3GPP TSG CN Plenary Meeting #19 **NP-030024**
12- 14 March 2003, Birmingham, UK

Source: **CN5 (OSA)**

Title: **Rel-4 CR 29.198-08 OSA API Part 8: Data session control**

Agenda item: **7.10**

Document for: **APPROVAL**

Doc-1st-Level	Spec	CR	Rev	Phase	Subject	Cat	Version-Current	Doc-2nd-Level	Workitem
NP-030024	29.198-08	018	-	Rel-4	Correction of status of methods to Data Session Control interfaces	F	4.5.0	N5-021017	OSA1
NP-030024	29.198-08	019	-	Rel-5	Addition of status of methods to Data Session Control interfaces	A	5.1.0	N5-021024	OSA2
NP-030024	29.198-08	020	-	Rel-4	Corrections to Data Session Control Types	F	4.5.0	N5-021125	OSA1
NP-030024	29.198-08	021	-	Rel-5	Corrections to data types in Data Session Control	A	5.1.0	N5-021126	OSA2

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-08** **CR** **018** ⌘ **rev** **-** ⌘ Current version: **4.5.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☒

Title:	⌘	Correction of status of methods to Data Session Control interfaces
Source:	⌘	N5
Work item code:	⌘	OSA1
Date:	⌘	31/10/2002
Category:	⌘	**F**
Release:	⌘	*REL-4*

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)*
Rel-6	*(Release 6)*

Reason for change:	⌘	There is no requirement in the standard about the necessity to implement all or only some of the methods defined for an interface.
Summary of change:	⌘	Add a statement that clarifies which methods are mandatory and which are optional.
Consequences if not approved:	⌘	Application developers will not know which methods will actually be available.

Clauses affected:	⌘	4, 8			
Other specs affected:	⌘	**Y**	**N**		⌘
			X	Other core specifications	
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

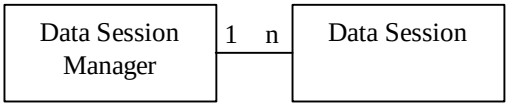
4 Data Session Control SCF

The Data Session control network service capability feature consists of two interfaces:

1) Data Session manager, containing management functions for data session related issues;

2) Data Session, containing methods to control a session.

A session can be controlled by one Data Session Manager only. Data Session Manager can control several sessions.



NOTE: The term "data session" is used in a broad sense to describe a data connection/session. For example, it comprises a PDP context in GPRS.

Figure 1: Data Session control interfaces usage relationship

The Data Session Control service capability features are described in terms of the methods in the Data Session Control interfaces. Table 1 gives an overview of the Data Session Control methods and to which interfaces these methods belong.

Table 1: Overview of Data Session Control interfaces and their methods

Data Session Manager	Data Session
createNotification	connectReq
destroyNotification	connectRes
dataSessionNotificationInterrupted	connectErr
dataSessionNotificationContinued	release
reportNotification	superviseDataSessionReq
dataSessionAborted	superviseDataSessionRes
getNotification	superviseDataSessionErr
changeNotification	dataSessionFaultDetected
	setAdviceofCharge
	setDataSessionChargePlan

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

The following clauses describe each aspect of the Data Session Control Service Capability Feature (SCF).

The order is as follows:

The Sequence diagrams give the reader a practical idea of how each of the SCF is implemented.

The Class relationships clause show how each of the interfaces applicable to the SCF, relate to one another.

The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part.

The State Transition Diagrams (STD) show the transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.

The Data Definitions clause show a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part of this specification.

4.1 General requirements on support of methods

An implementation of this API which supports or implements a method described in the present document, shall support or implement the functionality described for that method, for at least one valid set of values for the parameters of that method.

Where a method is not supported by an implementation of a Service interface, the exception P_METHOD_NOT_SUPPORTED shall be returned to any call of that method.

Where a method is not supported by an implementation of an Application interface, a call to that method shall be possible, and no exception shall be returned.

8 Data Session Control Interface Classes

The Data Session Control provides a means to control per data session basis the establishment of a new data session. This means especially in the GPRS context that the establishment of a PDP session is modelled not the attach/detach mode. Change of terminal location is assumed to be managed by the underlying network and is therefore not part of the model. The underlying assumption is that a terminal initiates a data session and the application can reject the request for data session establishment, can continue the establishment or can continue and change the destination as requested by the terminal.

The modelling is similar to the Generic Call Control but assumes a simpler underlying state model. An IpDataSessionControlManager object and an IpDataSession object are the interfaces used by the application, whereas the IpAppDataSessionControlManager and the IpAppDataSession interfaces are implemented by the application.

8.1 Interface Class IpAppDataSession

Inherits from: IpInterface.

The application side of the data session interface is used to handle data session request responses and state reports.

<<Interface>> IpAppDataSession
connectRes (dataSessionID: in TpSessionID, eventReport: in TpDataSessionReport, assignmentID: in TpAssignmentID): void connectErr (dataSessionID: in TpSessionID, errorIndication: in TpDataSessionError, assignmentID: in TpAssignmentID): void superviseDataSessionRes (dataSessionID: in TpSessionID, report: in TpDataSessionSuperviseReport, usedVolume: in TpDataSessionSuperviseVolume, qualityOfService: in TpDataSessionQosClass): void superviseDataSessionErr (dataSessionID: in TpSessionID, errorIndication: in TpDataSessionError): void dataSessionFaultDetected (dataSessionID: in TpSessionID, fault: in TpDataSessionFault): void

Method

`connectRes()`

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

Parameters

`dataSessionID: in TpSessionID`

Specifies the session ID of the data session.

`eventReport: in TpDataSessionReport`

Specifies the result of the request to connect the data session. It includes the network event, date and time, monitoring mode, negotiated quality of service and event specific information such as release cause.

assignmentID: in TpAssignmentID

Method

connectErr()

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

Parameters

dataSessionID: in TpSessionID

Specifies the session ID.

errorIndication: in TpDataSessionError

Specifies the error which led to the original request failing.

assignmentID: in TpAssignmentID

Method

superviseDataSessionRes()

This asynchronous method reports a data session supervision event to the application. In addition, it may also be used to notify the application of a newly negotiated set of Quality of Service parameters during the active life of the data session.

Parameters

dataSessionID: in TpSessionID

Specifies the data session.

report: in TpDataSessionSuperviseReport

Specifies the situation, which triggered the sending of the data session supervision response.

usedVolume: in TpDataSessionSuperviseVolume

Specifies the used volume for the data session supervision (in the same unit as specified in the request).

qualityOfService: in TpDataSessionQosClass

Specifies the newly negotiated Quality of Service parameters for the data session.

