3GPP TSG CN Plenary Meeting #12 **Tdoc NP-010325**
Stockholm, Sweden, 13th - 15th June 2001

Source: CN5

Title: R99 CR to Open Service Architecture; Application Programming Interface - Part 1 (29.198)

Agenda item: **7.23** Any Other R99 WI **[OSA]**

Document for: APPROVAL

Doc-	Doc-	Spec	CR	Rev	Phase	Subject	Cat	Version-	Version-	Meeting	Workitem
NP-010325	N5-010355	29.198	048		R99	IDL Correction of TpCallEventCriteria	F	3.3.0	3.4.0	N5-11	OSA

TSG_CN5 (Open Service Access – OSA) Meeting #11 ***Tdoc N5-010355***
San Diego, CA, USA, 21 – 24 May 2001

CR-Form-v4

CHANGE REQUEST

⌘ 29.198 **CR 048** ⌘ ev **-** ⌘ Current version: **3.3.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM ☐ ME/UE ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	IDL Correction of TpCallEventCriteria
Source:	⌘	CN5
Work item code:	⌘	OSA ***Date:*** ⌘ 24/052001
Category:	⌘	**F** ***Release:*** ⌘ R99

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	There is a mistake in the IDL specification of the data type TpCallEventCriteria: The attribute MonitorMode is missing.
Summary of change:	⌘	MonitorMode is inserted according to the data type definition.
Consequences if not approved:	⌘	Impossible to specify the MonitorMode parameter when using the IDL.

Clauses affected:	⌘	9.3.1
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	The IDL of Release 4 is correct in this respect. This CR was proposed by Siemens.

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. (3GPP 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 3GPP 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 */
                    TpCallMonitorMode MonitorMode;

                };


                /* 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;
                    TpCallMonitorMode MonitorMode;
                };

                /* 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 initialization 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 Busy;
                    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 initialization 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 initialization 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 = 260;
                const TpInt32 P_GCCS_INVALID_NETWORK_STATE = 261;


                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
```