabilities	
locationNumber	
redirectionInformation	
iMSI	
subscriberState	
locationInformation	
callReferenceNumber	
serviceInteractionIndicatorsTwo	
mscAddress	
timeAndTimezone	
gsm-ForwardingPending	
initialDPargExtension :	
naCarrierInformation	
gmscAddress	
cause	
cug-Index	
cug-Interlock	
cug-OutgoingAccess	

From: **CAP *InitialDP* parameter *eventTypeBCSM***	To: ***callEventNotify* parameter *callEventName* in *eventInfo***
<no mapping available>	P_EVENT_NAME_UNDEFINED
<no mapping available>	P_EVENT_GCCS_OFFHOOK_EVENT
collectedInfo, termAttemptAuthorized	P_EVENT_GCCS_ADDRESS_COLLECTED_EVENT
analyzedInformation	P_EVENT_GCCS_ADDRESS_ANALYSED_EVENT
tBusy	P_EVENT_GCCS_CALLED_PARTY_BUSY
tBusy[5]	P_EVENT_GCCS_CALLED_PARTY_UNREACHABLE
tNoAnswer	P_EVENT_GCCS_NO_ANSWER_FROM_CALLED_PARTY
routeSelectFailure	P_EVENT_GCCS_ROUTE_SELECT_FAILURE
<no mapping available>	P_EVENT_GCCS_ANSWER_FROM_CALL_PARTY

Table 1 : eventTypeBCSM mapping to callEventName

[5] Depending on the value of the *cause* parameter in the *initialDPArg extensions* parameter of the InitialDP operation

6.2 Call

The generic call interface represents the interface to the generic call Service Capability Feature. It provides a structure to allow simple and complex call behaviour.

6.2.1 routeReq

routeReq is an asynchronous method which requests routing of the call (and inherently attached parties) to the destination party, via a passive call leg. Subsequent invocations of the ***routeCallToDestinationReq*** method are not allowed. This implies that all triggers, required by the application throughout the lifetime of the call, need to be armed in the parameter **responseRequested**.

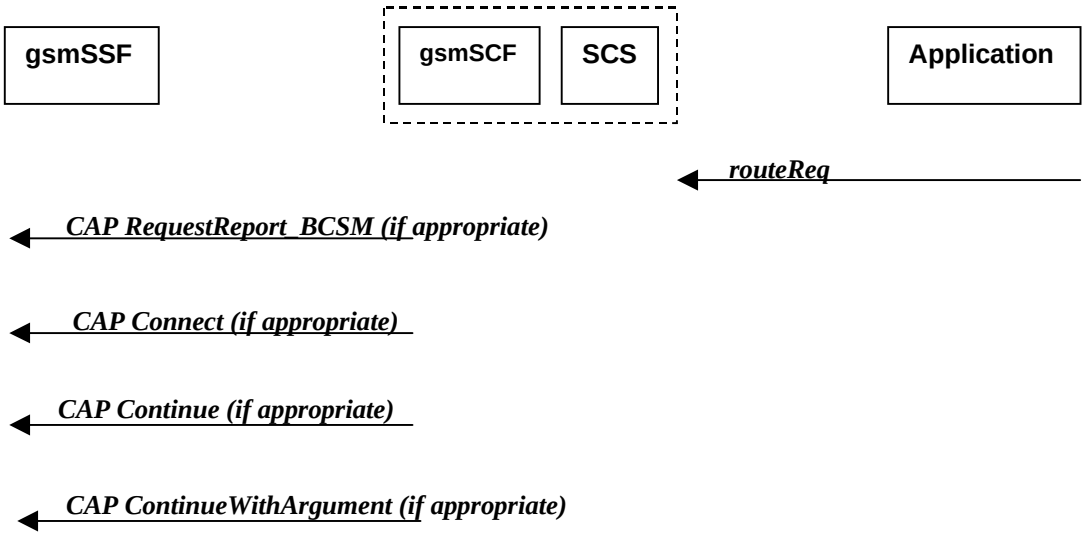


Figure 6-9: Call Flow for routeReq

Normal Operation

Three alternatives have been identified

1. The application changes the destination number

Pre-conditions	The application has been notified of a new call and the call object exists. The ***setCallChargePlan*** and ***getCallInfoReq*** methods may have been invoked
1	The application invokes the ***routeReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***RequestReportBSCM*** if the application needs to be informed about the outcome of the request
4	The gsmSCF sends a CAP ***Connect*** message

Parameter Mapping

From: ***routeReq***	To: CAP ***RequestReportBCSMEvent***
callSessionID	
responseRequested (TpCallReportRequestSet) :	**bcsmEvent** :
MonitorMode (TpCallMonitorMode, section 14)	**monitorMode**
CallReportType (TpCallReportType, section 14)	**eventTypeBCSM**

AdditionalReportCriteria (TpCallReportAdditionalCriteria) **:**	**dPSpecificCriteria** :
noAnswerDuration	**applicationTimer**
serviceCode	
	legID[6]
targetAddress	
originatingAddress	
originalDestinationAddress	
redirectingAddress	
appInfo	
callLegSessionID	

From: ***routeReq***	To: CAP ***Connect***
callSessionID	
responseRequested	
targetAddress	**destinationRoutingAddress**
originatingAddress	
originalDestinationAddress	**originalCalledPartyID**
redirectingAddress	**redirectingPartyID**
appInfo (TpCallAppInfoSet) :	
CallAppAlertingMechanism	**alertingPattern**
CallAppNetworkAccessType	
CallAppInterworkingIndicators	**serviceInteractionIndicatorsTwo**
CallAppTeleService	
CallAppBearerService	
CallAppPartyCategory	**callingPartysCategory**
PresentationAddress	**genericNumbers**[7]
CallAppGenericInfo	
CallAppAdditionalAddress	**genericNumbers**
callLegSessionID	
	redirectionInformation

6 the legID for both the originating and the terminating leg are required for the disconnect event

7 operator specific function if CallAppAdditionalAddress is not used to map the genericNumbers parameter

	suppressionOfAnnouncement
	oCSIApplicable
	na-Info :
	naCarrierInformation
	naOliInfo
	naChargeNumber
	connectArgExtension :
	cug-Interlock
	cug-OutgoingAccess
	nonCug-Call

2. The application does not modify the destination address and does not provide any Application Information

Pre-conditions	The application has been notified of a new call and the call object exists. The ***setCallChargePlan*** and ***getCallInfoReq*** methods may have been invoked
1	The application invokes the ***routeReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***RequestReportBSCM*** if the application needs to be informed about the outcome of the request
4	The gsmSCF sends a CAP ***Continue*** message

Parameter Mapping

From: ***routeReq***	To: CAP ***RequestReportBCSMEvent***
callSessionID	
responseRequested (TpCallReportRequestSet) :	**bcsmEvent** :
MonitorMode (TpCallMonitorMode, section 14)	**monitorMode**
CallReportType (TpCallReportType, section 14)	**eventTypeBCSM**
AdditionalReportCriteria (TpCallReportAdditionalCriteria **:**	**dPSpecificCriteria** :
noAnswerDuration	**applicationTimer**
serviceCode	
	legID[8]
targetAddress	
originatingAddress	

[8] the legID for both the originating and the terminating leg are required for the disconnect event

originalDestinationAddress	
redirectingAddress	
appInfo	
callLegSessionID	

From: ***routeReq***	To: CAP ***Continue***
callSessionID	
responseRequested	
targetAddress	
originatingAddress	
originalDestinationAddress	
redirectingAddress	
appInfo	
callLegSessionID	

3. The application does not modify the destination party number but modifies Application information

Pre-conditions	The application has been notified of a new call and the call object exists. The ***setCallChargePlan*** and ***getCallInfoReq*** methods may have been invoked
1	The application invokes the ***routeReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***RequestReportBSCM*** if the application needs to be informed about the outcome of the request
4	The gsmSCF sends a CAP ***ContinueWithArgument*** message

Parameter Mapping

From: ***routeReq***	To: CAP ***RequestReportBCSMEvent***
callSessionID	
responseRequested (TpCallReportRequestSet) :	**bcsmEvent** :
MonitorMode (TpCallMonitorMode, section 14)	**monitorMode**
CallReportType (TpCallReportType, section 14)	**eventTypeBCSM**
AdditionalReportCriteria (TpCallReportAdditionalCriteria) **:**	**dPSpecificCriteria** :
noAnswerDuration	**applicationTimer**
serviceCode	

	legID[9]
targetAddress	
originatingAddress	
originalDestinationAddress	
redirectingAddress	
appInfo	
callLegSessionID	

From: ***routeReq***	To: CAP ***ContinueWithArgument***
callSessionID	
responseRequested	
targetAddress	
originatingAddress	
originalDestinationAddress	
redirectingAddress	
appInfo :	
CallAppAlertingMechanism	**alerting Pattern**
CallAppNetworkAccessType	
CallAppInterworkingIndicators	**serviceInteractionIndicatorsTwo**
CallAppTeleService	
CallAppBearerService	
CallAppPartyCategory	**callingPartysCategory**
PresentationAddress	**genericNumbers**[10]
CallAppGenericInfo	
CallAppAdditionalAddress	**genericNumbers**
callLegSessionID	
	suppressionOfAnnouncement
	na-Info :
	naCarrierInformation
	naOliInfo
	naChargeNumber

9 the legID for both the originating and the terminating leg are required for the disconnect event

10 operator specific function if CallAppAdditionalAddress is not used to map the genericNumbers parameter

	continueWithArgumentArgExtension :
	cug-Interlock
	cug-OutgoingAccess
	nonCug-Call

6.2.2 routeRes

routeRes is an asynchronous method which indicates that the request to route the call to the destination was successful, and indicates the response of the destination party (for example, the call was answered, not answered, refused due to busy, etc.). For every trigger that was armed in the parameter **responseRequested** of the ***routeReq*** a ***routeRes*** method may be invoked.

Figure 6-10: Call Flow for routeRes

Normal Operation

Pre-conditions	Call routing attempted
1	If event reports have been requested, the gsmSSF sends a CAP ***EventReportBCSM*** to the gsmSCF
2	The gsmSCF sends an equivalent message to the SCS
3	The SCS invokes the ***routeRes*** method

Parameter Mapping

From: CAP ***EventReportBCSM***	To: ***routeRes***
	callSessionID
	eventReport :
miscCallInfo	**MonitorMode**
	CallEventTime
eventTypeBCSM **legID**	**CallReportType** (TpCallReportType, section14)
eventSpecificInformationBCSM	**AdditionalReportInfo (**TpCallAdditionalReportInfo, section 14)
	callLegSessionID

6.2.3 routeErr

routeErr is an asynchronous method which indicates that the request to route the call to the destination party was unsuccessful – the call could not be routed to the destination party (for example, the network was unable to route the call, parameters were incorrect, the request was refused, etc).

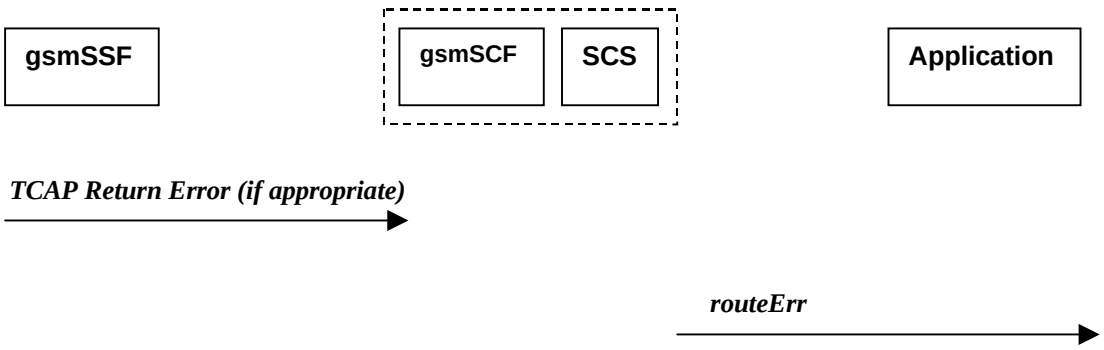


Figure 6-11: Call Flow for routeErr

Normal Operation

Two scenarios are possible

1. The gsmSCF receives a message from the gsmSSF indicating an error

Pre-conditions	**Call routing attempted**
1	The gsmSSF detects a call routing failure and sends an appropriate TCAP message returning an error to the gsmSCF

2	The gsmSCF sends an equivalent message to the SCS
3	The SCS detects an error with the ***routeReq*** method, or receives a TCAP Return Error, and invokes the ***routeErr*** method

2. The gsmSCF detects there is an error in the message from the SCS

Pre-conditions	**Call routing attempted**
1	The gsmSCF detects an error in the parameters of the internal message from the SCS requesting a ***routeReq***
2	The gsmSCF sends an equivalent message to the SCS
3	The SCS invokes the ***routeErr*** method

Parameter Mapping

From: TCAP ***Return Error***	To: ***routeErr***
	callSessionID
TC-U-ERROR TC-U-REJECT	**error**
	callLegSessionID

6.2.4 release

release is a method used to request the release of the call and associated objects.

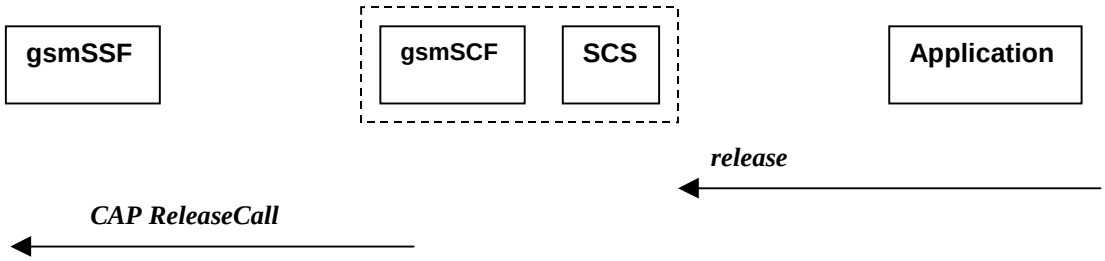


Figure 6-12: Call Flow for release

Normal Operation

Pre-conditions	**Call is in progress**
1	The application invokes the ***releaseCall*** method
2	The SCS sends an equivalent message to the gsmSCF
3	The gsmSCF invokes the CAP ***ReleaseCall*** operation

Parameter Mapping

From: ***release***	To: ***CAP ReleaseCall***

callSessionID	
cause (TpCallReleaseCause) :	
value (specified in ITU-T Q.850)	**Cause**
location	

6.2.5 deassignCall

deassignCall is a method that requests that the relationship between the application and the call and associated objects be de-assigned. It leaves the call in progress, however, it purges the specified call object so that the application has no further control of call processing. If a call is de-assigned that has event reports or call information reports requested, then these reports will be disabled and any related information discarded.

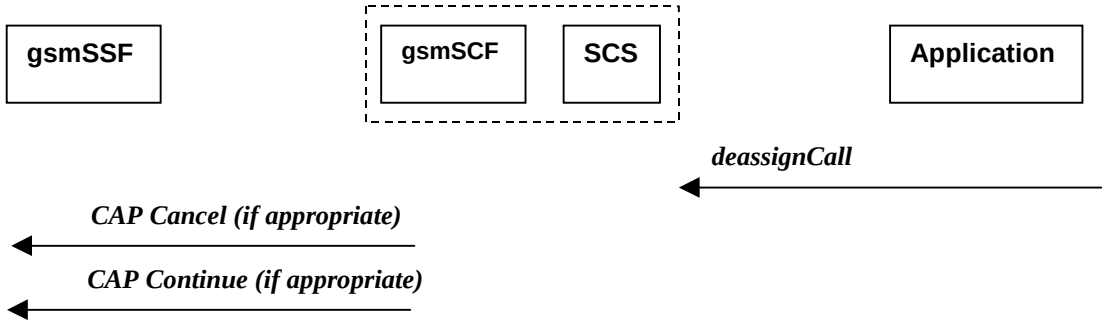


Figure 6-13: Call Flow for deassignCall

Normal Operation

Pre-conditions	
1	The application invokes the ***deassignCall*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***Cancel*** operation to the gsmSSF if there are any reports pending.
4	The gsmSCF may send a CAP ***Continue*** to allow the interrupted call processing to continue. This is not sent if the call has already been established.

Parameter Mapping

From: ***deassignCall***	To: CAP ***Cancel***
	AllRequests
callSessionID	

From: ***deassignCall***	To: CAP ***Continue***
callSessionID	

6.2.6 getCallInfoReq

getCallInfoReq is an asynchronous method that requests information associated with the call to be provided at the appropriate time (for example, to calculate charging). This method must be invoked before the call is routed to a target address. The call object will exist after the call is ended if information is required to be sent to the application at the end of the call. The information will be sent after any call event report.

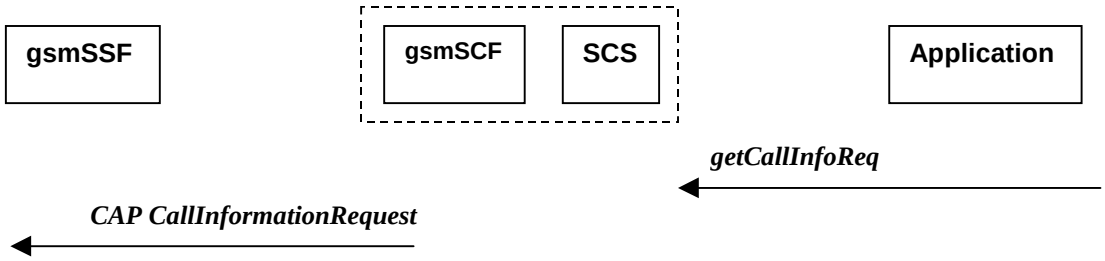


Figure 6-14: Call Flow for getCallInfoReq

Normal Operation

Pre-conditions	
1	The application invokes the ***getCallInfoReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***CallInformationRequest*** operation to the gsmSSF

Parameter Mapping

From: ***getCallInfoReq***	To: CAP ***CallInformationRequest***
callSessionID	
	RequestedInformationTypeList
callInfoRequested (TpCallInfoType) :	**RequestedInformationType**
P_CALL_INFO_UNDEFINED	
P_CALL_INFO_TIMES	**callAttemptElapsedTime** **callStopTime** **callConnectedElapsedTime**
P_CALL_INFO_RELEASE_CAUSE	**releaseCause**
P_CALL_INFO_INTERMEDIATE	
	LegID

6.2.7 getCallInfoRes

getCallInfoRes is an asynchronous method that reports all the necessary information requested by the application, for example to calculate charging.