Method

superviseDataSessionErr()

This asynchronous method reports a data session supervision error to the application.

Parameters

dataSessionID: in TpSessionID

Specifies the data session ID.

errorIndication: in TpDataSessionError

Specifies the error which led to the original request failing.

Method

dataSessionFaultDetected()

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

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

Parameters

dataSessionID: in TpSessionID

Specifies the data session ID of the Data Session object in which the fault has been detected

fault: in TpDataSessionFault

Specifies the fault that has been detected.

8.2 Interface Class IpAppDataSessionControlManager

Inherits from: IpInterface.

The data session control manager application interface provides the application data session control management functions to the data session control SCF.

<<Interface>> IpAppDataSessionControlManager
dataSessionAborted (dataSession: in TpSessionID): void reportNotification (dataSessionReference: in TpDataSessionIdentifier, eventInfo: in TpDataSessionEventInfo, assignmentID: in TpAssignmentID): IpAppDataSessionRef dataSessionNotificationContinued (): void dataSessionNotificationInterrupted (): void

Method

dataSessionAborted()

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

Parameters

dataSession: in TpSessionID

Specifies the session ID of the data session that has aborted or terminated abnormally.

Method
reportNotification()

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

Returns appDataSession: Specifies a reference to the application object which implements the callback interface for the new data session.

Parameters

dataSessionReference: in TpDataSessionIdentifier

Specifies the session ID and the reference to the Data Session object to which the notification relates. This parameter will be null if the notification is being given in NOTIFY mode.

eventInfo: in TpDataSessionEventInfo

Specifies data associated with this event. This data includes the destination address provided by the end-user and the quality of service requested or negotiated for the data session.

assignmentID: in TpAssignmentID

Specifies the assignment id which was returned by the createNotification() method. The application can use assignment ID to associate events with event-specific criteria and to act accordingly.

Returns

IpAppDataSessionRef

Method
dataSessionNotificationContinued()

This method indicates to the application that all event notifications are resumed.

Parameters
No Parameters were identified for this method.

Method
dataSessionNotificationInterrupted()

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

Parameters
No Parameters were identified for this method.

8.3 Interface Class IpDataSession

Inherits from: IpService.

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

This interface shall be implemented by a Data Session Control SCF.

As a minimum requirement, the connectReq(), release(), deassignDataSession() and continueProcessing() methods shall be implemented.

<<Interface>> IpDataSession
connectReq (dataSessionID: in TpSessionID, responseRequested: in TpDataSessionReportRequestSet, targetAddress: in TpAddress): TpAssignmentID release (dataSessionID: in TpSessionID, cause: in TpDataSessionReleaseCause): void superviseDataSessionReq (dataSessionID: in TpSessionID, treatment: in TpDataSessionSuperviseTreatment, bytes: in TpDataSessionSuperviseVolume): void setDataSessionChargePlan (dataSessionID: in TpSessionID, dataSessionChargePlan: in TpDataSessionChargePlan): void setAdviceOfCharge (dataSessionID: in TpSessionID, aoCInfo: in TpAoCInfo, tariffSwitch: in TpDuration): void deassignDataSession (dataSessionID: in TpSessionID): void continueProcessing (dataSessionID: in TpSessionID): void

Method

connectReq()

This asynchronous method requests the connection of a data session with the destination party (specified in the parameter TargetAddress). The Data Session object is not automatically deleted if the destination party disconnects from the data session.

Returns assignmentID: Specifies the ID assigned to the request. The same ID will be returned in the connectRes or Err. This allows the application to correlate the request and the result.

Parameters

dataSessionID: in TpSessionID

Specifies the session ID.

responseRequested: in TpDataSessionReportRequestSet

Specifies the set of observed data session events that will result in a connectRes() being generated.

targetAddress: in TpAddress

Specifies the address of destination party.

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_ADDRESS, P_INVALID_SESSION_ID

Method

release()

This method requests the release of the data session and associated objects.

Parameters

dataSessionID: in TpSessionID

Specifies the session.

cause: in TpDataSessionReleaseCause

Specifies the cause of the release.

Raises

TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_SESSION_ID

Method

superviseDataSessionReq()

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

Parameters

dataSessionID: in TpSessionID

Specifies the data session.

treatment: in TpDataSessionSuperviseTreatment

Specifies how the network should react after the granted data volume has been sent.

bytes: in TpDataSessionSuperviseVolume

Specifies the granted number of bytes that can be transmitted for the data session.

Raises

TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_SESSION_ID

Method

setDataSessionChargePlan()

Allows an application to include charging information in network generated CDR.

Parameters

dataSessionID: in TpSessionID

Specifies the session ID of the data session.

dataSessionChargePlan: in TpDataSessionChargePlan

Specifies the charge plan used.

Raises

TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_SESSION_ID

Method

setAdviceOfCharge()

This method allows the application to determine the charging information that will be sent to the end-users terminal.

Parameters

dataSessionID: in TpSessionID

Specifies the session ID of the data session.

aoCInfo: in TpAoCInfo

Specifies two sets of Advice of Charge parameter according to GSM.

tariffSwitch: in TpDuration

Specifies the tariff switch that signifies when the second set of AoC parameters becomes valid.

Raises

TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_TIME_AND_DATE_FORMAT

Method

deassignDataSession()

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

The application should always either release or deassign the data session when it is finished with the data session, unless dataSessionFaultDetected is received by the application.

Parameters

dataSessionID: in TpSessionID

Specifies the session ID of the data session.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

Method

continueProcessing()

This operation continues processing of the data session. Applications can invoke this operation after session handling was interrupted due to detection of a notification or event the application subscribed its interest in.

