3GPP TSG_CN **Tdoc NP-000519**
Plenary Meeting #9, Oahu, Hawaii
20th – 22nd September 2000.

Source: **TSG_N WG 5**

Title: **CRs to R99 Work Item OSA - corrections to 29.198**

Agenda item: **8.23.5**

Document for: **APPROVAL**

Introduction:

This document contains 12 CRs on R99 Work Item **OSA** that has been agreed by TSG_N WG5, and is forwarded to TSG_N Plenary meeting #9 for approval.

Spec	CR	Rev	Doc-2nd-Level	Phase	Subject	Cat	Ver_C	Ver_N
29.198	001	1	N5-000118	R99	Improvement of User Interaction STDs	F	3.0.0	3.1.0
29.198	003	2	N5-000120	R99	Renumbering of GCCS exceptions	F	3.0.0	3.1.0
29.198	004	1	N5-000121	R99	Remove of E.164 Mobile and correction of numbering in TpAddressPlan	F	3.0.0	3.1.0
29.198	005		N5-000132	R99	Common IDL interfaces for Generic Call Control and Generic User Interaction between 3GPP, ETSI SPAN3 and Parlay	F	3.0.0	3.1.0
29.198	006		N5-000133	R99	Correction to table with overview of IDL files	F	3.0.0	3.1.0
29.198	007		N5-000134	R99	Reduction in name scoping in IDL for createUICall operation on IpUICall interface	F	3.0.0	3.1.0
29.198	008	2	N5-000171	R99	Alignment of Framework with Parlay 2.1, improvement on business entity identification	F	3.0.0	3.1.0
29.198	009	2	N5-000172	R99	Alignment of Framework with Parlay 2.1, correction of missing service token	F	3.0.0	3.1.0
29.198	010	2	N5-000173	R99	Alignment of Framework with Parlay 2.1, parameter name and data-type alignments	F	3.0.0	3.1.0
29.198	011	1	N5-000138	R99	Alignment of Framework with Parlay 2.1, one interface per application correction	F	3.0.0	3.1.0
29.198	012	1	N5-000139	R99	Alignment of Framework with Parlay 2.1, only one error returned in load manager query	F	3.0.0	3.1.0
29.198	013	1	N5-000140	R99	Alignment of Framework with Parlay 2.1, missing operation fwUnavailableInd in IpAppFaultManager.	F	3.0.0	3.1.0

3GPP Meeting CN5 #4
Retz, 10-11 July 2000

Document N5-000118
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	001R1	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*	for approval	X	strategic		*(for SMG use only)*
	for information		non-strategic		

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM		ME		UTRAN / Radio		Core Network	X

Source:	N5			**Date:**	6 July 2000
Subject:	Improvement of User Interaction STDs				
Work item:	OSA				

Category: *(only one category shall be marked with an X)*			**Release:**	
	F Correction	X		Phase 2
	A Corresponds to a correction in an earlier release			Release 96
	B Addition of feature			Release 97
	C Functional modification of feature			Release 98
	D Editorial modification			Release 99 X
				Release 00

Reason for change:

Improvement of the state transition diagrams for the UI and UICall object.

Changes are:
- introduction of the “Finished” state
- transition from “Release Pending” to “Active” in case the final request was not successful.
- more clear separation between events that don’t cause a state transition from the application side and events from the network side. The first are shown as a self-transition with a semi-circle while the latter are shown as self-transitions on the bottom right of a state.
- addition of more descriptive text, explaining the transitions.

Clauses affected: 7.3.2, 7.3.3

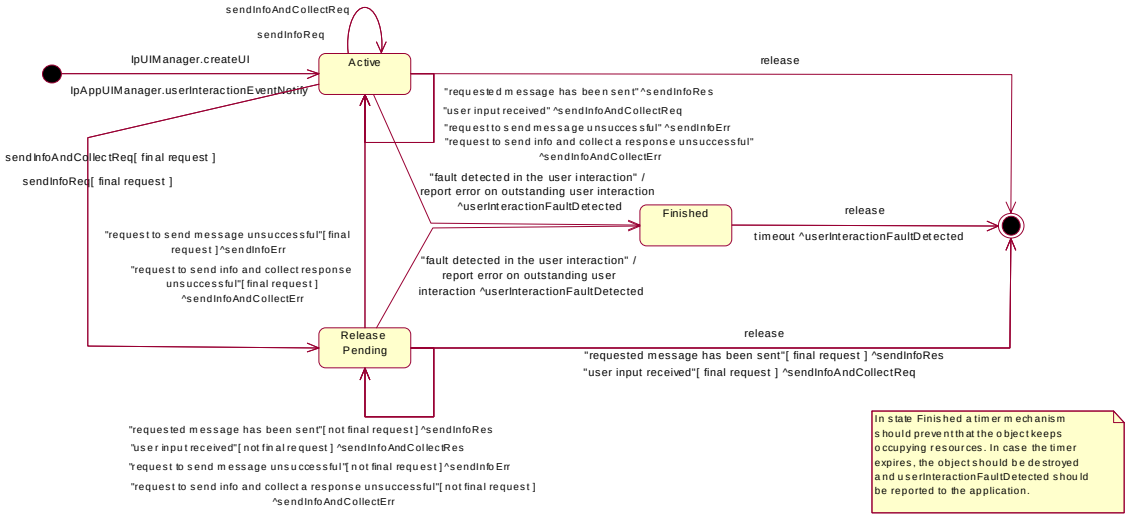
Other specs affected:			
Other 3G core specifications		→ List of CRs:	
Other GSM core specifications		→ List of CRs:	
MS test specifications		→ List of CRs:	
BSS test specifications		→ List of CRs:	
O&M specifications		→ List of CRs:	

Other comments:


help.doc

<---------- double-click here for help and instructions on how to create a CR.

7.3.2 UI


sendInfoAndCollectReq
sendInfoReq
IpUIManager.createUI
IpAppUIManager.userInteractionEventNotify
Active
release
"requested message has been sent" ^sendInfoRes
"user input received" ^sendInfoAndCollectReq
"request to send message unsuccessful" ^sendInfoErr
"request to send info and collect a response unsuccessful" ^sendInfoAndCollectErr
sendInfoAndCollectReq[final request]
sendInfoReq[final request]
"fault detected in the user interaction" / report error on outstanding user interaction ^userInteractionFaultDetected
Finished
release
timeout ^userInteractionFaultDetected
"request to send message unsuccessful"[final request] ^sendInfoErr
"request to send info and collect response unsuccessful"[final request] ^sendInfoAndCollectErr
"fault detected in the user interaction" / report error on outstanding user interaction ^userInteractionFaultDetected
Release Pending
release
"requested message has been sent"[final request] ^sendInfoRes
"user input received"[final request] ^sendInfoAndCollectReq
"requested message has been sent"[not final request] ^sendInfoRes
"user input received"[not final request] ^sendInfoAndCollectRes
"request to send message unsuccessful"[not final request] ^sendInfoErr
"request to send info and collect a response unsuccessful"[not final request] ^sendInfoAndCollectErr
In state Finished a timer mechanism should prevent that the object keeps occupying resources. In case the timer expires, the object should be destroyed and userInteractionFaultDetected should be reported to the application.

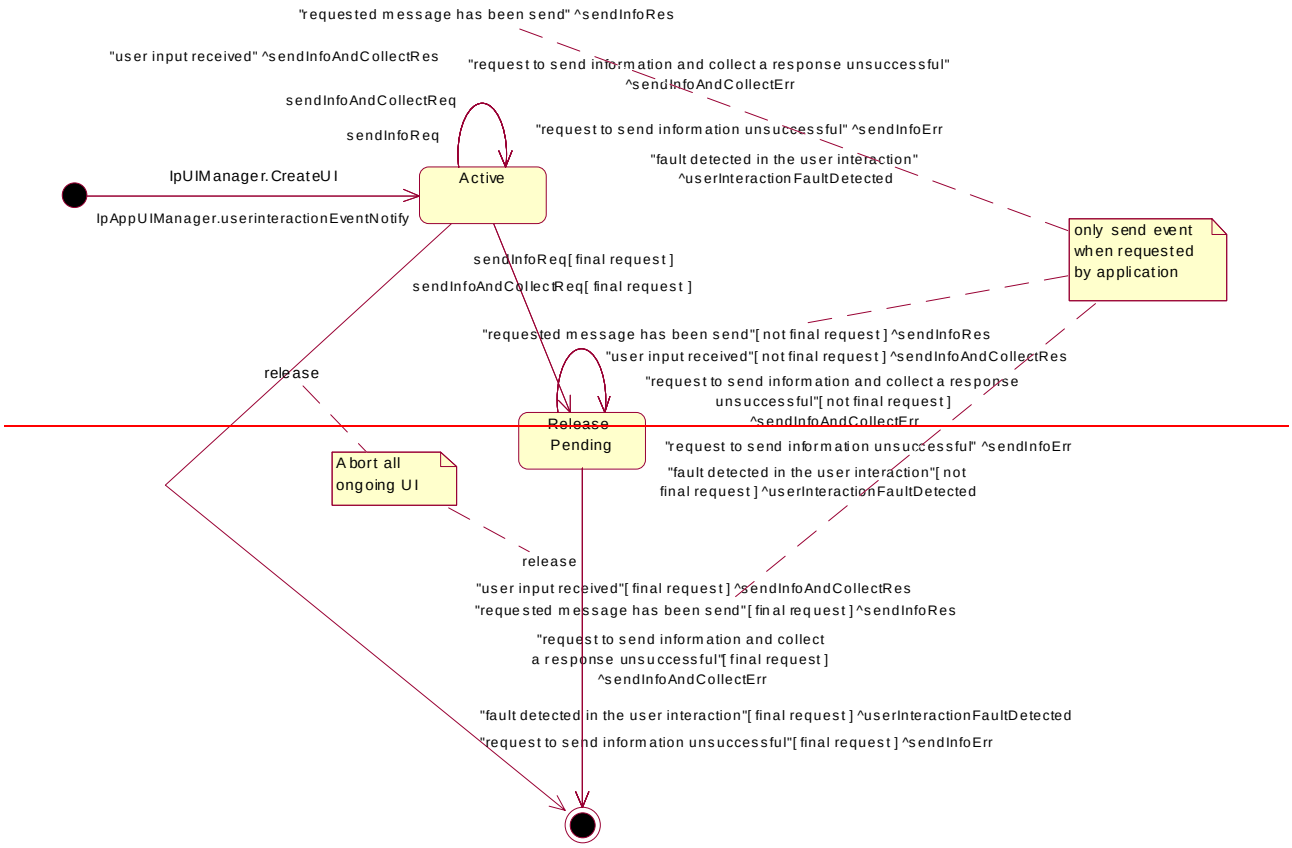


Figure 7-14: State Transition Diagram for UI

7.3.2.1 Active state

In this state the UI object is available for requesting messages to be send to the network.

In case a fault is detected on the user interaction (e.g. a link failure to the IVR system), userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

7.3.2.2 Release Pending state

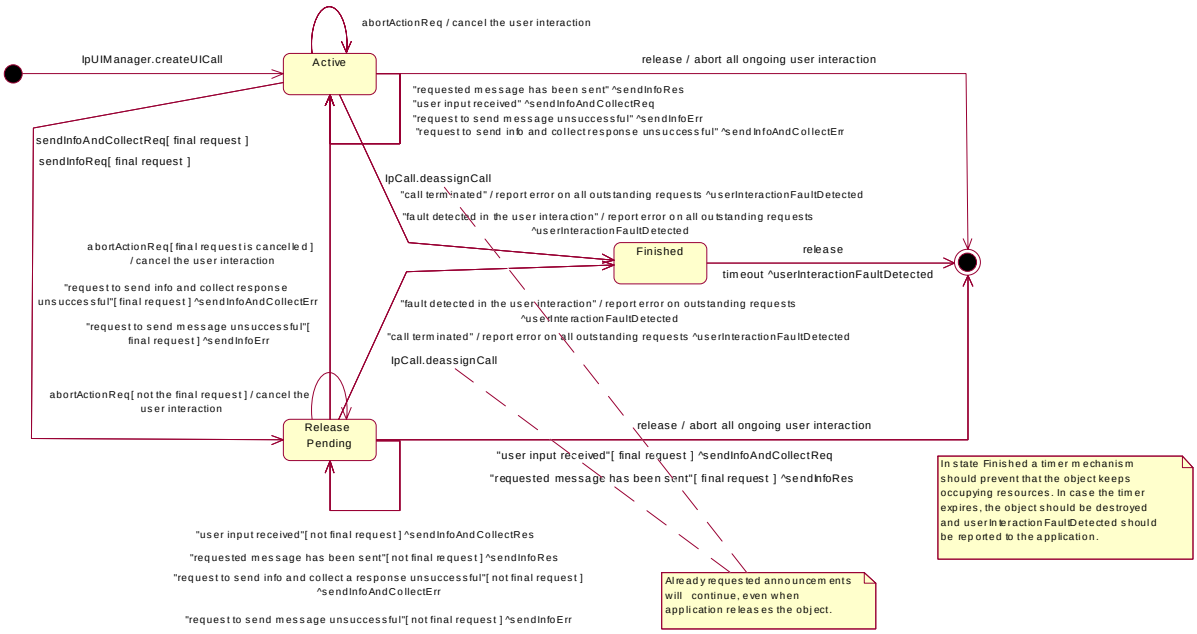
A transition to this state is made when the Application has indicated that after a certain message no further messages need to be sent~~d~~ to the end-user. There are, however, still a number of messages that are not yet completed. When the last message is sent or when the last user interaction has been obtained, the UI object is destroyed.
In case the final request failed or the application requested to abort the final request, a transition is made back to the Active state.

In case a fault is detected on the user interaction (e.g. a link failure to the IVR system), userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

7.3.3.3 Finished

In this state the user interaction has ended. The application can only release the UI object. Note that the application has to release the object itself as good OO practice requires that when an object is created on behalf of a certain entity, this entity is also responsible for destroying it when the object is no longer needed

7.3.3 UI Call


abortActionReq / cancel the user interaction
IpUIManager.createUICall
Active
release / abort all ongoing user interaction
"requested message has been sent" ^sendInfoRes
"user input received" ^sendInfoAndCollectReq
"request to send message unsuccessful" ^sendInfoErr
"request to send info and collect response unsuccessful" ^sendInfoAndCollectErr
sendInfoAndCollectReq[final request]
sendInfoReq[final request]
IpCall.deassignCall
"call terminated" / report error on all outstanding requests ^userInteractionFaultDetected
"fault detected in the user interaction" / report error on all outstanding requests ^userInteractionFaultDetected
abortActionReq[final request is cancelled] / cancel the user interaction
Finished
release
timeout ^userInteractionFaultDetected
"request to send info and collect response unsuccessful"[final request] ^sendInfoAndCollectErr
"request to send message unsuccessful"[final request] ^sendInfoErr
"fault detected in the user interaction" / report error on outstanding requests ^userInteractionFaultDetected
"call terminated" / report error on all outstanding requests ^userInteractionFaultDetected
IpCall.deassignCall
abortActionReq[not the final request] / cancel the user interaction
Release Pending
release / abort all ongoing user interaction
"user input received"[final request] ^sendInfoAndCollectReq
"requested message has been sent"[final request] ^sendInfoRes
In state Finished a timer mechanism should prevent that the object keeps occupying resources. In case the timer expires, the object should be destroyed and userInteractionFaultDetected should be reported to the application.
"user input received"[not final request] ^sendInfoAndCollectRes
"requested message has been sent"[not final request] ^sendInfoRes
"request to send info and collect a response unsuccessful"[not final request] ^sendInfoAndCollectErr
"request to send message unsuccessful"[not final request] ^sendInfoErr
Already requested announcements will continue, even when application releases the object.

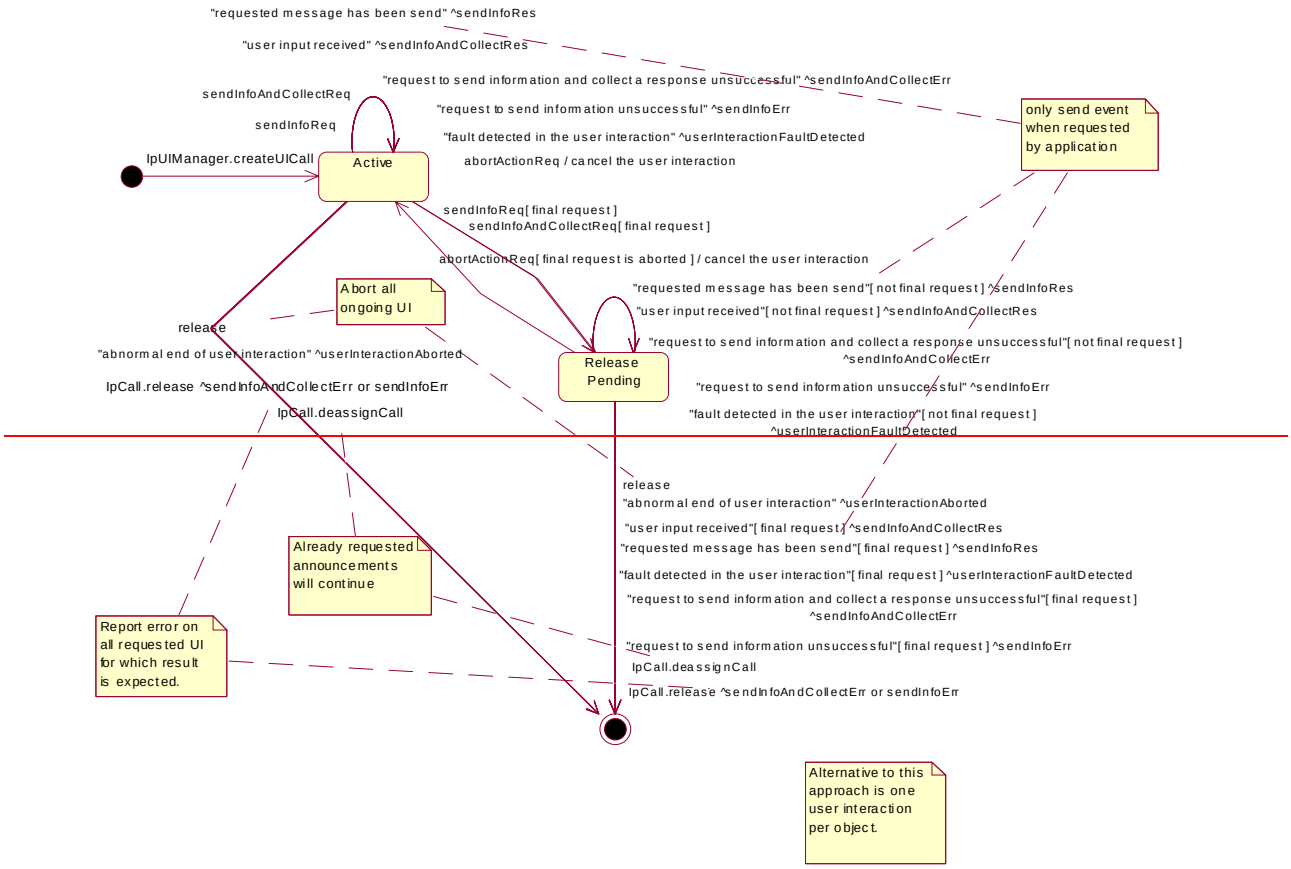


Figure 7-15: State Transition Diagram for UICall

7.3.3.1 Active state

In this state a UICall object is available for announcements to be played to an end-user or obtaining information from the end-user.