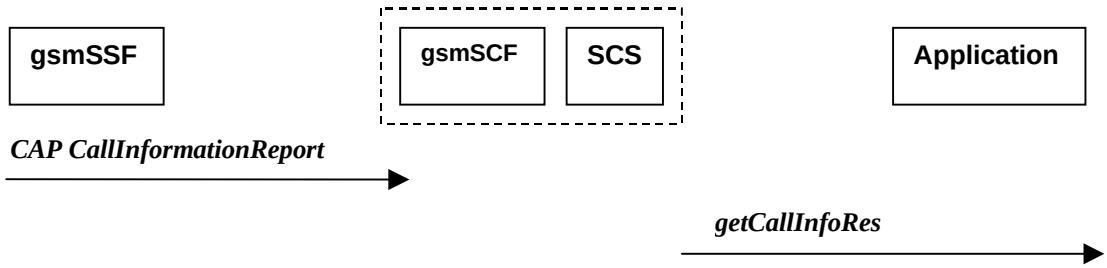


Figure 6-15: Call Flow for getCallInfoRes

Normal Operation

Pre-conditions	Call is in progress
1	The gsmSCF receives a CAP ***CallInformationReport*** from the gsmSSF.
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS identifies the correct application and invokes the ***getCallInfoRes*** method

Parameter Mapping

From: CAP ***CallInformationReport***	To: ***getCallInfoRes***
	callSessionID
requestedInformationList	**callInfoReport** :
requestedInformationType :	**CallInfoType**
	P_CALL_INFO_UNDEFINED
callAttemptElapsedTime **callStopTime** **callConnectedElapsedTime**	**P_CALL_INFO_TIMES**
releaseCause	**P_CALL_INFO_RELEASE_CAUSE**
	P_CALL_INFO_INTERMEDIATE
requestedInformationValue :	
	CallInitiationStartTime
callStopTimeValue	**CallEndTime**
	CallConnectedToResourceTime
	CallConnectedToDestinationTime
releaseCauseValue	**Cause**
LegID	

6.2.8 getCallInfoErr

getCallInfoErr is an asynchronous method that reports that the original request was erroneous, or resulted in an error condition.

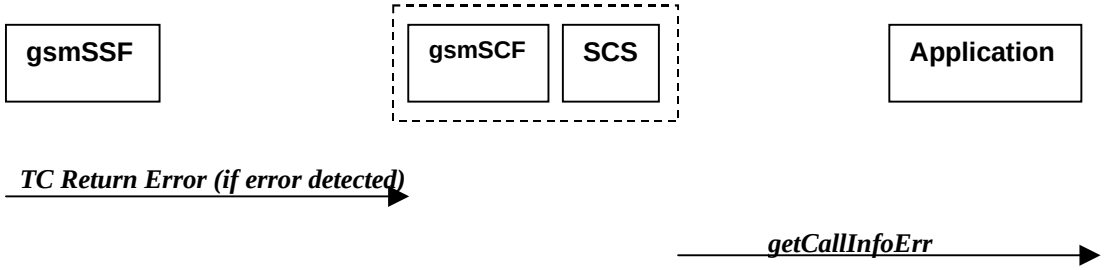


Figure 6-16: Call Flow for getCallInfoErr

Normal Operation

Pre-conditions	The application has requested information associated with a call via the ***getCallInfoReq*** method
1	A call terminates abnormally and the gsmSSF sends an error in a TCAP message to the gsmSCF , or aborts the TCAP dialogue
2	The gsmSCF sends an equivalent message to the SCS
3	The SCS identifies the correct applications that requested the call information and invokes the ***getCallInfoErr*** method.

Parameter Mapping

From:	To: ***getCallInfoErr***
	callSessionID
TC Primitives TC-U-ABORT TC-P-ABORT TC-NOTICE TC-U-ERROR TC-L-CANCEL TC-U-CANCEL TC-L-REJECT TC-R-REJECT TC-U-REJECT	**errorIndication**

6.2.9 superviseCallReq

superviseCallReq is a method that is called by the application to supervise a call. The application can set a granted connection time for this call. If an application calls this method before it calls a ***routeReq()*** or a user interaction method the time measurement will start as soon as the call is answered by the B-party or the user interaction system.

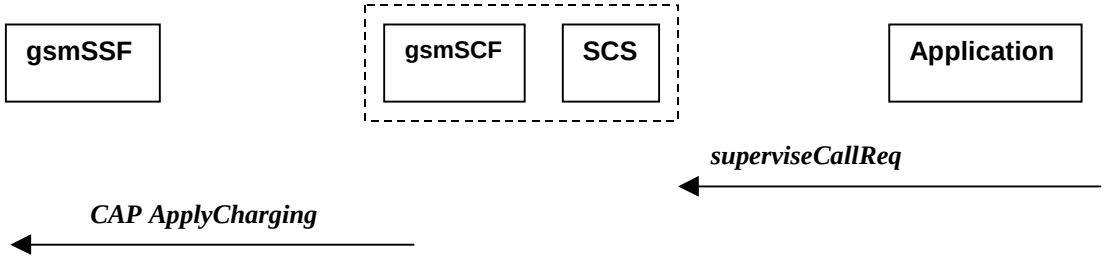


Figure 6-17: Call Flow for superviseCallReq

Normal Operation

Pre-conditions	
1	The application invokes the ***superviseCallReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***ApplyCharging*** message to the gsmSSF

Parameter Mapping

From: ***superviseCallReq***	To: CAP ***ApplyCharging***
callSessionID	
	AchBillingCharging Characteristics :
time	**timeDurationCharging** - **maxCallPeriodDuration**
	- **tariffSwitchInterval**
treatment (TpCallSuperviseTreatment) :	**timeDurationCharging**
P_CALL_SUPERVISE_RELEASE	- **releaseIfdurationExceeded**
P_CALL_SUPERVISE_RESPOND	
P_CALL_SUPERVISE_APPLY_TONE	- **tone**
	PartyToCharge

6.2.10 superviseCallRes

superviseCallRes is an asynchronous method that reports a call supervision event to the application.

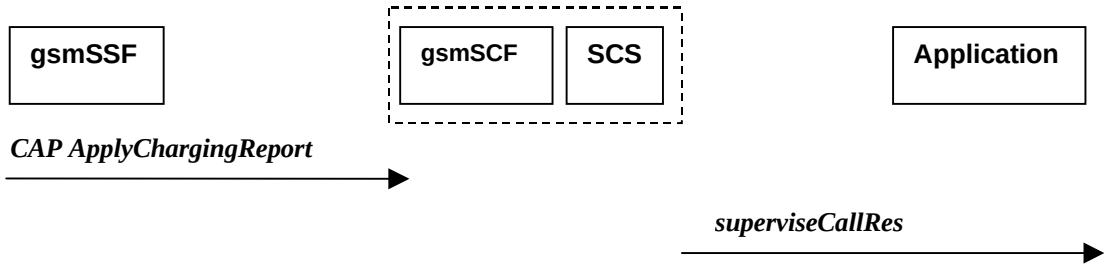


Figure 6-18: Call Flow for superviseCallRes

Normal Operation

Pre-conditions	The application has invoked the supervise Call method
1	The gsmSCF receives an CAP ***ApplyChargingReport*** from the gsmSSF
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS identifies the correct application and invokes the ***superviseCallRes*** method.

Parameter Mapping

From: CAP ***ApplyChargingReport***	To: ***superviseCallRes***
	callSessionID
CallResult	**report** (TpCallSuperviseReport) :
- **CallReleasedAtTcpExpiry**	- **P_CALL_SUPERVISE_TIMEOUT**
- **CallActive**	- **P_CALL_SUPERVISE_CALL_ENDED**
	- **P_CALL_SUPERVISE_TONE_APPLIED**
	- **P_CALL_SUPERVISE_UI_FINISHED**
CallResult - **TimeInformation**	**usedTime**
CallResult - **PartyToCharge**	

6.2.11 superviseCallErr

superviseCallErr is an asynchronous method that reports a call supervision error to the application.

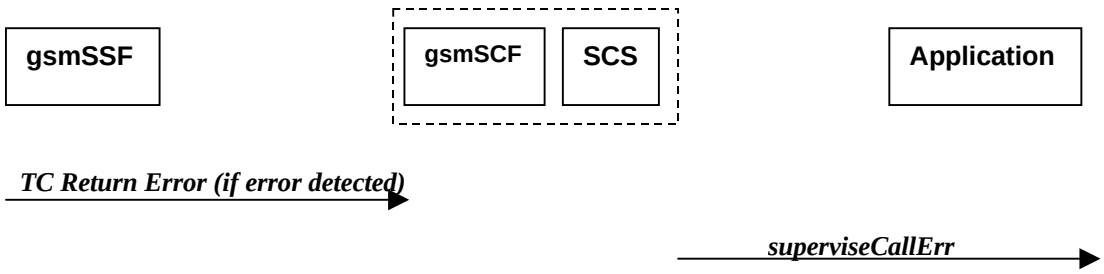


Figure 6-19: Call Flow for superviseCallErr

Normal Operation

Pre-conditions	The application has requested information associated with a call via the ***superviseCallReq*** method

1	A call terminates abnormally and the gsmSSF sends an error in a TCAP message to the gsmSCF , or aborts the TCAP dialogue
2	The gsmSCF sends an equivalent message to the SCS
3	The SCS identifies the correct applications that requested the call information and invokes the ***superviseCallErr*** method.

Parameter Mapping

From:	To: ***superviseCallErr***
	callSessionID
TC Primitives TC-U-ABORT TC-P-ABORT TC-NOTICE TC-U-ERROR TC-L-CANCEL TC-U-CANCEL TC-L-REJECT TC-R-REJECT TC-U-REJECT	**errorIndication**

6.2.12 setAdviceOfCharge

setAdviceOfCharge is a method that allows the application to determine the charging information that will be send to the end-users terminal.

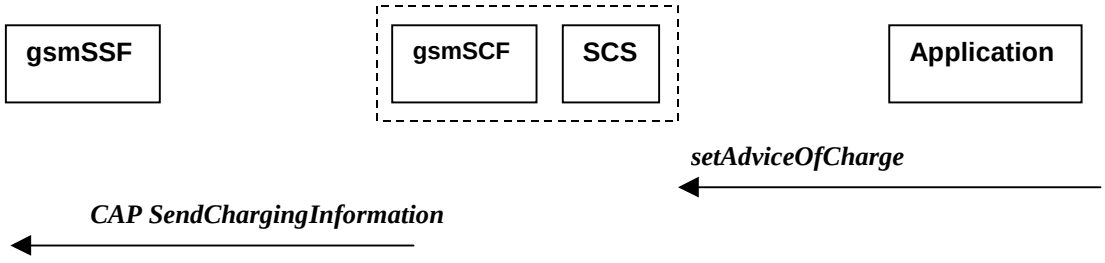


Figure 6-20: Call Flow for setAdviceOfCharge

Normal Operation

Pre-conditions	
1	The application invokes the ***setAdviceOfCharge*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***SendChargingInformation*** message to the SSF

Parameter Mapping

From: ***setAdviceOfCharge***	To: CAP ***SendChargingInformation***
callSessionID	
aOCInfo: - **CurrentCAI**	**SCIBillingChargingCharateristics** **aOCBeforeAnswer** **aOCInitial** **- or -** **SCIBillingChargingCharateristics** **aOCAfterAnswer** **cAI-GSM0224**
- **NextCAI**	**SCIBillingChargingCharateristics** **aOCBeforeAnswer** **aOCSubsequent** **cAI-GSM0224**
tariffSwitch	**SCIBillingChargingCharateristics** **aOCBeforeAnswer** **aOCSubsequent** **tariffSwitchInterval** **- or -** **SCIBillingChargingCharateristics** **aOCAfterAnswer** **tariffSwitchInterval**
	partyToCharge

6.2.13 setCallChargePlan

setCallChargePlan is a method that allows the application to include charging information in network generated CDR.

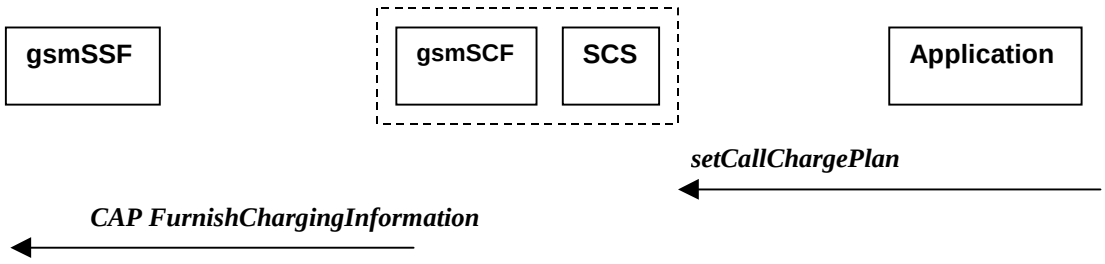


Figure 6-21: Call Flow for setCallChargePlan

Normal Operation

Pre-conditions	
1	The application invokes the ***setCallChargePlan***
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***FurnishChargingInformation*** message to the SSP

Parameter Mapping

From: ***setCallChargePlan***	To: CAP ***FurnishChargingInformation***
callSessionID	
callChargePlan **ChargeOrderType (choice)** **ChargePerTime** **InitialCharge** **CurrentChargePerMinute** **NextChargePerMinute** **NetworkCharge** **Currency** **AdditionalInfo**	**FCIBillingChargingCharacteristics** **fCIBCCCAMELsequence1** **freeFormatData**
	FCIBillingChargingCharacteristics **fCIBCCCAMELsequence1** **partyToCharge**
	FCIBillingChargingCharacteristics **fCIBCCCAMELsequence1** **appendFreeFormatData**

An alternative scenario would be to map ***setCallChargePlan*** method to the CAP ***ApplyCharging*** protocol operation.

6.2.14 callFaultDetected

callFaultDetected indicates to the application that a fault has been detected in the call.

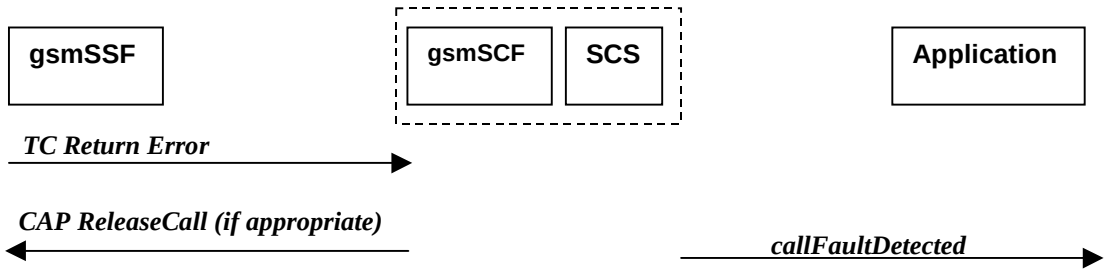


Figure 6-22: Call Flow for callFaultDetected

Normal Operation

Pre-conditions	A call exists and the SCS detects an error. No ***routeReq*** method has been invoked yet.
1	The gsmSSF may detect a fault and sends an appropriate dialogue error message to the gsmSCF
2	The gsmSCF may detect a fault an send an error message to the SCS
3	The SCS detects a fault and invokes the ***callFaultDetected*** method
4	The SCS sends an equivalent message to the gsmSCF if appropriate
5	The gsmSCF sends a CAP ***ReleaseCall*** if appropriate

Parameter Mapping

From: ***Dialogue Error***	To: ***callFaultDetected***
	callSessionID
TC_U_ABORT	**fault**

6.2.14 callEnded

callEnded will be invoked when the call has ended. Furthermore, the operation contains an indication on the reason why the call has been ended. Also the operation will always be invoked when the call has ended and not only when the application has requested its interest in this event.

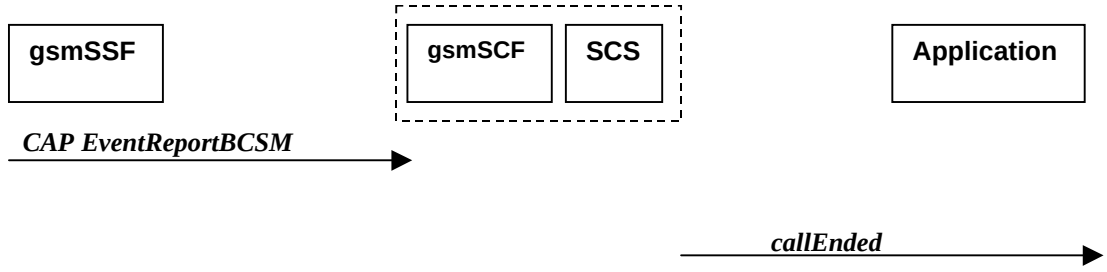


Figure 6-23: Call Flow for callEnded

Normal Operation

Pre-conditions	There is an application monitoring the call in some way.
1	The gsmSSF detects a release from the calling or called party leg. CAP eventReportBCSM is sent if requested by the gsmSCF The BCSM event indicated may be either abandon or disconnect depending on the phase of the call.

2	The gsmSCF sends an equivalent message to the SCS
3	The SCS invokes the ***callEnded*** method.

Parameter Mapping

From: ***CAP EventReportBCSM***	To: ***callEnded***
	callSessionID
eventTypeBCSM	
	report
legID	**callLegSessionID**
eventSpecificInformationBCSM: **releaseCause**	**cause**
miscCallInfo	

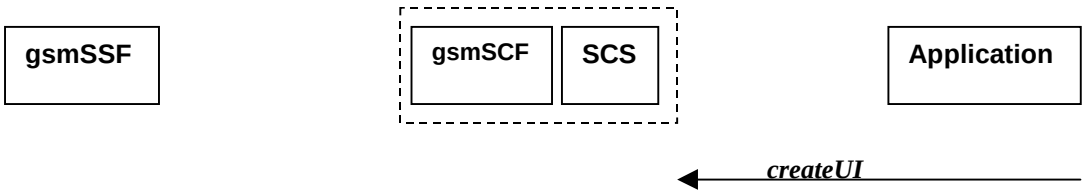
7 Generic Message Transfer Service CAMEL Call Flows

7.1 User Interaction

The User Interaction interface is used by applications to interact with end users. The API only supports Call User Interaction.

7.1.1 createUI

createUI is a method that is used to create a new (non call related) user interaction object.



Note: There are no associated CAP call flows

Figure 7-1: Call Flow for createUI

Normal Operation

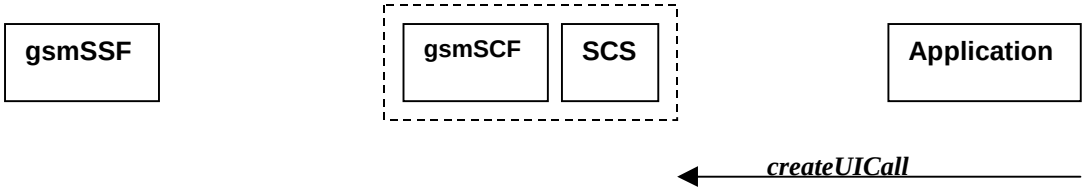
Pre-conditions	The application has been instructed to initiate a non call related User Interaction
1	The application invokes the ***createUI*** method
2	The SCS creates a new UI object

Parameter Mapping

None.

7.1.2 createUICall

createUICall is a method that is used to create a new call related user interaction object.



Note: There are no associated CAP call flows

Figure 7-2: Call Flow for createUICall

Normal Operation

Pre-conditions	The application has been requested to initiate a call related User Interaction
1	The application invokes the ***createUICall*** method
2	The SCS creates a new ***UICall*** object

Parameter Mapping

None.

7.1.3 enableUINotification

enableUINotification is a method that enables the reception of a user initiated user interaction.

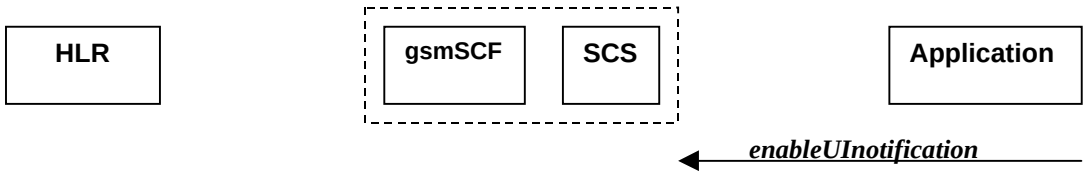


Figure 7-3: Call Flow for enableUINotification

Normal Operation

Pre-conditions	An agreement is established between the network operator and the service provider for the event notification to be enabled
1	The application invokes the ***enableUINotification*** method
2	The SCS stores the request.