Parameters

dataSessionID: in TpSessionID

Specifies the session ID of the data session.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

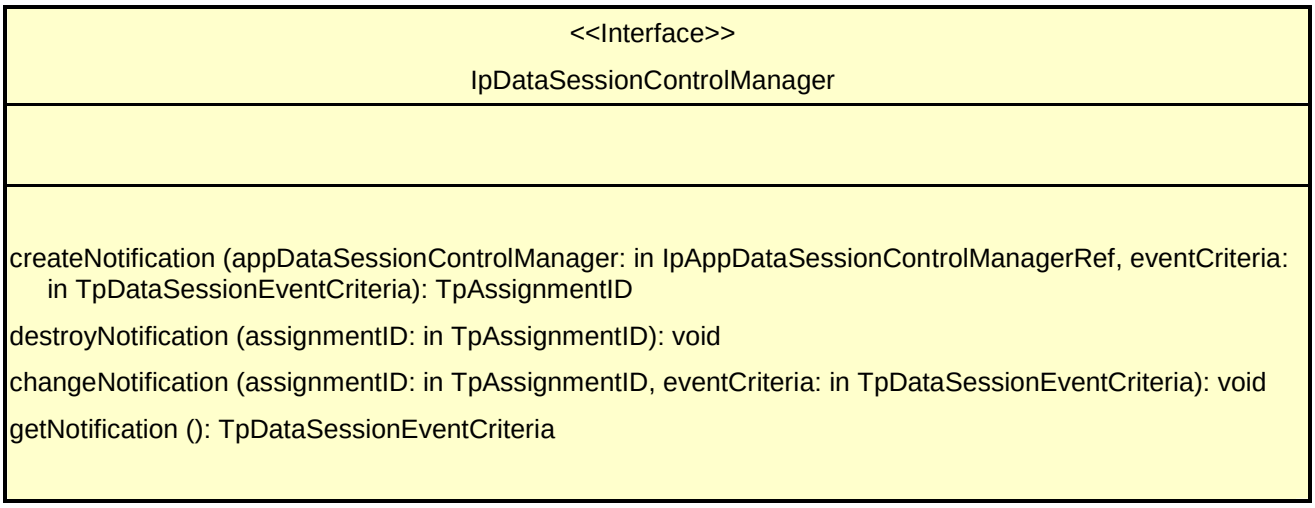
8.4 Interface Class IpDataSessionControlManager

Inherits from: IpService.

This interface is the "SCF manager" interface for Data Session Control.

This interface shall be implemented by a Data Session Control SCF.

As a minimum requirement, the createNotification() and destroyNotification() methods shall be implemented.



Method

createNotification()

This method is used to enable data session notifications so that events can be sent to the application. This is the first step an application has to do to get initial notifications of data session happening in the network. When such an event happens, the application will be informed by reportNotification(). In case the application is interested in other events during the context of a particular data session it has to use the connectReq() method on the data session object. The application will get access to the data session object when it receives the reportNotification().

The createNotification method is purely intended for applications to indicate their interest to be notified when certain data session events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a data session is setup to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria, the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the createNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().

Returns assignmentID: Specifies the ID assigned by the Data Session Manager object for this newly-enabled event notification.

Parameters

appDataSessionControlManager: in IpAppDataSessionControlManagerRef

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

eventCriteria: in TpDataSessionEventCriteria

Specifies the event specific criteria used by the application to define the event required. Individual addresses or address ranges may be specified for destination and/or origination. Examples of events are "Data Session set up".

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_ADDRESS, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE

Method

destroyNotification()

This method is used by the application to disable data session notifications.

Parameters

assignmentID: in TpAssignmentID

Specifies the assignment ID given by the data session manager object when the previous createNotification() was done.

Raises

TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID

Method

changeNotification()

This method is used by the application to change the event criteria introduced with the createNotification method. Any stored notification request associated with the specified assignmentID will be replaced with the specified events requested.

Parameters

assignmentID: in TpAssignmentID

Specifies the ID assigned by the manager interface for the event notification.

eventCriteria: in TpDataSessionEventCriteria

Specifies the new set of event criteria used by the application to define the event required. Only events that meet these criteria are reported.

Raises

TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE

Method
getNotification()

This method is used by the application to query the event criteria set with createNotification or changeNotification.

Returns eventCriteria: Specifies the event criteria used by the application to define the event required. Only events that meet these requirements are reported.

Parameters
No Parameters were identified for this method.

Returns

TpDataSessionEventCriteria

Raises

TpCommonExceptions, P_SERVICE_INFORMATION_MISSING, P_SERVICE_FAULT_ENCOUNTERED, P_INVALID_NETWORK_STATE

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-08** **CR** **021** ⌘ **rev** **-** ⌘ Current version: **5.1.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Corrections to data types in Data Session Control
Source:	⌘	N5
Work item code:	⌘	OSA2
Date:	⌘	31/10/2002
Category:	⌘	**A**
Release:	⌘	*REL-5*

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)*
Rel-6	*(Release 6)*

Reason for change:	⌘	The data definitions in TS 29.198-08 contain a number of minor errors: One of the constant values of TpDataSessionEventName is missing in the IDL and WSDL; TpDataSessionEventCriteria in the Word document has an element with an incorrect name; TpDataSessionFault in the Word document thas an element with an incorrect name
Summary of change:	⌘	Correct the Word document, the IDL and WSDL to remove these errors.
Consequences if not approved:	⌘	A contradiction will exist between the IDL, the WSDL and the Word document. If no alignment is made, some developers will chose one version of the types, others the other, and interworking problems will arise.

Clauses affected:	⌘	11.2.3, 11.2.14, Annex A, Annex B			
Other specs affected:	⌘	**Y**	**N**		
			X	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm.
Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

11.2.1 TpDataSessionEventName

Defines the names of events being notified with a new call request. The following events are supported. The values may be combined by a logical 'OR' function when requesting the notifications. Additional events that can be requested / received during the call process are found in the TpDataSessionReportType data-type.

Name	Value	Description
P_EVENT_NAME_UNDEFINED	0	Undefined
P_EVENT_DSCS_SETUP	1	The data session is going to be setup.
P_EVENT_DSCS_ESTABLISHED	2	The data session is established by the network.
P_EVENT_DSCS_QOS_CHANGED	4	A change in QoS class has taken place during the life of the data session.

11.2.2 TpDataSessionMonitorMode

Defines the mode that the call will monitor for events, or the mode that the call is in following a detected event.

Name	Value	Description
P_DATA_SESSION_MONITOR_MODE_INTERRUPT	0	The data session event is intercepted by the data session control service and data session establishment is interrupted. The application is notified of the event and data session establishment resumes following an appropriate API call or network event (such as a data session release)
P_DATA_SESSION_MONITOR_MODE_NOTIFY	1	The data session event is detected by the data session control service but not intercepted. The application is notified of the event and data session establishment continues
P_DATA_SESSION_MONITOR_MODE_DO_NOT_MONITOR	2	Do not monitor for the event

11.2.3 TpDataSessionEventCriteria

Defines the Sequence of Data Elements that specify the criteria for a event notification.

Of the addresses only the Plan and the AddrString are used for the purpose of matching the notifications against the criteria.

Sequence Element Name	Sequence Element Type	Description
DestinationAddress	TpAddressRange	Defines the destination address or address range for which the notification is requested.
Origination~~Originating~~Address	TpAddressRange	Defines the origination address or a address range for which the notification is requested.
DataSessionEventName	TpDataSessionEventName	Name of the event(s)
MonitorMode	TpDataSessionMonitorMode	Defines the mode that the Data Session is in following the notification. Monitor mode P_DATA_SESSION_MONITOR_MODE_DO_NOT_MONITOR is not a legal value here.