When the application de-assigns the related Call object, a transition is made to the Finished state. However, all requested announcements will continue, even when the application releases the UICall object.
When the related call is due to some reason terminated, a transition is made to the Finished state, the operation userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.
In case a fault is detected on the user interaction (e.g. a link failure to the IVR system), userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

7.3.3.2 Release Pending state

A transition to this state is made when the Application has indicated that after a certain announcement no further announcements need to be played to the end-user. There are, however, still a number of announcements that are not yet completed. When the last announcement is played or when the last user interaction has been obtained, the UICall object is destroyed.
In case the final request failed or the application requested to abort the final request, a transition is made back to the Active state.

When the application de-assigns the related Call object, a transition is made to the Finished state. However, all requested announcements will continue, even when the application releases the UICall object.
When the related call is due to some reason terminated, a transition is made to the Finished state, the operation userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

In case a fault is detected on the user interaction (e.g. a link failure to the IVR system), userInteractionFaultDetected() will be invoked on the application and an error will be reported on all outstanding requests.

7.3.3.4 Finished

In this state the user interaction has ended. The application can only release the UICall object. Note that the application has to release the object itself as good OO practice requires that when an object is created on behalf of a certain entity, this entity is also responsible for destroying it when the object is no longer needed

3GPP Meeting CN5 #4
Retz, 10-11 July 2000

Document N5-000120
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	003R2	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*	for approval	X	strategic		*(for SMG use only)*
	for information		non-strategic		

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)* (U)SIM [ ] ME [ ] UTRAN / Radio [ ] Core Network [X]

Source: N5 **Date:** 6 July 2000

Subject: Correction of numbering in TpResultInfo

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	X	**Release:**	Phase 2	
A	Corresponds to a correction in an earlier release			Release 96	
B	Addition of feature			Release 97	
C	Functional modification of feature			Release 98	
D	Editorial modification			Release 99	X
				Release 00	

Reason for change: Correction of a few numbering errors in the TpResultInfo

Clauses affected: 8.1.4.8, 9

Other specs affected:

Other 3G core specifications		→ List of CRs:
Other GSM core specifications		→ List of CRs:
MS test specifications		→ List of CRs:
BSS test specifications		→ List of CRs:
O&M specifications		→ List of CRs:

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

8.1.4.8 TpResultInfo

Defines further information relating to the result of the method, such as error codes.

Name	Value	Description
P_RESULT_INFO_UNDEFINED	0000h	No further information present
P_INVALID_APPLICATION_ID	0001h	Invalid application ID
P_INVALID_CLIENT_CAPABILITY	0002h	Invalid client capability
P_INVALID_AGREEMENT_TEXT	0003h	Invalid agreement text
P_INVALID_SIGNING_ALGORITHM	0004h	Invalid signing algorithm
P_INVALID_INTERFACE_ID	0005h	Invalid interface ID
P_INVALID_SERVICE_ID	0006h	Invalid service capability feature ID
P_INVALID_EVENT_TYPE	0007h	Invalid event type
P_SERVICE_NOT_ENABLED	0008h	The service capability feature ID does not correspond to a SCF that has been enabled
P_INVALID_ASSIGNMENT_ID	0009h	The assignment ID does not correspond to one of the valid assignment IDs
P_INVALID_PARAMETER	000Ah	The method has been called with an invalid parameter
P_INVALID_PARAMETER_VALUE	000Bh	A method parameter has an invalid value
P_PARAMETER_MISSING	000Ch	A required parameter has not been specified in the method call
P_RESOURCES_UNAVAILABLE	000Dh	The required resources in the network are not available
P_TASK_REFUSED	000Eh	The requested method has been refused
P_TASK_CANCELLED	000Fh	The requested method has been cancelled
P_INVALID_DATE_TIME_FORMAT	0010h	Invalid date and time format provided
P_NO_CALLBACK_ADDRESS_SET	0011h	The requested method has been refused because no callback address is set
P_INVALID_TERMINATION_TEXT	0012h	Invalid termination text
P_INVALID_SERVICE_TOKEN	0013h	The service capability feature token does not correspond to a token that had been issued, or the issued token has expired
P_INVALID_AUTHENTICATION	0014h	The client has not been correctly authenticated
P_INVALID_SERVICE_PROPERTY	0015h	Invalid service capability feature property
P_METHOD_NOT_SUPPORTED	001AB h	The method is not allowed or supported within the context of the current SCF agreement.
General security errors		
P_USER_NOT_SUBSCRIBED	0030h	A service (or application) is unauthorised to access information and request SCFs with regards to users that are **not** subscribed to it.
P_APPLICATION_NOT_ACTIVATED	0031h	A service (or application) is unauthorised to access information and request SCFs with regards to its subscribed users that have **deactivated** that particular service (or application).
P_USER_PRIVACY	0032h	A service (or application) is unauthorised to access information and request an SCF with regards to its subscribed users that have **set** their privacy flag regarding that particular SCF.
P_GCCS_SERVICE_INFORMATION_MISSING	0100h	Information relating to the Call Control SCF could not be found
P_GCCS_SERVICE_FAULT_ENCOUNTERED	0101h	Fault detected in the Call Control SCF
P_GCCS_UNEXPECTED_SEQUENCE	0102h	Unexpected sequence of methods, i.e., the sequence does not match the specified state diagrams for the call or the call leg.
P_GCCS_INVALID_ADDDRESS	0103h	Invalid address specified
P_GCCS_INVALID_CRITERIA	01045h	Invalid criteria specified
P_GCCS_INVALID_NETWORK_STATE	01056h	Although the sequence of method calls is allowed by the OSA gateway, the underlying protocol can not support it. E.g., in some protocols some methods are only allowed by the protocol, when the call processing is suspended, e.g., after reporting an event that was monitored in interrupt mode.

P_GUIS_INVALID_CRITERIA	0300h	Invalid criteria specified
P_GUIS_ILLEGAL_ID	0301h	Information id specified is invalid
P_GUIS_ID_NOT_FOUND	0302h	A legal information id is not known to the User Interaction SCF
P_GUIS_ILLEGAL_RANGE	0303h	The values for minimum and maximum collection length are out of range.
P_GUIS_INVALID_COLLECTION_CRITERIA	0304h	Invalid collection criteria specified
P_GUIS_INVALID_NETWORK_STATE	03056h	Although the sequence of method calls is allowed by the OSA gateway, the underlying protocol can not support it. E.g., in some protocols some methods are only allowed by the protocol, when the call processing is suspended, e.g., after reporting an event that was monitored in interrupt mode.
P_GUIS_UNEXPECTED_SEQUENCE	03067h	Unexpected sequence of methods, i.e., the sequence does not match the specified state diagrams.
P_DSCS_SERVICE_INFORMATION_MISSING	0400h	Information relating to the Data Session Control SCF could not be found
P_DSCS_SERVICE_FAULT_ENCOUNTERED	0401h	Fault detected in the Data Session Control SCF
P_DSCS_UNEXPECTED_SEQUENCE	0402h	Unexpected sequence of methods, i.e., the sequence does not match the specified state diagrams for the data session.
P_DSCS_INVALID_ADDDRESS	0403h	Invalid address specified
P_DSCS_INVALID_STATE	0404h	Invalid state specified
P_DSCS_INVALID_CRITERIA	0405h	Invalid criteria specified
P_DSCS_INVALID_NETWORK_STATE	0406h	Although the sequence of method calls is allowed by the OSA gateway, the underlying protocol can not support it.

This change should also be reflected in the IDL (chapter 9).

```
		// Defines the general Parlay exception values
			enum TpGeneralExceptionType
			{
				P_RESULT_INFO_UNDEFINED,    // No further information present
				P_INVALID_APPLICATION_ID,   // Invalid application ID
				P_INVALID_CLIENT_CAPABILITY,// Invalid client capability
				P_INVALID_AGREEMENT_TEXT,   // Invalid agreement text
				P_INVALID_SIGNING_ALGORITHM,// Invalid signing algorithm
				P_INVALID_INTERFACE_NAME,   // Invalid interface name
				P_INVALID_SERVICE_ID,       // Invalid service capability
feature ID
				P_INVALID_EVENT_TYPE,       // Invalid event type
				P_SERVICE_NOT_ENABLED,      // The SCF ID does not correspond
                                                        // to a SCF
that has been enabled
				P_INVALID_ASSIGNMENT_ID,    // The assignment ID does not
                                                        // correspond
to one of the valid assignment IDs
				P_INVALID_PARAMETER,        // The method has been called with
an
                                                        // invalid
parameter
				P_INVALID_PARAMETER_VALUE,  // A method parameter has an invalid
value
				P_PARAMETER_MISSING,        // A required parameter has not been
                                                        // specified
in the method call
				P_RESOURCES_UNAVAILABLE,    // The required resources in the
                                                        // network
are not available
				P_TASK_REFUSED,             // The requested method has been
refused
				P_TASK_CANCELLED,           // The requested method has been
cancelled
				P_INVALID_DATE_TIME_FORMAT, // Invalid date and time format
provided
				P_NO_CALLBACK_ADDRESS_SET,  // The requested method has been
refused
                                                        // because no
callback address is set
				P_INVALID_TERMINATION_TEXT, // Invalid termination text
				P_INVALID_SERVICE_TOKEN,    // The SCF token does not correspond
to a
```

```
                                                                        // token that
had been issued, or the issued token
                                                                        // has
expired.
                        P_INVALID_AUTHENTICATION,   // The client has not been correctly
authenticated
                        P_INVALID_SERVICE_PROPERTY, // Invalid service capability
feature property.
                        P_METHOD_NOT_SUPPORTED      // The method is not allowed or
supported within
                                                                        // the
context of the current SCF agreement.
                };

                exception TpGeneralException
                {
                        TpGeneralExceptionType exceptionType;
                };

                // Defines the GCCS OSA exception values
                enum TpGCCSExceptionType
                {
                        P_GCCS_SERVICE_INFORMATION_MISSING,// Information relating to
the Call
                                                                                //
Control SCF could not be found
                        P_GCCS_SERVICE_FAULT_ENCOUNTERED,  // Fault detected in the Call
Control SCF
                        P_GCCS_UNEXPECTED_SEQUENCE,  // Unexpected sequence of methods,
i.e.,
                                                                        // the
sequence does not match the specified
                                                                        // state
diagrams for the call or the call leg.
                        P_GCCS_INVALID_ADDDRESS,     // Invalid address specified
                        P_GCCS_INVALID_STATE,        // Invalid state specified
                        P_GCCS_INVALID_CRITERIA,     // Invalid criteria specified
                        P_GCCS_INVALID_NETWORK_STATE,// Although the sequence of method
calls is
                                                                        // allowed
by the OSA gateway, the underlying
                                                                        // protocol
can not support it. E.g., in some
                                                                        // protocols
some methods are only allowed by
                                                                        // the
protocol, when the call processing is
                                                                        //
suspended, e.g., after reporting an event
                                                                        // that was
monitored in interrupt mode.
                        P_GCCS_NETWORK_DEASSIGN // The relation between the network and
the OSA
                                                                        // gateway is
terminated. Therefore, the gateway
                                                                        // can no longer
influence the call. This can happen
                                                                        // after the last
requested report is sent to the
                                                                        // application. To
prevent this error, the application
                                                                        // should ensure
that it has requested events which
                                                                        // are not yet
reported.
                };

                exception TpGCCSException
                {
                        TpGCCSExceptionType exceptionType;
                };

                // Defined the GUIS OSA exception values
                enum TpGUISExceptionType
                {
                        P_GUIS_INVALID_CRITERIA, // Invalid criteria specified
                        P_GUIS_ILLEGAL_ID,       // Information id specified is invalid
                        P_GUIS_ID_NOT_FOUND,     // A legal information id is not known
to the User
                                                                        // Interaction SCF
                        P_GUIS_ILLEGAL_RANGE,    // The values for minimum and maximum
collection
                                                                        // length are out
of range.
                        P_GUIS_INVALID_COLLECTION_CRITERIA, // Invalid collection
criteria specified
                        P_GUIS_NETWORK_DEASSIGN, // The relation between the network and
the OSA
```

~~// gateway is terminated. Therefore, the gateway // can no longer perform UI operations. This can // happen after the last requested report is sent // to the application. To prevent this error, the /// application should ensure that it has requested // events which are not yet reported.~~

P_GUIS_INVALID_NETWORK_STATE // Although the sequence of method calls is // allowed by the OSA gateway, the underlying // protocol can not support it. E.g., in some // protocols some methods are only allowed by // the protocol, when the call processing is // suspended, e.g., after reporting an event // that was monitored in interrupt mode.

};

exception TpGUISException
{
TpGUISExceptionType exceptionType;
};

3GPP Meeting CN5 #4
Retz, 10-11 July 2000

Document N5-000121
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	004R1	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	CN#09	for approval	X	strategic		(for SMG use only)
list expected approval meeting # here ↑		for information		non-strategic		

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM		ME		UTRAN / Radio		Core Network	X

Source: N5 **Date:** 6 July 2000

Subject: Removal of E.164 Mobile and correction of numbering in TpAddressPlan

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	X	**Release:**	Phase 2	
A	Corresponds to a correction in an earlier release			Release 96	
B	Addition of feature			Release 97	
C	Functional modification of feature			Release 98	
D	Editorial modification			Release 99	X
				Release 00	

Reason for change: The TpAddressPlan data-type currently contains a type P_ADDRESS_PLAN_E164_MOBILE. As there is no special E.164 numbering for the mobile network this type should be removed.
Furthermore, the types in TpAddressPlan should be numbered in a sequential order.

Clauses affected: 8.1.4.14, 9

Other specs affected:

Other 3G core specifications		→ List of CRs:	
Other GSM core specifications		→ List of CRs:	
MS test specifications		→ List of CRs:	
BSS test specifications		→ List of CRs:	
O&M specifications		→ List of CRs:	

Other comments: Note: In Parlay 2.1 the value 9 is reserved to P_ADDRESS_PLAN_MSMAIL. In the Antwerpen meeting we came to the conclusion that MSMAIL should be the same as SMTP and therefore the MSMAIL was removed. We also agreed to keep the same numbering as Parlay, therefore the number 9 in 3GPP is not used.

help.doc

<--------- double-click here for help and instructions on how to create a CR.

8.1.4.14 TpAddressPlan

Defines the address plan (or numbering plan) used. It is also used to indicate whether an address is actually defined in a Address data element.

Name	Value	Description
P_ADDRESS_PLAN_NOT_PRESENT	-1	No Address Present
P_ADDRESS_PLAN_UNDEFINED	0	Undefined
P_ADDRESS_PLAN_IP	1	IP
P_ADDRESS_PLAN_MULTICAST	2	Multicast
P_ADDRESS_PLAN_UNICAST	3	Unicast
P_ADDRESS_PLAN_E164	4	E.164
~~P_ADDRESS_PLAN_E164_MOBILE~~	~~5~~	~~E.164 Mobile~~
P_ADDRESS_PLAN_AESA	5~~6~~	AESA
P_ADDRESS_PLAN_URL	6~~7~~	URL
P_ADDRESS_PLAN_NSAP	7~~8~~	NSAP
P_ADDRESS_PLAN_SMTP	8~~9~~	SMTP
P_ADDRESS_PLAN_X400	10~~1~~	X.400

The changes should also be reflected in the IDL (chapter 9):

```
// Defines the address plan (or numbering plan) used. It is also used to indicate
                // whether an address is actually defined in a TAddress data element
                enum TpAddressPlan
                {
                        P_ADDRESS_PLAN_NOT_PRESENT,  // No Address Present
                        P_ADDRESS_PLAN_UNDEFINED,    // Undefined
                        P_ADDRESS_PLAN_IP,           // IP
                        P_ADDRESS_PLAN_MULTICAST,    // Multicast
                        P_ADDRESS_PLAN_UNICAST,      // Unicast
                        P_ADDRESS_PLAN_E164,         // E.164
~~                      P_ADDRESS_PLAN_E164_MOBILE,  // E.164 Mobile~~
                        P_ADDRESS_PLAN_AESA,         // AESA
                        P_ADDRESS_PLAN_URL,          // URL
                        P_ADDRESS_PLAN_NSAP,         // NSAP
                        P_ADDRESS_PLAN_SMTP,         // SMTP
                        P_ADDRESS_PLAN_NOT_USED,
                        P_ADDRESS_PLAN_X400          // X.400
                };
```

3GPP Meeting CN5 #4
Retz, 10-11 July 2000

Document N5-000132
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	005	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*	for approval	X	strategic		*(for SMG use only)*
	for information		non-strategic		

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)* (U)SIM [ ] ME [ ] UTRAN / Radio [ ] Core Network [X]

Source: N5 **Date:** 27 July 2000

Subject: Common IDL interfaces for Generic Call Control and Generic User Interaction between 3GPP, ETSI SPAN3 and Parlay

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	X	**Release:**	Phase 2	
A	Corresponds to a correction in an earlier release			Release 96	
B	Addition of feature			Release 97	
C	Functional modification of feature			Release 98	
D	Editorial modification			Release 99	X
				Release 00	

Reason for change:

The Generic Call Control SCF is one of the capability features of 3GPP OSA R99. Because of a great drive for harmonisation between 3GPP, SPAN3 and Parlay, the Generic Call Control API is in principle common between the three. However, at this moment the IDL for Generic Call Control in TS 29.198 is specific to 3GPP release 99, because e.g. the operation for createCall() has been removed as there is no support for service initiated calls in release 99.
When the IDL is different from IDL specified by Parlay or ETSI-SPAN3, this will lead in fact to different APIs for the developer community.

In order to ensure common API's for fixed and wireless access, the OSA API work is done jointly between ETSI SPAN3 and 3GPP CN5. It has been agreed that the joint N5/SPAN3 specification should contain one common IDL. Since the Generic Call Control of R99 is also applicable for fixed access, the common IDL for Generic Call Control and User Interaction should be the complete IDL, including the operations:

- createCall() and setCallLoadControl() on the IpCallControlManager
- callOverloadEncountered() and callOverloadCeased() on the IpAppCallControlManager,
- getMoreDialledDigitsReq() on the IpCall,
- getMoreDialledDigitsRes() and getMoreDialledDigitsErr() on the IpAppCall,
- recordMessageReq() on the IpUICall,
- recordMessageRes() and recordMessageErr() on the IpAppUICall.

This CR proposes that in the TS 29.198 the complete Generic Call Control IDL is included as well with the addition of a remark that operations that are not supported will throw the exception for method not supported (P_METHOD_NOT_SUPPORTED in TpResultInfo).

Clauses affected: 9

Other specs affected:	Other 3G core specifications		→ List of CRs:	
	Other GSM core specifications		→ List of CRs:	
	MS test specifications		→ List of CRs:	
	BSS test specifications		→ List of CRs:	
	O&M specifications		→ List of CRs:	

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

9. IDL Interface Definitions

The OSA API definitions have been divided into several CORBA modules. The common data definitions are placed in the root module while each of the specific service capability feature API definitions are being assigned their own module directly under that root. Each specific SCF functions, like User Status, have their data and interface definitions collocated. This structure has the advantage that explicit scoping is kept to a minimum. The IDLs defined for the specific SCFs assumes that the OSA common definitions (interfaces and data) are provided in the org.threegpp.osa module within a file name called OSA.idl

Module Name	**Description**	**IDL file name**
org.threegpp.osa	Common data/interface definitions	OSA.idl
org.threegpp.osa.mm	Common mobility data definitions (root)	MM.idl
org.threegpp.osa.mm.ul	Network User Location (UL)	MMul.idl
org.threegpp.osa.mm.us	User Status (US)	MMus.idl
org.threegpp.osa.cc	Call Control	CC.idl
org.threegpp.osa.ui	User Interaction	UI.idl
org.threegpp.osa.termcap	Terminal Capabilities	TERMCAP.idl

Some of the interfaces contain more operations than defined in the interface classes of Chapter 6. These operations must return a “Method not supported” exception in case the interface is implemented on a SCS based on this specification.