Parameter Mapping

None.

7.1.4 disableUINotification

disableUINotification is a method that allows the application to remove notification for UI related actions previously set.

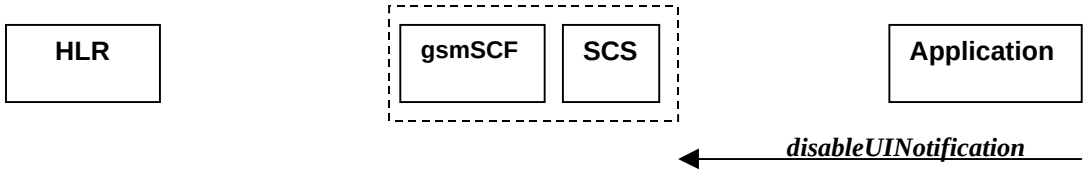


Figure 7-4: Call Flow for disableUINotification

Normal Operation

Pre-conditions	An agreement is established between the network operator and the service provider for the event notification to be disabled
1	The application invokes the ***disableUINotification*** method
2	The request is disabled in the SCS.

Parameter Mapping

None.

7.1.5 userInteractionEventNotify

userInteractionEventNotify is a method that notifies the application of a user initiated request for user interaction.

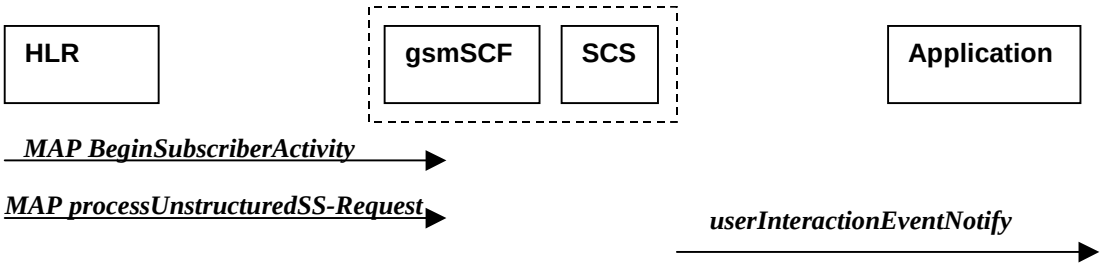


Figure 7-5: Call Flow for userInteractionEventNotify

1.Normal Operation

Pre-conditions	
1	The gsmSCF receives a MAP ***processUnstructuredSS-Request*** message from the HLR. This operation may be preceded by MAP ***beginSubscriberActivity***[11].
2	The gsmSCF sends an equivalent internal message to the SCS

[11] The MAP beginSubscriberActivity is sent in case of MAP version 1.

3	The SCS identified the correct application that enable the notification request from the subscriber and invokes the ***userInteractionEventNotify*** method

Parameter Mapping

From: **processUnstructuredSS-Request**	To: ***userInteractionEventNotify***
	ui
	eventInfo (TpCallEventInfo) :
msisdn	**OriginatingAddress**
	DestinationAddress
	ServiceCode
	DataTypeIndication
ussd-DataCodingScheme **ussd-String**	**DataString**
	assignmentID
	appInterface (output)

7.1.6 userInteractionAborted

userInteractionAborted is a method that indicates to the application that the User Interaction service instance has terminated or closed abnormally. No further communication will be possible between the User Interaction service instance and the application.

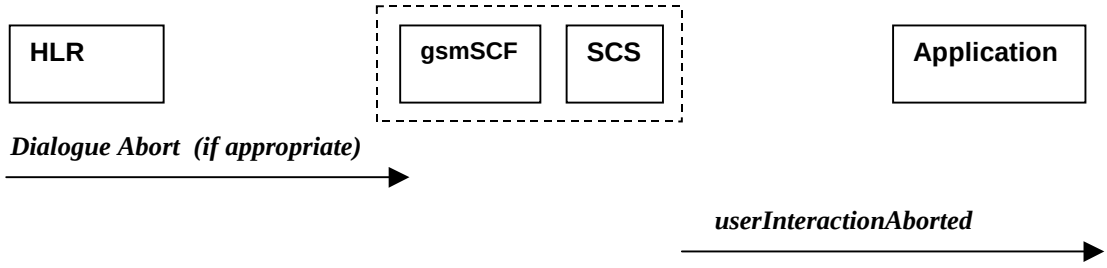


Figure 7-6: Call Flow for userInteractionAborted (scenario 1)

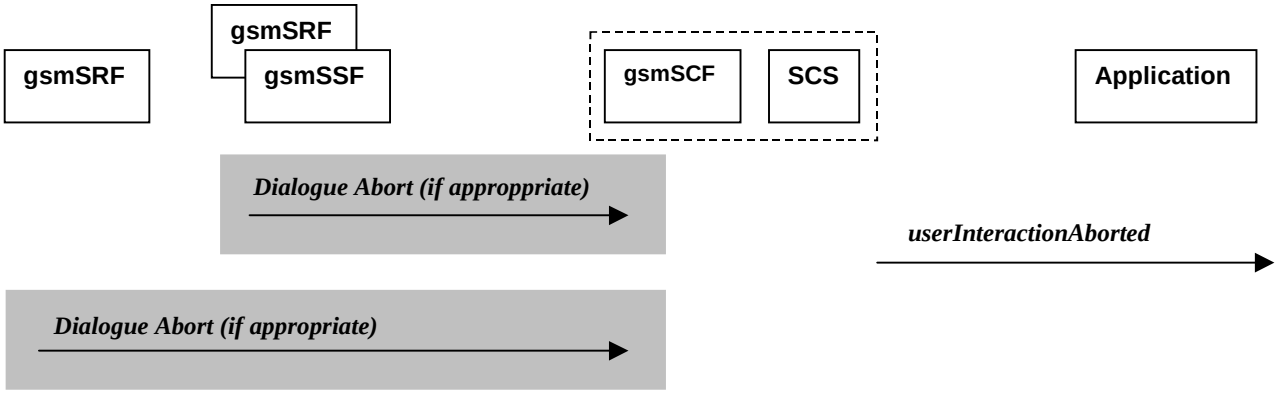


Figure 7-7: Call Flow for userInteractionAborted (scenario 2)

Parameter Mapping

None.

7.1.7 userInteractionNotificationInterrupted

userInteractionNotificationInterrupted is a method that indicates to the application that all user interaction event notifications have been temporarily interrupted.

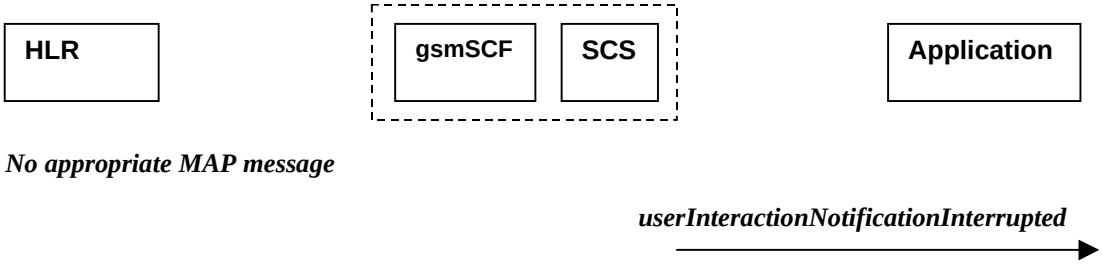


Figure 7-8: Call Flow for userInteractionNotificationInterrupted

Normal Operation

Pre-conditions	User interaction event notifications have been enabled using the ***enableUINotification*** method on the UIManager interface
1	The SCS has detected, or has been informed of, a fault which prevents further user interaction events from being notified
2	The SCS invokes the ***userInteractionNotificationInterrupted*** method

Parameter Mapping

None.

7.1.8 userInteractionNotificationContinued

userInteractionNotificationContinued is a method that indicates to the application that user interaction event notifications will again be possible.

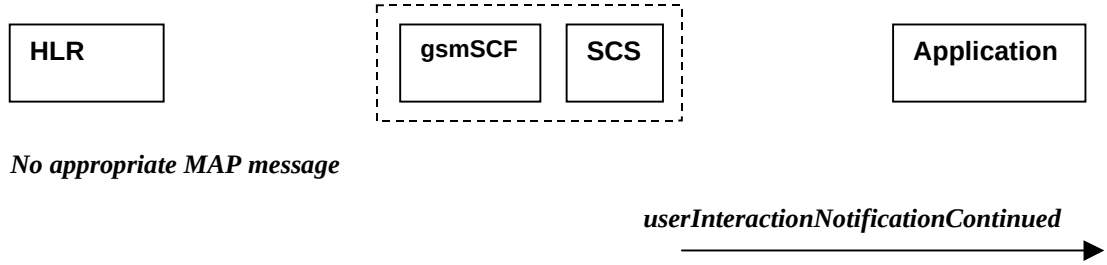


Figure 7-9: Call Flow for userInteractionNotificationContinued

Normal Operation

Pre-conditions	User interaction event notifications have been interrupted and ***userInteractionNotificationInterrupted*** method has been invoked**.**
1	The SCS detects that user interaction event notifications are again possible.
2	The SCS invokes the ***userInteractionNotificationContinued*** method.

Parameter Mapping

None.

7.1.9 userInteractionFaultDetected

userInteractionFaultDetected is a method that indicates to the application that a fault has been detected in the user interaction. This method is invoked e.g. if the call has been deassigned.

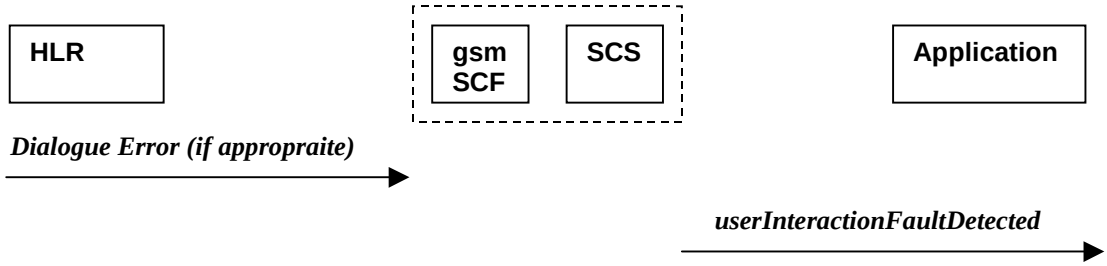


Figure 7-10: Call Flow for userInteractionFaultDetected (scenario 1)

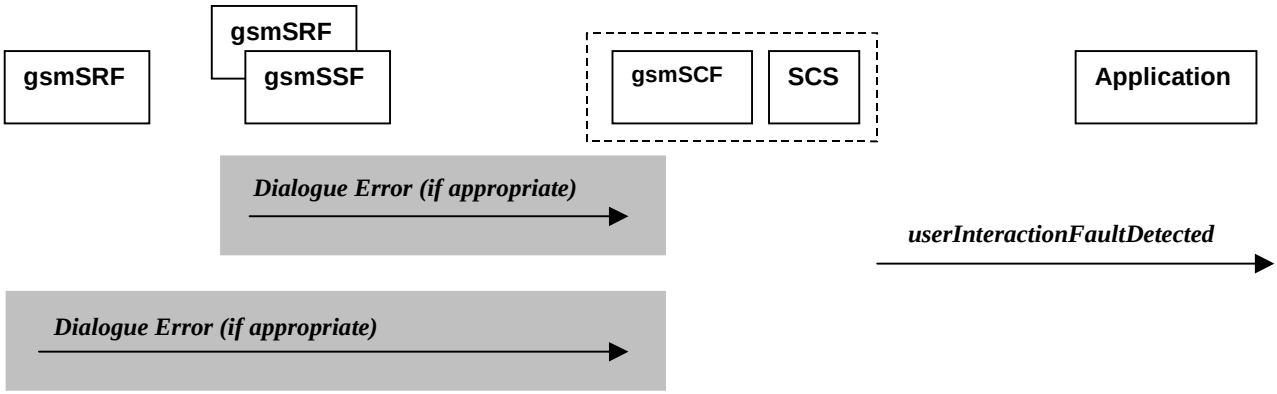


Figure 7-11: Call Flow for userInteractionFaultDetected (scenario 2)

Normal Operation

Three Alternatives have been identified
1. USSD based interaction between the MS and the gsmSCF

Pre-conditions	USSD interaction is in progress and a dialogue is running between the HLR and gsmSCF
1	The gsmSCF detects or receives an indication that the there is an error in the user interaction
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS invokes the ***userInteractionFaultDetected*** method to the appropriate application

2. Interaction between a gsmSRF internal to the gsmSSF and the gsmSCF

Pre-conditions	User interaction is in progress between the gsmSRF and the gsmSCF
1	The gsmSCF detects or receives an indication that there is an error in the user interaction
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS invokes the ***userInteractionFaultDetected*** method to the appropriate application

3. Interaction between a gsmSRF and the gsmSCF

Pre-conditions	User interaction is in progress between the gsmSRF and the gsmSCF
1	The gsmSCF detects or receives an indication that the there is an error in the user interaction
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS invokes the ***userInteractionFaultDetected*** method to the appropriate application

Parameter Mapping

From: Dialogue Error	To: ***userInteractionFaultDetected***
	userInteractionIdentifier
	fault
ReturnError	

7.1.10 sendInfoReq

sendInfoReq is an asynchronous method that sends information to the user.

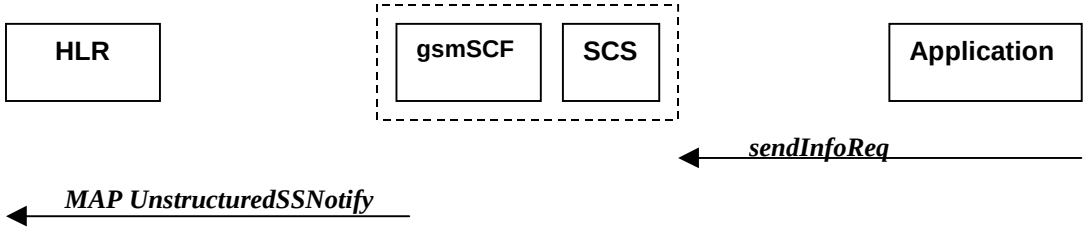


Figure 7-12: Call Flow for sendInfoReq (scenario 1)

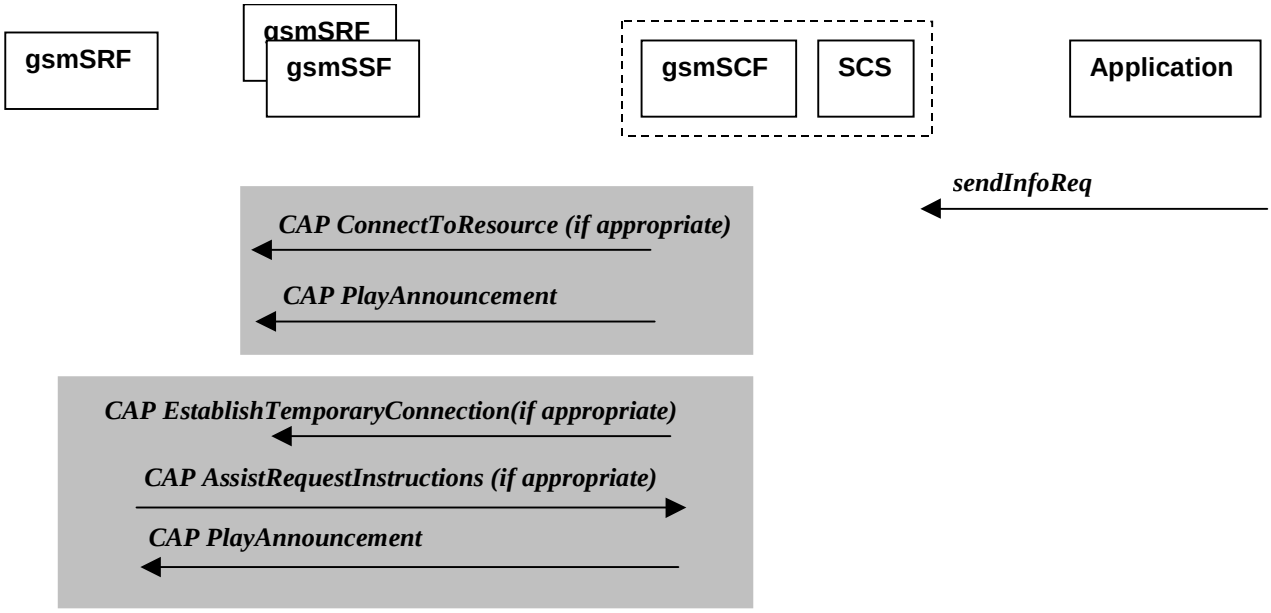


Figure 7-13: Call Flow for sendInfoReq (scenario 2)

Normal Operation

Three Alternatives have been identified

1. USSD based interaction between the MS and the gsmSCF

Pre-conditions	USSD interaction
1	The application invokes the sendInfo method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a MAP ***UnstructuredSS-Notify*** message to the HLR. If processUnstructuredSS-Request was previously received its result component may be sent containing ussd-DataCodingScheme and ussd-String. Note : For call-related USSD cases, the USSD is sent to the calling party.

2. Interaction between a gsmSRF internal to the gsmSSF and the gsmSCF

Pre-conditions	
1	The application invokes the ***sendInfoReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF is aware of a gsmSRF internal to the gsmSSF. The gsmSCF sends CAP ***ConnectToResource***, and CAP ***PlayAnnouncement*** messages the gsmSSF Note : The user interaction shall apply to all parties connected to the call segment for the user interactions initiated by the connectToResource and establishTemporaryConnection operations.

3. Interaction between a gsmSRF internal to the gsmSSF and the gsmSCF

Pre-conditions	
1	The application invokes the ***sendInfoReq*** method

2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF is aware of an external gsmSRF. The gsmSCF sends CAP ***EstablishTemporaryConnection*** message the gsmSSF.
4	On receipt of the CAP ***AssistRequestInstructions*** message from the gsmSRF, the gsmSCF sends the CAP ***PlayAnnouncement*** message to the gsmSRF. Note : The user interaction shall apply to all parties connected to the call segment for the user interactions initiated by the ***connectToResource*** and ***establishTemporaryConnection*** operations.

Parameter Mapping

From: ***sendInfoReq***	To: MAP ***unstructuredSS-Notify***
userInteractionSessionID	
info (choice)	
infoID	
infoData	**ussd-DataCodingScheme** **ussd-String**
infoAddress	
variableInfoSet	
repeatIndicator	
responseRequested	
assignmentID	
	alertingPattern
	msisdn

From: ***sendInfoReq***	To: CAP ***PlayAnnouncement***
userInteractionSessionID	
info (choice) **infoID**	**InformationToSend** (choice) **inbandInfo** **messageID** (choice) **elementaryMessageID** **text** **messageContent** **attributes** **elementaryMessageIDs** **variableMessage** **elementaryMessageID** **variableParts** (sequence of the following choices**)**

	integer **number** **time** **date** **price** **numberOfRepetitions** **duration** **interval** **tone** **toneID** **duration**
infoData	
infoAddress	
variableInfoSet	The contents are directly mapped to **variableParts** above
variablePartInteger	
variablePartAddress	
variablePartTime	
variablePartDate	
variablePartPrice	
repeatIndicator	This integer value is directly mapped to **numberOfRepetitions** above
	disconnectFromIPForbidden (according to **responseRequested**)
responseRequested	**requestAnnouncementComplete**
assignmentID	

7.1.11 sendInfoRes

sendInfoRes is an asynchronous method that informs the application about the start or the completion of a ***sendInfoReq()***. This response is called only if the application has requested a response.