. . .

11.2.14 TpDataSessionFault

Defines the cause of the data session fault detected.

Name	Value	Description
P_DATA_SESSION_FAULT_UNDEFINED	0	Undefined
P_DATA_SESSION_FAULT_USER_ABORTED	1	User has finalised the data session before any message could be sent by the application
P_DATA_SESSION_TIMEOUT_ON_RELEASE	2	This fault occurs when the final report has been sent to the application, but the application did not explicitly release data session object, within a specified time. The timer value is operator specific.
P_DATA_SESSION_TIMEOUT_ON_INTERRUPT	3	This fault occurs when the 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. The timer value is operator specific.

Annex A (normative): OMG IDL Description of Data Session Control SCF

The OMG IDL representation of this interface specification is contained in a text file (dsc.idl contained in archive 2919808IDL.ZIP) which accompanies the present document.

```
typedef TpInt32 TpDataSessionEventName;

const TpInt32 P_EVENT_NAME_UNDEFINED = 0;
const TpInt32 P_EVENT_DSCS_SETUP = 1;
const TpInt32 P_EVENT_DSCS_ESTABLISHED = 2;
const TpInt32 P_EVENT_NAME_QOD_CHANGED = 4;
const TpInt32 P_EVENT_DSCS_QOS_CHANGED = 4;
```

...

```
struct TpDataSessionEventCriteria {
    TpAddressRange DestinationAddress;
    TpAddressRange OriginationAddress;
    TpDataSessionEventName DataSessionEventName;
    TpDataSessionMonitorMode MonitorMode;
};
```

...

```
enum TpDataSessionFault {
    P_DATA_SESSION_FAULT_UNDEFINED,
    P_DATA_SESSION_FAULT_USER_ABORTED,
    P_DATA_SESSION_TIMEOUT_ON_RELEASE,
    P_DATA_SESSION_TIMEOUT_ON_INTERRUPT
};
```

Annex B (informative): W3C WSDL Description of Data Session Control SCF

The W3C WSDL representation of this specification is contained in a text file (dsc.wsdl contained in archive 2919808WSDL.ZIP) which accompanies the present document.

```
<xsd:simpleType name="TpDataSessionEventName">
    <xsd:restriction base="osaxsd:TpInt32"/>
</xsd:simpleType>

<xsd:simpleType name="P_EVENT_NAME_UNDEFINED">
    <xsd:restriction base="osaxsd:TpInt32">
        <xsd:minInclusive value="0"/>
        <xsd:maxInclusive value="0"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="P_EVENT_DSCS_SETUP">
    <xsd:restriction base="osaxsd:TpInt32">
        <xsd:minInclusive value="1"/>
        <xsd:maxInclusive value="1"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="P_EVENT_DSCS_ESTABLISHED">
    <xsd:restriction base="osaxsd:TpInt32">
        <xsd:minInclusive value="2"/>
        <xsd:maxInclusive value="2"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="P_EVENT_NAME_QOD_CHANGED">
    <xsd:restriction base="osaxsd:TpInt32">
        <xsd:minInclusive value="4"/>
        <xsd:maxInclusive value="4"/>
    </xsd:restriction>
</xsd:simpleType>


<xsd:simpleType name="P_EVENT_DSCS_QOS_CHANGED">
    <xsd:restriction base="osaxsd:TpInt32">
        <xsd:minInclusive value="4"/>
        <xsd:maxInclusive value="4"/>
    </xsd:restriction>
</xsd:simpleType>
```

. . .

```
<xsd:complexType name="TpDataSessionEventCriteria">
    <xsd:sequence>
        <xsd:element name="DestinationAddress" type="osaxsd:TpAddressRange"/>
        <xsd:element name="OriginationAddress" type="osaxsd:TpAddressRange"/>
        <xsd:element name="DataSessionEventName" type="dscxsd:TpDataSessionEventName"/>
        <xsd:element name="MonitorMode" type="dscxsd:TpDataSessionMonitorMode"/>
    </xsd:sequence>
</xsd:complexType>
```

. . .

```
<xsd:simpleType name="TpDataSessionFault">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="P_DATA_SESSION_FAULT_UNDEFINED"/>
        <xsd:enumeration value="P_DATA_SESSION_FAULT_USER_ABORTED"/>
        <xsd:enumeration value="P_DATA_SESSION_TIMEOUT_ON_RELEASE"/>
        <xsd:enumeration value="P_DATA_SESSION_TIMEOUT_ON_INTERRUPT"/>
```

```
        </xsd:restriction>
    </xsd:simpleType>
```

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-08** **CR** **020** ⌘ **rev** **-** ⌘ Current version: **4.5.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network ☒

Title:	⌘	Corrections to Data Session Control Types
Source:	⌘	N5
Work item code:	⌘	OSA1
Date:	⌘	31/10/2002
Category:	⌘	**F**
Release:	⌘	*REL-4*

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)*
Rel-6	*(Release 6)*

Reason for change:	⌘	The data definitions in TS 29.198-08 contain a number of minor errors: One of the constant values of TpDataSessionEventName is missing in the IDL; TpDataSessionEventCriteria in the Word document has an element with an incorrect name; TpDataSessionFault in the Word document thas an element with an incorrect name
Summary of change:	⌘	Correct both the IDL and the Word document to remove these errors.
Consequences if not approved:	⌘	A contradiction will exist between the IDL and the Word document. If no alignment is made, some developers will chose one version of the types, others the other, and interworking problems will arise.

Clauses affected:	⌘	Annex A, 11.2.3, 11.2.14			
Other specs affected:	⌘	**Y**	**N**		
			X	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

11.2.1 TpDataSessionEventName

Defines the names of events being notified with a new call request. The following events are supported. The values may be combined by a logical 'OR' function when requesting the notifications. Additional events that can be requested / received during the call process are found in the TpDataSessionReportType data-type.

Name	Value	Description
P_EVENT_NAME_UNDEFINED	0	Undefined
P_EVENT_DSCS_SETUP	1	The data session is going to be setup.
P_EVENT_DSCS_ESTABLISHED	2	The data session is established by the network.
P_EVENT_DSCS_QOS_CHANGED	4	A change in QoS class has taken place during the life of the data session.

11.2.2 TpDataSessionMonitorMode

Defines the mode that the call will monitor for events, or the mode that the call is in following a detected event.