Below impact on the Call Control and User interaction is shown:

9.3 Call Control

9.3.1 Common Data Types for Call Control

```
// source file: CC.idl
// Generic Call Data description

#ifndef __OSA_CC_DEFINED
#define __OSA_CC_DEFINED

#include <OSA.idl>
#include <UI.idl>

module org
{
 module threegpp
 {
     module osa
     {
       module cc
       {
        /* Defines the mechanism that will be used to alert a called party. */
        typedef TpInt32 TpCallAlertingMechanism;

        /* Defines the bearer service associated with the call. */
        enum TpCallBearerService
        {
                P_CALL_BEARER_SERVICE_UNKNOWN,                                  /* Bearer
       capability information
                                                                 unknown at this time*/
                P_CALL_BEARER_SERVICE_SPEECH,                                  /* Speech*/
                P_CALL_BEARER_SERVICE_DIGITALUNRESTRICTED,      /* Unrestricted digital
       information*/
                P_CALL_BEARER_SERVICE_DIGITALRESTRICTED,           /* Restricted digital
       information*/
                P_CALL_BEARER_SERVICE_AUDIO,                              /* 3.1 kHz audio*/
                P_CALL_BEARER_SERVICE_DIGITALUNRESTRICTEDTONES, /* Unrestricted digital
       information
                                                                 with tones/announcements*/
                P_CALL_BEARER_SERVICE_VIDEO                              /*Video*/
        };



        /*This data defines the bearer capabilities associated with the call. (3G TS 24.002)
       This
            information is network operator specific and may not always be available because
       there
            is no standard protocol to retrieve the information */
        enum TpCallNetworkAccessType
        {
                P_CALL_NETWORK_ACCESS_TYPE_UNKNOWN,         /* Network type information unknown
       at this time */
                P_CALL_NETWORK_ACCESS_TYPE_POT,                    /* POTS */
                P_CALL_NETWORK_ACCESS_TYPE_ISDN,            /* ISDN */
                P_CALL_NETWORK_ACCESS_TYPE_DIALUPINTERNET,  /* Dial-up Internet  */
                P_CALL_NETWORK_ACCESS_TYPE_XDSL,            /* xDSL */
                P_CALL_NETWORK_ACCESS_TYPE_WIRELESS         /* Wireless */
        };

        /* Defines the category of a calling or called party (e.g. call priority, payphone,
             prepaid).*/
        enum TpCallPartyCategory
        {
                P_CALL_PARTY_CATEGORY_UNKNOWN,              /*calling party's category unknown
       at this time*/
                P_CALL_PARTY_CATEGORY_OPERATOR_F,    /* operator, language French*/
                P_CALL_PARTY_CATEGORY_OPERATOR_E,    /* operator, language English*/
                P_CALL_PARTY_CATEGORY_OPERATOR_G,    /* operator, language German*/
                P_CALL_PARTY_CATEGORY_OPERATOR_R,    /* operator, language Russian*/
                P_CALL_PARTY_CATEGORY_OPERATOR_S,    /* operator, language Spanish*/
                P_CALL_PARTY_CATEGORY_ORDINARY_SUB,  /* ordinary calling subscriber*/
                P_CALL_PARTY_CATEGORY_PRIORITY_SUB,  /* calling subscriber with priority*/
                P_CALL_PARTY_CATEGORY_DATA_CALL,     /* data call (voice band data) */
```

```
        P_CALL_PARTY_CATEGORY_TEST_CALL,     /* test call*/
        P_CALL_PARTY_CATEGORY_PAYPHONE              /* payphone*/
 };

 /* This data type defines the tele-service associated with the call. (Q.763: User
Teleservice Information,  Q.931: High Layer Compatitibility Information, and 3G TS
22.003)Defines the tele-service associated with the call (e.g. speech, video, fax, file
transfer, browsing). */
 enum TpCallTeleService
 {
        P_CALL_TELE_SERVICE_UNKNOWN,         /* Teleservice information unknown at this
time*/
        P_CALL_TELE_SERVICE_TELEPHONY,              /* Telephony */
        P_CALL_TELE_SERVICE_FAX_2_3,         /* Facsimile Group 2/3 */
        P_CALL_TELE_SERVICE_FAX_4_I,         /* Facsimile Group 4, Class I */
        P_CALL_TELE_SERVICE_FAX_4_II_III,    /* Facsimile Group 4, Classes II and III
*/
        P_CALL_TELE_SERVICE_VIDEOTEX_SYN,    /* Syntax based Videotex */
        P_CALL_TELE_SERVICE_VIDEOTEX_INT,    /* International Videotex interworking via
gateways or interworking units */
        P_CALL_TELE_SERVICE_TELEX,           /* Telex service*/
        P_CALL_TELE_SERVICE_MHS,             /* Message Handling Systems */
        P_CALL_TELE_SERVICE_OSI,             /* OSI application*/
        P_CALL_TELE_SERVICE_FTAM,            /* FTAM application*/
        P_CALL_TELE_SERVICE_VIDEO,           /* Videotelephony*/
        P_CALL_TELE_SERVICE_VIDEO_CONF,      /* Videoconferencing*/
        P_CALL_TELE_SERVICE_AUDIOGRAPH_CONF,        /* Audiographic conferencing*/
        P_CALL_TELE_SERVICE_MULTIMEDIA,      /* Multimedia services*/
        P_CALL_TELE_SERVICE_CS_INI_H221,     /* Capability set of initial channel of
H.221*/
        P_CALL_TELE_SERVICE_CS_SUB_H221,     /* Capability set of subsequent channel of
H.221*/
        P_CALL_TELE_SERVICE_CS_INI_CALL,     /* Capability set of initial channel
associated with an active 3.1 kHz audio or speech call.*/
        P_CALL_TELE_SERVICE_DATATRAFFIC,     /* Data traffic.*/
        P_CALL_TELE_SERVICE_EMERGENCY_CALLS,        /* Emergency Calls*/
        P_CALL_TELE_SERVICE_SMS_MT_PP,       /* Short message MT/PP*/
        P_CALL_TELE_SERVICE_SMS_MO_PP,       /* Short message MO/PP*/
        P_CALL_TELE_SERVICE_CELL_BROADCAST,         /* Cell Broadcast Service*/
        P_CALL_TELE_SERVICE_ALT_SPEECH_FAX_3,       /* Alternate speech and facsimile
group 3*/
        P_CALL_TELE_SERVICE_AUTOMATIC_FAX_3,        /* Automatic Facsimile group 3*/
        P_CALL_TELE_SERVICE_VOICE_GROUP_CALL,       /* Voice Group Call Service*/
        P_CALL_TELE_SERVICE_VOICE_BROADCAST  /* Voice Broadcast Service*/
 };


 /* Defines a specific call event report type. */
 enum TpCallAppInfoType
 {
        P_CALL_APP_UNDEFINED,               /* Undefined */
        P_CALL_APP_ALERTING_MECHANISM,      /* The alerting mechanism or pattern to use
*/
        P_CALL_APP_NETWORK_ACCESS_TYPE,     /* The network access type (e.g. ISDN) */
        P_CALL_APP_TELE_SERVICE,            /* Indicates the tele-service (e.g. speech)
and related info such as clearing programme */
        P_CALL_APP_BEARER_SERVICE,          /* Indicates the bearer service (e.g.
64kb/s unrestricted data). */
        P_CALL_APP_PARTY_CATEGORY,          /* The category of the calling or called
party */
        P_CALL_APP_PRESENTATION_ADDRESS,    /* The address to be presented to other
call parties */
        P_CALL_APP_GENERIC_INFO,            /* Carries unspecified application-SCF
information */
        P_CALL_APP_ADDITIONAL_ADDRESS       /* Indicates an additional address */
 };

 /* Defines the Tagged Choice of Data Elements that specify call application-related
specific information. */
 union TpCallAppInfo switch(TpCallAppInfoType)
 {
        case P_CALL_APP_TELE_SERVICE:
        TpCallTeleService CallAppTeleService;
        case P_CALL_APP_BEARER_SERVICE:
        TpCallBearerService CallAppBearerService;
        case P_CALL_APP_PARTY_CATEGORY:
        TpCallPartyCategory CallAppPartyCategory;
        case P_CALL_APP_PRESENTATION_ADDRESS:
```

```
        TpAddress CallAppPresentationAddress;
        case P_CALL_APP_GENERIC_INFO:
        TpString CallAppGenericInfo;
        case P_CALL_APP_ADDITIONAL_ADDRESS:
        TpAddress CallAppAdditionalAddress;
        case P_CALL_APP_ALERTING_MECHANISM:
        TpCallAlertingMechanism CallAppAlertingMechanism;
        case P_CALL_APP_NETWORK_ACCESS_TYPE:
        TpCallNetworkAccessType CallAppNetworkAccessType;
};

typedef sequence <TpCallAppInfo> TpCallAppInfoSet;

enum TpCallChargeOrderCategory
{
        P_CALL_CHARGE_PER_TIME, /* Charge per time*/
        P_CALL_CHARGE_NETWORK /* Operator specific charge plan specification, e.g.
charging table name / charging table entry*/
};

/* Defines the Tagged Choice of Data Elements that specify the charge plan for the
call. */

union TpCallChargeOrder switch(TpCallChargeOrderCategory)
{
        case P_CALL_CHARGE_PER_TIME:  TpChargePerTime ChargePerTime;
        case P_CALL_CHARGE_NETWORK:   TpString NetworkCharge;
};



/* Defines the Sequence of Data Elements that specify the charge plan for the call This
data type is identical to a TpString, and defines the call charge plan to be used for
the call. The values of this data type are operator specific. */
struct TpCallChargePlan
{
        TpCallChargeOrder ChargeOrderType;
        TpString Currency;
        TpString AdditionalInfo;
};

const TpInt32 P_EVENT_NAME_UNDEFINED = 0;                    // Undefined
const TpInt32 P_EVENT_GCCS_OFFHOOK_EVENT = 1;                // Offhook event
const TpInt32 P_EVENT_GCCS_ADDRESS_COLLECTED_EVENT = 2;      // Address information
collected
const TpInt32 P_EVENT_GCCS_ADDRESS_ANALYSED_EVENT = 4;       // Address information is
analysed
const TpInt32 P_EVENT_GCCS_CALLED_PARTY_BUSY = 8;            // Called party is busy
const TpInt32 P_EVENT_GCCS_CALLED_PARTY_UNREACHABLE = 16;    // Called party is
unreachable
const TpInt32 P_EVENT_GCCS_NO_ANSWER_FROM_CALLED_PARTY = 32; // No answer from called
party
const TpInt32 P_EVENT_GCCS_ROUTE_SELECT_FAILURE = 64;        // Failure in routing the
call
const TpInt32 P_EVENT_GCCS_ANSWER_FROM_CALL_PARTY = 128;     // Party answered call

typedef TpInt32 TpCallEventName; /*Defines the names of event being notified. */

enum TpCallNotificationType
{
        P_ORIGINATING, // The notification is related to the originating user in the
call.
        P_TERMINATING  // The notification is related to the terminating user in the
call.
};

struct TpCallEventCriteria
{
        TpAddressRange DestinationAddress;          /*Destination address or address
range*/
        TpAddressRange OriginationAddress;          /*Origination address or address
range */
        TpCallEventName CallEventName;                /*Name of the event(s) */
        TpCallNotificationType CallNotificationType;  /*Indicates whether the criteria
are related to the originating or terminating user in the call */
};
```

```
 /* Defines a sequence of data elements that specify a requested call event notification
criteria with the associated assignmentID  */
 struct TpCallEventCriteriaResult
 {
        TpCallEventCriteria EventCriteria;
        TpInt32 AssignmentID;
 };

 /* Defines a set of TpCallEventCriteriaResult */
 typedef sequence <TpCallEventCriteriaResult> TpCallEventCriteriaResultSet;

 //Defines the type of notification.
 //Indicates whether it is related to the originating of the terminating user in the
call.
 struct TpCallEventInfo
 {
        TpAddress DestinationAddress;
        TpAddress OriginatingAddress;
        TpAddress OriginalDestinationAddress;
        TpAddress RedirectingAddress;
        TpCallAppInfoSet CallAppInfo;
        TpCallEventName CallEventName;
        TpCallNotificationType CallNotificationType;
 };

 /* Defines the Sequence of Data Elements that specify the cause of the release of a
call.*/
 struct TpCallReleaseCause {
 TpInt32 Value;
 TpInt32 Location;
 };

 /* Defines the Sequence of Data Elements that specify the reason for the call ending.*/
 struct TpCallEndedReport
 {
        TpSessionID CallLegSessionID;
        TpCallReleaseCause Cause;
 };

 /* Defines a specific call error. */
 enum TpCallErrorType
 {
        P_CALL_ERROR_UNDEFINED,         /* Undefined */
        P_CALL_ERROR_INVALID_ADDRESS,   /* The operation failed because an invalid
address was given */
        P_CALL_ERROR_INVALID_STATE     /* The call was not in a valid state for the
requested operation */
 };

 /* Defines the Tagged Choice of Data Elements that specify additional call error and
call error specific information. This is also used to specify call leg errors and call
information errors. */
 union TpCallAdditionalErrorInfo switch(TpCallErrorType)
 {
        case P_CALL_ERROR_INVALID_ADDRESS: TpAddressError CallErrorInvalidAddress;
        default: short Dummy;   // allows initialisation of the union in the default
case
 };

 /* Defines the Sequence of Data Elements that specify the additional information
relating to an undefined call error. */
 struct TpCallError
 {
        TpCallAdditionalErrorInfo AdditionalErrorInfo;
        TpCallErrorType ErrorType;
        TpDateAndTime ErrorTime;
 };

 /* Defines the cause of the call fault detected. */
 enum TpCallFault
 {
        P_CALL_FAULT_UNDEFINED,     /* Undefined */

        P_CALL_TIMEOUT_ON_RELEASE,  /* Final report has been sent to the application,
but the application did not explicitly release or deassign the call object, within a
specified time. */
```

```
        P_CALL_TIMEOUT_ON_INTERRUPT /* Application did not instruct the gateway how to
handle the call within a specified time, after the gateway reported an event that was
requested by the application in interrupt mode.*/
 };

 /* Defines the type of call information requested and reported */
 const TpInt32 P_CALL_INFO_UNDEFINED = 0;         /* Undefined */
 const TpInt32 P_CALL_INFO_TIMES = 1;             /* Relevant call times */
 const TpInt32 P_CALL_INFO_RELEASE_CAUSE = 2;     /* Call release cause. */
 const TpInt32 P_CALL_INFO_INTERMEDIATE = 4;      /* Send only intermediate reports
(i.e., when a party leaves the call). */

 typedef TpInt32 TpCallInfoType;

 /* Defines the Sequence of Data Elements that specify the call information requested.
Information that was not requested may be undefined or not present. */
 struct TpCallInfoReport
 {
        TpCallInfoType CallInfoType;
        TpDateAndTime CallInitiationStartTime;
        TpDateAndTime CallConnectedToResourceTime;
        TpDateAndTime CallConnectedToDestinationTime;
        TpDateAndTime CallEndTime;
        TpCallReleaseCause Cause;
 };

 /* Defines the mode that the call will monitor for events, or the mode that the call is
in following a detected event. */
 enum TpCallMonitorMode
 {
        P_CALL_MONITOR_MODE_INTERRUPT,     /* The call event is intercepted by the call
control SCF and call processing is interrupted. The application is notified of the
event and call processing resumes following an appropriate API call or network event
(such as a call release) */
        P_CALL_MONITOR_MODE_NOTIFY,        /* The call event is detected by the call
control SCF but not intercepted. The application is notified of the event and call
processing continues */
        P_CALL_MONITOR_MODE_DO_NOT_MONITOR /* Do not monitor for the event */
 };

 /* Defines the type of call overload that has been detected (and possibly acted upon)
by the network. */
 enum TpCallOverloadType
 {
        P_CALL_OVERLOAD_TYPE_UNDEFINED,   /* Infinite interval (do not admit any calls)
*/
        P_CALL_OVERLOAD_TYPE_NEW_CALLS,   /* New calls to the application are causing
overload (i.e. inbound overload) */
        P_CALL_OVERLOAD_TYPE_ROUTED_CALLS /* Calls being routed to destination or
origination addresses by the application are causing overload (i.e. outbound overload)
*/
 };

 /* Defines a specific call event report type. */
 enum TpCallReportType
 {
        P_CALL_REPORT_UNDEFINED,       /* Undefined */
        P_CALL_REPORT_PROGRESS,        /* Call routing progress event */
        P_CALL_REPORT_ALERTING,        /* Call alerting at  address */
        P_CALL_REPORT_ANSWER,          /* Call answered at address */
        P_CALL_REPORT_BUSY,            /* Called address refused call due to busy */
        P_CALL_REPORT_NO_ANSWER,       /* No answer at called address */
        P_CALL_REPORT_DISCONNECT,      /* Call disconnect requested by address */
        P_CALL_REPORT_REDIRECTED,
        P_CALL_REPORT_SERVICE_CODE,
        P_CALL_REPORT_ROUTING_FAILURE

 };

 /* Defines the Tagged Choice of Data Elements that specify additional call report
information. */
 union TpCallAdditionalReportInfo switch(TpCallReportType)
 {
        case P_CALL_REPORT_BUSY: TpCallReleaseCause RefuseBusy;
        case P_CALL_REPORT_DISCONNECT: TpCallReleaseCause CallDisconnect;
        case P_CALL_REPORT_REDIRECTED: TpAddress ForwardAddress;
        case P_CALL_REPORT_SERVICE_CODE: TpCallReleaseCause ServiceCode;
        case P_CALL_REPORT_ROUTING_FAILURE: TpCallReleaseCause RoutingFailure;
```

```
        default:  short Dummy;   // allows initialisation of the union in the default
case
 };

 struct TpCallReport
 {
        TpCallMonitorMode MonitorMode;
        TpDateAndTime CallEventTime;
        TpCallReportType CallReportType;
        TpCallAdditionalReportInfo AdditionalReportInfo;
 };

 /* Defines the different types of service codes that can be received during the call.*/
 enum TpCallServiceCodeType
 {
        P_CALL_SERVICE_CODE_UNDEFINED,       /* The type of service code is unknown.
The corresponding string is operator specific.*/
        P_CALL_SERVICE_CODE_DIGITS, /* The user entered a digit sequence during the
call. The corresponding string is an ascii representation of the received digits. */
        P_CALL_SERVICE_CODE_FACILITY, /*  A facility information element is received.
The corresponding string contains the facility information element as defined in ITU
Q.932*/
        P_CALL_SERVICE_CODE_U2U, /* A user-to-user message was received. The associated
string contains the content of the user-to-user information element. */
        P_CALL_SERVICE_CODE_HOOKFLASH,       /* The user performed a hookflash,
optionally followed by some digits. The corresponding string is an ascii representation
of the entered digits. */
        P_CALL_SERVICE_CODE_RECALL /* The user pressed the register recall button,
optionally followed by some digits. The corresponding string is an ascii representation
of the entered digits. */
        };

 /* Defines the Sequence of Data Elements that specify the service code and type of
service code received during a call. The service code type defines how the value string
should be interpreted. Defines the service code received during a call. For example,
this may be a digit sequence, user-user information, recall, flash-hook or ISDN
Facility Information Element. This data type is identical to a TpString. The coding of
this data type is operator specific. */
 struct TpCallServiceCode
 {
        TpCallServiceCodeType CallServiceCodeType;
        TpString ServiceCodeValue;
 };


 /* Defines the Tagged Choice of Data Elements that specify specific criteria. */
 union TpCallAdditionalReportCriteria switch(TpCallReportType)
 {
        case P_CALL_REPORT_NO_ANSWER: TpDuration NoAnswerDuration;
        case P_CALL_REPORT_SERVICE_CODE: TpCallServiceCode ServiceCode;
        default: short Dummy;   // allows initialisation of the union in the default
case
 };

 /* Defines the Sequence of Data Elements that specify the criteria relating to call
report requests. */
 struct TpCallReportRequest
 {
        TpCallMonitorMode MonitorMode;
        TpCallReportType CallReportType;
        TpCallAdditionalReportCriteria AdditionalReportCriteria;
 };

 /* Defines a Numbered Set of Data Elements of TpCallReportRequest. */
 typedef sequence <TpCallReportRequest> TpCallReportRequestSet;

 const TpInt32 P_CALL_SUPERVISE_TIMEOUT = 1;      /* The call supervision timer has
expired. */
 const TpInt32 P_CALL_SUPERVISE_CALL_ENDED = 2;   /* The call has ended, either due to
timer expiry or calling or called party release. In case the called party disconnects
but a follow-on call can still be made also this indication is used.*/
 const TpInt32 P_CALL_SUPERVISE_TONE_APPLIED = 4; /* A warning  tone has been applied.
*/
 const TpInt32 P_CALL_SUPERVISE_UI_FINISHED = 8;  /* The user interaction has finished
*/

 /* Defines the responses from the call control SCF for calls that are supervised:*/
```

```
typedef TpInt32 TpCallSuperviseReport;

const TpInt32 P_CALL_SUPERVISE_RELEASE = 1;      /* Release the call when the call
supervision timer expires. */
const TpInt32 P_CALL_SUPERVISE_RESPOND = 2;      /* Notify the application when the
call supervision timer expires. */
const TpInt32 P_CALL_SUPERVISE_APPLY_TONE = 4;   /* Send a warning tone to the
controlling party when the call supervision timer expires. If call release is
requested, then the call will be released following the tone after an administered time
period */


/* Defines the following treatment of the call by the call control SCF when the call
supervision timer expires.*/
typedef TpInt32 TpCallSuperviseTreatment;


/* Define the possible Exceptions. */
const TpInt32 P_GCCS_SERVICE_INFORMATION_MISSING = 256;
const TpInt32 P_GCCS_SERVICE_FAULT_ENCOUNTERED = 257;
const TpInt32 P_GCCS_UNEXPECTED_SEQUENCE = 258;
const TpInt32 P_GCCS_INVALID_ADDDRESS = 259;
const TpInt32 P_GCCS_INVALID_CRITERIA = 261;
const TpInt32 P_GCCS_INVALID_NETWORK_STATE = 262;


exception TpGCCSException
{
        TpInt32 exceptionType;
};

/* The next data type is not used for an SCF implementation based
   on this specification: */
typedef TpInt32 TpCallLoadControlIntervalRate;

/* The next data type is not used for an SCF implementation based
   on this specification: */
const TpInt32 P_CALL_LOAD_CONTROL_ADMIT_NO_CALLS = 0;


/* The next data type is not used for an SCF implementation based
   on this specification: */
enum TpCallLoadControlMechanismType {
  P_CALL_LOAD_CONTROL_PER_INTERVAL
};


/* The next data type is not used for an SCF implementation based
   on this specification: */
union TpCallLoadControlMechanism switch(TpCallLoadControlMechanismType) {
  case P_CALL_LOAD_CONTROL_PER_INTERVAL:
    TpCallLoadControlIntervalRate CallLoadControlPerInterval;
};


/* The next data type is not used for an SCF implementation based
   on this specification: */
enum TpCallTreatmentType {
  P_CALL_TREATMENT_DEFAULT,
  P_CALL_TREATMENT_RELEASE,
  P_CALL_TREATMENT_SIAR
};


/* The next data type is not used for an SCF implementation based
   on this specification: */
union TpCallAdditionalTreatmentInfo switch(TpCallTreatmentType) {
  case P_CALL_TREATMENT_SIAR: ui::TpUIInfo InformationToSend;
  default: short Dummy;
};


/* The next data type is not used for an SCF implementation based
   on this specification: */
struct TpCallTreatment {
  TpCallTreatmentType CallTreatmentType;
  TpCallReleaseCause ReleaseCause;
  TpCallAdditionalTreatmentInfo AdditionalTreatmentInfo;
```

```
      };

       }; // end module cc
      }; // end module osa
    }; // end module threegpp
 }; // end module org

#endif

// END file CC.idl
```