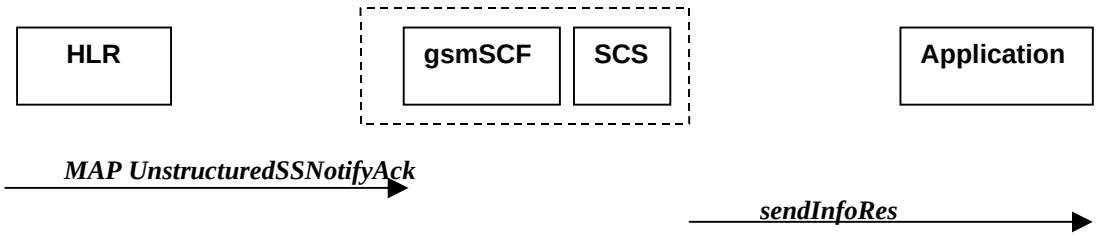


Figure 7-14: Call Flow for sendInfoRes (scenario 1)

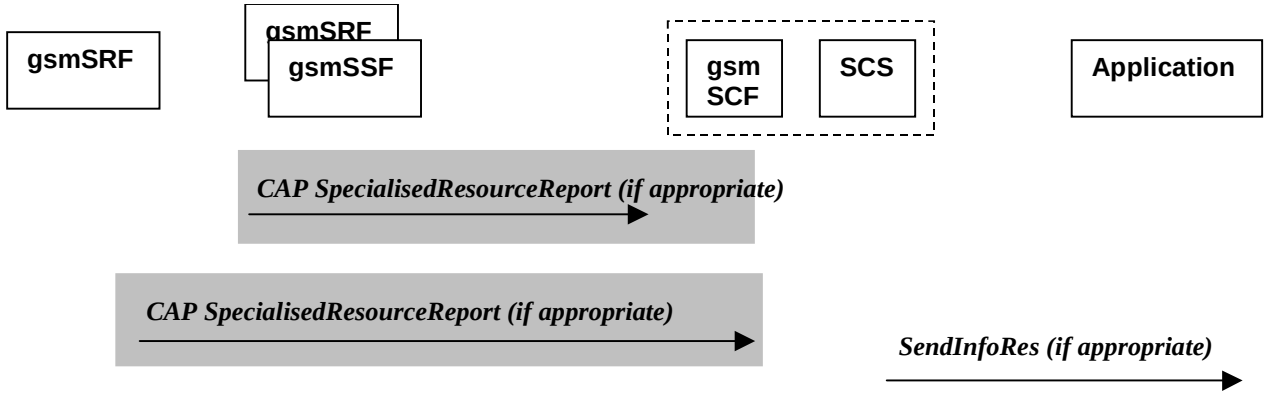


Figure 7-15: Call Flow for sendInfoRes (scenario 2)

Normal Operation

Three Alternatives have been identified

1. USSD based interaction between the MS and the gsmSCF

Pre-conditions	The application has previously invoked the ***sendInfoReq*** method and has requested a notification
1	The gsmSCF receives an MAP unstructured***SS-Notify acknowledgement*** message from the HLR
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS identifies the correct application and invokes the ***sendInfoRes*** method

2. Interaction between a gsmSRF internal to the gsmSSF and the gsmSCF

Pre-conditions	The application has previously invoked the ***sendInfoReq*** method and has requested a notification
1	The gsmSCF receives a CAP ***SpecialisedResourceReport*** message from the gsmSSF indicating that the announcement has been played to the subscriber
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS identifies the correct application and invokes the ***sendInfoRes*** method

3. Interaction between a gsmSRF internal to the gsmSSF and the gsmSCF

Pre-conditions	The application has previously invoked the ***sendInfoReq*** method and has requested a notification
1	The gsmSCF receives a CAP ***SpecialisedResourceReport*** message from the gsmSRF indicating that the announcement has been played to the subscriber
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS identifies the correct application and invokes the ***sendInfoRes*** method

Parameter Mapping

From: CAP ***SpecialisedResourceReport***	To: ***sendInfoRes***
	userInteractionSessionID
	assignmentID
	response

7.1.12 sendInfoErr

sendInfoErr is an asynchronous method that indicates that the request to send information was unsuccessful.

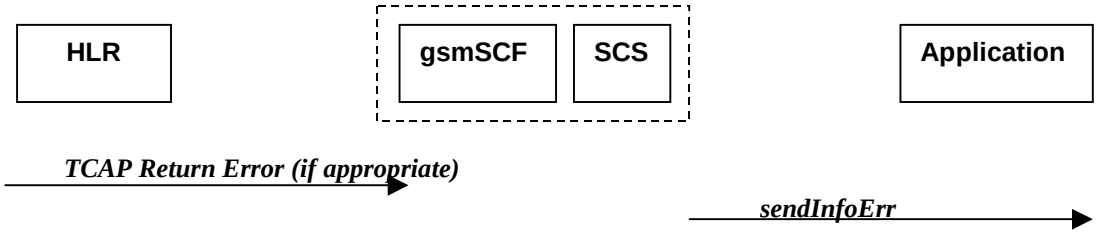


Figure 7-16: Call Flow for sendInfoErr (scenario 1)

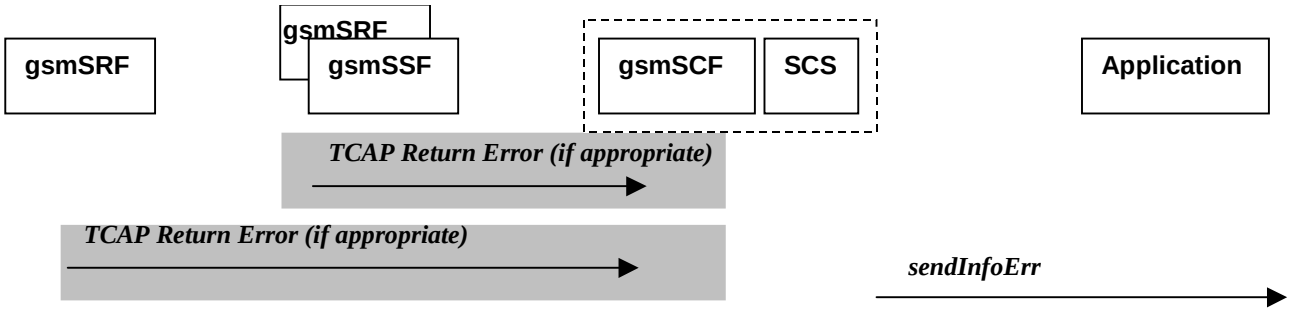


Figure 7-17: Call Flow for sendInfoErr (scenario 2)

Normal Operation

For:

1. USSD based interaction between the MS and the CSE

2. Interaction between a gsmSRF internal to the gsmSSF and the CSE

3. Interaction between a gsmSRF internal to the gsmSSF and the CSE

Pre-conditions	The application has previously invoked the ***sendInfoReq*** method
1	The gsmSCF receives a message from the either the HLR, the gsmSSF or the gsmSRF indicating an error in the previous ***sendInfoReq*** method. Alternatively the gsmSCF may internal detect that the application has incorrectly sent the information
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS identifies the correct application and invokes the ***sendInfoErr*** method

Parameter Mapping

From: TCAP ***Return Error***	To: ***sendInfoErr***
	userInteractionSessionID
InvokeID	**assignmentID**
Error	**error**

7.1.13 sendInfoAndCollectReq

sendInfoAndCollectReq is an asynchronous method that plays an announcement or sends other information to the user and collects some information from the user. The announcement usually prompts for a number of characters (for example, these are digits or text strings such as "YES" if the user's terminal device is a phone).

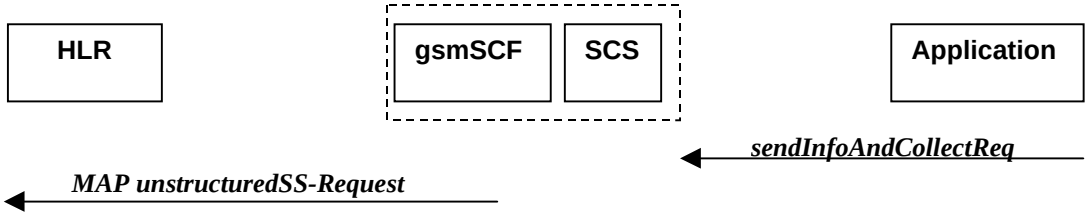


Figure 7-18: Call Flow for sendInfoAndCollectReq (scenario 1)

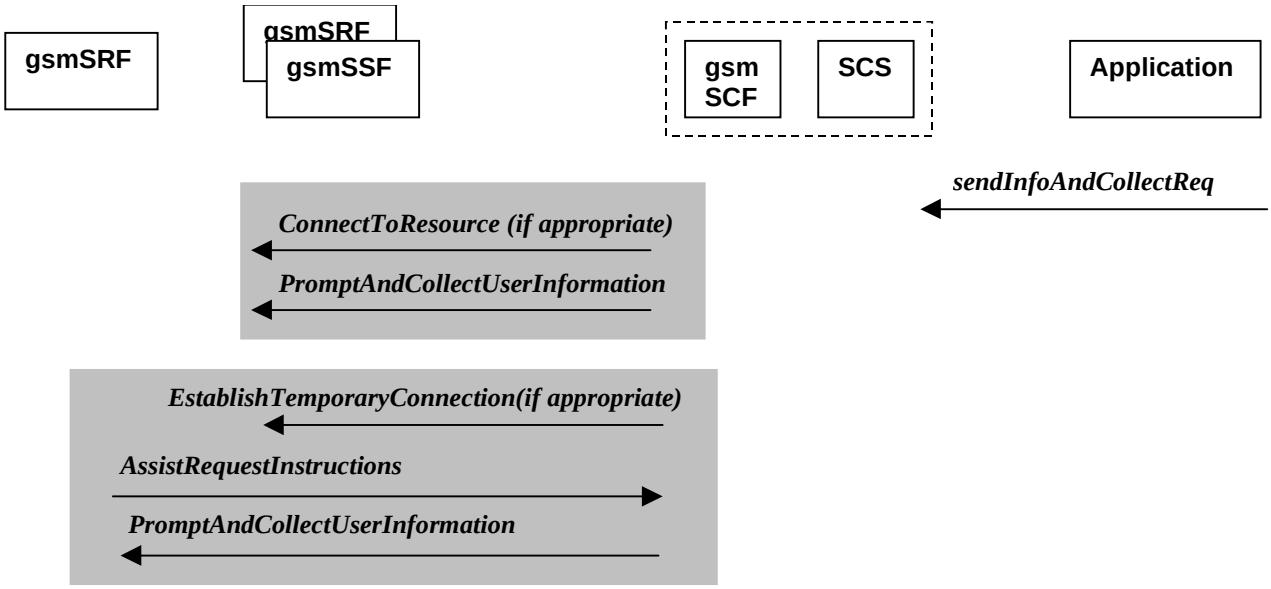


Figure 7-19: Call Flow for sendInfoAndCollectReq (scenario 2)

Normal Operation

Three Alternatives have been identified
1. USSD based interaction between the MS and the gsmSCF

Pre-conditions	USSD interaction
1	The application invokes the ***sendInfoAndCollectReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a MAP ***unstructuredSS-Request message.***

2. Interaction between a gsmSRF internal to the gsmSSF and the gsmSCF

Pre-conditions	
1	The application invokes the ***sendInfoAndCollectReq*** method

2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF is aware of a gsmSRF internal to the gsmSSF. The gsmSCF sends CAP ***ConnectToResource*** and ***PromptAndCollectUserInformation*** messages the gsmSSF

3. Interaction between a gsmSRF internal to the gsmSSF and the gsmSCF

Pre-conditions	
1	The application invokes the ***sendInfoAndCollectReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF is aware of an external gsmSRF. The gsmSCF sends CAP ***EstablishTemporaryConnection***, message the gsmSSF.
4	On receipt of the CAP ***AssistRequestInstructions*** message from the gsmSRF, the gsmSCF sends the CAP ***PromptAndCollectUserInformation*** message to the gsmSRF

Parameter Mapping

From: ***sendInfoAndCollectReq***	**To: MAP *unstructuredSS-Request***
userInteractionSessionID	
info (choice)	
infoID	
infoData	**ussd-DataCodingScheme** **ussd-String**
infoAddress	
variableInfo	
criteria	
responseRequested	
	alertingPattern
	msisdn
assignmentID	

From: ***sendInfoAndCollectReq***	To: CAP ***PromptAndCollectUserInformation***
userInteractionSessionID	
	disconnectFromIPForbidden (always true)
info (choice)	
infoID	**InformationToSend** (choice) **inbandInfo** **messageID** (choice)

	elementaryMessageID **text** **messageContent** **attributes** **elementaryMessageIDs** **variableMessage** **elementaryMessageID** **variableParts** (sequence of the following choices**)** **integer** **number** **time** **date** **price** **numberOfRepetitions** **duration** **interval** **tone** **toneID** **duration**
infoData	
infoAddress	
variableInfo	The contents are directly mapped to **variableParts** above
variablePartInteger	
variablePartAddress	
variablePartTime	
variablePartDate	
variablePartPrice	
criteria	**collectedInfo**
	collectedDigits
minLength	**minimumNbOfDigits**
maxLength	**maximumNbOfDigits**
endSequence	**endOfReplyDigit**
	cancelDigit
	startDigit
startTimeout	**firstDigitTimeOut**
interCharTimeout	**interDigitTimeOut**

	errorTreatment
	interruptableAnnInd
	voiceInformation
	voiceBack
responseRequested	
assignmentID	

7.1.14 sendInfoAndCollectRes

sendInfoAndCollectRes is an asynchronous method that returns the information collected to the application.

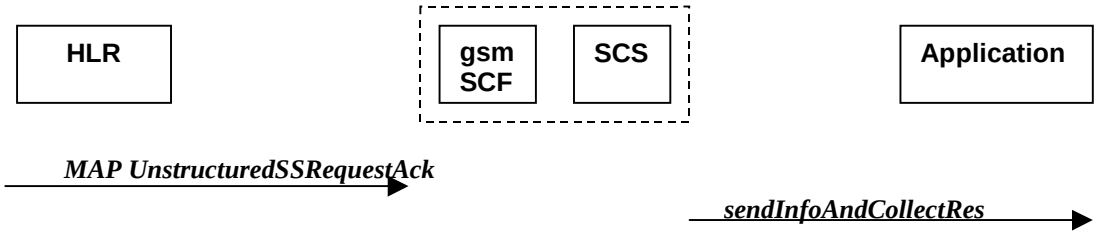


Figure 7-20: Call Flow for sendInfoAndCollectRes (scenario 1)

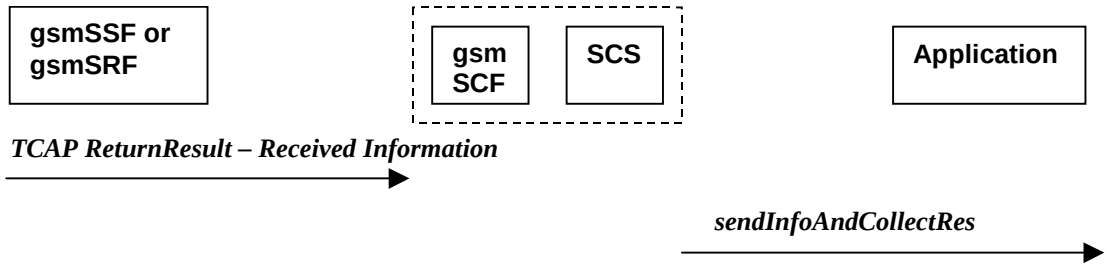


Figure 7-21: Call Flow for sendInfoAndCollectRes (scenario 2)

Normal Operation

Two Alternatives have been identified

1. USSD based interaction between the MS and the gsmSCF

Pre-conditions	The application has invoked a ***sendInfoAndCollectReq***()
1	The gsmSCF receives a MAP ***UnstructuredSS-Request acknowledgement*** message from the HLR
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS invokes the ***sendInfoAndCollectRes*** method to the correct applications

2. Interaction with an gsmSRF internal to gsmSSF or external

Pre-conditions	The application has invoked a ***sendInfoAndCollectReq***()
1	The gsmSCF receives a TCAP **ReturnResult** from the gsmSSF or the gsmSRF depending on whether a direct or indirect gsmSRF is used containing the Received Information.
2	The gsmSCF sends an equivalent internal operation to the SCS

3	The SCS identifies the correct application instance and invokes the ***sendinfoAndCollectRes*** method

Parameter Mapping

From: MAP ***unstructuredSS-Request acknowledgement***	To: ***sendInfoAndCollectRes***
	userInteractionSessionID
	assignmentID
	response
ussd-DataCodingScheme **ussd-String**	**info**

From: TCAP ***Return Result (Received Information)***	To: ***sendInfoAndCollectRes***
	userInteractionSessionID
	assignmentID
	response
DigitsResponse	**info (only the digits are mapped)**

7.1.15 sendInfoAndCollectErr

sendInfoAndCollectErr is an asynchronous method that indicates that the request to send information and collect a response was unsuccessful.

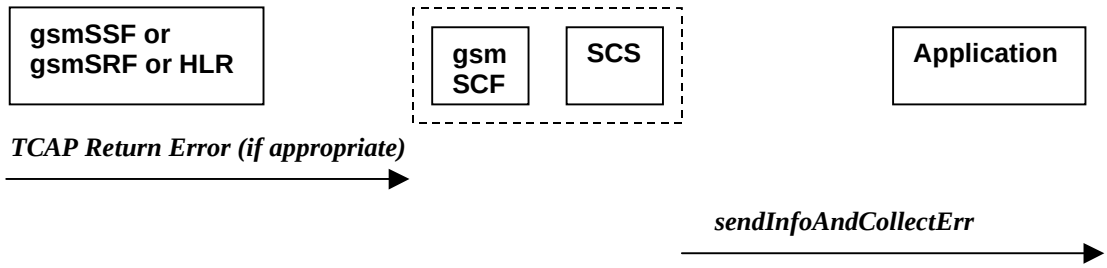


Figure 7-22: Call Flow for sendInfoAndCollectErr

Normal Operation

Two Alternatives have been identified

1. USSD based interaction between the MS and the gsmSCF

Pre-conditions	The application has invoked a ***sendInfoAndCollectReq***()
1	The gsmSCF detects an error in the ***sendInfoAndCollectReq*** method or receives a message form the HLR indicating an error that there is an error in ***sendInfoAndCollectReq*** method
2	The gsmSCF sends an equivalent internal message to the SCS
3	The SCS invokes the ***sendInfoAndCollectErr*** method to the correct application

2. Interaction with an gsmSRF internal to gsmSSF or external gsmSRF

Pre-conditions	The application has invoked a ***sendInfoAndCollectReq***()
1	The gsmSCF either detects and error or receives a TCAP ***Error*** from the gsmSSF or the gsmSRF depending on whether a direct or indirect gsmSRF is used
2	The gsmSCF sends an equivalent internal operation to the SCS
3	The SCS identifies the correct application instance and invokes the ***sendInfoAndCollectErr*** method

Parameter Mapping

From: TCAP ***Return Error***	To: ***sendInfoAndCollectErr***
	userInteractionSessionID
	assignmentID
error	**error**

7.1.16 release

release is a method that requests that the relationship between the application and the user interaction object be released. It causes the release of the used user interaction resources and interrupts any ongoing user interaction.