Name	Value	Description
P_DATA_SESSION_MONITOR_MODE_INTERRUPT	0	The data session event is intercepted by the data session control service and data session establishment is interrupted. The application is notified of the event and data session establishment resumes following an appropriate API call or network event (such as a data session release)
P_DATA_SESSION_MONITOR_MODE_NOTIFY	1	The data session event is detected by the data session control service but not intercepted. The application is notified of the event and data session establishment continues
P_DATA_SESSION_MONITOR_MODE_DO_NOT_MONITOR	2	Do not monitor for the event

11.2.3 TpDataSessionEventCriteria

Defines the Sequence of Data Elements that specify the criteria for a event notification.

Of the addresses only the Plan and the AddrString are used for the purpose of matching the notifications against the criteria.

Sequence Element Name	Sequence Element Type	Description
DestinationAddress	TpAddressRange	Defines the destination address or address range for which the notification is requested.
~~OriginatingAddress~~OriginationAddress	TpAddressRange	Defines the origination address or a address range for which the notification is requested.
DataSessionEventName	TpDataSessionEventName	Name of the event(s)
MonitorMode	TpDataSessionMonitorMode	Defines the mode that the Data Session is in following the notification. Monitor mode P_DATA_SESSION_MONITOR_MODE_DO_NOT_MONITOR is not a legal value here.

. . .

11.2.14 TpDataSessionFault

Defines the cause of the data session fault detected.

Name	Value	Description
P_DATA_SESSION_FAULT_UNDEFINED	0	Undefined
P_DATA_SESSION_FAULT_USER_ABORTED	1	User has finalised the data session before any message could be sent by the application
P_DATA_SESSION_TIMEOUT_ON_RELEASE	2	This fault occurs when the final report has been sent to the application, but the application did not explicitly release data session object, within a specified time. The timer value is operator specific.
P_DATA_SESSION_TIMEOUT_ON_INTERRUPT	3	This fault occurs when the 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. The timer value is operator specific.

Annex A (normative): OMG IDL Description of Data Session Control SCF

The OMG IDL representation of this interface specification is contained in a text file (dsc.idl contained in archive 2919808IDL.ZIP) which accompanies the present document.

```
typedef TpInt32 TpDataSessionEventName;

const TpInt32 P_EVENT_NAME_UNDEFINED = 0;
const TpInt32 P_EVENT_DSCS_SETUP = 1;
const TpInt32 P_EVENT_DSCS_ESTABLISHED = 2;
const TpInt32 P_EVENT_NAME_QOD_CHANGED = 4;
const TpInt32 P_EVENT_DSCS_QOS_CHANGED = 4;
```

...

```
struct TpDataSessionEventCriteria {
    TpAddressRange DestinationAddress;
    TpAddressRange OriginationAddress;
    TpDataSessionEventName DataSessionEventName;
    TpDataSessionMonitorMode MonitorMode;
};
```

...

```
enum TpDataSessionFault {
    P_DATA_SESSION_FAULT_UNDEFINED,
    P_DATA_SESSION_FAULT_USER_ABORTED,
    P_DATA_SESSION_TIMEOUT_ON_RELEASE,
    P_DATA_SESSION_TIMEOUT_ON_INTERRUPT
};
```

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-021024**
Meeting #21, Dublin, IRELAND, 28 – 31 October 2002

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-08** **CR** **019** ⌘ **rev** **-** ⌘ Current version: **5.1.0** ⌘

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

Proposed change affects: UICC apps⌘ ☐ ME ☐ Radio Access Network ☐ Core Network **X**

Title:	⌘	Addition of status of methods to Data Session Control interfaces
Source:	⌘	N5
Work item code:	⌘	OSA2 ***Date:*** ⌘ 31/10/2002
Category:	⌘	**A** ***Release:*** ⌘ *REL-5*

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)
Rel-6 (Release 6)

Reason for change:	⌘	There is no requirement in the standard about the necessity to implement all or only some of the methods defined for an interface.
Summary of change:	⌘	Add a statement that clarifies which methods are mandatory and which are optional.
Consequences if not approved:	⌘	Application developers will not know which methods will actually be available.

Clauses affected:	⌘	4, 8			
Other specs affected:	⌘	**Y**	**N**		⌘
			X	Other core specifications	
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

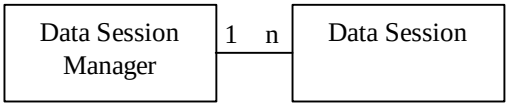
4 Data Session Control SCF

The Data Session control network service capability feature consists of two interfaces:

1) Data Session manager, containing management functions for data session related issues;

2) Data Session, containing methods to control a session.

A session can be controlled by one Data Session Manager only. Data Session Manager can control several sessions.



NOTE: The term "data session" is used in a broad sense to describe a data connection/session. For example, it comprises a PDP context in GPRS.

Figure 1: Data Session control interfaces usage relationship

The Data Session Control service capability features are described in terms of the methods in the Data Session Control interfaces. Table 1 gives an overview of the Data Session Control methods and to which interfaces these methods belong.

Table 1: Overview of Data Session Control interfaces and their methods

Data Session Manager	Data Session
createNotification	connectReq
destroyNotification	connectRes
dataSessionNotificationInterrupted	connectErr
dataSessionNotificationContinued	release
reportNotification	superviseDataSessionReq
dataSessionAborted	superviseDataSessionRes
getNotification	superviseDataSessionErr
changeNotification	dataSessionFaultDetected
enableNotifications	setAdviceofCharge
disableNotifications	setDataSessionChargePlan

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

The following clauses describe each aspect of the Data Session Control Service Capability Feature (SCF).

The order is as follows:

The Sequence diagrams give the reader a practical idea of how each of the SCF is implemented.

The Class relationships clause show how each of the interfaces applicable to the SCF, relate to one another.

The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part.

The State Transition Diagrams (STD) show the transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.

The Data Definitions clause show a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part of this specification.

4.1 General requirements on support of methods

An implementation of this API which supports or implements a method described in the present document, shall support or implement the functionality described for that method, for at least one valid set of values for the parameters of that method.

Where a method is not supported by an implementation of a Service interface, the exception P_METHOD_NOT_SUPPORTED shall be returned to any call of that method.

Where a method is not supported by an implementation of an Application interface, a call to that method shall be possible, and no exception shall be returned.

8 Data Session Control Interface Classes

The Data Session Control provides a means to control per data session basis the establishment of a new data session. This means especially in the GPRS context that the establishment of a PDP session is modelled not the attach/detach mode. Change of terminal location is assumed to be managed by the underlying network and is therefore not part of the model. The underlying assumption is that a terminal initiates a data session and the application can reject the request for data session establishment, can continue the establishment or can continue and change the destination as requested by the terminal.