9.3.2 Generic Call Control IDL

```
// source file: GCC.idl
// GenericCall Interface description

#ifndef __OSA_CC_GCC_DEFINED
#define __OSA_CC_GCC_DEFINED

#include <CC.idl>

module org {
 module threegpp {
  module osa {
   module cc {
    module gcc {


  interface IpAppCallControlManager; // forward definition
  interface IpAppCall;               // forward definition
  interface IpCall;                  // forward definition

/* Sequence of Data Elements that unambiguously specify the Generic Call object */
  struct TpCallIdentifier {
    IpCall CallReference;
    TpSessionID CallSessionID;
  };

  /* This interface is the SCF manager' interface for Generic Call Control. */
  interface IpCallControlManager : IpService {
    /* This method is used to enable call notifications. */
    void enableCallNotification (
      in IpAppCallControlManager appInterface,
      in TpCallEventCriteria eventCriteria,
      out TpAssignmentID assignmentID
      )
    raises (TpGCCSException, TpGeneralException);

    /* This method is used by the application to disable call notifications.*/
    void disableCallNotification (
      in TpAssignmentID assignmentID
      )
    raises (TpGCCSException, TpGeneralException);

    void changeCallNotification (
      in TpAssignmentID assignmentID,
      in TpCallEventCriteria eventCriteria
      )
    raises (TpGCCSException, TpGeneralException);

    void getCriteria (
      out TpCallEventCriteriaResultSet eventCriteria
      )
    raises (TpGCCSException, TpGeneralException);

    /* The next operation is not supported for Release 99 and must
       return the exception "Method not supported" when invoked on a SCF
       implementation based on this specification: */
    void createCall (
       in IpAppCall appCall,
       out TpCallIdentifier callReference
       )
    raises (TpGCCSException,TpGeneralException);

    /* The next operation is not supported for Release 99 and must
       return the exception "Method not supported" when invoked on a SCF
       implementation based on this specification: */
    void setCallLoadControl (
```

```
      in TpDuration duration,
      in TpCallLoadControlMechanism mechanism,
      in TpCallTreatment treatment,
      in TpAddressRange addressRange,
      out TpAssignmentID assignmentID
      )
    raises (TpGCCSException, TpGeneralException);


};

/* This interface provides the means to control a simple call. */
interface IpCall : IpService {
  /* This method requests routing of the call to the destination party.*/
 void routeReq (
    in TpSessionID callSessionID,
    in TpCallReportRequestSet responseRequested,
    in TpAddress targetAddress,
    in TpAddress originatingAddress,
    in TpAddress originalDestinationAddress,
    in TpAddress redirectingAddress,
    in TpCallAppInfoSet appInfo,
    out TpSessionID callLegSessionID
    )
  raises (TpGCCSException, TpGeneralException);

  /* This method requests the release of the call and associated objects.*/
  void release (
    in TpSessionID callSessionID,
    in TpCallReleaseCause cause
    )
  raises (TpGCCSException, TpGeneralException);

  /* This method requests that the relationship between the application and
     the call and associated objects be de-assigned.  */
  void deassignCall (
    in TpSessionID callSessionID
    )
  raises (TpGCCSException, TpGeneralException);

  /* This method requests information associated with the call.*/
  void getCallInfoReq (
    in TpSessionID callSessionID,
    in TpCallInfoType callInfoRequested
    )
  raises (TpGCCSException, TpGeneralException);

  /* Set an operator specific charge plan for the call.  */
  void setCallChargePlan (
    in TpSessionID callSessionID,
    in TpCallChargePlan callChargePlan
    )
  raises (TpGCCSException, TpGeneralException);

  /* The application calls this method to supervise a call.  */
  void superviseCallReq (
    in TpSessionID callSessionID,
    in TpDuration time,
    in TpCallSuperviseTreatment treatment

    )
  raises (TpGCCSException, TpGeneralException);

  void setAdviceOfCharge(
    in TpSessionID callSessionID,
    in TpAoCInfo aOCInfo,
    in TpDuration tariffSwitch
    )
  raises (TpGCCSException, TpGeneralException);

  /* The next operation is not supported for Release 99 and must
     return the exception "Method not supported" when invoked on a SCF
     implementation based on this specification: */
  void getMoreDialledDigitsReq (
    in TpSessionID callSessionID,
    in TpInt32 length
    )
  raises (TpGeneralException, TpGCCSException);
```

```
};

/* Sequence of Data Elements that unambiguously specify the Generic Call object */
struct TpCallIdentifier {
  IpCall CallReference;
  TpSessionID CallSessionID;
};

/* The generic call control manager application interface provides the
   application call control management functions to the generic call control
   SCF. */
interface IpAppCallControlManager : IpOsa {
  void callAborted (
    in TpSessionID callReference
    )
  raises (TpGCCSException, TpGeneralException);

  /* This method notifies the application of the arrival of a call-related event. */
  void callEventNotify (
    in TpCallIdentifier callReference,
    in TpCallEventInfo eventInfo,
    in TpAssignmentID assignmentID,
    out IpAppCall appInterface
    )
  raises (TpGCCSException, TpGeneralException);

   /* This method indicates to the application that all event notifications
      have been terminated .*/
  void callNotificationInterrupted ()
    raises (TpGCCSException, TpGeneralException);

  void callNotificationContinued ()
    raises (TpGCCSException, TpGeneralException);

  /* The next operation is not supported for Release 99 and must
     return the exception "Method not supported" when invoked on a SCF
     implementation based on this specification: */
  void callOverloadEncountered (
    in TpAssignmentID assignmentID
    )
  raises (TpGeneralException,TpGCCSException);

  /* The next operation is not supported for Release 99 and must
     return the exception "Method not supported" when invoked on a SCF
     implementation based on this specification: */
  void callOverloadCeased (
    in TpAssignmentID assignmentID
    )
  raises (TpGeneralException,TpGCCSException);
};

/* The application side of the simple call interface is used to handle call
   request responses and state reports. */
interface IpAppCall : IpOsa {
  /* This method indicates that the request to route the call to the
     destination was successful.*/
  void routeRes (
    in TpSessionID callSessionID,
    in TpCallReport eventReport,
    in TpSessionID callLegSessionID
    )
  raises (TpGCCSException, TpGeneralException);

  /* This method indicates that the request to route the call to the
     destination party was unsuccessful. */
  void routeErr (
    in TpSessionID callSessionID,
    in TpCallError errorIndication,
    in TpSessionID callLegSessionID
    )
  raises (TpGCCSException, TpGeneralException);

  /* This method reports all necessary information requested by the
     application, for example to calculate charging.*/
  void getCallInfoRes (
```

```
      in TpSessionID callSessionID,
      in TpCallInfoReport callInfoReport
      )
    raises (TpGCCSException, TpGeneralException);

    /* This asynchronous method reports that the original request was erroneous,
       or resulted in an error condition.*/
    void getCallInfoErr (
      in TpSessionID callSessionID,
      in TpCallError errorIndication
      )
    raises (TpGCCSException, TpGeneralException);

    /* This asynchronous method reports a call supervision event to the application.*/
    void superviseCallRes (
      in TpSessionID callSessionID,
      in TpCallSuperviseReport report,
      in TpDuration usedTime
)
    raises (TpGCCSException, TpGeneralException);

    /* This asynchronous method reports a call supervision error to the application.*/
    void superviseCallErr (
      in TpSessionID callSessionID,
      in TpCallError errorIndication
      )
    raises (TpGCCSException, TpGeneralException);

    /* This method indicates to the application that a fault in the network has
       been detected.*/
    void callFaultDetected (
      in TpSessionID callSessionID,
      in TpCallFault fault
      )
    raises (TpGCCSException, TpGeneralException);

void callEnded (
      in TpSessionID callSessionID,
      in TpCallEndedReport report
      )
    raises (TpGCCSException, TpGeneralException);

    /* The next operation is not supported for Release 99 and must
       return the exception "Method not supported" when invoked on a SCF
       implementation based on this specification: */
    void getMoreDialledDigitsRes (
      in TpSessionID callSessionID,
      in TpString digits
      )
    raises (TpGeneralException,TpGCCSException);

    /* The next operation is not supported for Release 99 and must
       return the exception "Method not supported" when invoked on a SCF
       implementation based on this specification: */
    void getMoreDialledDigitsErr (
      in TpSessionID callSessionID,
      in TpCallError errorIndication
      )
    raises (TpGeneralException,TpGCCSException);
  };

    }; // end module gcc
   }; // end module cc
  }; // end module osa
 }; // end module threegpp
}; // end module org

#endif

// END file GCC.idl
```

9.3.3 Enhanced Call Control IDL

The IDL in this section is only supplied in order to make the User Interaction IDL compile.

With the createUICall() method on the UIManager object it is possible to associate the UICall object to a Call object as well as a CallLeg object. The CallLeg object is not used in this specification. However the IDL for this interface has to be supplied otherwise the User Interaction IDL will not compile.

```
// source file: ECC.idl

#ifndef __OSA_CC_ECC_DEFINED
#define __OSA_CC_ECC_DEFINED

#include <GCC.idl>

module org {
 module threegpp {
  module osa {
   module cc {
    module ecc {



   typedef TpInt32 TpMediaType;

   const TpInt32 P_AUDIO = 1;
   const TpInt32 P_VIDEO = 2;
   const TpInt32 P_DATA = 4;

   typedef TpInt32 TpAudioCapabilitiesType;

   typedef TpInt32 TpVideoCapabilitiesType;

   typedef TpInt32 TpDataCapabilities;

   union TpChannelDataTypeRequest switch(TpMediaType) {
        case P_DATA: TpDataCapabilities Data;
        case P_VIDEO: TpVideoCapabilitiesType Video;
        case P_AUDIO: TpAudioCapabilitiesType Audio;
   };

   typedef TpChannelDataTypeRequest TpChannelDataType;

   enum TpChannelDirection {
        P_INCOMING,
        P_OUTGOING
   };


   struct TpChannelRequest {
        TpChannelDataTypeRequest DataTypeRequest;
        TpChannelDirection Direction;
   };

   typedef sequence <TpChannelRequest> TpChannelRequestSet;

enum TpCallLegType {
   P_CALL_LEG_TYPE_UNDEFINED,
   P_CALL_LEG_TYPE_CONTROLLING,
   P_CALL_LEG_TYPE_PASSIVE
 };

enum TpCallLegInfoType {
   P_CALL_LEG_INFO_UNDEFINED,
   P_CALL_LEG_INFO_ADDRESS,
   P_CALL_LEG_INFO_RELEASE_CAUSE,
   P_CALL_LEG_INFO_APPINFO,
   P_CALL_LEG_INFO_TIMES
 };

 interface IpMMChannel : IpService {
   void close (
        in TpSessionID channelSessionID
   )
 raises (TpGeneralException,TpGCCSException);

 };

   struct TpChannel {
        TpChannelDirection Direction;
```

```
        IpMMChannel Channel;
        TpChannelDataType DataType;
        TpInt32 ChannelNumber;
    };

    typedef sequence <TpChannel> TpChannelSet;


interface IpCallLeg : IpService {
  void routeCallLegToOrigination (
      in TpSessionID callLegSessionID,
      in TpAddress targetAddress,
      in TpAddress originatingAddress,
      in TpAddress originalCalledAddress,
      in TpAddress redirectingAddress,
      in TpCallAppInfoSet appInfo
      )
  raises (TpGeneralException,TpGCCSException);

  void routeCallLegToDestination (
      in TpSessionID callLegSessionID,
      in TpAddress targetAddress,
      in TpAddress originatingAddress,
      in TpAddress originalCalledAddress,
      in TpAddress redirectingAddress,
      in TpCallAppInfoSet appInfo
      )
  raises (TpGeneralException,TpGCCSException);

  void eventReportReq (
      in TpSessionID callLegSessionID,
      in TpCallReportRequestSet eventReportsRequested
      )
  raises (TpGeneralException,TpGCCSException);

  void release (
      in TpSessionID callLegSessionID,
      in TpCallReleaseCause cause
      )
  raises (TpGeneralException,TpGCCSException);

  void getInfoReq (
      in TpSessionID callLegSessionID,
      in TpCallLegInfoType callLegInfoRequested
      )
  raises (TpGeneralException,TpGCCSException);

  void getType (
      in TpSessionID callLegSessionID,
      out TpCallLegType callLegType
      )
  raises (TpGeneralException,TpGCCSException);

  void getCall (
      in TpSessionID callLegSessionID,
      out org::threegpp::osa::cc::gcc::TpCallIdentifier callReference
      )
  raises (TpGeneralException,TpGCCSException);

  void mediaChannelAllow (
      in TpSessionID callLegSessionID,
      in TpSessionIDSet channelList
      )
  raises (TpGeneralException,TpGCCSException);

  void getMediaChannels (
      in TpSessionID callLegSessionID,
      out TpChannelSet channels
      )
  raises (TpGeneralException,TpGCCSException);

  void mediaChannelMonitorReq (
      in TpSessionID callLegSessionID,
      in TpChannelRequestSet channelEventCriteria,
      in TpCallMonitorMode monitorMode
      )
  raises (TpGeneralException,TpGCCSException);
};
```

```
  struct TpCallLegIdentifier {
    TpSessionID CallLegSessionID;
    IpCallLeg CallLegReference;
  };


    }; // end module ecc
   }; // end module cc
  }; // end module osa
 }; // end module threegpp
}; // end module org

#endif

// END file ECC.idl
```