Call Flow

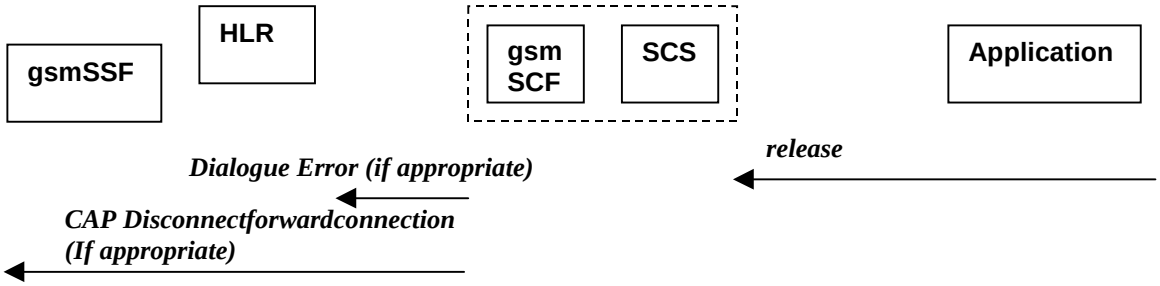


Figure 7-23: Call Flow for release

Normal Operation

Two Alternatives have been identified

1. USSD based interaction

Pre-conditions	The gsmSCF has an open dialogue with the HLR
1	The application invokes a ***release***
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a TCAP ***Abort*** message to the HLR if appropriate.

2. Interaction with a gsmSRF internal to gsmSSF or external gsmSRF

Pre-conditions	The application has previously invoked the ***sendInfoAndCollectErr***. The gsmSCF is waiting for a response form the user
1	The application invokes a release

2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***DisconnectForwardConnection*** to the gsmSSF

Parameter Mapping

From: ***release***	To: Dialogue Error
userInteractionSessionID	
	TC-U-ABORT **TC-P-ABORT**

From: ***release***	To: CAP ***DisconnectForwardConnection***
userInteractionSessionID	

7.1.17 abortActionReq

abortActionReq is an asynchronous method that aborts a user interaction operation, e.g. a ***sendInfoReq***, from the specified call. The call remains otherwise unaffected. The user interaction call service interrupts the current action on the specified call.

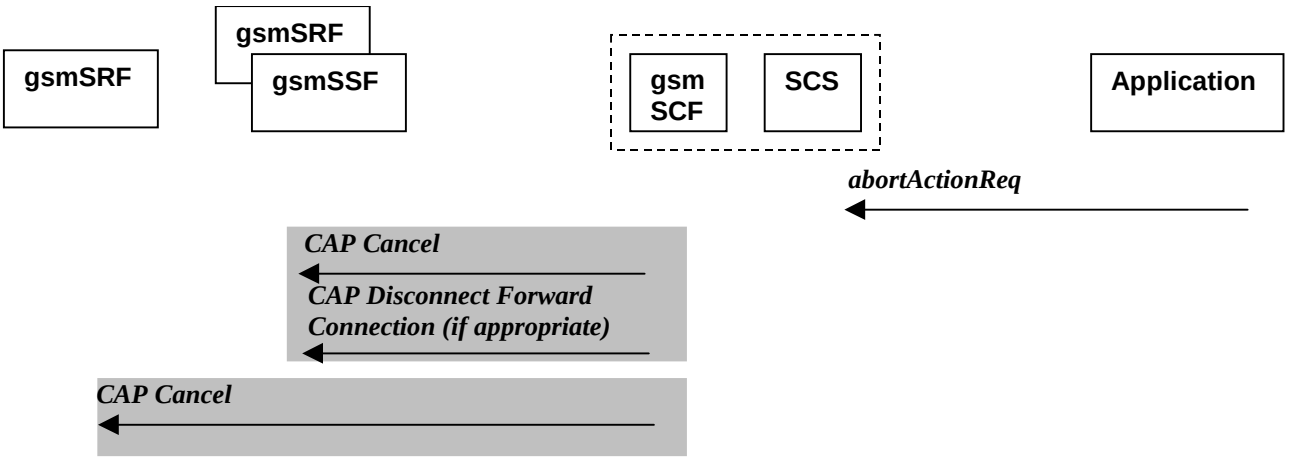


Figure 7-24: Call Flow for abortActionReq

Normal Operation

Pre-conditions	The application has previously invoked the ***sendInfoAndCollectErr***. The gsmSCF is waiting for a response form the user
1	The application invokes a ***abortActionReq***
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***Cancel*** message to the gsmSSF or the gsmSRF as appropriate and may send a CAP ***DisconnectForwardConnection*** to the gsmSSF if appropriate

Parameter Mapping

From: ***abortActionReq***	To: CAP ***Cancel***

userInteractionSessionID	
assignmentID	**InvokeID**
	allRequests

7.1.18 abortActionRes

abortActionRes is an asynchronous method that confirms that the request to abort a user interaction operation on a call was successful.

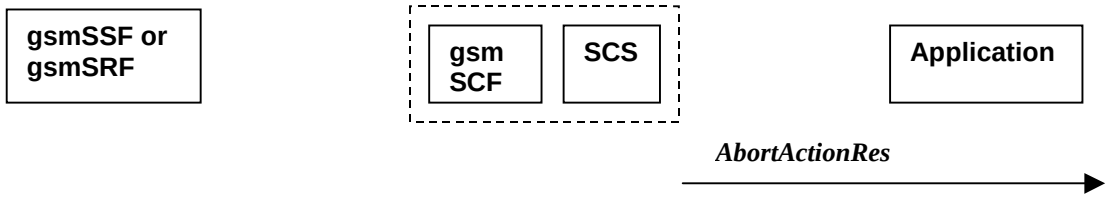


Figure 7-25: Call Flow for abortActionRes

There is no equivalent CAP/MAP mapping message

Normal Operation

Pre-conditions	The application has previously invoked the ***abortActionRes***. The gsmSCF has sent the necessary instruction to the gsmSSF or the gsmSRF and is running a timer awaiting for any possible error return message. This timer expires and no errors are returned
2	The gsmSCF determines that the CAP ***Cancel*** operation was successful. The gsmSCF sends an equivalent internal message to the SCS
3	The SCS invokes the ***abortActionRes*** method to the appropriate application.

7.1.19 abortActionErr

abortActionErr is an asynchronous method that indicates that the request to abort a user interaction on a call resulted in an error.

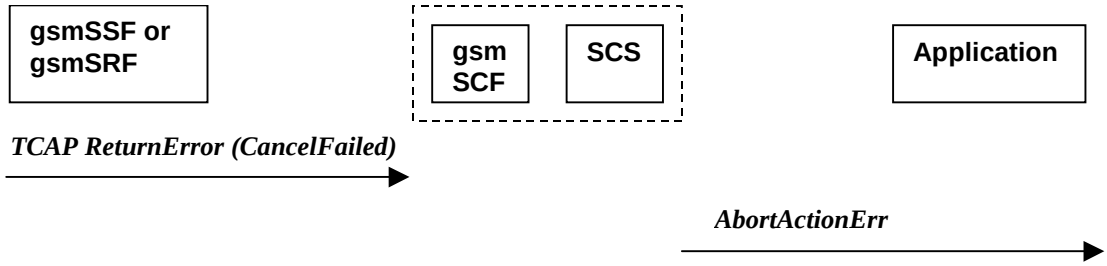


Figure 7-26: Call Flow for abortActionErr

Parameter Mapping

From: TCAP error primitive	To: ***abortActionErr***
	userInteractionSessionID

	assignmentID
TC-U-ERROR	**error**

8 Generic Message Transfer Service WAP Call Flows

8.1 User Interaction

No mapping of parameters is defined for the case where the sending of information is realised via WGP/WPP. The reason for this is that the WAP Forum does not specify a mapping either from the Push Access Protocol (used between Application Server and WGP/WPP) onto the Push Over-the-Air Protocol (used between WGP/WPP and terminal).

8.1.1 sendInfoReq

When the sendInfoReq is used to send a text message (e.g. URL or textual notification) to the terminal, the SCS can use the WAP Gateway/Push Proxy (WGP/WPP) as underlying mechanism to deliver the message to the terminal.

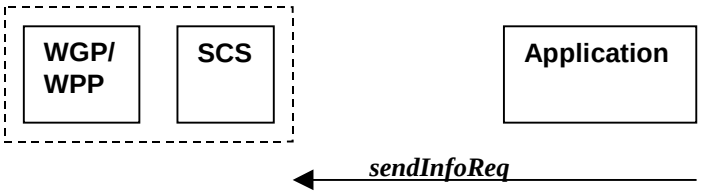


Figure 8-1: Call Flow for sendInfoReq

Normal Operation

1. Sending of messages via the WGP/WPP

Pre-conditions	
1	The application invokes the ***sendInfo*** method
2	The SCS sends an equivalent internal message to the WGP/WPP

8.1.2 sendInfoRes

sendInfoRes is an asynchronous method that informs the application about the start or the completion of a **sendInfoReq()**. This response is called only if the application has requested a response.

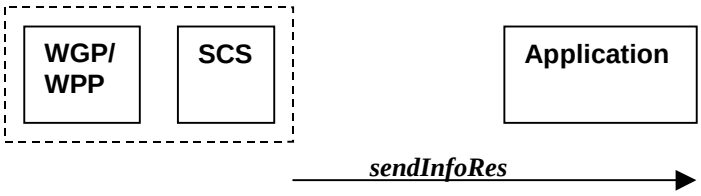


Figure 8-2: Call Flow for sendInfoRes

Normal Operation

1. Sending of messages via the WGP/WPP

Pre-conditions	The application has previously invoked the ***sendInfoReq*** method and has requested a notification
1	The SCS receives an internal message from the WGP/WPP
2	The SCS identifies the correct application and invokes the ***sendInfoRes*** method

8.1.3 sendInfoErr

sendInfoErr is an asynchronous method that indicates that the request to send information was unsuccessful.

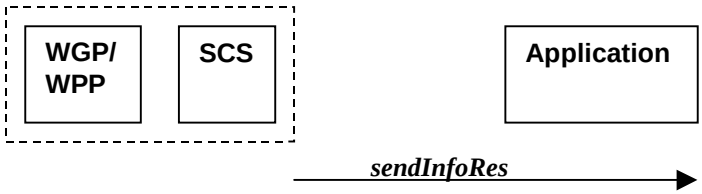


Figure 8-3: Call Flow for sendInfoRes

Normal Operation

1. Sending of messages via the WGP/WPP

Pre-conditions	The application has previously invoked the ***sendInfoReq*** method
1	The WGP/WPP sends an internal message to the SCS
2	The SCS identifies the correct application and invokes the ***sendInfoErr*** method

9 User Status Service CAMEL Flows

The User Status (US) interface class allows applications to obtain the status of mobile telephony users.

9.1 triggeredStatusReportingStartReq

TriggeredStatusReportingStartReq is a method that is used to subscribe to triggered user status notifications so that events can be sent to the application.

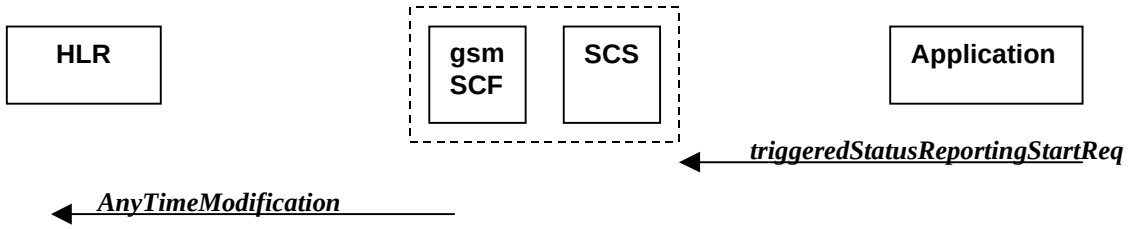


Figure 9-1: Call Flow for triggeredStatusReportingStartReq

Normal Operation

Pre-conditions	**An agreement is established between the network operator and the service provider for the event notification to be enabled**
1	The application invokes the ***triggeredStatusReportingStartReq*** method
2	The gsmSCF sends a MAP ***AnyTimeModification*** to the HLR in order to activate the CAMEL Subscription Information (M-CSI). In case the Status Report is requested for multiple users, multiple ATM requests are sent to the HLR.

Parameter Mapping

From: ***triggeredStatusReportingStartReq***	To: MAP ***AnyTimeModification***
appStatus	
users	**subscriberIdentity** **modificationInstruction in modificationRequestFor-CSI has value 'activate', for M-CSI (Mobility CAMEL Subscription Information)**
assignmentID	
	gsmSCF-Address

9.2 triggeredStatusReportingStop

triggeredStatusReportingStop is a method that is used by the application to disable triggered user status notifications.

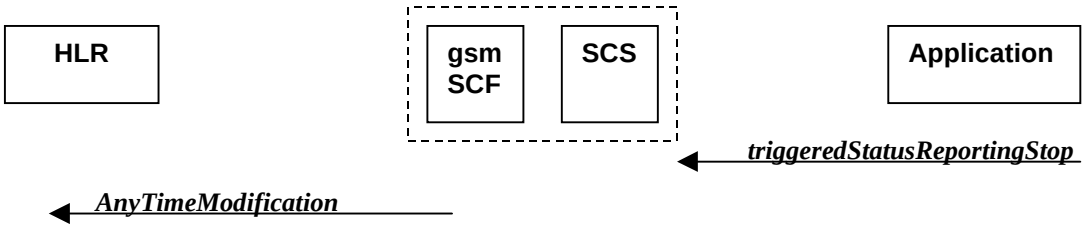


Figure 9-2: Call Flow for triggeredStatusReportingStop

Normal Operation

Pre-conditions	**An agreement is established between the network operator and the service provider for the status notification to be disabled**
1	The application invokes the ***triggeredStatusReportingStop*** method
2	The gsmSCF sends a MAP ***AnyTimeModificaitonRequest*** to the HLR in order to de-activate the CAMEL Subscription Information (M-CSI). In case stopping Status Reporting is requested for multiple users, multiple ATM requests are sent to the HLR.

Parameter Mapping

From: ***triggeredStatusReportingStop***	To: MAP ***AnyTimeModification***
stopRequest	**subscriberIdentity**

assignmentID **stopScope** **users**	**(either extracted from assignmentID, or mapped from 'users')** **modificationInstruction in modificationRequestFor-CSI has value 'deactivate', for M-CSI (Mobility CAMEL Subscription Information)**
	gsmSCF-Address

9.3 statusReportReq

statusReportReq is a method that is used by the application to request a user status report. Note that this can be requested for multiple users at the same time.

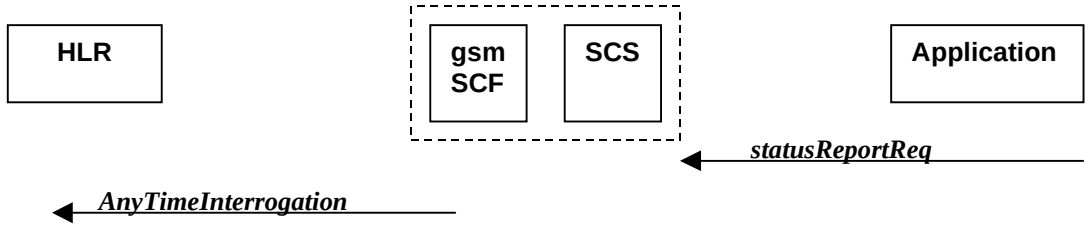


Figure 9-3: Call Flow for statusReportReq

Normal Operation

Pre-conditions	
1	The application invokes the ***statusReportReq*** method
2	The gsmSCF sends a MAP ***AnyTimeInterrogateRequest*** to the HLR in order to request the subscriber status In case the Status Report is requested for multiple users, multiple ATI requests are sent to the HLR.

Parameter Mapping

From: ***statusReportReq***	To: MAP ***AnyTimeInterrogation***
	Invoke id
appStatus	
users	**subscriberIdentity**
	requestedInfo (sequence of optional indicators, of which only subscriberState is present)
	gsmSCF-Address
assignmentID	

9.4 statusReportRes

statusReportRes is a method that is used by the HLR/SCS towards the application, in response to an earlier request for a user status report. Note that this can be requested for multiple users at the same time.

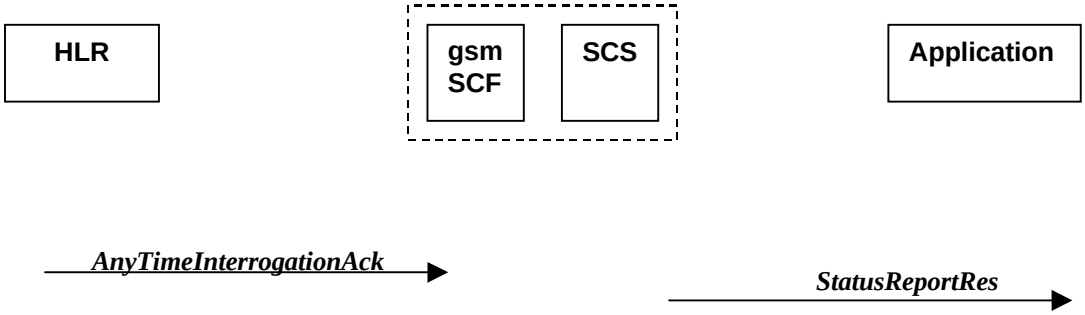


Figure 9-4: Call Flow for statusReportRes

Normal Operation

Pre-conditions	The application has invoked a ***statusReportReq*** method, and this request has been forwarded to the HLR.
1	The HLR sends a MAP ***AnyTimeInterrogationAck*** to the HLR/SCS in response to the earlier request.
2	The gsmSCF/SCS respond to the application via ***StatusReportRes***. In case the Status Report was requested for multiple users, multiple ATI acknowledgements are collected in the gsmSCF/SCS before a response is sent back to the Application.

Parameter Mapping

To: ***statusReportRes***	From: MAP ***AnyTimeInterrogationAck***
	Invoke id
assignmentID	
status	
userID	
statusCode	
	subscriberInfo (sequence of optional parameters, of which only subscriberState present)
status	**subscriberState**

9.5 triggeredStatusReport

triggeredStatusReport is a method that is used to notify the application of the arrival of a requested user status report event.

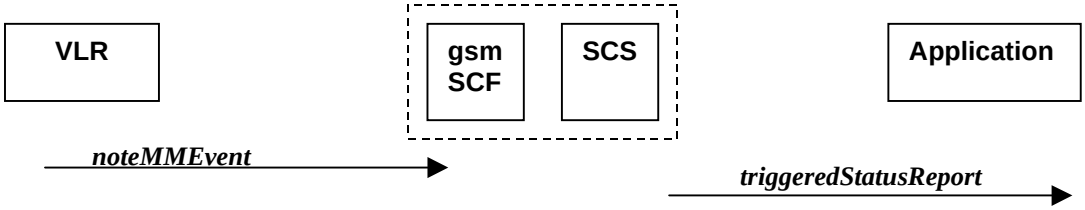


Figure 9-5: Call Flow for triggeredStatusReport

Normal Operation

Pre-conditions	The Application has requested triggeredStatusReporting
1	The VLR sends a MAP ***noteMM-Event*** message to the CSE/SCS
2	The SCS sends a ***triggeredStatusReport*** to the Application

Parameter Mapping

To ***triggeredStatusReport***	From: MAP ***noteMM-Event***
status	
userID	**msisdn**
statusCode	
status	**event-Met**
	serviceKey
	imsi
assignmentID	

10 User Status Service core-MAP Flows

The User Status (US) interface class allows applications to obtain the status of mobile telephony users.