The modelling is similar to the Generic Call Control but assumes a simpler underlying state model. An IpDataSessionControlManager object and an IpDataSession object are the interfaces used by the application, whereas the IpAppDataSessionControlManager and the IpAppDataSession interfaces are implemented by the application.

8.1 Interface Class IpAppDataSession

Inherits from: IpInterface.

The application side of the data session interface is used to handle data session request responses and state reports.

<<Interface>> IpAppDataSession
connectRes (dataSessionID : in TpSessionID, eventReport : in TpDataSessionReport, assignmentID : in TpAssignmentID) : void connectErr (dataSessionID : in TpSessionID, errorIndication : in TpDataSessionError, assignmentID : in TpAssignmentID) : void superviseDataSessionRes (dataSessionID : in TpSessionID, report : in TpDataSessionSuperviseReport, usedVolume : in TpDataSessionSuperviseVolume, qualityOfService : in TpDataSessionQosClass) : void superviseDataSessionErr (dataSessionID : in TpSessionID, errorIndication : in TpDataSessionError) : void dataSessionFaultDetected (dataSessionID : in TpSessionID, fault : in TpDataSessionFault) : void

8.1.1 Method connectRes()

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

Parameters

`dataSessionID : in TpSessionID`

Specifies the session ID of the data session.

`eventReport : in TpDataSessionReport`

Specifies the result of the request to connect the data session. It includes the network event, date and time, monitoring mode, negotiated quality of service and event specific information such as release cause.

`assignmentID : in TpAssignmentID`

8.1.2 Method connectErr()

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

Parameters

dataSessionID : in TpSessionID

Specifies the session ID.

errorIndication : in TpDataSessionError

Specifies the error which led to the original request failing.

assignmentID : in TpAssignmentID

8.1.3 Method superviseDataSessionRes()

This asynchronous method reports a data session supervision event to the application. In addition, it may also be used to notify the application of a newly negotiated set of Quality of Service parameters during the active life of the data session.

Parameters

dataSessionID : in TpSessionID

Specifies the data session.

report : in TpDataSessionSuperviseReport

Specifies the situation, which triggered the sending of the data session supervision response.

usedVolume : in TpDataSessionSuperviseVolume

Specifies the used volume for the data session supervision (in the same unit as specified in the request).

qualityOfService : in TpDataSessionQosClass

Specifies the newly negotiated Quality of Service parameters for the data session.

8.1.4 Method superviseDataSessionErr()

This asynchronous method reports a data session supervision error to the application.

Parameters

dataSessionID : in TpSessionID

Specifies the data session ID.

errorIndication : in TpDataSessionError

Specifies the error which led to the original request failing.

8.1.5 Method dataSessionFaultDetected()

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

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

Parameters

`dataSessionID : in TpSessionID`

Specifies the data session ID of the Data Session object in which the fault has been detected

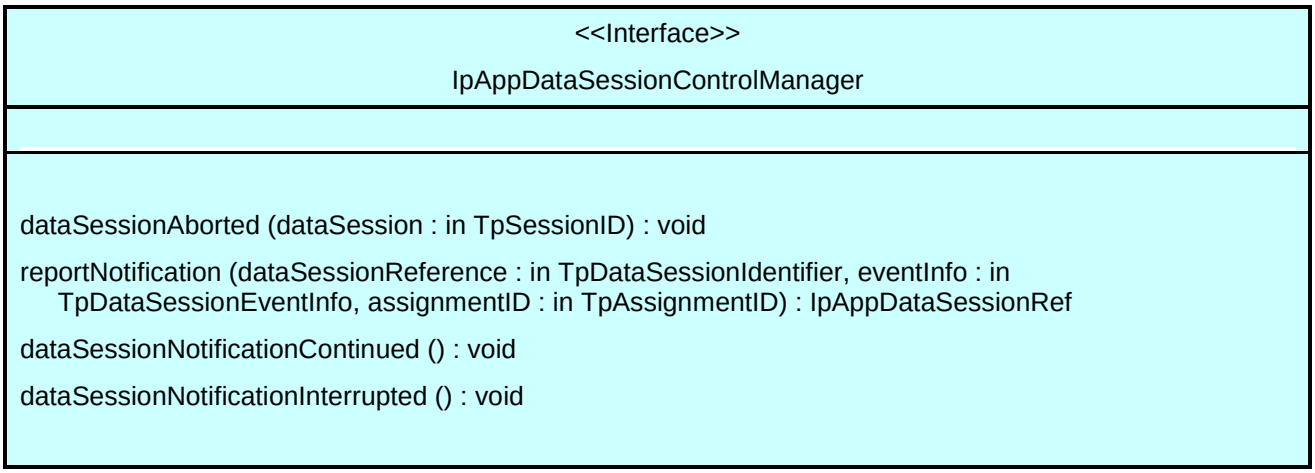
`fault : in TpDataSessionFault`

Specifies the fault that has been detected.

8.2 Interface Class IpAppDataSessionControlManager

Inherits from: IpInterface.

The data session control manager application interface provides the application data session control management functions to the data session control SCF.



8.2.1 Method dataSessionAborted()

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

Parameters

`dataSession : in TpSessionID`

Specifies the session ID of the data session that has aborted or terminated abnormally.

8.2.2 Method reportNotification()

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

If this method is invoked with a monitor mode of P_DATA_SESSION_MONITOR_MODE_INTERRUPT, then the application has control of the data session. If the application does nothing with the data session within a specified time period (the duration of which forms a part of the service level agreement), then the data session in the network shall be released and dataSessionFaultDetected() shall be invoked, giving a fault code of P_DATA_SESSION_TIMEOUT_ON_INTERRUPT.

Returns appDataSession : Specifies a reference to the application object which implements the callback interface for the new data session. If the application has previously explicitly passed a reference to the IpAppDataSession interface using a setCallback() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallback().

This parameter will be null if the notification is in NOTIFY mode.

Parameters

`dataSessionReference : in TpDataSessionIdentifier`

Specifies the session ID and the reference to the Data Session object to which the notification relates. If the notification is being given in NOTIFY mode, this parameter shall be ignored by the application client implementation, and consequently the implementation of the SCS entity invoking reportNotification may populate this parameter as it chooses.

`eventInfo : in TpDataSessionEventInfo`

Specifies data associated with this event. This data includes the destination address provided by the end-user and the quality of service requested or negotiated for the data session.

`assignmentID : in TpAssignmentID`

Specifies the assignment id which was returned by the createNotification() method. The application can use assignment ID to associate events with event-specific criteria and to act accordingly.