9.4 User Interaction IDL

9.4.1 Common data types for User Interaction

```
// source file: UI.idl
// User Interaction data description

#ifndef __OSA_UI_DEFINED
#define __OSA_UI_DEFINED

#include <OSA.idl>

module org {
 module threegpp {
  module osa {
   module ui {

 /* Defines the additional properties for the collection of information */
 struct TpUICollectCriteria {
  TpInt32 MinLength;            /* minimum number of characters to collect */
  TpInt32 MaxLength;           /* maxmum number of characters to collect */
  TpString EndSequence;        /* character(s) which terminate an input of variable length.
  */
  TpDuration StartTimeout;      /* defines a duration (in seconds) */
  TpDuration InterCharTimeout;  /* value for the inter-character time-out timer. */
 };

 /* Defines the UI call error codes. */
 enum TpUIError {
  P_UI_ERROR_UNDEFINED,               /* Undefined error */
  P_UI_ERROR_ILLEGAL_ID,              /* The information id specified is invalid */
  P_UI_ERROR_ID_NOT_FOUND,            /* Information id is not known to the the User
  Interaction SCFs */
  P_UI_ERROR_RESOURCE_UNAVAILABLE,    /* Resources used by the User Interaction SCFs are
  unavailable. */
  P_UI_ERROR_ILLEGAL_RANGE,           /* The values for manimum and maximum collection length
  are out of range */
  P_UI_ERROR_IMPROPER_CALLER_RESPONSE, /* Improper user response */
  P_UI_ERROR_ABANDON,                 /* Specified leg is disconnected before the send
  information completed */
  P_UI_ERROR_NO_OPERATION_ACTIVE,     /* No active user interaction for the specified leg. */
  P_UI_ERROR_NO_SPACE_AVAILABLE       /* There is no more storage capacity to record the
  message.*/
 };

/* Defines the type of the dataString parameter in the method userInteractionEventNotify */
 enum TpUIEventInfoDataType {
  P_UI_EVENT_DATA_TYPE_UNDEFINED,            /* Undefined */
  P_UI_EVENT_DATA_TYPE_UNSPECIFIED,          /* Unspecified data */
  P_UI_EVENT_DATA_TYPE_TEXT,                /* Text */
  P_UI_EVENT_DATA_TYPE_USSD_DATA             /* USSD data starting with coding scheme */
 };


/* Defines the Sequence of Data Elements that specify the additional criteria for receiving a
  UI notification */
```

```
struct TpUIEventCriteria {
 TpAddressRange OriginatingAddress;  /* Address of the end-user for which notification shall
 be handled */
 TpAddressRange DestinationAddress;
 TpString ServiceCode;  /* 2 digit code indicating the UI to be triggered. */
};

/* Defines the Sequence of Data Elements that specify a UI notification */
struct TpUIEventInfo {
 TpAddress OriginatingAddress; /* Address of the end-user for which notification shall be
 handled */
 TpAddress DestinationAddress;
TpString ServiceCode; /* 2 digit code indicating the UI to be triggered.  */
TpUIEventInfoDataType DataTypeIndication;
TpString DataString;
};

/* Defines the cause of the UI fault detected. */
enum TpUIFault {
 P_UI_FAULT_UNDEFINED,       /* Undefined */
 P_UI_CALL_DEASSIGNED        /* The related Call object has been deassigned. */
};

/* Defines the type of  information send to the end-user */
enum TpUIInfoType {
 P_UI_INFO_ID,               /* The information consists of an ID */
 P_UI_INFO_DATA,     /* The information consists of a data string */
 P_UI_INFO_ADDRESS   /* The information consists of a URL. */
};

/* Defines the Tagged Choice of Data Elements that specifies the information to be send to a
 end-user. */
union TpUIInfo switch(TpUIInfoType) {
 case P_UI_INFO_ID: TpInt32 InfoID;         /*Defines the ID of the user information script
 or stream to send to an end-user.*/
 case P_UI_INFO_DATA: TpString InfoData;    /*Defines the data to be sent to an end-user's
 terminal.*/
 case P_UI_INFO_ADDRESS: TpURL InfoAddress; /*Defines the URL of the text or stream to be
 sent to an end-user's terminal*/
};

/* Defines the criteria for recording of messages */
struct TpUIMessageCriteria {
TpString EndSequence;      /* Defines the character(s) which terminate an input of variable
 length. */
TpDuration MaxMessageTime; /* Specifies the maximum allowed duration in seconds. */
TpInt32 MaxMessageSize;    /* Specifies the maximum allowed size in bytes of the message. */
};

/* Defines the UI call reports if a response was requested. */
enum TpUIReport {
 P_UI_REPORT_UNDEFINED,             /* Undefined report */
 P_UI_REPORT_ANNOUNCEMENT_ENDED,    /* Confirmation that the announcement has ended */
 P_UI_REPORT_LEGAL_INPUT,           /* Information collected., meeting the specified
 criteria. */
 P_UI_REPORT_NO_INPUT,              /* User immediately entered the delimiter character. No
 valid information has been returned */
 P_UI_REPORT_TIMEOUT,              /* User did not input any response before the input
 timeout expired */
 P_UI_REPORT_MESSAGE_STORED,        /* A message has been stored successfully */
 P_UI_REPORT_MESSAGE_NOT_STORED     /* The message has not been stored successfully */
};

/* Defines the situations for which a response is expected following the user interaction. */
const TpInt32 P_UI_RESPONSE_REQUIRED = 1; /* A response must be sent when the request  has
 completed. */
const TpInt32 P_UI_LAST_ANNOUNCEMENT_IN_A_ROW = 2; /* This is the final announcement within a
 sequence. */
const TpInt32 P_UI_FINAL_REQUEST = 4; /* This is the final request. */

typedef TpInt32 TpUIResponseRequest; /* Defines the situations for which a response is
 expected following the user interaction. */


/* Defines the type of the variable parts in the information to send to the user. */
enum TpUIVariablePartType {
 P_UI_VARIABLE_PART_INT,     /* Variable part is of type integer */
 P_UI_VARIABLE_PART_ADDRESS, /* Variable part is of type address */
```

```
  P_UI_VARIABLE_PART_TIME,    /* Variable part is of type time */
  P_UI_VARIABLE_PART_DATE,    /* Variable part is of type date */
  P_UI_VARIABLE_PART_PRICE    /* Variable part is of type price */
 };


 /* Defines the Tagged Choice of Data Elements that specify the variable parts in the
  information to send to the user. */
 union TpUIVariableInfo switch(TpUIVariablePartType) {
  case P_UI_VARIABLE_PART_INT: TpInt32 VariablePartInteger;
  case P_UI_VARIABLE_PART_ADDRESS: TpString VariablePartAddress;
  case P_UI_VARIABLE_PART_TIME: TpTime VariablePartTime;
  case P_UI_VARIABLE_PART_DATE: TpDate VariablePartDate;
  case P_UI_VARIABLE_PART_PRICE: TpPrice VariablePartPrice;
 };

/* Defines a Numbered Set of Data Elements of TpUIVariableInfo. */
  typedef sequence <TpUIVariableInfo> TpUIVariableInfoSet;

 /* Define the possible Exceptions. */
 exception TpGUISException {
  TpInt32 exceptionType;
 };

 const TpInt32 P_GUIS_INVALID_CRITERIA = 768;            /* Invalid criteria specified */
 const TpInt32 P_GUIS_ILLEGAL_ID = 769;                 /* Information id specified is invalid
  */
 const TpInt32 P_GUIS_ID_NOT_FOUND = 770;               /* Information id is not known to the
  User Interaction Service */
 const TpInt32 P_GUIS_ILLEGAL_RANGE = 771;              /* The values for minimum and maximum
  collection length are out of range */
 const TpInt32 P_GUIS_INVALID_COLLECTION_CRITERIA = 772; /* Invalid collection criteria
  specified */
 const TpInt32 P_GUIS_INVALID_NETWORK_STATE = 774;       /* Although the sequence of method
  calls is allowed by the gateway, the underlying protocol can not support it. */
 const TpInt32 P_GUIS_UNEXPECTED_SEQUENCE = 775;        /* Although the sequence of method
  calls is allowed by the gateway, the underlying protocol can not support it. */

   }; // end module ui
  }; // end module osa
 }; // end module threegpp
}; // end module org

#endif

// END file UI.idl
```

9.4.2 Generic User Interaction IDL

```
// source file: GUI.idl
// GUIS Interface description

#ifndef __OSA_UI_GUI_DEFINED
#define __OSA_UI_GUI_DEFINED

#include <UI.idl>
#include <ECC.idl>

module org {
 module threegpp {
  module osa {
   module ui {
    module gui {

  interface IpAppUIManager; // forward definition;
  interface IpAppUI;        // forward definition;
  interface IpAppUICall;    // forward definition;

  /* The Generic User Interaction SCF Interface provides functions to send
     information to, or gather information from the user. */
  interface IpUI : IpService {
    /* This method plays an announcement or sends other information to the user.*/
    void sendInfoReq (
      in TpSessionID userInteractionSessionID,
      in TpUIInfo info,
      in TpUIVariableInfoSet variableInfo,
      in TpInt32 repeatIndicator,
```

```
    in TpUIResponseRequest responseRequested,
    out TpAssignmentID assignmentID
  )
  raises (TpGUISException, TpGeneralException);

  /* This method plays an announcement or sends other information to the user
     and collects some information from the user.  */
  void sendInfoAndCollectReq (
    in TpSessionID userInteractionSessionID,
    in TpUIInfo info,
    in TpUIVariableInfoSet variableInfo,
    in TpUICollectCriteria criteria,
    in TpUIResponseRequest responseRequested,
    out TpAssignmentID assignmentID
  )
  raises (TpGUISException, TpGeneralException);

  /* This method requests that the relationship between the application and
     the user interaction object be released. */
  void release (
    in TpSessionID userInteractionSessionID
  )
  raises (TpGUISException, TpGeneralException);

};

/* Defines the Sequence of Data Elements that unambiguously specify the UI object */
struct TpUIIdentifier {
  TpSessionID UserInteractionSessionID;
  IpUI UIRef;
};


/* The Call User Interaction Service Interface provides functions to send
   information to, or gather information from,  the user. */
interface IpUICall : IpUI {
  /* This asynchronous method aborts the specified user interaction operation. */
  void abortActionReq (
    in TpSessionID userInteractionSessionID,
    in TpAssignmentID assignmentID
  )
  raises (TpGUISException, TpGeneralException);


  /* The next operation is not supported for Release 99 and must
     return the exception "Method not supported" when invoked on a SCF
     implementation based on this specification: */
  void recordMessageReq (
    in TpSessionID userInteractionSessionID,
    in TpUIInfo info,
    in TpUIMessageCriteria criteria,
    out TpAssignmentID assignmentID
  )
  raises (TpGUISException, TpGeneralException);

};

/* Defines the Sequence of Data Elements that unambiguously specify the UICall object. */
struct TpUICallIdentifier {
  IpUICall UICallRef;
  TpSessionID UserInteractionSessionID;
};

/* This interface is the 'SCF manager' interface for the Generic User Interaction SCF. */
interface IpUIManager : IpService {
  /* This method is used to create a new user interaction object for non-call related
     purposes */
  void createUI (
    in IpAppUI appUI,
    in TpAddress userAddress,
    out TpUIIdentifier userInteraction
  )
  raises (TpGUISException, TpGeneralException);

  /* This method is used to create a new user interaction object for call related purposes.
     */
  void createUICall (
    in IpAppUICall appUI,
```

```
    in org::threegpp::osa::cc::gcc::TpCallIdentifier callIdentifier,
    in org::threegpp::osa::cc::ecc::TpCallLegIdentifier callLegIdentifier,
    out TpUICallIdentifier userInteraction
  )
  raises (TpGUISException, TpGeneralException);

  /* This method is used to enable the reception of user initiated user interaction. */
  void enableUINotification (
    in IpAppUIManager appInterface,
    in TpUIEventCriteria eventCriteria,
    out TpAssignmentID assignmentID
  )
  raises (TpGUISException, TpGeneralException);

  /* This method is used by the application to disable UI notifications. */
  void disableUINotification (
    in TpAssignmentID assignmentID
  )
  raises (TpGUISException, TpGeneralException);
};


/* The Generic User Interaction SCF manager application interface provides
   the application call management functions to the Generic User Interaction SCF. */
interface IpAppUIManager : IpOsa {
  /* This method indicates to the application that the User Interaction SCF
     instance has terminated or closed abnormally.  */
  void userInteractionAborted (
    in TpUIIdentifier userInteraction
  )
  raises (TpGUISException, TpGeneralException);

  /* This method notifies the application of an user initiated request for user interaction.
     */
  void userInteractionEventNotify (
    in TpUIIdentifier ui,
    in TpUIEventInfo eventInfo,
    in TpAssignmentID assignmentID,
    out IpAppUI appInterface
  )
  raises (TpGUISException, TpGeneralException);

  void userInteractionNotificationInterrupted ()
  raises (TpGUISException, TpGeneralException);

  void userInteractionNotificationContinued ()
  raises (TpGUISException, TpGeneralException);

};

/* The User Interaction Application Interface is used to handle generic user
   interaction request responses and reports. */
interface IpAppUI : IpOsa {
  /* This method informs the application about the start or the completion of a
     sendInfoCallReq(). */
  void sendInfoRes (
    in TpSessionID userInteractionSessionID,
    in TpAssignmentID assignmentID,
    in TpUIReport response
  )
  raises (TpGUISException, TpGeneralException);

  /* This asynchronous method indicates that the request to send information was
     unsuccessful. */
  void sendInfoErr (
    in TpSessionID userInteractionSessionID,
    in TpAssignmentID assignmentID,
    in TpUIError error
  )
  raises (TpGUISException, TpGeneralException);

  /* This asynchronous method returns the information collected to the application. */
  void sendInfoAndCollectRes (
    in TpSessionID userInteractionSessionID,
    in TpAssignmentID assignmentID,
    in TpUIReport response,
    in TpString info
  )
```

```
    raises (TpGUISException, TpGeneralException);

    /* This asynchronous method indicates that the request to send information
       and collect a response was unsuccessful. */
    void sendInfoAndCollectErr (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID,
      in TpUIError error
    )
    raises (TpGUISException, TpGeneralException);

    /* This method indicates to the application that a fault has been detected in the user
       interaction. */
    void userInteractionFaultDetected (
      in TpSessionID userInteractionSessionID,
      in TpUIFault fault
    )
    raises (TpGUISException, TpGeneralException);
  };


  /* The Call User Interaction Application Interface is used to handle call user
     interaction request responses and reports. */
  interface IpAppUICall : IpAppUI {
    /* This method confirms that the request to abort a user interaction operation on a call
       was successful. */
    void abortActionRes (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID
    )
    raises (TpGUISException, TpGeneralException);

    /* This asynchronous method indicates that the request to abort a user interaction
       operation on a call resulted in an error.*/
    void abortActionErr (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID,
      in TpUIError error
    )
    raises (TpGUISException, TpGeneralException);

    /* The next operation is not supported for Release 99 and must
       return the exception "Method not supported" when invoked on a SCF
       implementation based on this specification: */
    void recordMessageRes (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID,
      in TpUIReport response,
      in TpInt32 messageID
    )
    raises (TpGUISException, TpGeneralException);

    /* The next operation is not supported for Release 99 and must
       return the exception "Method not supported" when invoked on a SCF
       implementation based on this specification: */
    void recordMessageErr (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID,
      in TpUIError error
    )
    raises (TpGUISException, TpGeneralException);
  };


   }; // end module gui
  }; // end module ui
 }; // end module osa
}; // end module threegpp
}; // end module org


#endif
```

// END file GUI.idl

3GPP Meeting CN5 #4
Retz, 10-11 July 2000

Document N5-000133
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	006	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*	for approval	X	strategic		*(for SMG use only)*
	for information		non-strategic		

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)* (U)SIM [ ] ME [ ] UTRAN / Radio [ ] Core Network [X]

Source: N5 **Date:** 27 July 2000

Subject: Correction to table with overview of IDL files

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	X	**Release:**	Phase 2	
A	Corresponds to a correction in an earlier release			Release 96	
B	Addition of feature			Release 97	
C	Functional modification of feature			Release 98	
D	Editorial modification			Release 99	X
				Release 00	

Reason for change: The table in chapter 9 showing the different name spaces and IDL files is not in line with the real IDL.

Clauses affected: 9

Other specs affected:

Other 3G core specifications	[ ]	→ List of CRs:
Other GSM core specifications	[ ]	→ List of CRs:
MS test specifications	[ ]	→ List of CRs:
BSS test specifications	[ ]	→ List of CRs:
O&M specifications	[ ]	→ List of CRs:

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

9. IDL Interface Definitions

The OSA API definitions have been divided into several CORBA modules. The common data definitions are placed in the root module while each of the specific service capability feature API definitions are being assigned their own module directly under that root. Each specific SCF functions, like User Status, have their data and interface definitions collocated. This structure has the advantage that explicit scoping is kept to a minimum. The IDLs defined for the specific SCFs assumes that the OSA common definitions (interfaces and data) are provided in the org.threegpp.osa module within a file name called OSA.idl

Module Name	**Description**	**IDL file name**
org.threegpp.osa	Common data/interface definitions	OSA.idl
org.threegpp.osa.fw	common Framework data-types	FW.idl
org.threegpp.osa.fw.discovery	Discovery data-types and interfaces	DISC.idl
org.threegpp.osa.fw.trust_and_security	Trust and Security date-types and interfaces	TandS.idl
org.threegpp.osa.fw.integrity	Integrity management data-types and interfaces	IM.idl
org.threegpp.osa.fw.registration	Registration data-types and interfaces	REG.idl
~~org.threegpp.osa.mm~~	~~Common mobility data definitions (root)~~	~~MM.idl~~
~~org.threegpp.osa.mm.ul~~	~~Network User Location (UL)~~	~~MMul.idl~~
~~org.threegpp.osa.mm.us~~	~~User Status (US)~~	~~MMus.idl~~
org.threegpp.osa.cc	Call Control data-types	CC.idl
org.threegpp.osa.cc.gcc	Generic Call Control interfaces	GCC.idl
org.threegpp.osa.cc.ecc	data-types and interfaces specific for Enhanced Call Control. This is only needed to compile the User Interaction IDL	ECC.idl
org.threegpp.osa.ui	User Interaction data-types	UI.idl
org.threegpp.osa.ui.gui	User Interaction interfaces	GUI.idl
org.threegpp.osa.dsc	Data Session data-types and interfaces	DSC.idl
org.threegpp.osa.mm	Common mobility data definitions (root)	MM.idl
org.threegpp.osa.mm.ul	Network User Location (UL)	MMul.idl
org.threegpp.osa.mm.us	User Status (US)	MMus.idl
org.threegpp.osa.termcap	Terminal Capabilities	TERMCAP.idl

3GPP Meeting CN5 #4
Retz, 10-11 July 2000

Document N5-000134
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	007	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to:	CN#09	for approval	X	strategic		(for SMG use only)
list expected approval meeting # here ↑		for information		non-strategic		

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM		ME		UTRAN / Radio		Core Network	X

Source: N5 **Date:** 27 July 2000

Subject: Reduction in name scoping in IDL for createUICall operation on IpUICall interface

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	X	**Release:**	Phase 2	
A	Corresponds to a correction in an earlier release			Release 96	
B	Addition of feature			Release 97	
C	Functional modification of feature			Release 98	
D	Editorial modification			Release 99	X
				Release 00	

Reason for change: The createUICall operation on the IpUICall interface has parameters of data-types that are defined in other places of the naming tree.
However, the explicit name scoping up to the root of the namespace (org.3gpp.osa.xx.yy) that is currently in the IDL is not necessary, it is sufficient to limit the scoping to xx.yy

Clauses affected: 9.4.2

Other specs affected:

Other 3G core specifications		→ List of CRs:	
Other GSM core specifications		→ List of CRs:	
MS test specifications		→ List of CRs:	
BSS test specifications		→ List of CRs:	
O&M specifications		→ List of CRs:	