10.1 statusReportReq

statusReportReq is a method that is used by the application to request a user status report. Note that this can be requested for multiple users at the same time

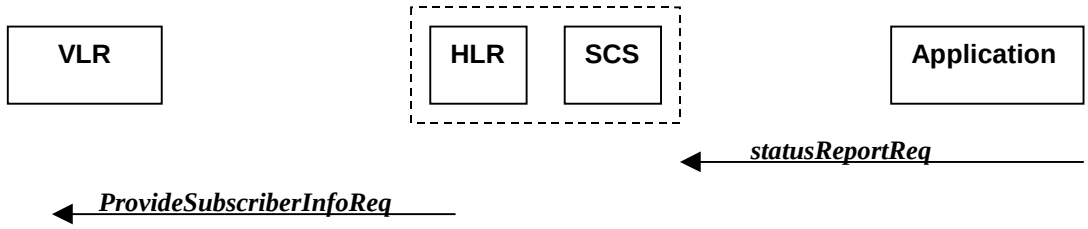


Figure 10-1: Call Flow for statusReportReq

Normal Operation

Pre-conditions	
1	The application invokes the ***statusReportReq*** method
2	The HLR sends a MAP ***ProvideSubscriberInfoRequest*** to the VLR in order to request the subscriber status In case the Status Report is requested for multiple users, multiple PSI requests are sent to the VLR.

Parameter Mapping

From: ***statusReportReq***	To: MAP ***ProvideSubscriberInfo***
	Invoke id
appStatus	
users	**imsi (deduced from information in 'users')**
	requestedInfo (sequence of optional indicators, of which only subscriberState is present)
assignmentID	

10.2 statusReportRes

statusReportRes is a method that is used by the HLR/SCS towards the application, in response to an earlier request for a user status report. Note that this can be requested for multiple users at the same time

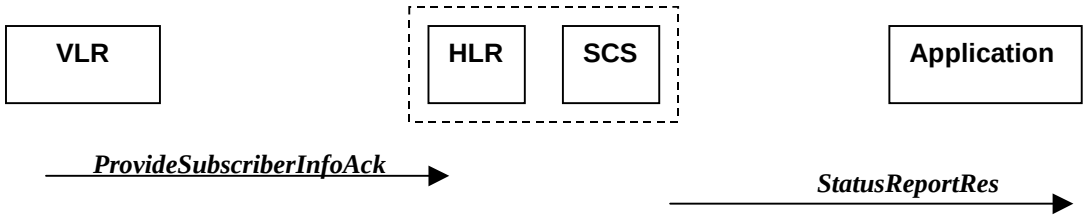


Figure 10-2: Call Flow for statusReportRes

Normal Operation

Pre-conditions	The application has invoked a ***statusReportReq*** method, and this request has been forwarded to the VLR.
1	The VLR sends a MAP ***ProvideSubscriberInfoAck*** to the HLR/SCS in response to the earlier request.
2	The HLR/SCS respond to the application via ***StatusReportRes***. In case the Status Report was requested for multiple users, multiple PSI acknowledgements are collected in the HLR/SCS before a response is sent back to the Application.

Parameter Mapping

To: ***statusReportRes***	From: MAP ***ProvideSubscriberInfoAck***
	Invoke id
assignmentID	
status	
userID	
statusCode	
	subscriberInfo (sequence of optional parameters, of which only subscriberState present)
status	**subscriberState**

11 Network User Location Call Flows

The Network User Location (NUL) provides location information, based on network-related information

Using the NUL functions, an application programmer can request the VLR number, the Location Area Identifier, geodetic Location Information and the Cell Global Identification and other mobile telephony specific location information, if the network is able to support the corresponding capability

11.1 locationReportReq

locationReportReq is a method used by the application to request for mobile-related location information on one or several users[12].

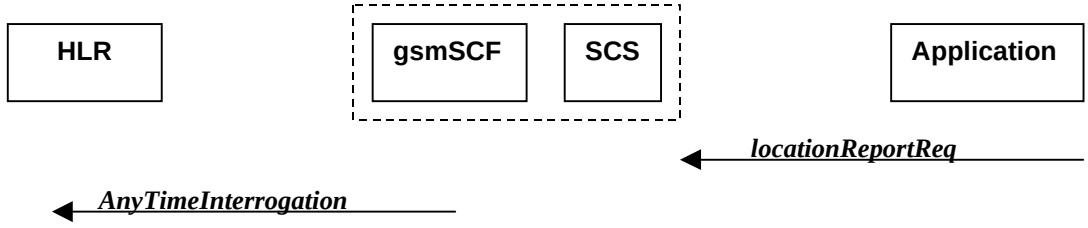


Figure 11-1: Call Flow for locationReportReq

Normal Operation

Pre-conditions	**An agreement is established between the network operator and the service provider for the *locationReportReq* to be enabled**
1	The application invoked the ***locationReportReq*** method
2	The gsmSCF sends a MAP ***AnyTimeInterrogationReq*** to the HLR.

Parameter Mapping

From: ***locationReportReq***	To: MAP ***AnyTimeInterrogationReq***
	invokeID
appLocationCamel	
users	**subscriberIdentity**
	gsmSCF-Address
	requestedInfo (sequence of optional indicators, of which only locationInformation is present)
assignmentID	

[12] note that a request of location information for several users has to be mapped to several MAP-operation-requests

11.2 locationReportRes

locationReportRes is a method that delivers a mobile location report towards the application. The report contains mobile-related location information for one or several users[13].

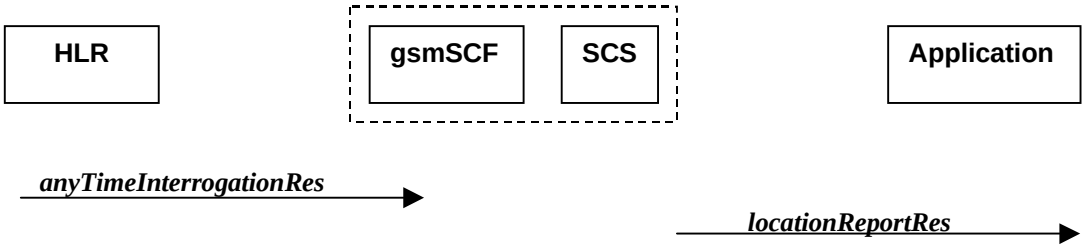


Figure 11-2: Call Flow for locationReportRes

Normal Operation

Pre-conditions	
1	The application invoked the ***locationReportReq*** method
2	The SCS responds to the application via a ***locationReportRes*** method invocation

Parameter Mapping

From: MAP ***AnyTimeInterrogationAck***	To: ***locationReportRes***
invokeId	
	assignmentID
subscriberInfo (sequence of optional parameters, of which only locationInformation is present)	
locationInformation	**locations**
	UserID
	StatusCode
geographicalInformation **geodeticInformation**	**GeographicalPosition (geodeticInformation is mapped if present, otherwise geographicInformation is used)**
ageOfLocationInformation	**Timestamp (calculated from ageOfLocationInfo)**
vlr-number	**VlrNumber**
locationNumber	**LocationNumber**
cellGlobalIdorServiceAreaIdOrLai	**CellidOrLai**
extensionContainer	
selectedLSA-Id	
msc-Number	

[13] note that a request of location information for several users has to be mapped to several MAP-operation-requests

currentLocationRetrieved	

11.3 locationReportErr

locationReportErr is a method that indicates that the location report request has failed.

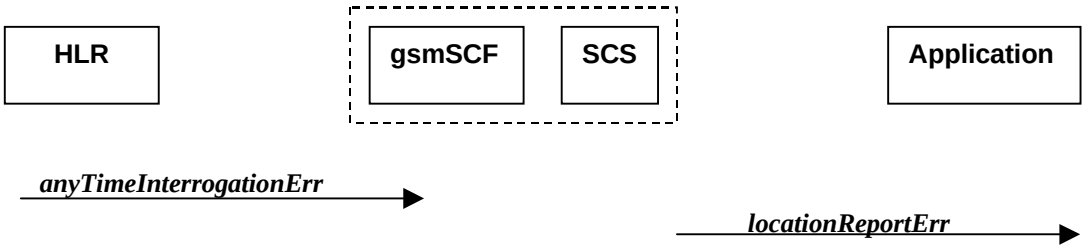


Figure 11-3: Call Flow for locationReportErr

Normal Operation

Pre-conditions	
1	The application invoked the ***locationReportReq*** method

Parameter Mapping

From: MAP ***anyTimeInterrogationErr***	To: ***locationReportErr***
	assignmentID
SystemFailure **ATI-NotAllowed** **DataMissing** **UnexpectedDataValue** **UnknownSubscriber**	**cause**
	diagnostic

11.4 periodicLocationReportingStartReq

periodicLocationReportingStartReq is a method used by the application to request for periodic mobile location reports on one or several users[14].

[14] note that a request of location information for several users has to be mapped to several MAP-operation-requests

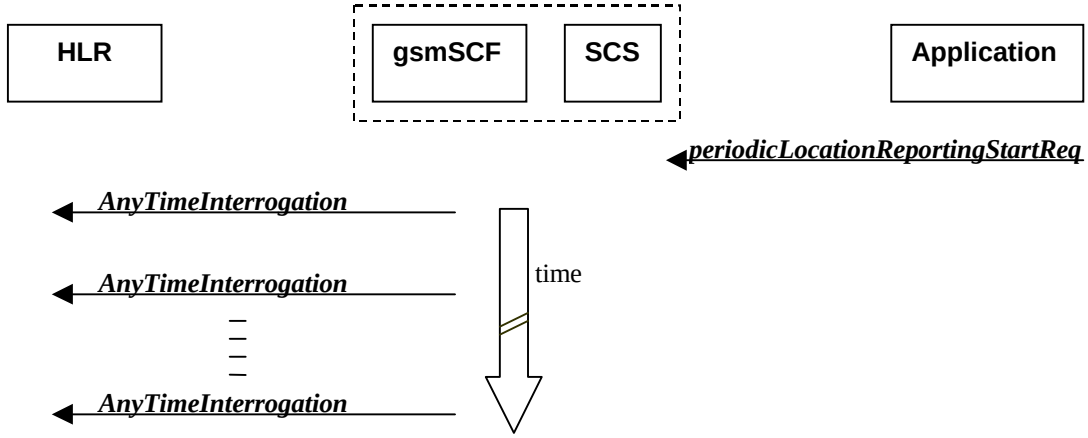


Figure 11-4: Call Flow for periodicLocationReportingStartReq

Normal Operation

Pre-conditions	**An agreement is established between the network operator and the service provider for the *periodicLocationReportingStartReq* to be enabled**
1	The application invoked the ***periodicLocationReportingStartReq*** method
2	The gsmSCF sends a MAP ***AnyTimeInterrogationReq*** to the HLR, and repeats this according to the requested time interval..

Parameter Mapping

From: ***periodicLocationReportingStartReq***	To: MAP ***AnyTimeInterrogationReq***
	invokeID
appLocation	
users	**subscriberIdentity**
	gsmSCF-Address
	requestedInfo (sequence of optional indicators, of which only locationInformation is present)
reportingInterval	
assignmentID	

11.5 periodicLocationReportingStop

periodicLocationReportingStop is a method used by the application to stop the sending of periodic mobile location reports for one or several users[15].

[15] note that a request of location information for several users has to be mapped to several MAP-operation-requests

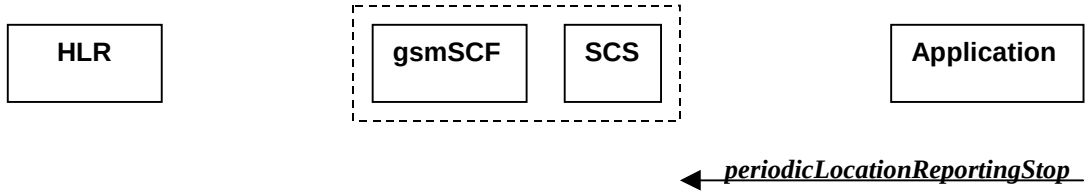


Figure 11-5: Call Flow for periodicLocationReportingStop

Normal Operation

Pre-conditions	
1	The application invoked the ***periodicLocationReportingStartReq*** method
2	The gsmSCF stops the periodic sending of MAP ***AnyTimeInterrogationReq*** to the HLR, for the subscribers as indicated in the stop request (for details of StopRequest see e.g. with triggeredLocationReportingStop)..

Parameter Mapping

None.

11.6 periodicLocationReport

periodicLocationReport is a method that provides periodic delivery of mobile location reports. The reports are containing mobile-related location information for one or several users[16].

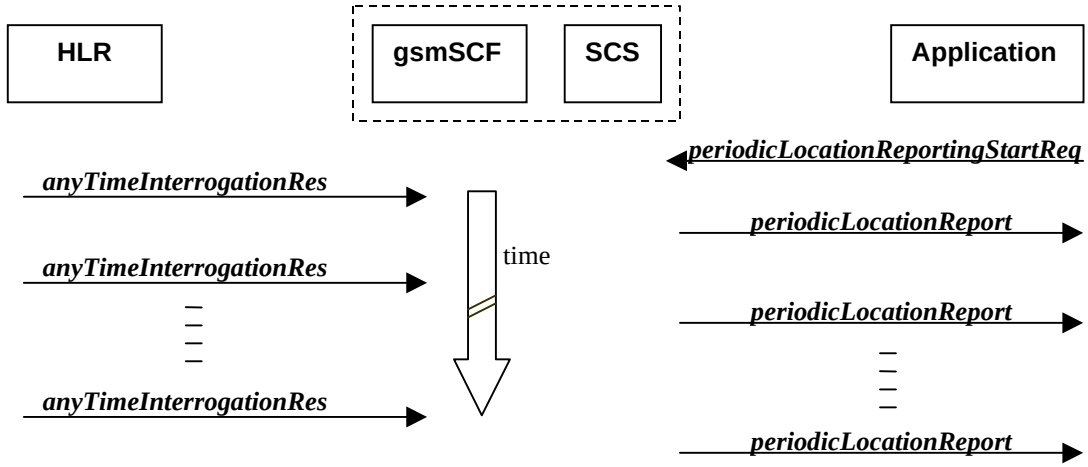


Figure 11-6: Call Flow for periodicLocationReport

Normal Operation

Pre-conditions	
1	The application invoked the ***periodicLocationReportingStartReq*** method

Parameter Mapping

[16] note that a request of location information for several users has to be mapped to several MAP-operation-requests

From: MAP ***AnyTimeInterrogationAck***	To: ***locationReportRes***
invokeID	**assignmentID**
subscriberInfo (sequence of optional parameters, of which only is present)	
locationInformation	**locations**
	UserID
	StatusCode
geographicalInformation **geodeticInformation**	**GeographicalPosition** **(geodeticInformation is mapped if present, otherwise geographicInformation is used)**
ageOfLocationInfromation	**Timestamp**
vlr-number	**VlrNumber**
locationNumber	**LocationNumber**
cellGlobalIdorServiceAreaIdOrLai	**CellidOrLai**
extensionContainer	
selectedLSA-Id	
msc-Number	
currentLocationRetrieved	

11.7 periodicLocationReportErr

periodicLocationReportErr is a method that indicates that the requested periodic location report has failed. Note that errors only concerning individual users are reported in the ordinary periodicLocationReport() message.

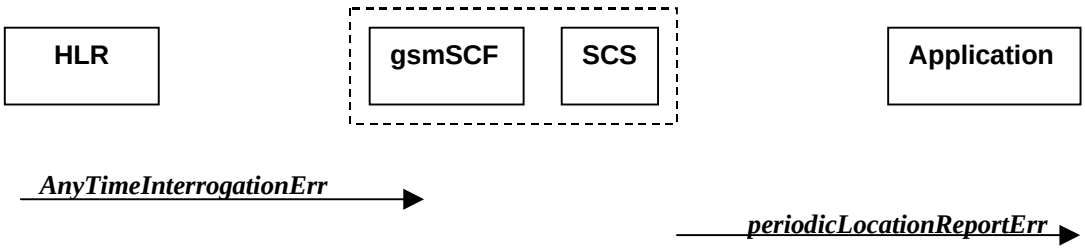


Figure 11-7: Call Flow for periodicLocationReportErr

Normal Operation

Pre-conditions	
1	The application invoked the ***periodicLocationReportingStartReq*** method

Parameter Mapping

From: MAP ***anyTimeInterrogationErr***	To: ***periodicLocationReportErr***

	assignmentID
SystemFailure **ATI-NotAllowed** **DataMissing** **UnexpectedDataValue** **UnknownSubscriber**	**cause**
	diagnostic
gsmSCF-Address	

11.8 triggeredLocationReportingStartReq

triggeredLocationReportingStartReq is a method used by the application to request for user location reports, containing mobile related information, when the location is changed (the report is triggered by the location change, e.g. change of VLR number, change of Global Cell Identification or other location information if available).

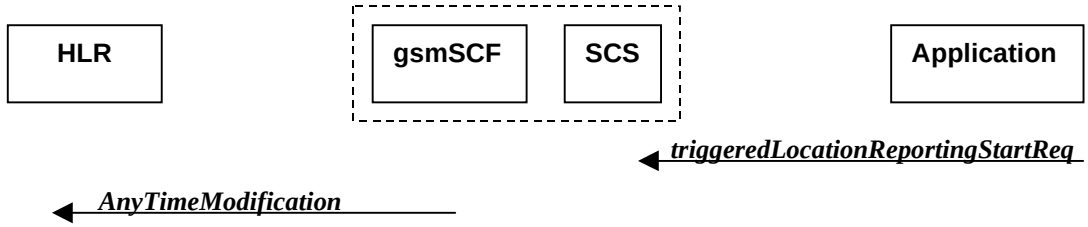


Figure 11-8: Call Flow for triggeredLocationReportingStartReq

Normal Operation

Pre-conditions	**An agreement is established between the network operator and the service provider for the *triggeredLocationReportingStartReq* to be disabled**
1	The application invoked the ***triggeredLocationReportingStartReq*** method
2	The gsmSCF sends a MAP ***AnyTimeModificationReq*** to the HLR in order to activate the CAMEL subscription Information (M-CSI). In case the Location Report is requested for multiple users, multiple ATM requests are sent to the HLR.

Parameter Mapping

From: ***triggeredLocationReportingStartReq***	To: MAP ***AnyTimeModificationReq***
appLocation	
users	**subscriberIdentity** **modificationInstruction in modificationRequestFor-CSI has value 'activate', for M-CSI (Mobility CAMEL Subscription Information)**
	gsmSCF-Address
triggers	

11.9 triggeredLocationReportingStop

triggeredLocationReportingStop is a method used by the application to request that triggered mobile location reporting should stop.