Returns

`IpAppDataSessionRef`

8.2.3 Method dataSessionNotificationContinued()

This method indicates to the application that all event notifications are resumed.

Parameters

No Parameters were identified for this method

8.2.4 Method dataSessionNotificationInterrupted()

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

Parameters

No Parameters were identified for this method

8.3 Interface Class IpDataSession

Inherits from: IpService.

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

This interface shall be implemented by a Data Session Control SCF.

As a minimum requirement, the connectReq(), release(), deassignDataSession() and continueProcessing() methods shall be implemented.

<<Interface>> IpDataSession
connectReq (dataSessionID : in TpSessionID, responseRequested : in TpDataSessionReportRequestSet, targetAddress : in TpAddress) : TpAssignmentID release (dataSessionID : in TpSessionID, cause : in TpDataSessionReleaseCause) : void superviseDataSessionReq (dataSessionID : in TpSessionID, treatment : in TpDataSessionSuperviseTreatment, bytes : in TpDataSessionSuperviseVolume) : void setDataSessionChargePlan (dataSessionID : in TpSessionID, dataSessionChargePlan : in TpDataSessionChargePlan) : void setAdviceOfCharge (dataSessionID : in TpSessionID, aoCInfo : in TpAoCInfo, tariffSwitch : in TpDuration) : void deassignDataSession (dataSessionID : in TpSessionID) : void continueProcessing (dataSessionID : in TpSessionID) : void

8.3.1 Method connectReq()

This asynchronous method requests the connection of a data session with the destination party (specified in the parameter TargetAddress). The Data Session object is not automatically deleted if the destination party disconnects from the data session.

Returns assignmentID : Specifies the ID assigned to the request. The same ID will be returned in the connectRes or Err. This allows the application to correlate the request and the result.

Parameters

`dataSessionID : in TpSessionID`

Specifies the session ID.

`responseRequested : in TpDataSessionReportRequestSet`

Specifies the set of observed data session events that will result in a connectRes() being generated.

`targetAddress : in TpAddress`

Specifies the address of destination party.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_ADDRESS, P_INVALID_SESSION_ID`

8.3.2 Method release()

This method requests the release of the data session and associated objects.

Parameters

dataSessionID : in TpSessionID

Specifies the session.

cause : in TpDataSessionReleaseCause

Specifies the cause of the release.

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_SESSION_ID

8.3.3 Method superviseDataSessionReq()

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

Parameters

dataSessionID : in TpSessionID

Specifies the data session.

treatment : in TpDataSessionSuperviseTreatment

Specifies how the network should react after the granted data volume has been sent.

bytes : in TpDataSessionSuperviseVolume

Specifies the granted number of bytes that can be transmitted for the data session.

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_SESSION_ID

8.3.4 Method setDataSessionChargePlan()

Allows an application to include charging information in network generated CDR.

Parameters

dataSessionID : in TpSessionID

Specifies the session ID of the data session.

dataSessionChargePlan : in TpDataSessionChargePlan

Specifies the charge plan used.

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_SESSION_ID

8.3.5 Method setAdviceOfCharge()

This method allows the application to determine the charging information that will be sent to the end-users terminal.

Parameters

dataSessionID : in TpSessionID

Specifies the session ID of the data session.

aoCInfo : in TpAoCInfo

Specifies two sets of Advice of Charge parameter according to GSM.

tariffSwitch : in TpDuration

Specifies the tariff switch that signifies when the second set of AoC parameters becomes valid.

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_TIME_AND_DATE_FORMAT

8.3.6 Method deassignDataSession()

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

The application should always either release or deassign the data session when it is finished with the data session, unless dataSessionFaultDetected is received by the application.

Parameters

dataSessionID : in TpSessionID

Specifies the session ID of the data session.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

8.3.7 Method continueProcessing()

This operation continues processing of the data session. Applications can invoke this operation after session handling was interrupted due to detection of a notification or event the application subscribed its interest in.

Parameters

dataSessionID : in TpSessionID

Specifies the session ID of the data session.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_NETWORK_STATE

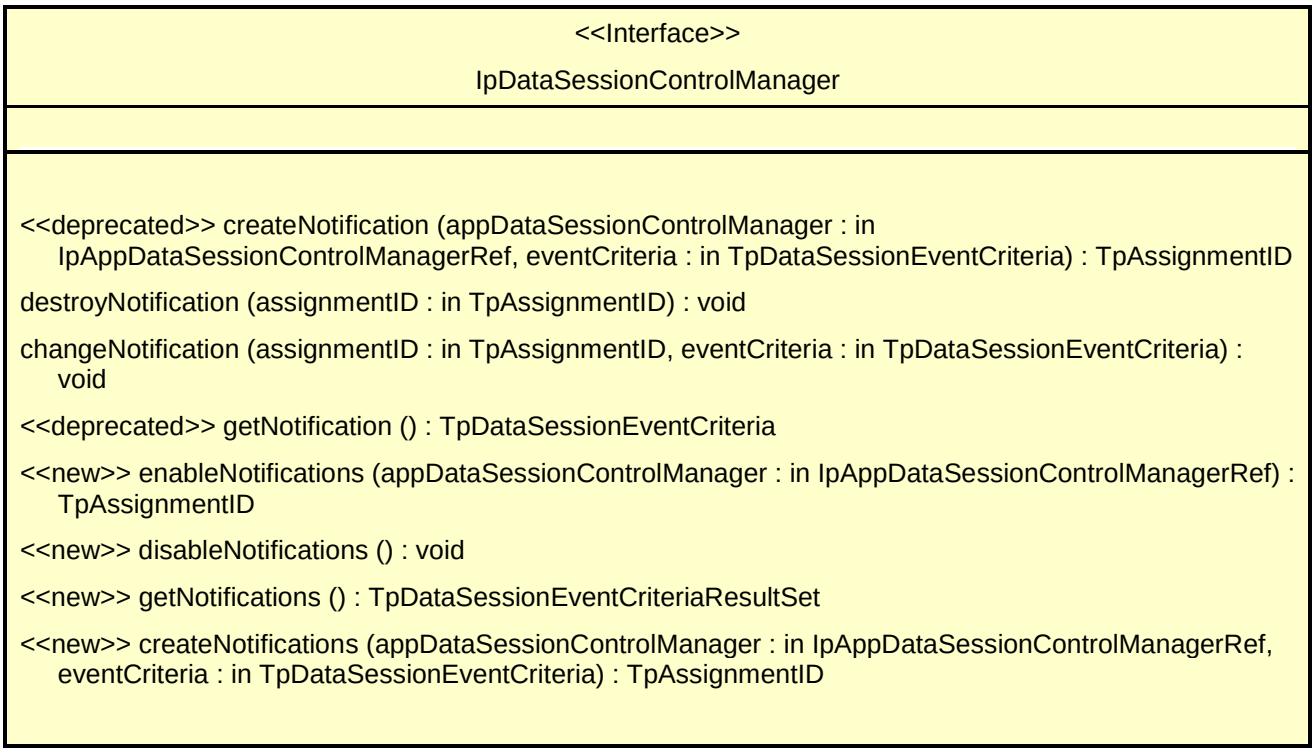
8.4 Interface Class IpDataSessionControlManager

Inherits from: IpService.