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

9.4.2 Generic User Interaction IDL

```
// source file: GUI.idl
// GUIS Interface description

#ifndef __OSA_UI_GUI_DEFINED
#define __OSA_UI_GUI_DEFINED

#include <UI.idl>
#include <ECC.idl>

module org {
 module threegpp {
  module osa {
   module ui {
    module gui {

  interface IpAppUIManager; // forward definition;
  interface IpAppUI;        // forward definition;
  interface IpAppUICall;    // forward definition;

  /* The Generic User Interaction SCF Interface provides functions to send
     information to, or gather information from the user. */
  interface IpUI : IpService {
    /* This method plays an announcement or sends other information to the user.*/
    void sendInfoReq (
      in TpSessionID userInteractionSessionID,
      in TpUIInfo info,
      in TpUIVariableInfoSet variableInfo,
      in TpInt32 repeatIndicator,
      in TpUIResponseRequest responseRequested,
      out TpAssignmentID assignmentID
    )
    raises (TpGUISException, TpGeneralException);

    /* This method plays an announcement or sends other information to the user
       and collects some information from the user.  */
    void sendInfoAndCollectReq (
      in TpSessionID userInteractionSessionID,
      in TpUIInfo info,
      in TpUIVariableInfoSet variableInfo,
      in TpUICollectCriteria criteria,
      in TpUIResponseRequest responseRequested,
      out TpAssignmentID assignmentID
    )
    raises (TpGUISException, TpGeneralException);

    /* This method requests that the relationship between the application and
       the user interaction object be released. */
    void release (
      in TpSessionID userInteractionSessionID
    )
    raises (TpGUISException, TpGeneralException);

  };

  /* Defines the Sequence of Data Elements that unambiguously specify the UI object */
  struct TpUIIdentifier {
    TpSessionID UserInteractionSessionID;
    IpUI UIRef;
  };


  /* The Call User Interaction Service Interface provides functions to send
     information to, or gather information from,  the user. */
  interface IpUICall : IpUI {
    /* This asynchronous method aborts the specified user interaction operation. */
    void abortActionReq (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID
    )
    raises (TpGUISException, TpGeneralException);
  };
```

```
/* Defines the Sequence of Data Elements that unambiguously specify the UICall object. */
struct TpUICallIdentifier {
  IpUICall UICallRef;
  TpSessionID UserInteractionSessionID;
};

/* This interface is the 'SCF manager' interface for the Generic User Interaction SCF. */
interface IpUIManager : IpService {
  /* This method is used to create a new user interaction object for non-call related
     purposes */
  void createUI (
    in IpAppUI appUI,
    in TpAddress userAddress,
    out TpUIIdentifier userInteraction
  )
  raises (TpGUISException, TpGeneralException);

  /* This method is used to create a new user interaction object for call related purposes.
     */
  void createUICall (
    in IpAppUICall appUI,
    in org::threegpp::osa::cc::gcc::TpCallIdentifier callIdentifier,
    in org::threegpp::osa::cc::ecc::TpCallLegIdentifier callLegIdentifier,
    out TpUICallIdentifier userInteraction
  )
  raises (TpGUISException, TpGeneralException);

  /* This method is used to enable the reception of user initiated user interaction. */
  void enableUINotification (
    in IpAppUIManager appInterface,
    in TpUIEventCriteria eventCriteria,
    out TpAssignmentID assignmentID
  )
  raises (TpGUISException, TpGeneralException);

  /* This method is used by the application to disable UI notifications. */
  void disableUINotification (
    in TpAssignmentID assignmentID
  )
  raises (TpGUISException, TpGeneralException);
};


/* The Generic User Interaction SCF manager application interface provides
   the application call management functions to the Generic User Interaction SCF. */
interface IpAppUIManager : IpOsa {
  /* This method indicates to the application that the User Interaction SCF
     instance has terminated or closed abnormally.  */
  void userInteractionAborted (
    in TpUIIdentifier userInteraction
  )
  raises (TpGUISException, TpGeneralException);

  /* This method notifies the application of an user initiated request for user interaction.
     */
  void userInteractionEventNotify (
    in TpUIIdentifier ui,
    in TpUIEventInfo eventInfo,
    in TpAssignmentID assignmentID,
    out IpAppUI appInterface
  )
  raises (TpGUISException, TpGeneralException);

  void userInteractionNotificationInterrupted ()
  raises (TpGUISException, TpGeneralException);

  void userInteractionNotificationContinued ()
  raises (TpGUISException, TpGeneralException);

};

/* The User Interaction Application Interface is used to handle generic user
   interaction request responses and reports. */
interface IpAppUI : IpOsa {
  /* This method informs the application about the start or the completion of a
     sendInfoCallReq(). */
  void sendInfoRes (
```

```
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID,
      in TpUIReport response
    )
    raises (TpGUISException, TpGeneralException);

    /* This asynchronous method indicates that the request to send information was
       unsuccessful. */
    void sendInfoErr (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID,
      in TpUIError error
    )
    raises (TpGUISException, TpGeneralException);

    /* This asynchronous method returns the information collected to the application. */
    void sendInfoAndCollectRes (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID,
      in TpUIReport response,
      in TpString info
    )
    raises (TpGUISException, TpGeneralException);

    /* This asynchronous method indicates that the request to send information
       and collect a response was unsuccessful. */
    void sendInfoAndCollectErr (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID,
      in TpUIError error
    )
    raises (TpGUISException, TpGeneralException);

    /* This method indicates to the application that a fault has been detected in the user
       interaction. */
    void userInteractionFaultDetected (
      in TpSessionID userInteractionSessionID,
      in TpUIFault fault
    )
    raises (TpGUISException, TpGeneralException);
  };


  /* The Call User Interaction Application Interface is used to handle call user
     interaction request responses and reports. */
  interface IpAppUICall : IpAppUI {
    /* This method confirms that the request to abort a user interaction operation on a call
       was successful. */
    void abortActionRes (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID
    )
    raises (TpGUISException, TpGeneralException);

    /* This asynchronous method indicates that the request to abort a user interaction
       operation on a call resulted in an error.*/
    void abortActionErr (
      in TpSessionID userInteractionSessionID,
      in TpAssignmentID assignmentID,
      in TpUIError error
    )
    raises (TpGUISException, TpGeneralException);
  };


    }; // end module gui
   }; // end module ui
  }; // end module osa
 }; // end module threegpp
}; // end module org


#endif
```

// END file GUI.idl

3GPP Meeting CN5 #5
Bristol, 5-7 Sept 2000

Document N5-000171
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	008R2	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*

for approval [X] for information [ ]

strategic [ ] non-strategic [ ] *(for SMG use only)*

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)* (U)SIM [ ] ME [ ] UTRAN / Radio [ ] Core Network [X]

Source: N5 **Date:** 27 August 2000

Subject: Alignment of Framework with Parlay 2.1, improvement on business entity identification

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	[X]	**Release:**	Phase 2	[ ]
A	Corresponds to a correction in an earlier release	[ ]		Release 96	[ ]
B	Addition of feature	[ ]		Release 97	[ ]
C	Functional modification of feature	[ ]		Release 98	[ ]
D	Editorial modification	[ ]		Release 99	[X]
				Release 00	[ ]

Reason for change: The FW of TS 29.198 contains a few differences compared to Parlay 2.1, mainly in the area of different parameter names and data-types.
This CR relates to the introduction of a data-type for indicating the business entity for the purpose of specifying the framework or the type of entity attempting to access the framework.

Clauses affected: 6.2, 8.2, 9.2

Other specs affected:

Other 3G core specifications	[ ]	→ List of CRs:
Other GSM core specifications	[ ]	→ List of CRs:
MS test specifications	[ ]	→ List of CRs:
BSS test specifications	[ ]	→ List of CRs:
O&M specifications	[ ]	→ List of CRs:

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

6.2.3.1 IpInitial

<<Interface>> IpInitial
initiateAuthentication(appDomain: in TpAuthDomain ~~clientAppID: in TpClientAppID~~, authType : in TpAuthType, ~~appAuthInterface: in IpOsaRef, fwAuthInterface :out TpFwAuthRef~~ fwDomain: out TpAuthDomainRef) : TpResult requestAccess(accessType: in TpAccessType, appAccessInterface; in IpOsaRef, fwAccessInterface: out IpOsaRefRef): TpResult

6.2.5 Service Factory

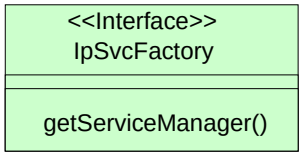


Figure 6-8: Service Factory Class Diagram

<<Interface>> IpSvcFactory
getServiceManager(application : in TpDomain~~ClientApp~~ID, serviceManager : out IpOSA) : TpResult

8.2 Framework Data Definitions

8.2.1.2 TpClientAppIDList

This data type defines a Numbered Set of Data Elements of type TpClientAppID.

TpDomainID

Defines the Tagged Choice of Data Elements that specify either the framework or the type of entity attempting to access the framework.

	Tag Element Type	
	TpDomainIDType	

Tag Element Value	Choice Element Type	Choice Element Name
P_FW	TpFwID	FwID
P_CLIENT_APPLICATION	TpClientAppID	ClientAppID
P_ENT_OP	TpEntOpID	EntOpID
P_REGISTERED_SERVICE	TpServiceID	ServiceID
P_SERVICE_SUPPLIER	TpServiceSupplierID	ServiceSupplierID

TpDomainIDType

Defines either the framework or the type of entity attempting to access the framework

Name	Value	Description
P_FW	0	The framework
P_CLIENT_APPLICATION	1	A client application
P_ENT_OP	2	An enterprise operator
P_REGISTERED_SERVICE	3	A registered service
P_SERVICE_SUPPLIER	4	A service supplier

8.2.1.3 TpEntOpID

This data type is identical to TpString and is defined as a string of characters that identifies an enterprise operator. In conjunction with the application it uniquely identifies the enterprise operator which uses a particular OSA Service Capability Feature.

8.2.1.4 TpEntOpIDList

This data type defines a Numbered Set of Data Elements of type TpEntOpID.

TpFwID

This data type is identical to TpString and identifies the Framework to a client application (or Service Capability Feature)

8.2.1.23 TpServicePropertyList

This data type defines a Numbered Set of Data Elements of type TpServiceProperty.

TpServiceSupplierID

This is an identifier for a service supplier. It is used to identify the supplier to the framework. This data type is identical to TpString.

8.2.2.4 TpAuthCapabilityList

This data type is identical to a TpString. It is a string of multiple TpAuthCapability concatenated using a comma (,)as the separation character.

TpAuthDomain

This is Sequence of Data Elements containing the domain identifier, and a reference to the authentication interface of the domain

Sequence Element Name	Sequence Element Type
DomainID	TpDomainID
AuthInterface	IpOSARef

9.2.1 Common Data Types for the Framework

```
#include <OSA.idl>

module org{

module threegpp{

module osa{

module fw{


        typedef TpString       TpClientAppID;        // Identifies the client appl to the
              framework.

        typedef sequence       <TpClientAppID> TpClientAppIDList;

       /* Defines either the framework or the type of entity attempting to access the
       framework
       The framework
       A client application
       An enterprise operator
       A registered service
       A service supplier */
       enum TpDomainIDType
       {
        P_FW,
        P_CLIENT_APPLICATION,
        P_ENT_OP,
        P_REGISTERED_SERVICE,
        P_SERVICE_SUPPLIER
       };


        typedef TpString TpEntOpID;

        typedef sequence < TpEntOpID >         TpEntOpIDList;


       typedef TpString TpFwID;

       typedef TpString TpServiceSupplierID;

       /* Defines the Tagged Choice of Data Elements that specify either the framework or the
       type of entity
       attempting to access the framework.
       Tag Element Type
       TpDomainIDType */

       struct TpDomainID
       {
        TpFwID FwID;
        TpClientAppID ClientAppID;
        TpEntOpID EntOpID;
        TpServiceID ServiceID;
        TpServiceSupplierID ServiceSupplierID;
       };

        typedef  TpString  TpServiceID;      // A string of characters, generated automatically
                           by the
                                        // Framework and comprising a TpUniqueServiceNumber,
                                        // TpServiceNameString, and a number of relevant
                                        // TpServiceSpecString, concatenated using a forward
                                        // separator (/), that uniquely identifies an
                                     instance of a
                                        // SCF interface.

        typedef sequence <TpServiceID>              TpServiceIDList;

       typedef TpString              TpServiceNameString;         // Uniquely identifies the
   name of an SCF
                                                 // interface. For OSA release 99 the
                                              following
                                                 // values have been defined: NULL (no SCF
                                              name),
```

```
                                                // P_CALL_CONTROL, P_USER_INTERACTION,
                                                // P_USER_LOCATION, P_TERMINAL_CAPABILITIES
                                              and
                                                // P_USER_STATUS.

        typedef TpString             TpServiceSpecString;         // Uniquely identifies the
name of a SCF
                                                // specialisation interface. For OSA release
                                        99
                                                // the following values have been defined:
                                        NULL
                                                // no SCF specialisation) and P_CALL.


        typedef TpString             TpUniqueServiceNumber;       // A string of characters
that represents a
                                                   // unique number.
        enum TpServicePropertyMode {
                NORMAL,                                    // The value of the corresponding
SCF property type may optionally be
                                // provided.
       MANDATORY,                           // The value of the corresponding SCF property
                    type must be provided
                                // at SCF registration.
                READONLY,                                  // The value of the corresponding
SCF property is optional, nut once
                                // given a value it may not be modified.
                MANDATORY_READONLY           // The value of the corresponding SCF property
type must be provided
                                // and may not be modified subsequently.
        };

        typedef TpString             TpServicePropertyTypeName;

        typedef TpString             TpServicePropertyName;

        typedef sequence <TpServicePropertyName>    TpServicePropertyNameList;

        typedef TpString             TpServicePropertyValue;

        typedef sequence <TpServicePropertyValue>   TpServicePropertyValueList;

        struct TpServiceProperty {                          // Describes a SCF property
                TpServicePropertyName        ServicePropertyName;
                TpServicePropertyValueList   ServicePropertyValueList;
                TpServicePropertyMode        ServicePropertyMode;
        };

        typedef sequence <TpServiceProperty>        TpServicePropertyList;

        typedef TpString             TpServiceTypeName;

        typedef sequence <TpServiceTypeName>        TpServiceTypeNameList;

        struct  TpService {                         // Describes a registered SCF.
                TpServiceID          ServiceID;
                TpServicePropertyList ServicePropertyList;
               };

        typedef sequence <TpService>  TpServiceList;

        struct TpServiceDescription {               // Describes the properties of a
registered SCF.
                TpServiceTypeName     ServiceTypeName;
                TpServicePropertyList ServicePropertyList;
        };

        struct TpPropertyStruct {                                  // Describes a SCF property.
                TpServiceTypeName                          ServicePropertyName;
                TpServicePropertyMode        ServicePropertyMode;
                TpServicePropertyTypeName    ServicePropertyTypeName;
        };

        typedef sequence <TpPropertyStruct>  TpPropertyStructList;

        struct TpServiceTypeDescription {                  // Describes a SCF type.
                TpPropertyStructList               PropertyStructList;
                TpServiceTypeNameList        ServiceTypeNameList;
```

```
                TpBoolean                                       EnabledOrDisabled;
        };

};};};};
```

9.2.3 Trust and Security Management IDL

```
#include <fw.idl>


module org{
module threegpp{
module osa{
module fw{
module trust_and_security{

      /*******************************************************************************************
/
      //                                Data definitions
//
      /*******************************************************************************************
/

        typedef TpString              TpAccessType;         // The type of access interface
requested by the client
                                          // application. For OSA release 99 the following
                                       values
                                          // have been defined: NULL (indicates the default
                                       access
                                          // type) and P_ACCESS.

        typedef TpString              TpAuthType;                   // The type of
authentication mechanism requested by the
                                          // client. For OSA release 99 the following values
                                       have
                                          // been defined: NULL (indicates OSA authentication),
                                          // P_AUTHENTICATION (indicates use of the OSA
                                          // authentication interfaces.

        typedef TpString      TpAuthCapability;             // The authentication capabilities
that could be supported
                                          // by the OSA. For OSA release 99 the following
                                       values
                                          // have been defined: NULL (indicates no client
                                          // capabilities, P_DES_56, P_RSA_512 and P_RSA_1024).

        typedef TpString              TpAuthCapabilityList; // A string of multiple
TpAuthCapability
                                                  // concatenated using a commas.

 struct TpAuthDomain
       {
        TpDomainID DomainID;
        IpOSA AuthInterface;
       };

typedef TpString              TpInterfaceName;      // Identifies the names of the framework
SCFs that are be
                                          // supported by the OSA API. For release 99 these are
                                       NULL,
                                          // P_DISCOVERY, P_OAM,
                                       P_TRUST_AND_SECURITY_MANAGEMENT
                                          // P_INTEGRITY_MANAGEMENT.

        struct TpServiceAccessControl {
                TpString              Policy;               // Access control policy
information controlling access to the
                                    // service feature.
                TpString                  TrustLevel;           // The level of trust that the
network operator has assigned to the
                                   // client application.
        };

        typedef TpString              TpServiceToken;       // Uniquely identifies a SCF.

        struct TpSignatureAndServiceMgrRef {
```

```
		TpString		DigitalSignature;		// The digital signature of
the Framework for the service
					// agreement.
		IpOsa		ServiceMgrInterface;
	};

	typedef TpString		TpSigningAlgorithm;		// Identifies the signing
algorithm that must be

						// used. For OSA release 99 the follwing values
have

						// been defined: NULL (indicates no signing
algorithm

						// is required), P_MD5_RSA_512 and
P_MD5_RSA_1024.


	typedef TpString		TpFwID;


	struct TpFwAuth {

		TpFwID  FwID;

		IpOsa   FwAuthInterface;

	};


      /*************************************************************************************
/
      //                                  Interface definitions
//
      /*************************************************************************************
/

      /* The Initial Framework interface is used by the client application to initiate the
mutual
      authentication with the Framework and, when this is finished successfully, to request
access
      to it. */
      interface IpInitial : IpOsa {

         /* This method is invoked by the client application to start the process of mutual
         authentication with the framework, and request the use of a specific authentication
method.
         */
         void initiateAuthentication (
         in TpAuthDomain appDomainTpClientAppID clientAppID,		// Identifies the
client to the framework.
         in TpAuthType authType,			// Allows the client application to
request a
                                                // specific type of authentication mechanism.
         in IpOsa appAuthInterface,			// Provides a reference to the
client application
                                                // authentication interface.
         out TpAuthDomain fwDomainFwAuth fwAuthInterface		// Provides a framework
identifier, and a reference
                                                // to framework authentication interface.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application, once mutual authentication is
         achieved, to request access to the framework and specify the type of access desired.
*/
         void requestAccess (
         in TpAccessType accessType,		// Identifies the type of access interface
requested by
                                                // the client application.
         in IpOsa appAccessInterface,		// Provides a reference to the access interface
                                                of the
                                                // client application.
         out IpOsa fwAccessInterface			// Provides a reference to call the
access interface of
                                                // the framework.
```

```
         ) raises (TpGeneralException);

      };


      /* The Access Framework interface is used by the client application to perform the
      mechanisms
      necessary for it to obtain access to SCFs. */
      interface IpAccess : IpOsa {

         /* This method is invoked by the client application to obtain interface references to
         other
         framework interfaces. */
         void obtainInterface (
         in TpInterfaceName interfaceName,   // The name of the framework interface to which a
                                             // reference to the interface is requested.
         out IpOsa fwInterface                           // The requested interface
reference.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to obtain interface references to
         other
         framework interfaces, when it is required to supply a callback interface to the
         framework. */
         void obtainInterfaceWithCallback (
         in TpInterfaceName interfaceName,          // The name of the framework interface to
which
                                                 // a reference to the interface is
                                              requested.
         in IpOsa appInterface,                                  // This is the reference to
the client application
                                                 // interface which is used for callbacks.
         out IpOsa fwInterface                                       // The requested
interface reference.
         ) raises (TpGeneralException);

         /* This method may be invoked by the client application to check whether it has been
         granted permission to access the specified SCF and, if granted, the level of trust
that
         will be applied. */
         void accessCheck (
         in TpString securityContext,                            // A group of security
relevant
                                                              // attributes.
         in TpString securityDomain,                             // The security domain in
which
                                                              // the client application is
                                                              // operating.
         in TpString group,                                       // Used to define the access
                                                              // rights associated with all
                                                              // clients that belong to that
                                                              // group.
         in TpString serviceAccessTypes,                          // Defined by the specific
                                                              // security model in use.
         out TpServiceAccessControl serviceAccessControl          // The access control policy
                                                              // information controlling
                                                              // access to the service
                                                              // capability feature, and the
                                                              // trustLevel that the network
                                                              // operator has assigned to the
                                                              client
                                                              // application.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to identify the SCF that it
wishes
         to use. */
         void selectService (
         in TpServiceID serviceID,                       // Identifies the SCF.
         in TpServicePropertyList serviceProperties,     // List the properties that the SCF
                                                      // should support.
         out TpServiceToken serviceToken                 // A free format text token
                                                   returned by
                                                      // the framework, which can be signed
                                                       as
                                                      // part of a service agreement.
         ) raises (TpGeneralException);
```

```
        /* This method is invoked by the client application to request that the framework
       sign an
        agreement on the SCF, which allows the client application to use the SCF. */
        void signServiceAgreement (
        in TpServiceToken serviceToken,                          // Used to identify the SCF
                                                                  // instance requested by
                                                                 the
                                                                  // client application.
        in TpString agreementText,                               // The agreement text to be
                                                                  // signed by the
                                                           framework.
        in TpSigningAlgorithm signingAlgorithm,                  // The algorithm used to
compute
                                                                  // the digital signature.
        out TpSignatureAndServiceMgrRef signatureAndServiceMgr    // A reference to a
structure
                                                                  // that contains the
                                                           digital
                                                                  // signature of the
                                                           framework
                                                                  // for the service
                                                           agreement,
                                                                  // and a reference to the
                                                                  // SCF manager interface
                                                           of
                                                                  // the SCF.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to terminate an agreement for the
        specified SCF. */
        void terminateServiceAgreement (
        in TpServiceToken serviceToken,     // Identifies the service agreement to be
terminated.
        in TpString terminationText,        // Describes the reason for the termination of the
                                             // service agreement.
        in TpString digitalSignature        // Used by the framework to check that the
                                             // terminationText has been signed by the client.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to end the access session
        with the Framework. */
        void endAccess () raises (TpGeneralException);

      };



      /* The Access client application interface is used by the Framework to perform the steps
      that
      are necessary in order to allow it to SCF access. */
      interface IpAppAccess : IpOsa {

        /* This method is invoked by the Framework to request that client application sign an
        agreement on a specified SCF. */
        void signServiceAgreement (
        in TpServiceToken serviceToken,            // Identifies the SCF instance to which
                                                     // this service agreement corresponds.
        in TpString agreementText,                 // Agreement text that has to be signed by
the
                                                     // client application.
        in TpSigningAlgorithm signingAlgorithm,           // Algorithm used to compute the
digital
                                                     // signature.
        out TpString digitalSignature                     // Signed version of a hash of the
service
                                                     // token and agreement text given by the
                                                     // framework.
        ) raises (TpGeneralException);

        /* This method is invoked by the Framework to terminate an agreement for a specified
        SCF. */
        void terminateServiceAgreement (
        in TpServiceToken serviceToken,            // Identifies the SCF agreement to be
terminated.
        in TpString terminationText,       // Describes the reason for the termination.
        in TpString digitalSignature       // Used by the Framework to confirm its identity
to the
                                            // client.
```

```
         ) raises (TpGeneralException);

         /* This method is invoked by the Framework to end the client application's access
session
         with the framework. */
         void terminateAccess (
         in TpString terminationText,              // Describes the reason for the
termination of
                                                     // the access session.
         in TpSigningAlgorithm signingAlgorithm,           // The algorithm used to compute
the digital
                                                     // signature.
         in TpString digitalSignature                      // Used by the Framework to confirm
its
                                                     // identity to the client.
         ) raises (TpGeneralException);

      };



      /* The Authentication Framework interface is used by client application to perform its
part of
      the mutual authentication process with the Framework necessary to be allowed to use any
of the
      other interfaces supported by the Framework. */
      interface IpAuthentication : IpOsa {

         /* This method is invoked by the client application to start the authentication
process,
         informed the Framework of the authentication mechanisms it supports, and be informed
by its
         of its preferred choice. */
         void selectAuthMethod (
         in TpAuthCapabilityList authCapability,           // Informs the Framework of the
authentication
                                                     // mechanisms supported by the client
                                                     // application.
         out TpAuthCapability prescribedMethod             // Indicates the mechanism
preferred by the
                                                     // framework.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to authenticate the framework
using the
         mechanism indicated in the parameter prescribedMethod. */
         void authenticate (
         in TpAuthCapability prescribedMethod,             // Specifies the method accepted by
that the
                                                  // framework for authentication.
         in TpString challenge,                   // The challenge presented by the client
                                                  // application to be responded to by the
                                                  // framework.
         out TpString response                     // The response of the framework to the
                                                  // challenge of the client application.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to to abort the authentication
         process.*/
         void abortAuthentication() raises (TpGeneralException);

      };



      /* The Authentication client application interface is used by the Framework to
      authenticate

      the client application. */

      interface IpAppAuthentication : IpOsa {

         /* This method is invoked by the Framework to authenticate the client application
using the
         mechanism indicated in prescribedMethod. */
         void authenticate (
         in TpAuthCapability prescribedMethod,             // The agreed authentication
method.
```

```
         in TpString challenge,                                                    // The
challenge presented by the Framework.
         out TpString response
         ) raises (TpGeneralException);

         /* This method is invoked by the Framework to abort the authentication process. */
         void abortAuthentication() raises (TpGeneralException);

      };

};};};};};
```