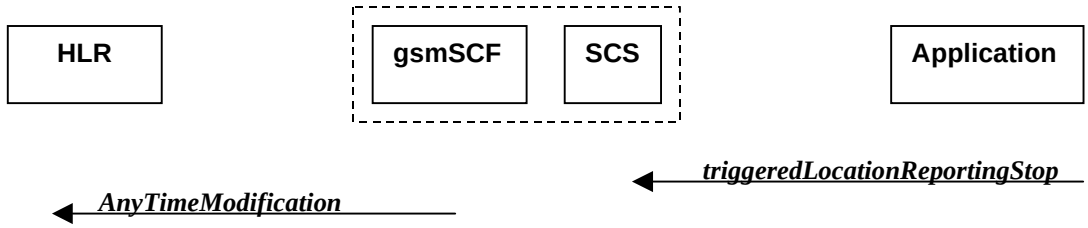


Figure 11-9: Call Flow for triggeredLocationReportingStop

Normal Operation

Pre-conditions	
1	The application has initiated a **triggered Location Report *assignment*** method
2	The gsmSCF sends a MAP ***AnyTimeModificationReq*** to the HLR in order to de-activate the CAMEL subscription Information (M-CSI). In case stopping of triggered location reporting is requested for multiple users, multiple ATM requests are sent to the HLR.

Parameter Mapping

From: ***triggeredLocationReportingStop***	To: MAP ***AnyTimeModificationReq***
stopRequest **assignmentID** **stopScope** **users**	**subscriberIdentity** **(either extracted from assignmentID,** **or mapped from 'users')** **modificationInstruction in ModificationRequestFor-CSI has value 'deactivate', for M-CSI** **(Mobility CAMEL Subscription Information)**
	gsmSCF-Address

11.10 triggeredLocationReport

triggeredLocationReport is a method providing the delivery of a report that is indicating that one or several user's mobile location has changed.

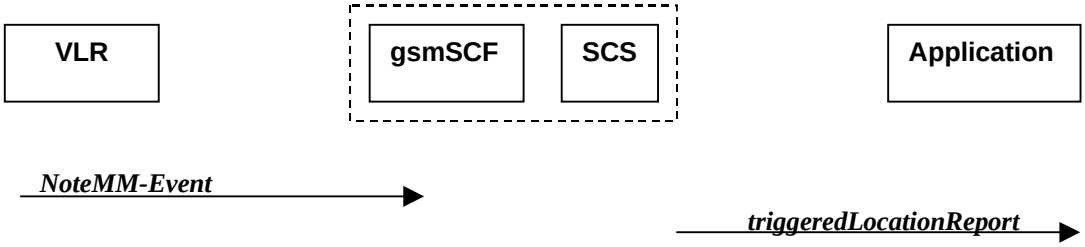


Figure 11-10: Call Flow for triggeredLocationReport

Normal Operation

Pre-conditions	
1	The application invoked the ***triggeredLocationReportingStartReq*** method

Parameter Mapping

From: MAP ***NoteMM-Event***	To: ***triggeredLocationReport***
	assignmentID
serviceKey	
imsi	
msisdn	
locationInformation	**location**
	UserID (from msisdn)
	StatusCode
geographicalInformation **geodeticInformation**	**GeographicalPosition**
ageOfLocationInformation	**Timestamp (calculated from ageOfLocationInfo)**
vlr-number	**VlrNumber**
locationNumber	**LocationNumber**
cellGlobalIdorServiceAreaIdOrLai	**CellidOrLai**
extensionContainer	
selectedLSA-Id	
msc-Number	
currentLocationRetrieved	
eventMet	**criterion**

11.11 triggeredLocationReportErr

triggeredLocationReportErr is a method indicates that a requested triggered location report has failed. Note that errors only concerning individual users are reported in the ordinary ***triggeredLocationReport*** message.

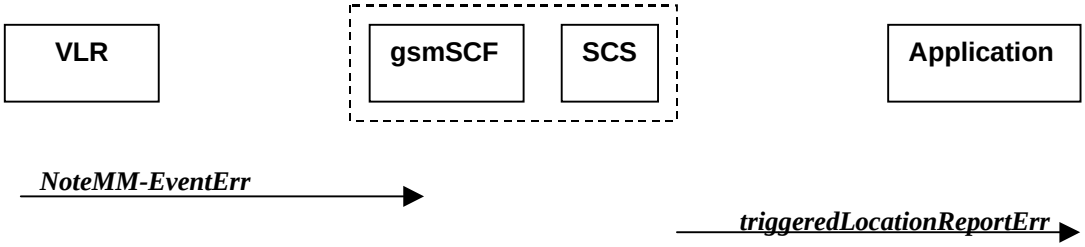


Figure 11-11: Call Flow for triggeredLocationReportErr

Normal Operation

Pre-conditions	
1	The application invoked the ***triggeredLocationReportingStartReq*** method
2	The gsmSCF sends a MAP ***AnyTimeModificationReq*** to the HLR

Parameter Mapping

From: MAP ***NoteMM-EventErr***	To: ***triggeredLocationReportErr***
	assignmentID
dataMissing **unexpectedDataValue** **unknownSubscriber** **MM-EventNotSupported**	**cause**
	diagnostic

12 Terminal Capabilities WAP Call Flows

The Terminal Capabilities SCF allows the application to request Terminal Capabilities.

12.1 getTerminalCapabilities

getTerminalCapabilities is a method that will result in the SCS asking the WAP Gateway/Push Proxy (WGP/WPP) to return the terminal capabilities. The ***getTerminalCapabilities*** method is a synchronous method and therefore no arrow is shown from SCS towards Application.

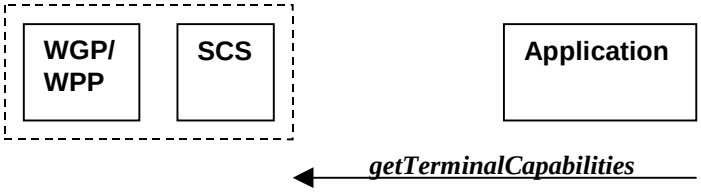


Figure 12-1: Call Flow for getTerminalCapabilities

Parameter mapping

No mapping of parameters is defined. The reason for this is that the WAP Forum does not specify a mapping either from the Push Access Protocol (used between Application Server and WGP/WPP) onto the Push Over-the-Air Protocol (used between WGP/WPP and terminal).

13 Data Session Control Service CAMEL Call Flows

13.1 Data Session Manager

The session manager interface provides the management functions to the data session service capability features. The application programmer can use this interface to enable or disable data session-related event notifications.

In order to ensure that the mobility events are transparent to the Data Session SCF, the same gsmSCF address must be used in the GPRS-CSI for the detection points: PDP Context Establishment, PDP Context Establishment Acknowledge and Change of Position.

13.1.1 enableDataSessionNotification

enableDataSessionNotification is used to enable data session-related notifications to be sent to the application.

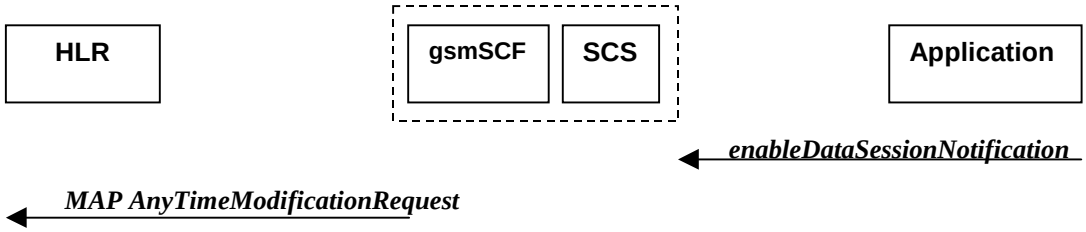


Figure 13-1: Call Flow for enableDataSessionNotification

Normal Operation

Pre-conditions	An agreement is established between the network operator and the service provider for the event notification to be enabled
1	The application invokes the ***enableDataSessionNotification*** method
2	The gsmSCF sends a MAP ***AnyTimeModification*** to the HLR in order to activate the necessary CAMEL Subscription Information (GPRS-CSI) Note : CAMEL phase 3 only allows for activation/deactivation of the CSI and not modification of the contents of the CSIs.

Parameter Mapping

From: ***enableDataSessionNotification***	To: MAP ***AnyTimeModification***
appInterface	
	-
eventCriteria `OriginatingAddress`	**GPRS CAMEL Subscription Information** **GPRS-CSI** **gsmSCF Address**

assignmentID	

13.1.2 disableDataSessionNotification

disableDataSessionNotification is used by the application to disable data session notifications.

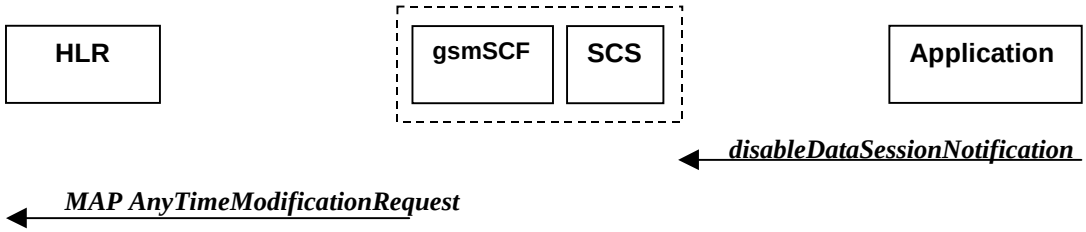


Figure 13-2: Call Flow for disableDataSessionNotification

Normal Operation

Pre-conditions	An agreement is established between the network operator and the service provider for the event notification to be disabled
1	The application invokes the ***disableDataSessionNotification*** method
2	The gsmSCF sends a MAP ***AnyTimeModification*** to the HLR in order to de-activate the necessary CAMEL Subscription Information. Note that CAMEL Phase 3 only allows the capability to activate/deactivate CSI and not to modify the triggering information.

Parameter Mapping

From: ***disableDataSessionNotification***	To: MAP ***AnyTimeModification***
eventCriteria `OriginatingAddress`	**GPRS CAMEL Subscription Information** **GPRS-CSI** **gsmSCF Address**
assignmentID	

13.1.3 dataSessionEventNotify

dataSessionEventNotify notifies the application of the arrival of a data session-related event.

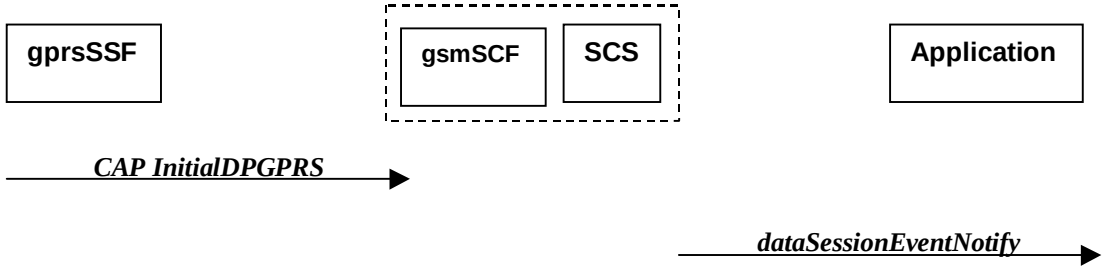


Figure 13-3: Call Flow for dataSessionEventNotify

Normal Operation

Pre-conditions	Call notifications have been enabled using the ***enableDataSessionNotification*** method on the Data Session Manager interface
1	A data session request arrives at the gsmSSF causing initial triggering to the gsmSCF CAP ***InitialDPGPRS***
2	The gsmSCF recognizes the need for an API service and passes the triggering information to the SCS
3	The SCS identifies the application responsible for handling the data session and invokes the ***dataSessionEventNotify*** method

Parameter Mapping

From: CAP ***InitialDPGPRS***	To: ***dataSessionlEventNotify***
serviceKey	
gPRSEventType	
mSISDN **accessPointName**	**eventInfo** **OriginatingAddress** **DestinationAddress**
iMSI	
timeAndTimeZone	
gPRSMSClass	
pDPType	
qualityOfService	
routeingAreaIdentity	
chargeID	
sGSNCapabilities	
	assignmentID
	appInterface

13.1.4 dataSessionAborted

dataSessionAborted indicates to the application that the Data Session object has aborted or terminated abnormally. No further communication will be possible between the Data Session object and the application.

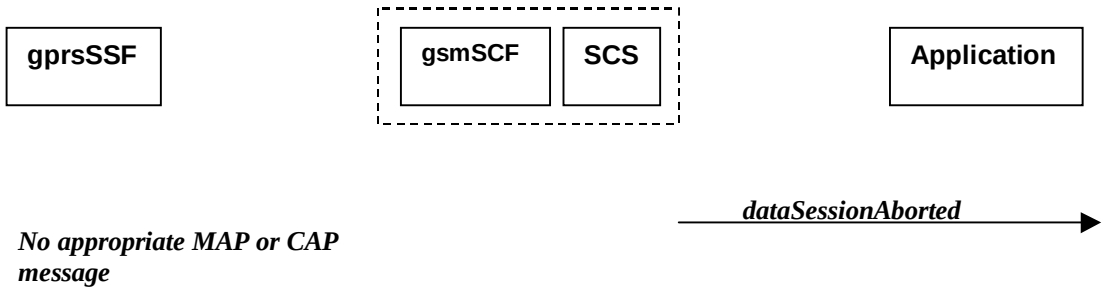


Figure 13-4: Call Flow for dataSessionAborted

Normal Operation

Pre-conditions	
1	The SCS detect a catastrophic failure in its communication with the gsmSCF
2	The SCS, invokes the ***dataSessionAborted*** method. The data session running in the network may continue and will not have been affected by this failure betweeen the gsmSCF and the SCS

Parameter Mapping

None.

13.1.5 dataSessionNotificationInterrupted

dataSessionlNotificationInterrupted indicates to the application that event notifications will no longer be sent (for example, due to faults detected).

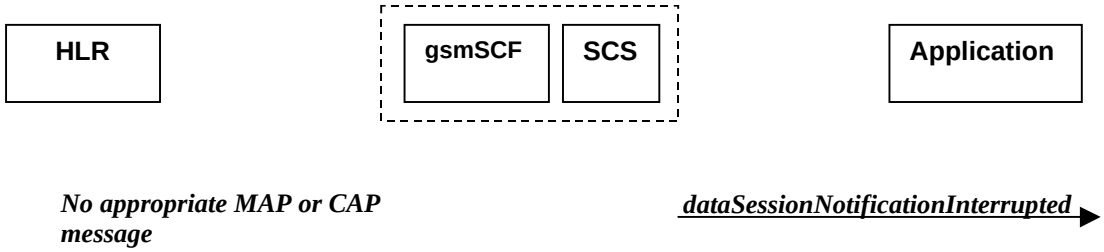


Figure 13-5: Call Flow for dataSessionNotificationInterruptedNormal Operation

Pre-conditions	Data session notifications have been enabled using the ***enableNotification*** method on the Data Session Manager interface
1	The SCS has detected, or has been informed of, a fault which prevents further events from being notified
2	The SCS invokes the ***dataSessionNotificationInterrupted*** method

Parameter Mapping

None.

13.1.6 dataSessionNotificationContinued

dataSessionNotificationContinued indicates to the application that all event notifications will be sent again.

Figure 13-6: Call Flow for dataSessionNotificationContinued

Normal Operation

Pre-conditions	Data session notifications have been interrupted and ***dataSessionNotificationInterrupted*** method has been invoked.
1	The SCS detects that data session notifications are again possible.
2	The SCS invokes the ***dataSessionNotificationContinued*** method

Parameter Mapping

None.

13.2 Data Session

The Data Session interface provides basic methods for applications to control data sessions.

13.2.1 ConnectReq

connectReq requests the connection of a data session with the destination party (specified in the parameter TargetAddress). The Data Session object is not automatically deleted if the destination party disconnects from the data session. The mapping to ***continueGPRS*** is also possible.

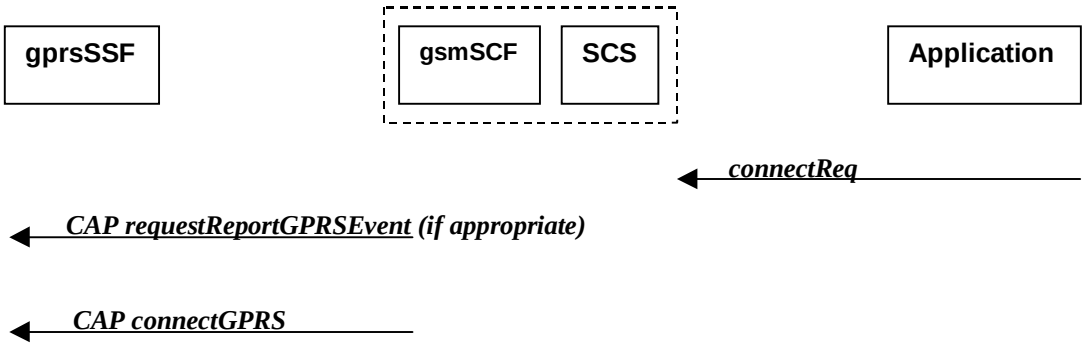


Figure 13-7: Call Flow for connectReq

Normal Operation

Pre-conditions	The application has been notified of a new data session and the data session object exists.
1	The application invokes the ***connectReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***requestReportGPRSEvent*** if the application needs to be informed

	about the outcome of the request
4	The gsmSCF sends a CAP ***connectGPRS*** message

Parameter Mapping

From: ***connectReq***	To: CAP ***requestReportGPRSEvent***
	gPRS-ReferenceNumber
dataSessionID	
responseRequested	**gPRSEvent**
targetAddress	
	pDPID
assignmentID	

From: ***connectReq***	To: CAP ***connectGPRS***
dataSessionID	
responseRequested	
targetAddress	**accessPointName**
	pdpID
assignmentID	

13.2.2 connectRes

connectRes indicates that the request to connect a data session with the destination party was successful, and indicates the response of the destination party (e.g. connected, disconnected).

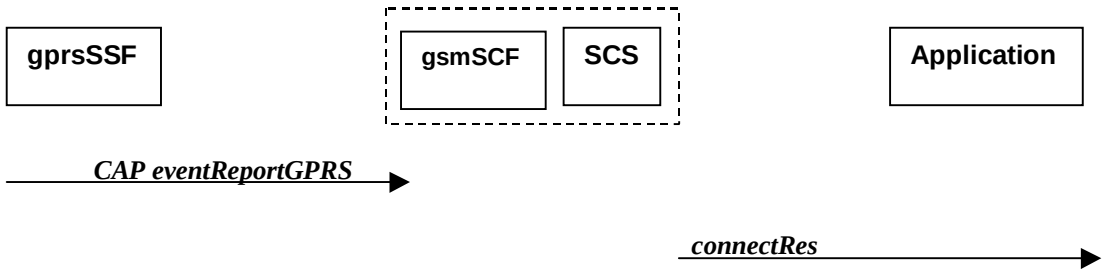


Figure 13-8: Call Flow for connectRes

Normal Operation

Pre-conditions	Data session routing attempted
1	If event reports have been requested, the gprsSSF sends a CAP ***eventReportGPRS*** to the gsmSCF
2	The gsmSCF sends an equivalent message to the SCS
3	The SCS invokes the ***connectRes*** method

Parameter Mapping

From: CAP ***eventReportGPRS***	To: ***connectRes***
	dataSessionID
gPRS-ReferenceNumber	
gPRSEventType **miscGPRSInfo** **gPRSEventSpecificInformation**	**eventReport**
pDPID	

13.2.3 connectErr

connectErr indicates that the request to connect a data session with the destination party was unsuccessful, e.g. an error detected in the network or the data session was abandoned.