This interface is the 'SCF manager' interface for Data Session Control.

This interface shall be implemented by a Data Session Control SCF.

As a minimum requirement, the createNotifications() and destroyNotification(), or the enableNotifications() and disableNotifications() methods shall be implemented.



8.4.1 Method <<deprecated>> createNotification()

This method is deprecated and will be removed in a later release. It is replaced with createNotifications().

This method is used to enable data session notifications so that events can be sent to the application. This is the first step an application has to do to get initial notifications of data session happening in the network. When such an event happens, the application will be informed by reportNotification(). In case the application is interested in other events during the context of a particular data session it has to use the connectReq() method on the data session object. The application will get access to the data session object when it receives the reportNotification().

The createNotification method is purely intended for applications to indicate their interest to be notified when certain data session events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a data session is setup to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used.

If a notification is requested by an application with monitor mode set to notify, then there is no need to check the rest of the criteria for overlapping with any existing request as the notify mode does not give control of a data session. Only one application can place an interrupt request if the criteria overlaps.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the createNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().

Returns assignmentID : Specifies the ID assigned by the Data Session Manager object for this newly-enabled event notification.

Parameters

appDataSessionControlManager : in IpAppDataSessionControlManagerRef

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

eventCriteria : in TpDataSessionEventCriteria

Specifies the event specific criteria used by the application to define the event required. Individual addresses or address ranges may be specified for destination and/or origination. Examples of events are "Data Session set up".

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE

8.4.2 Method destroyNotification()

This method is used by the application to disable data session notifications. This method only applies to notifications created with createNotification().

Parameters

assignmentID : in TpAssignmentID

Specifies the assignment ID given by the data session manager object when the previous createNotification() was done.

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID

8.4.3 Method changeNotification()

This method is used by the application to change the event criteria introduced with the createNotification method. Any stored notification request associated with the specified assignmentID will be replaced with the specified events requested.

Parameters

assignmentID : in TpAssignmentID

Specifies the ID assigned by the manager interface for the event notification.

eventCriteria : in TpDataSessionEventCriteria

Specifies the new set of event criteria used by the application to define the event required. Only events that meet these criteria are reported.

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE

8.4.4 Method <<deprecated>> getNotification()

This method is deprecated and its use is discouraged. It will be removed in a later release. It is replaced with getNotifications.

This method is used by the application to query the event criteria set with createNotification or changeNotification.

Returns eventCriteria : Specifies the event criteria used by the application to define the event required. Only events that meet these requirements are reported.

Parameters

No Parameters were identified for this method

Returns

TpDataSessionEventCriteria

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE

8.4.5 Method <<new>> enableNotifications()

This method is used to indicate that the application is able to receive which are provisioned from within the network (i.e. these notifications are NOT set using createNotification() but via, for instance, a network management system). If notifications provisioned for this application are created or changed, the application is unaware of this until the notification is reported.

If the same application requests to enable notifications for a second time with a different IpAppDataSessionControlManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. This means that the callback wil only be used in cases when the first callback specified by the application is unable to handle the callEventNotify (e.g. due to overload or failure).

When this method is used, it is still possible to use createNotification() for service provider provisioned notifications on the same interface as long as the criteria in the network and provided by createNotification() do not overlap. However, it is NOT recommended to use both mechanisms on the same service manager.

The methods changeNotification(), getNotification(), and destroyNotification() do not apply to notifications provisoned in the network and enabled using enableNotifications(). These only apply to notifications created using createNotification().

Returns assignmentID: Specifies the ID assigned by the manager interface for this operation. This ID is contained in any reportNotification() that relates to notifications provisioned from within the network. Repeated calls to enableNotifications() return the same assignment ID.

Parameters

appDataSessionControlManager : in IpAppDataSessionControlManagerRef

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface, which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

Returns

TpAssignmentID

Raises

TpCommonExceptions

8.4.6 Method <<new>> disableNotifications()

This method is used to indicate that the application is not able to receive notifications for which the provisioning has been done from within the network. (i.e. these notifications that are NOT set using createNotification() but via, for instance, a network management system). After this method is called, no such notifications are reported anymore.

Parameters

No Parameters were identified for this method

Raises

TpCommonExceptions

8.4.7 Method <<new>> getNotifications()

This method replaces getNotification()

This method is used by the application to query the event criteria set with createNotification or changeNotification.

Returns eventCriteria: the list of event criteria for the notifications requested by the application. If there is no information to return (e.g. no notifications requested by the application), an empty set (zero length) is returned.

Parameters

No Parameters were identified for this method

Returns

TpDataSessionEventCriteriaResultSet

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE

8.4.8 Method <<new>> createNotifications()

This method is deprecated and will be removed in a later release. It is replaced with createNotifications().

This method is used to enable data session notifications so that events can be sent to the application. This is the first step an application has to do to get initial notifications of data session happening in the network. When such an event happens, the application will be informed by reportNotification(). In case the application is interested in other events during the context of a particular data session it has to use the connectReq() method on the data session object. The application will get access to the data session object when it receives the reportNotification().

The createNotification method is purely intended for applications to indicate their interest to be notified when certain data session events take place. It is possible to subscribe to a certain event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a data session is setup to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used.

If a notification is requested by an application with monitor mode set to notify, then there is no need to check the rest of the criteria for overlapping with any existing request as the notify mode does not give control of a data session. Only one application can place an interrupt request if the criteria overlaps.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the createNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().

Returns assignmentID : Specifies the ID assigned by the Data Session Manager object for this newly-enabled event notification.

Parameters

appDataSessionControlManager : in IpAppDataSessionControlManagerRef

If this parameter is set (i.e. not NULL) it specifies a reference to the application interface which is used for callbacks. If set to NULL, the application interface defaults to the interface specified via the setCallback() method.

eventCriteria : in TpDataSessionEventCriteria

Specifies the event specific criteria used by the application to define the event required. Individual addresses or address ranges may be specified for destination and/or origination. Examples of events are "Data Session set up".

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE, P_INVALID_INTERFACE_TYPE