9.2.4 Registration IDL

```
#include <fw.idl>


module org{
module threegpp{
module osa{
module fw{
module registration{

      /*******************************************************************************
/
      //                                Interface definitions
//
      /*******************************************************************************
/

      /* The Service Registration Framework interface provides the methods used for the
      registration
      of network SCFs at the Framework. */
      interface IpServiceRegistration : IpOsa {

            /* This method is used to register a SCF in the Framework, for subsequent
            discovery by
            the applications. */
            void registerService (
            in TpServiceTypeName                  serviceTypeName,

            in TpServicePropertyList        servicePropertyList,
            out TpServiceID                           serviceID
            ) raises (TpGeneralException);

            /* This method informs the Framework of the availability of a service factory for
            a
            previously registered SCF. */
            void announceServiceAvailability (
            in TpServiceID          serviceID,
            in IpOsa                      serviceFactory
            ) raises (TpGeneralException);

            /* This method is used to remove a registered SCF from the Framework. */
            void unregisterService (
            in TpServiceID          serviceID
            ) raises  (TpGeneralException);

            /* This method is used to ebtain the decription of a certain SCF as it was
            registered in
            the Framework. */
            void describeService (
            in TpServiceID                            serviceID,
            out TpServiceDescription        serviceDescription
            ) raises (TpGeneralException);
               };

      /* The Service Factory Framework interface provides the Framework with access to a
      manager
      interface of a network SCF to be given to an application. */
      interface IpSvcFactory : IpOsa {

            /* This method returns an SCF manager interface reference for a specified
            application. */
            void getServiceManager (
            in TpDomainIDClientAppID        application,
            out IpOsa                       serviceManager
```

```
            ) raises (TpGeneralException);
               };
};};};};};
```

3GPP Meeting CN5 #5
Bristol, 5-7 Sept 2000

Document N5-000172
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	009R2	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*

for approval [X] for information [ ]

strategic [ ] non-strategic [ ] *(for SMG use only)*

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)* (U)SIM [ ] ME [ ] UTRAN / Radio [ ] Core Network [X]

Source: N5 **Date:** 27 August 2000

Subject: Alignment of Framework with Parlay 2.1, correction of missing service token

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	[X]	**Release:**	Phase 2	[ ]
A	Corresponds to a correction in an earlier release	[ ]		Release 96	[ ]
B	Addition of feature	[ ]		Release 97	[ ]
C	Functional modification of feature	[ ]		Release 98	[ ]
D	Editorial modification	[ ]		Release 99	[X]
				Release 00	[ ]

Reason for change: The FW of TS 29.198 contains a few differences compared to Parlay 2.1, mainly in the area of different parameter names and data-types.
Currently in TS 29.198 the service token parameter is missing for the operation accessCheck on the IpAccess interface. (A service token is created by the framework when an application selects a service capability feature to be used)

Clauses affected: 6.2, 9.2

Other specs affected:

Other 3G core specifications	[ ]	→ List of CRs:
Other GSM core specifications	[ ]	→ List of CRs:
MS test specifications	[ ]	→ List of CRs:
BSS test specifications	[ ]	→ List of CRs:
O&M specifications	[ ]	→ List of CRs:

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

6.2.3.4 IpAccess

<<Interface>> IpAccess
obtainInterface(interfaceName: in TpInterfaceName, fwInterface: out IpOsaRefRef): TpResult obtainInterfaceWithCallback(interfaceName: in TpInterfaceName, appInterface: in IpOsaRef, fwInterface: out IpOsaRefRef): TpResult accessCheck(serviceToken: in TpServiceToken, securityContext: in TpString, securityDomain: in TpString, group : in TpString, serviceAccessTypes: in TpString, serviceAccessControl: out TpServiceAccessControlRef): TpResult selectService(serviceID: in TpServiceID, serviceProperties: in TpServicePropertyList, serviceToken: out TpServiceTokenRef): TpResult signServiceAgreement(serviceToken: in TpServiceToken, agreementText: in TpString, signingAlgorithm: in TpSigningAlgorithm, signatureAndServiceMgr: out TpSignatureAndServiceMgrRef): TpResult terminateServiceAgreement(serviceToken: in TpServiceToken, terminationText: in TpString, digitalSignature: in TpString): TpResult endAccess(endAccessProperties: in TpPropertyList) : TpResult

9.2.3 Trust and Security Management IDL

```
#include <fw.idl>

..


      /* The Access Framework interface is used by the client application to perform the
      mechanisms
      necessary for it to obtain access to SCFs. */
      interface IpAccess : IpOsa {

         /* This method is invoked by the client application to obtain interface references to
         other
         framework interfaces. */
         void obtainInterface (
         in TpInterfaceName interfaceName,   // The name of the framework interface to which a
                                             // reference to the interface is requested.
         out IpOsa fwInterface                          // The requested interface
reference.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to obtain interface references to
         other
         framework interfaces, when it is required to supply a callback interface to the
         framework. */
         void obtainInterfaceWithCallback (
         in TpInterfaceName interfaceName,          // The name of the framework interface to
which
                                                 // a reference to the interface is
                                              requested.
         in IpOsa appInterface,                                 // This is the reference to
the client application
                                                 // interface which is used for callbacks.
         out IpOsa fwInterface                                          // The requested
interface reference.
         ) raises (TpGeneralException);

         /* This method may be invoked by the client application to check whether it has been
         granted permission to access the specified SCF and, if granted, the level of trust
that
         will be applied. */
         void accessCheck (
         in TpServiceToken serviceToken,
         in TpString securityContext,                           // A group of security
relevant
                                                             // attributes.
         in TpString securityDomain,                            // The security domain in
which
                                                             // the client application is
                                                             // operating.
         in TpString group,                                      // Used to define the access
                                                             // rights associated with all
                                                             // clients that belong to that
                                                             // group.
         in TpString serviceAccessTypes,                         // Defined by the specific
                                                             // security model in use.
         out TpServiceAccessControl serviceAccessControl         // The access control policy
                                                             // information controlling
                                                             // access to the service
                                                             // capability feature, and the
                                                             // trustLevel that the network
                                                             // operator has assigned to the
                                                             client
                                                             // application.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to identify the SCF that it
wishes
         to use. */
         void selectService (
         in TpServiceID serviceID,                          // Identifies the SCF.
         in TpServicePropertyList serviceProperties,        // List the properties that the SCF
                                                         // should support.
         out TpServiceToken serviceToken                    // A free format text token
                                                      returned by
```

```
                                                    // the framework, which can be signed
                                                as
                                                    // part of a service agreement.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to request that the framework
       sign an
        agreement on the SCF, which allows the client application to use the SCF. */
        void signServiceAgreement (
        in TpServiceToken serviceToken,                         // Used to identify the SCF
                                                                 // instance requested by
                                                                the
                                                                 // client application.
        in TpString agreementText,                              // The agreement text to be
                                                                 // signed by the
                                                        framework.
        in TpSigningAlgorithm signingAlgorithm,                 // The algorithm used to
compute
                                                                 // the digital signature.
        out TpSignatureAndServiceMgrRef signatureAndServiceMgr    // A reference to a
structure
                                                                 // that contains the
                                                        digital
                                                                 // signature of the
                                                        framework
                                                                 // for the service
                                                        agreement,
                                                                 // and a reference to the
                                                                 // SCF manager interface
                                                        of
                                                                 // the SCF.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to terminate an agreement for the
        specified SCF. */
        void terminateServiceAgreement (
        in TpServiceToken serviceToken,     // Identifies the service agreement to be
terminated.
        in TpString terminationText,        // Describes the reason for the termination of the
                                             // service agreement.
        in TpString digitalSignature        // Used by the framework to check that the
                                             // terminationText has been signed by the client.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to end the access session
        with the Framework. */
        void endAccess () raises (TpGeneralException);

      };

      ..
```

3GPP Meeting CN5 #5
Bristol, 5-7 Sept 2000

Document N5-000173
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	010R2	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*

for approval [X] for information [ ]

strategic [ ] non-strategic [ ] *(for SMG use only)*

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)* (U)SIM [ ] ME [ ] UTRAN / Radio [ ] Core Network [X]

Source: N5 **Date:** 27 August 2000

Subject: Alignment of Framework with Parlay 2.1, parameter name and data-type alignments

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	[X]	**Release:**	Phase 2	[ ]
A	Corresponds to a correction in an earlier release	[ ]		Release 96	[ ]
B	Addition of feature	[ ]		Release 97	[ ]
C	Functional modification of feature	[ ]		Release 98	[ ]
D	Editorial modification	[ ]		Release 99	[X]
				Release 00	[ ]

Reason for change: The FW of TS 29.198 contains a few differences compared to Parlay 2.1, mainly in the area of different parameter names and data-types.
This CR corrects differences in parameter names and data-types.

Clauses affected: 6.2, 8.2, 9.2

Other specs affected:

Other 3G core specifications	[ ]	→ List of CRs:
Other GSM core specifications	[ ]	→ List of CRs:
MS test specifications	[ ]	→ List of CRs:
BSS test specifications	[ ]	→ List of CRs:
O&M specifications	[ ]	→ List of CRs:

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

6.2.3.3 IpAuthentication

<<Interface>> IpAuthentication
selectAuthMethod (authCapsability: in TpAuthCapabiltyList, prescribedMethod: out TpAuthCapabilityRef) : TpResult authenticate (prescribedMethod: in TpAuthCapability, challenge: in TpString, response: out TpStringRef) : TpResult abortAuthentication() : TpResult

6.2.3.5 IpAppAccess

<<Interface>> IpAppAccess
signServiceAgreement(serviceToken: in TpServiceToken, agreementText: in TpString, signingAlgorithm: in TpSigningAlgorithm, digitalSignature: out TpStringRef): TpResult terminateServiceAgreement(serviceToken: in TpServiceToken, terminationText: in TpString, digitalSignature: in TpString): TpResult terminateAccess(terminationText: in TpString, signingAlgorithm: in TpSigningAlgorithm, digitalSignature: in TpStringRef) : TpResult

6.2.4.6 IpAppLoadManager

<<Interface>> IpAppLoadManager
queryAppLoadReq(serviceIDs: in TpServiceIdList, timeInterval : TpTimeInterval) : TpResult queryLoadRes(loadStatistics : in TpLoadStatisticList) : TpResult queryLoadErr(loadStatisticsError : in TpLoadStatisticErrorList) : TpResult disableLoadControl(serviceIDs: in TpServiceIdList) : TpResult enableLoadControl(loadStatistics : in TpLoadStatisticList) : TpResult resumeNotification() : TpResult suspendNotification() : TpResult

6.2.4.7 IpFaultManager

<<Interface>> IpFaultManager
activityTestReq(activityTestID: in TpActivityTestID, svcID: in TpServiceID): TpResult appActivityTestRes(activityTestID: in TpActivityTestID, activityTestResult: in TpActivityTestRes): TpResult svcUnavailableInd(serviceID: in TpServiceID): TpResult genFaultStatsRecordReq(timePeriod: in TpTimeInterval, serviceIDs~~List~~: in TpServiceIDList): TpResult

6.2.4.10 IpAppOAM

<<Interface>> IpAppOAM
systemDateTimeQuery(~~clientDateAndTime~~ systemDateAndTime: in TpDateAndTime, clientDateAndTime~~systemDateAndTime~~: out TpDateAndTimeRef) : TpResult

6.2.5 Service Factory

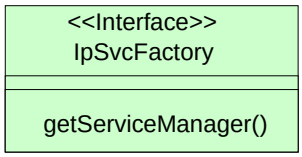


Figure 6-8: Service Factory Class Diagram

<<Interface>> IpSvcFactory
getServiceManager(application : in TpClientAppID, serviceManager : out IpServiceRefRef~~OSA~~) : TpResult

6.2.6 Service Registration

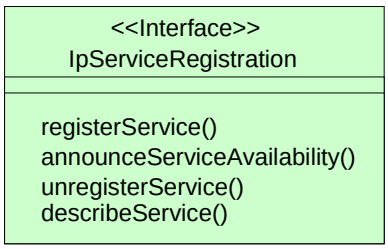


Figure 6-7: Service Registration Class Diagram

<<Interface>> IpServiceRegistration
registerService(serviceTypeName : in TpServiceTypeName, servicePropertyList : in TpServicePropertyList, serviceID : out TpServiceIDRef) : TpResult announceServiceAvailability(serviceID : in TpServiceID, serviceFactoryRef : in IpOSARef) : TpResult unregisterService(serviceID : in TpServiceID) : TpResult describeService(serviceID : in TpServiceID, serviceDescription : out TpServiceDescriptionRef) : TpResult

8.2.2.8 TpSignatureAndServiceMgr

This is a Sequence of Data Elements containing the digital signature of the framework for the service agreement, and a reference to the SCF manager interface of the SCF.

Sequence Element Name	Sequence Element Type
DigitalSignature	TpString~~Ref~~
ServiceMgrInterface	IpServiceRef

The digitalSignature is the signed version of a hash of the service token and agreement text given by the client application.
The ServiceMgrInterface is a reference to the SCF manager interface for the selected SCF.

8.2.3 Integrity Management Data Definitions

8.2.3.3 TpFaultStats~~Set~~

This defines the sequence of data elements which provide the statistics on a per fault type basis.

Sequence Element Name	Sequence Element Type	Description
Fault	TpInterfaceFault	
Occurrences	TpInt32	The number of separate instances of this fault
MaxDuration	TpInt32	The number of seconds duration of the longest fault
TotalDuration	TpInt32	The cumulative duration (all occurrences)
NumberOfClientsAffected	TpInt32	The number of clients informed of the fault by the Fw

Occurrences is the number of separate instances of this fault during the period. MaxDuration and TotalDuration are the number of seconds duration of the longest fault and the cumulative total during the period. NumberOfClientsAffected is the number of clients informed of the fault by the framework.