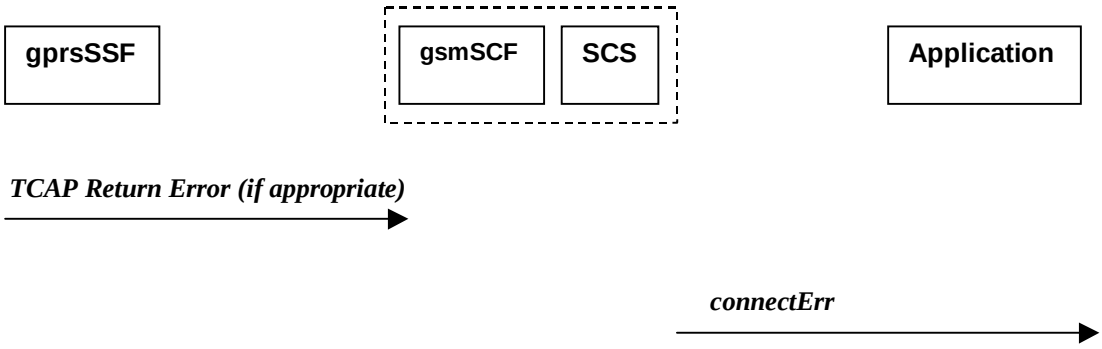


Figure 13-9: Call Flow for connectErr

Normal Operation

Two scenarios are possible

1. The gsmSCF receives a message from the gprsSSF indicating an error

Pre-conditions	**Data session routing attempted**
1	The gprsSSF detects a call routing failure and sends an appropriate TCAP message returning an error to the gsmSCF
2	The gsmSCF sends an equivalent message to the SCS
3	The SCS detects an error with the ***connectReq*** method, or receives a TCAP Return Error, and invokes the ***connectErr*** method

2. The gsmSCF detects there is an error in the message from the SCS

Pre-conditions	**Data session routing attempted**
1	The gsmSCF detects an error in the parameters of the internal message from the SCS requesting a ***connectReq***
2	The gsmSCF sends an equivalent message to the SCS
3	The SCS invokes the ***connectErr*** method

Parameter Mapping

From: TCAP ***Return Error***	To: ***routeCallToDestinationErr***
	dataSessionID
TC-U-ERROR TC-U-REJECT	**errorIndication**

13.2.4 release

release requests the release of the data session.

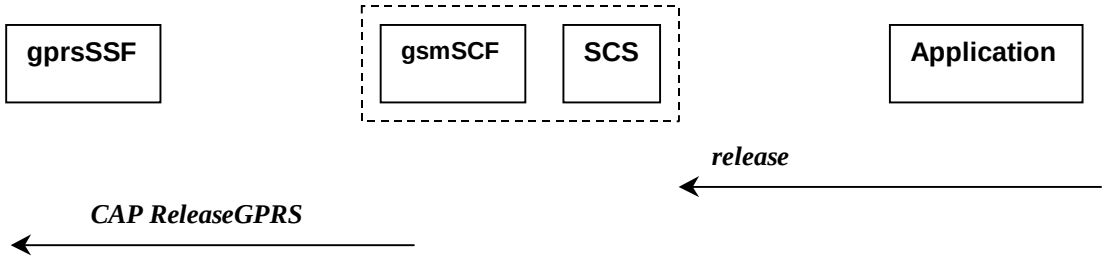


Figure 13-10: Call Flow for release

Normal Operation

Pre-conditions	**Data session is in progress**
1	The application invokes the ***release*** method
2	The SCS sends an equivalent message to the gsmSCF
3	The gsmSCF invokes the CAP ***ReleaseGPRS*** operation

Parameter Mapping

From: ***release***	To: CAP ***ReleaseGPRS***
dataSessionID	
	gPRS-ReferenceNumber
cause	**gPRSCause**
	pDPID

13.2.5 superviseDataSessionReq

superviseDataSessionReq is called by the application to supervise a data session. The application can set a granted data volume for this data session. If an application calls this function before it calls a `connectReq()` or a user interaction function the time measurement will start as soon as the data session is connected. The Data Session object will exist after the data session has been terminated if information is required to be sent to the application at the end of the data session.

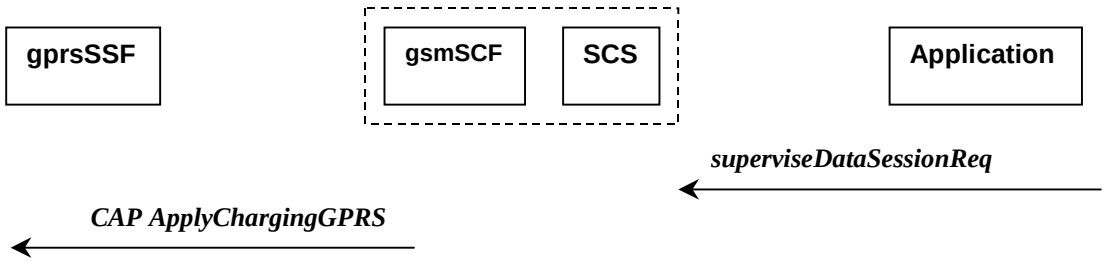


Figure 13-11: Call Flow for superviseDataSessionReq

Normal Operation

Pre-conditions	
1	The application invokes the ***superviseDataSessionReq*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ApplyChargingGPRS message to the gprsSSF

Parameter Mapping

From: ***superviseDataSessionReq***	To: CAP ***ApplyChargingGPRS***
dataSessionID	
	gPRS-ReferenceNumber
treatment	
bytes	**chargingCharacteristics** **maxTransferedVolume**
	pDPID

13.2.6 superviseDataSessionRes

superviseDataSessionRes is an asynchronous method that reports a data session supervision event to the application.

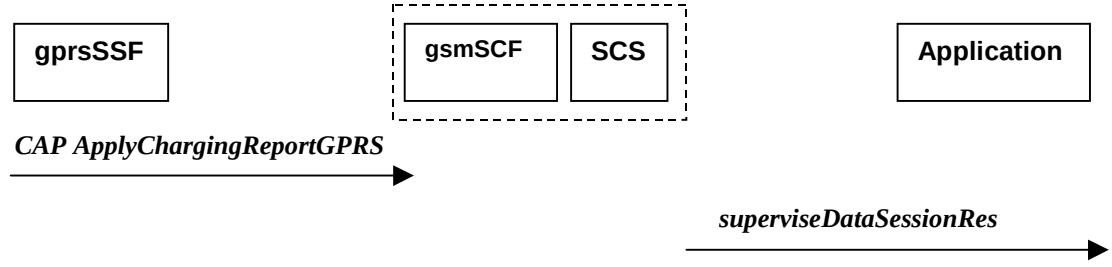


Figure 13-12: Call Flow for superviseDataSessionRes

Normal Operation

Pre-conditions	The application has invoked the ***superviseDataSessionReq*** method
1	The gsmSCF receives an CAP ***ApplyChargingReportGPRS*** from the gprsSSF
2	The gsmSCF sends an equivalent internal message to the SCS

3	The SCS identifies the correct application and invokes the ***superviseDataSessionRes*** method.

Parameter Mapping

From: CAP ***ApplyChargingReportGPRS***	To: ***superviseDataSessionRes***
	dataSessionID
gPRSReferenceNumber	
	report
chargingResult **transferedVolume**	**usedVolume**
qualityOfService	
pDPID	
active	

13.2.7 superviseDataSessionErr

superviseDataSessionErr is an asynchronous method that reports a data session supervision error to the application.

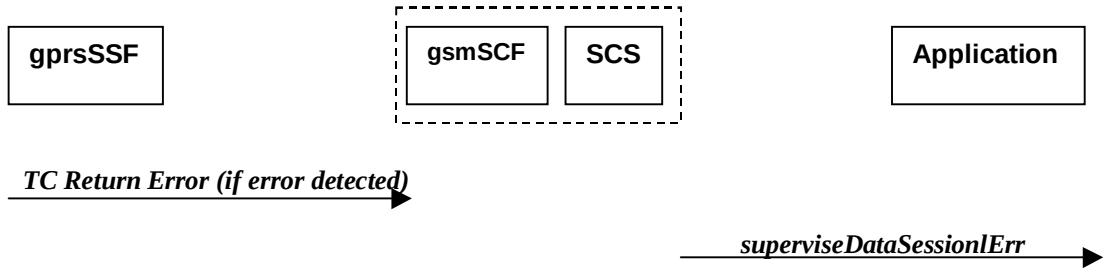


Figure 13-13: Call Flow for superviseDataSessionErr

Normal Operation

Pre-conditions	The application has requested information associated with a call via the ***superviseDataSessionReq*** method
1	A data session terminates abnormally and the gprsSSF sends an error in a TCAP message to the gsmSCF , or aborts the TCAP dialogue
2	The gsmSCF sends an equivalent message to the SCS
3	The SCS identifies the correct applications that requested the data session information and invokes the ***superviseDataSessionErr*** method.

Parameter Mapping

From: TCAP ***Return Error***	To: ***superviseCallErr***
	dataSessionID
TC Primitives TC-U-ABORT	**error**

TC-P-ABORT TC-NOTICE TC-U-ERROR TC-L-CANCEL TC-U-CANCEL TC-L-REJECT TC-R-REJECT TC-U-REJECT	

13.2.8 dataSessionFaultDetected

dataSessionFaultDetected indicates to the application that a fault in the network has been detected which can't be communicated by a network event, e.g., when the user aborts before any establishment method is called by the application.

The system purges the Data Session object. Therefore, the application has no further control of data session processing. No report will be forwarded to the application.

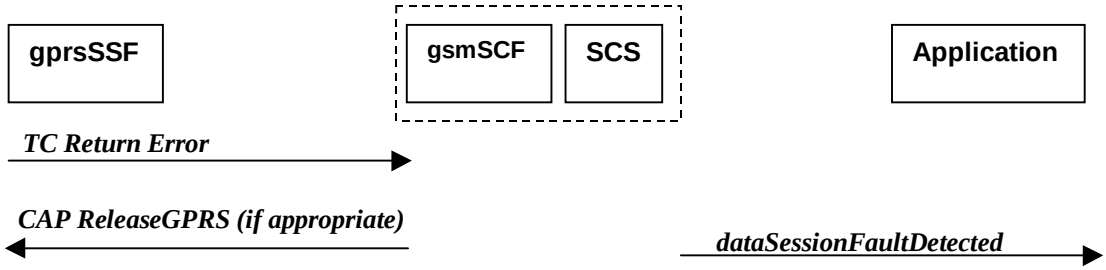


Figure 13-14: Call Flow for dataSessionFaultDetected

Normal Operation

Pre-conditions	A data session exists and the SCS detects an error. No ***connectReq*** method has been invoked yet.
1	The gprsSSF may detect a fault and sends an appropriate dialogue error message to the gsmSCF
2	The gsmSCF may detect a fault an send an error message to the SCS
3	The SCS detects a fault and invokes the ***dataSessionFaultDetected*** method
4	The SCS sends an equivalent message to the gsmSCF if appropriate
5	The gsmSCF sends a CAP ***ReleaseGPRS*** if appropriate

Parameter Mapping

From:***Dialogue Error***	To: ***dataSessionFaultDetected***
	dataSessionID
TC_U_ABORT	**fault**

13.2.9 setAdviceOfCharge

setAdviceOfCharge is a method that allows the application to determine the charging information that will be send to the end-users terminal.

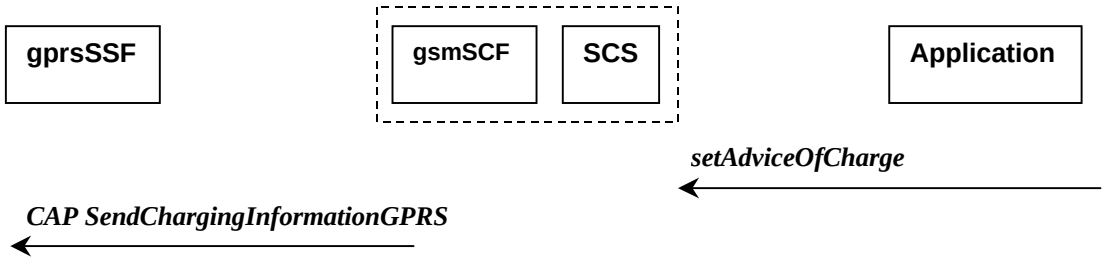


Figure 13-15: Call Flow for setAdviceOfCharge

Normal Operation

Pre-conditions	
1	The application invokes the ***setAdviceOfCharge*** method
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***SendChargingInformationGPRS*** message to the gprsSSF

Parameter Mapping

From: ***setAdviceOfCharge***	To: CAP ***SendChargingInformationGPRS***
sessionID	
aOCInfo: - **CurrentCAI**	**SCIGPRSBillingChargingCharateristics** **aOCGPRS** **aOCInitial**
- **NextCAI**	**SCIGPRSBillingChargingCharateristics** **aOCGPRS** **aOCSubsequent** **cAI-GSM0224**
tariffSwitch	**SCIGPRSBillingChargingCharateristics** **aOCGPRS** **aOCSubsequent** **tariffSwitchInterval**
	SCIGPRSBillingChargingCharateristics **aOCGPRS** **pDPID**

13.2.10 setDataSessionChargePlan

setDataSessionChargePlan is a method that allows the application to include charging information for data sessions in network generated CDR.

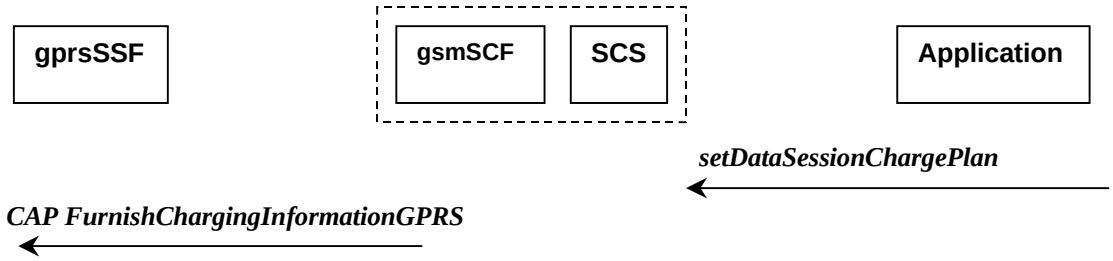


Figure 13-16: Call Flow for setDataSessionChargePlan

Normal Operation

Pre-conditions	
1	The application invokes the ***setDataSessionChargePlan***
2	The SCS sends an equivalent internal message to the gsmSCF
3	The gsmSCF sends a CAP ***FurnishChargingInformationGPRS*** message to the gprsSSF

Parameter Mapping

From: ***setDataSessionPlan***	To: CAP ***FurnishChargingInformationGPRS***
dataSessionID	
dataSessionChargePlan	**FCIGPRSBillingChargingCharacteritics** **fCIBCCCAMELsequence1** **freeFormatData**
	FCIGPRSBillingChargingCharacteritics **fCIBCCCAMELsequence1** **appendFreeFormatData**
	FCIGPRSBillingChargingCharacteritics **fCIBCCCAMELsequence1** **pDPID**

14 Detailed Parameter Mappings

This section contains detailed parameter mappings for data types that are used in the Parameter Mapping tables in the previous sections.

14.1 TpCallMonitorMode

TpCallMonotirMode	**monitorMode**
P_CALL_MONITOR_MODE_INTERRUPT	interrupted
P_CALL_MONITOR_MODE_NOTIFY	notifyAndContinue
P_CALL_MONITOR_MODE_DO_NOT_MONITOR	transparent

14.2 TpCallReportType

TpCallReportType	**eventTypeBCSM**
P_CALL_REPORT_UNDEFINED	analyzedInformation
P_CALL_REPORT_PROGRESS	<no mapping available>
P_CALL_REPORT_ALERTING	<no mapping available>
P_CALL_REPORT_ANSWER	oAnswer tAnswer
P_CALL_REPORT_REFUSED_BUSY	oCalledPartyBusy tBusy
P_CALL_REPORT_NO_ANSWER	oNoAnswer tNoAnswer
P_CALL_REPORT_DISCONNECT	tDisconnect
P_CALL_REPORT_REDIRECTED	<no mapping available>
P_CALL_REPORT_SERVICE_CODE	<no mapping available>
P_CALL_REPORT_ROUTING_FAILURE	routeSelectFailure

14.3 TpCallEventName

TpCallEventName	**eventTypeBCSM**
P_EVENT_NAME_UNDEFINED	<no mapping available>
P_EVENT_GCCS_OFFHOOK_EVENT	<no mapping available>
P_EVENT_GCCS_ADDRESS_COLLECTED_EVENT	O-CSI[17] O-BcsmTriggerDetectionPoint: collectedInfo T-CSI/VT-CSI: T-BcsmTriggerDetectionPoint: termAttemptAuthorized
P_EVENT_GCCS_ADDRESS_ANALYSED_EVENT	O-CSI O-BcsmTriggerDetectionPoint analysedInfo
P_EVENT_GCCS_CALLED_PARTY_BUSY	 T-CSI/VT-CSI: T-BcsmTriggerDetectionPoint: tBusy
P_EVENT_GCCS_CALLED_PARTY_UNREACHABLE	mapped to the cause value returned with TBusy : T-CSI/VT-CSI: T-BcsmTriggerDetectionPoint: tBusy
P_EVENT_GCCS_NO_ANSWER_FROM_CALLED_PARTY	T-CSI/VT-CSI: T-BcsmTriggerDetectionPoint:

[17] O-CSI applies when the value for CallNotificationType is P_ORIGINATING, T-CSI applies when the value for CallNotificationType is P_TERMINATING

	tNoAnswer
P_EVENT_GCCS_ROUTE_SELECT_FAILURE	O-CSI: O-BcsmTriggerDetectionPoint: routeSelectFailure
P_EVENT_GCCS_ANSWER_FROM_CALL_PARTY	T-CSI/VT-CSI: T-BcsmTriggerDetectionPoint: tAnswer

14.4 TpCallAdditionalReportInfo

TpCallAdditionalReportInfo	**eventSpecificInformationBCSM**
RefusedBusy	oCalledPartyBusy busyCause or tBusySpecificInfo busyCause callForwarded (no mapping)
CallDisconnect	oDisconnectSpecificInfo - releaseCause tDisconnectSpecificInfo - releaseCause
ForwardAddress	oAnswerSpecificInfo - destinationAddress - or-Call (no mapping) - forwardedCall (no mapping) tAnswerSpecificInfo - destinationAddress - or-Call (no mapping) - forwardedCall (no mapping)
ServiceCode	<no mapping available>
RoutingFailure	routeSelectFailureSpecificInfo - failureCause
	tNoAnswerSpecificInfo - callForwarded

Annex A (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
June 2000	CN#08	NP-000311			Approval of Report	2.0.0	3.0.0
Sept. 2000	CN#09	NP-000521	001	2	Mapping of CallNotification interupted CallNotification continue methods.	3.0.0	3.1.0
Sept. 2000	CN#09	NP-000521	002		callEnded method mapping correction	3.0.0	3.1.0
Sept. 2000	CN#09	NP-000521	003		Mapping of userInteractionNotificationInterrupted and userInteractionNotificationContinued methods	3.0.0	3.1.0

History

Document history		
V3.0.0	June 2000	Publication
V3.1.0	September 2000	Publication