TpFaultStatsSet

This data type defines a Numbered Set of Data Elements of type TpFaultStats

9.2 Framework IDL

9.2.3 Trust and Security Management IDL

```
      ..

      /* The Authentication Framework interface is used by client application to perform its
part of
      the mutual authentication process with the Framework necessary to be allowed to use any
of the
      other interfaces supported by the Framework. */
      interface IpAuthentication : IpOsa {

         /* This method is invoked by the client application to start the authentication
process,
         informed the Framework of the authentication mechanisms it supports, and be informed
by its
         of its preferred choice. */
         void selectAuthMethod (
         in TpAuthCapabilityList authsCapability,          // Informs the Framework of the
authentication
                                                          // mechanisms supported by the client
                                                          // application.
         out TpAuthCapability prescribedMethod            // Indicates the mechanism
preferred by the
                                                          // framework.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to authenticate the framework
using the
         mechanism indicated in the parameter prescribedMethod. */
         void authenticate (
          in TpAuthCapability prescribedMethod,            // Specifies the method accepted by
                                                                                    that the
                                                        // framework for authentication.
         in TpString challenge,                        // The challenge presented by the client
                                                        // application to be responded to by the
                                                        // framework.
         out TpString response                          // The response of the framework to the
                                                        // challenge of the client application.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to to abort the authentication
         process.*/
         void abortAuthentication() raises (TpGeneralException);

      };
```

9.2.4 Integrity Management IDL

```
..
      /* The Fault Manager Framework interface is used by the client application to inform the
      Framework of events that affect the integrity of the Framework and SCFs, and to request
      information about the integrity of the system. */
      interface IpFaultManager : IpOsa {

         /* This method may be invoked by the client application to test that the Framework or
a
         SCF is operational. */
         void activityTestReq (
         in TpActivityTestID activityTestID,         // Identifier provided by the client
                                                     // application to correlate the
                                                     // response with this request.
         in TpServiceID svcID,                             // Identifies for which SCF the
client
                                                      // application is requesting the
                                              activity test
                                                      // be done.
         in TpClientAppID appID                            // Identifies which client
application is
                                                      // requesting the activity test (and
                                               therefore
```

```
                                                    // which application receives the
                                              results).
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to return the result of a
previously
        requested activity test. */
        void appActivityTestRes (
        in TpActivityTestID activityTestID,         // Used by the Framework to correlate
this
                                                    // response with the original request.
        in TpActivityTestRes activityTestResult     // Result of the activity test.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to inform the Framework that it
can no
        longer use the indicated SCF. */
        void svcUnavailableInd (
        in TpServiceID serviceID,           // Identity of the SCF which can no longer be
used.
        in TpClientAppID appID              // Identity of the application sending the
indication.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to request fault statistics from
the
        Framework. */
        void genFaultStatsRecordReq (
        in TpTimeInterval timePeriod,             // The period over which the fault
statistics
                                                  // are to be generated.
        in TpServiceIDsList serviceIDList,        // The SCFs that the application would
like
                                                  // to have included in the general fault
                                                  // statistics record.
        in TpClientAppID appID                    // Identifies which client application is
                                                  // requesting the statistics record (and
                                                  // therefore should receive it).
        ) raises (TpGeneralException);

    };
```

9.2.4 Registration IDL

```
    /* The Service Factory Framework interface provides the Framework with access to a
    manager
    interface of a network SCF to be given to an application. */
    interface IpSvcFactory : IpOsa {

        /* This method returns an SCF manager interface reference for a specified
        application. */
        void getServiceManager (
        in TpClientAppID        application,
        out IpServiceOsa                        serviceManager
        ) raises (TpGeneralException);
           };
}
```

9.2.4 Integrity Management IDL

```
#include <fw.idl>


module org{
module threegpp{
module osa{
module fw{
module integrity{

    /*******************************************************************************
/
```

```
      //                                Data definitions
//
      /*******************************************************************************************
/

       typedef TpString             TpActivityTestRes;           // An implementation
specific result, whose values
                                                  // are Framework provider specific.

       struct TpTimeInterval {                     // A time interval.
              TpDateAndTime         StartTime;
              TpDateAndTime         StopTime;
       };


       enum TpInterfaceFault {                     // The cause of the interface fault
detected.
              INTERFACE_FAULT_UNDEFINED,                        // Undefined.
              INTERFACE_FAULT_LOCAL_FAILURE,              // A fault in the local API
software or hardware has been
                                          // detected.
              INTERFACE_FAULT_GATEWAY_FAILURE,            // A fault in the gateway API
software or hardware has been
                                          // detected.
              INTERFACE_FAULT_PROTOCOL_ERROR              // An error in the protocol used on
the client-gateway link
                                          // has been detected.
       };

       struct TpFaultStatsSet {                    // Statistics on a per fault type basis.
              TpInterfaceFault      Fault;
              TpInt32               Occurrences;                                        //
The number of separate instances of this fault
                                                  // during the period.
              TpInt32               MaxDuration;                                        //
The duration in seconds of the longest fault.
              TpInt32               TotalDuration;                                      //
The cumulative total during the period.
              TpInt32               NumberOfClientsAffected;            // Those informed of
the fault by the Framework.
       };

       typedef sequence <TpFaultStats> TpFaultStatsSet;

       struct TpFaultStatsRecord {          // The set of fault information records to be
returned for the
                                  // requested time period.
              TpTimeInterval        Period;
              TpFaultStatsSet       FaultRecords;
       };
```

3GPP Meeting CN5 #5
Bristol, 5-7 Sept 2000

Document N5-000138
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	011R1	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*

for approval [X] for information [ ]

strategic [ ] non-strategic [ ] *(for SMG use only)*

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)* (U)SIM [ ] ME [ ] UTRAN / Radio [ ] Core Network [X]

Source: N5 **Date:** 27 August 2000

Subject: Alignment of Framework with Parlay 2.1, one interface per application correction

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	[X]	**Release:**	Phase 2	[ ]
A	Corresponds to a correction in an earlier release	[ ]		Release 96	[ ]
B	Addition of feature	[ ]		Release 97	[ ]
C	Functional modification of feature	[ ]		Release 98	[ ]
D	Editorial modification	[ ]		Release 99	[X]
				Release 00	[ ]

Reason for change: The FW of TS 29.198 contains a few differences compared to Parlay 2.1, mainly in the area of different parameter names and data-types. In the TS29.198 some of operations of the load manager and fault manager have a parameter for indicating the client application that wants to use an operation. As the interface is created per application this parameter is not needed and should be removed.

Clauses affected: 6.2, 9.2

Other specs affected:

Other 3G core specifications	[ ]	→ List of CRs:
Other GSM core specifications	[ ]	→ List of CRs:
MS test specifications	[ ]	→ List of CRs:
BSS test specifications	[ ]	→ List of CRs:
O&M specifications	[ ]	→ List of CRs:

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

6.2.4.5 IpLoadManager

<<Interface>> IpLoadManager
reportLoad(~~requester : in TpClientAppID,~~ loadLevel : in TpLoadLevel) : TpResult queryLoadReq(~~requester : in TpClientAppID,~~ serviceIDs: in TpServiceIDList, timeInterval : in TpTimeInterval) : TpResult queryAppLoadRes(loadStatistics : in TpLoadStatisticList) : TpResult queryAppLoadErr(loadStatisticsError : in TpLoadStatisticErrorList) : TpResult registerLoadController(~~ requester : in TpClientAppID,~~ serviceIDs: in TpServiceIDList) : TpResult unregisterLoadController(~~ requester : in TpClientAppID,~~ serviceIDs: in TpServiceIDList) : TpResult resumeNotification(serviceIDs: in TpServiceIDList) : TpResult suspendNotification(serviceIDs: in TpServiceIDList) : TpResult

6.2.4.7 IpFaultManager

<<Interface>> IpFaultManager
activityTestReq(activityTestID: in TpActivityTestID, svcID: in TpServiceID~~, appID: in TpClientAppID~~): TpResult appActivityTestRes(activityTestID: in TpActivityTestID, activityTestResult: in TpActivityTestRes): TpResult svcUnavailableInd(serviceID: in TpServiceID~~, appID: in TpClientAppID~~): TpResult genFaultStatsRecordReq(timePeriod: in TpTimeInterval, serviceIDsList: in TpServiceIDList~~, appID: in TpClientAppID~~): TpResult

9.2.4 Integrity Management IDL

```
#include <fw.idl>

..


      /* The Load Manager Framework interface is used by the client application for load
balancing
      management. */
      interface IpLoadManager : IpOsa {

         /* This method is invoked by the client application to notify framework its current
load
         level (0,1, or 2) when the load level on the application has changed. */
         void reportLoad (
         in TpClientAppID requester,          // The identifier of the client application for
                                              // callbacks from the load balancing SCF.
         in TpLoadLevel loadLevel             // The application's load level.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to request load statistic records
for
         the framework and specified SCFs. */
         void queryLoadReq (
         in TpClientAppID requester,          // The identifier of the client application for
                                               // callbacks from the load balancing SCF.
         in TpServiceIDList serviceIDs,       // Specifies the framework and SCFs for which the
                                               // load statistics shall be reported.
         in TpTimeInterval timeInterval       // The time interval within which the load
statistics
                                               // are generated.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to report load statistics back to
the
         framework that requested the information. */
         void queryAppLoadRes (
         in TpLoadStatisticList loadStatistics      // The application's load statistics.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to return an error response to
the
         framework that requested the application's load statistics information. */
         void queryAppLoadErr (
         in TpLoadStatisticErrorList loadStatisticsError   // The error code associated with
the
                                                            // failed attempt to retrieve the
                                                            // application's load statistics.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to register the client
application for
         load management under various load conditions. */
         void registerLoadController (
         in TpClientAppID requester,                 // Identifies the client application for
callbacks
                                                  // from the load balancing SCF.
         in TpServiceIDList serviceIDs               // Specifies the framework and SCFs to be
                                                  // registered for load control.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to unregister for load
management. */
         void unregisterLoadController (
         in TpClientAppID requester,                 // Identifies the client application for
callbacks from
                                                  // the load balancing SCF.
         in TpServiceIDList serviceIDs            // Specifies the framework or SCFs to be
                                                  // unregistered for load control.
         ) raises (TpGeneralException);

         /* This method is invoked by the client application to resume load management
notifications
         to it from the framework and specified SCFs. */
         void resumeNotification (
         in TpServiceIDList serviceIDs       // Specifies the framework and SCFs for which
```

```
                                          // notifications are to be resumed.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to suspend load management
        notifications to it from the framework and specified SCFs, while it handles a
temporary
        load condition. */
        void suspendNotification (
        in TpServiceIDList serviceIDs      // Specifies the framework and SCFs for which
                                           // notifications are to be suspended.
        ) raises (TpGeneralException);

     };


     ..


     /* The Fault Manager Framework interface is used by the client application to inform the
     Framework of events that affect the integrity of the Framework and SCFs, and to request
     information about the integrity of the system. */
     interface IpFaultManager : IpOsa {

        /* This method may be invoked by the client application to test that the Framework or
a
        SCF is operational. */
        void activityTestReq (
        in TpActivityTestID activityTestID,        // Identifier provided by the client
                                                   // application to correlate the
                                                   // response with this request.
        in TpServiceID svcID,                            // Identifies for which SCF the
client
                                                    // application is requesting the
                                              activity test
                                                   // be done.
        in TpClientAppID appID                          // Identifies which client
application is
                                                   // requesting the activity test (and
                                               therefore
                                                   // which application receives the
                                               results).
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to return the result of a
previously
        requested activity test. */
        void appActivityTestRes (
        in TpActivityTestID activityTestID,        // Used by the Framework to correlate
this
                                                   // response with the original request.
        in TpActivityTestRes activityTestResult    // Result of the activity test.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to inform the Framework that it
can no
        longer use the indicated SCF. */
        void svcUnavailableInd (
        in TpServiceID serviceID,          // Identity of the SCF which can no longer be
used.
        in TpClientAppID appID             // Identity of the application sending the
indication.
        ) raises (TpGeneralException);

        /* This method is invoked by the client application to request fault statistics from
the
        Framework. */
        void genFaultStatsRecordReq (
        in TpTimeInterval timePeriod,              // The period over which the fault
statistics
                                                // are to be generated.
        in TpServiceIDList serviceIDList,          // The SCFs that the application would
like
                                                // to have included in the general fault
                                                // statistics record.
        in TpClientAppID appID                     // Identifies which client application is
                                                // requesting the statistics record (and
                                                // therefore should receive it).
```

```
    ) raises (TpGeneralException);
};
```

3GPP Meeting CN5 #5
Bristol, 5-7 Sept 2000

Document N5-000139
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	012R1	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*	for approval	X	strategic		*(for SMG use only)*
	for information		non-strategic		

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)*	(U)SIM		ME		UTRAN / Radio		Core Network	X

Source:	N5	**Date:**	27 August 2000

Subject: Alignment of Framework with Parlay 2.1, only one error returned in load manager query

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	X	**Release:**	Phase 2	
A	Corresponds to a correction in an earlier release			Release 96	
B	Addition of feature			Release 97	
C	Functional modification of feature			Release 98	
D	Editorial modification			Release 99	X
				Release 00	

Reason for change: The FW of TS 29.198 contains a few differences compared to Parlay 2.1, mainly in the area of different parameter names and data-types.
In case an error occurs during a load query, the load manager interfaces will return only one error indication and not a list of errors.

Clauses affected: 6.2.4.5, 6.2.4.6

Other specs affected:

Other 3G core specifications		→ List of CRs:
Other GSM core specifications		→ List of CRs:
MS test specifications		→ List of CRs:
BSS test specifications		→ List of CRs:
O&M specifications		→ List of CRs:

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

6.2.4.5 IpLoadManager

<<Interface>> IpLoadManager
reportLoad(requester : in TpClientAppID, loadLevel : in TpLoadLevel) : TpResult queryLoadReq(requester : in TpClientAppID, serviceIDs: in TpServiceIDList, timeInterval : in TpTimeInterval) : TpResult queryAppLoadRes(loadStatistics : in TpLoadStatisticList) : TpResult queryAppLoadErr(loadStatisticsError : in TpLoadStatisticError~~List~~) : TpResult registerLoadController(requester : in TpClientAppID, serviceIDs: in TpServiceIDList) : TpResult unregisterLoadController(requester : in TpClientAppID, serviceIDs: in TpServiceIDList) : TpResult resumeNotification(serviceIDs: in TpServiceIDList) : TpResult suspendNotification(serviceIDs: in TpServiceIDList) : TpResult

6.2.4.6 IpAppLoadManager

<<Interface>> IpAppLoadManager
queryAppLoadReq(serviceIDs: in TpServiceIdList, timeInterval : TpTimeInterval) : TpResult queryLoadRes(loadStatistics : in TpLoadStatisticList) : TpResult queryLoadErr(loadStatisticsError : in TpLoadStatisticError~~List~~) : TpResult disableLoadControl(serviceIDs: in TpServiceIdList) : TpResult enableLoadControl(loadStatistics : in TpLoadStatisticList) : TpResult resumeNotification() : TpResult suspendNotification() : TpResult

3GPP Meeting CN5 #5
Bristol, 5-7 Sept 2000

Document N5-000140
e.g. for 3GPP use the format TP-99xxx
or for SMG, use the format P-99-xxx

CHANGE REQUEST

Please see embedded help file at the bottom of this page for instructions on how to fill in this form correctly.

29.198	CR	013R1	Current Version:	3.0.0

GSM (AA.BB) or 3G (AA.BBB) specification number ↑ *↑ CR number as allocated by MCC support team*

For submission to: CN#09 *list expected approval meeting # here ↑*

for approval [X]
for information [ ]

strategic [ ]
non-strategic [ ]
(for SMG use only)

Form: CR cover sheet, version 2 for 3GPP and SMG *The latest version of this form is available from: ftp://ftp.3gpp.org/Information/CR-Form-v2.doc*

Proposed change affects: *(at least one should be marked with an X)* (U)SIM [ ] ME [ ] UTRAN / Radio [ ] Core Network [X]

Source: N5 **Date:** 27 August 2000

Subject: Alignment of Framework with Parlay 2.1, missing operation fwUnavailableInd in IpAppFaultManager.

Work item: OSA

Category: *(only one category shall be marked with an X)*

F	Correction	[X]
A	Corresponds to a correction in an earlier release	[ ]
B	Addition of feature	[ ]
C	Functional modification of feature	[ ]
D	Editorial modification	[ ]

Release:

Phase 2	[ ]
Release 96	[ ]
Release 97	[ ]
Release 98	[ ]
Release 99	[X]
Release 00	[ ]

Reason for change: The FW of TS 29.198 contains a few differences compared to Parlay 2.1, mainly in the area of different parameter names and data-types. The operation for indicating that the Framework is unavailable is missing in the IpAppFaultManager interface of TS 29.198

Clauses affected: 6.2, 8.2 9.2

Other specs affected:

Other 3G core specifications	[ ]	→ List of CRs:
Other GSM core specifications	[ ]	→ List of CRs:
MS test specifications	[ ]	→ List of CRs:
BSS test specifications	[ ]	→ List of CRs:
O&M specifications	[ ]	→ List of CRs:

Other comments:

help.doc

<--------- double-click here for help and instructions on how to create a CR.

6.2.4.8 IpAppFaultManager

<<Interface>> IpAppFaultManager
activityTestRes(activityTestID: in TpActivityTestID, activityTestResult: in TpActivityTestRes): TpResult appActivityTestReq(activityTestID: in TpActivityTestID): TpResult fwFaultReportInd(fault: in TpInterfaceFault): TpResult fwFaultRecoveryInd(fault: in TpInterfaceFault): TpResult fwUnavailableInd(reason: in TpFwUnavailReason): TpResult svcUnavailableInd(serviceID: in TpServiceID, reason: in TpSvcUnavailReason): TpResult genFaultStatsRecordRes(faultStatistics: in TpFaultStatsRecord, serviceIDs: in TpServiceIDList): TpResult

8.2.3.7 TpFWAPIUnavailReason

Defines the reason why the FrameworkAPI is unavailable.

Name	Value	Description
FWAPI_UNAVAILABLE_UNDEFINED	0	Undefined
FWAPI_UNAVAILABLE_LOCAL_FAILURE	1	The Local API software or hardware has failed
FWAPI_UNAVAILABLE_GATEWAY_FAILURE	2	The gateway API software or hardware has failed
FWAPI_UNAVAILABLE_OVERLOADED	3	The frameworkgateway is fully overloaded
FWAPI_UNAVAILABLE_CLOSED	4	The frameworkgateway has closed itself (e.g. to protect from fraud or malicious attack)
FWAPI_UNAVAILABLE_PROTOCOL_FAILURE	5	The protocol used on the client-gateway link has failed

9.2.4 Integrity Management IDL

```
      ..


      /* The Fault Manager client application interface is used by the Framework to inform the
      application of events that affect the integrity of the Framework, SCF or client
      application. */
      interface IpAppFaultManager : IpOsa {

         /* This method is invoked by the Framework, in response to an activityTestReq, to
return
         the result of the activity test in this method. */
         void activityTestRes (
         in TpActivityTestID activityTestID,                    // The identifier provided
to correlate this
                                                  // response with the original request.
         in TpActivityTestRes activityTestResult           // Result of the activity test.
         ) raises (TpGeneralException);

         /* This method is invoked by the Framework to request that the client application
carries
         out an activity test to check that is it operating correctly. */
         void appActivityTestReq (
         in TpActivityTestID activityTestID        // The identifier provided to correlate
this
                                             // response with the original request.
         ) raises (TpGeneralException);

         /* This method is invoked by the Framework to notify the client application of a
failure
         within the Framework. */
         void fwFaultReportInd (
         in TpInterfaceFault fault                 // The fault that has been detected.
         ) raises (TpGeneralException);

         /* This method is invoked by the Framework to notify the client application that a
         previously reported fault has been rectified. */
         void fwFaultRecoveryInd (
         in TpInterfaceFault fault          // The fault from which the framework has
recovered.
         ) raises (TpGeneralException);

         void fwUnavailableInd (
         in TpFwUnavailReason reason
         ) raises (TpGeneralException);

         /* This method is invoked by the Framework to inform the client application that it
can no
         longer use the indicated SCF due to a failure. */
         void svcUnavailableInd (
         in TpServiceID serviceID,          // Identity of the SCF which can no longer be
used.
         in TpSvcUnavailReason reason       // The reason why the SCF is no longer available.
         ) raises (TpGeneralException);

         /* This method is invoked by the Framework to provide fault statistics to a client

         application in response to a genFaultStatsRecordReq. */

         void genFaultStatsRecordRes (
         in TpFaultStatsRecord faultStatistics,     // The fault statistics record.
         in TpServiceIDList serviceIDs              // The SCFs that have been included in the
                                                      // general fault statistics record.
         ) raises (TpGeneralException);

      };
```