3GPP TSG CN Plenary Meeting #16 **NP-020179**
5th - 7th June 2002. Marco Island, USA.

Source: **CN5 (OSA)**

Title: **Rel-4 CRs 29.198-03 OSA API Part 3: Framework**

Agenda item: **7.10**

Document for: **APPROVAL**

Doc-1st -Level	Spec	CR	Rev	Pha	Subject	Cat	Ver Curr	Ver New	Doc-2nd -Level	Work item
NP-020179	29.198-03	030	-	Rel-4	Solving the problem in the OSA Framework with method appUnavailableInd() in a scenario with multiple service sessions per access session	F	4.4.0	4.5.0	N5-020471	OSA1
NP-020179	29.198-03	031	-	Rel-4	Adding missing mandatory method (authenticationSucceeded) to sequence flow	F	4.4.0	4.5.0	N5-020494	OSA1

joint API group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-020471**
Meeting #18, Budapest, HUNGARY, 13 – 17 May 2002

CR-Form-v5

CHANGE REQUEST

⌘ **29.198-03** **CR** **030** ⌘ **rev** **-** ⌘ Current version: **4.4.0** ⌘

*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:	⌘	Solving the problem in the OSA Framework with method appUnavailableInd() in a scenario with multiple service sessions per access session
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	17/05/2002
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)

Reason for change:	⌘	Since the method call IpAppFaultManager:appUnavialableInd() does not pass any parameters to the client application, there is no way for a client (who has multiple service sessions) to determine which service session is in jeopardy. If a client application signs two or more service agreements (different services) using the same access session, when the framework calls the clients appUnavailableInd() method, the client will not know which service is at risk.
Summary of change:	⌘	Introduce a new serviceID parameter for appUnavailableInd().
Consequences if not approved:	⌘	When invoking appUnavailableInd(), there is no way for a client (who has multiple service sessions) to determine which service session is in jeopardy

Clauses affected:	⌘	7.3.3.1
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ 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/3G_Specs/CRs.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.

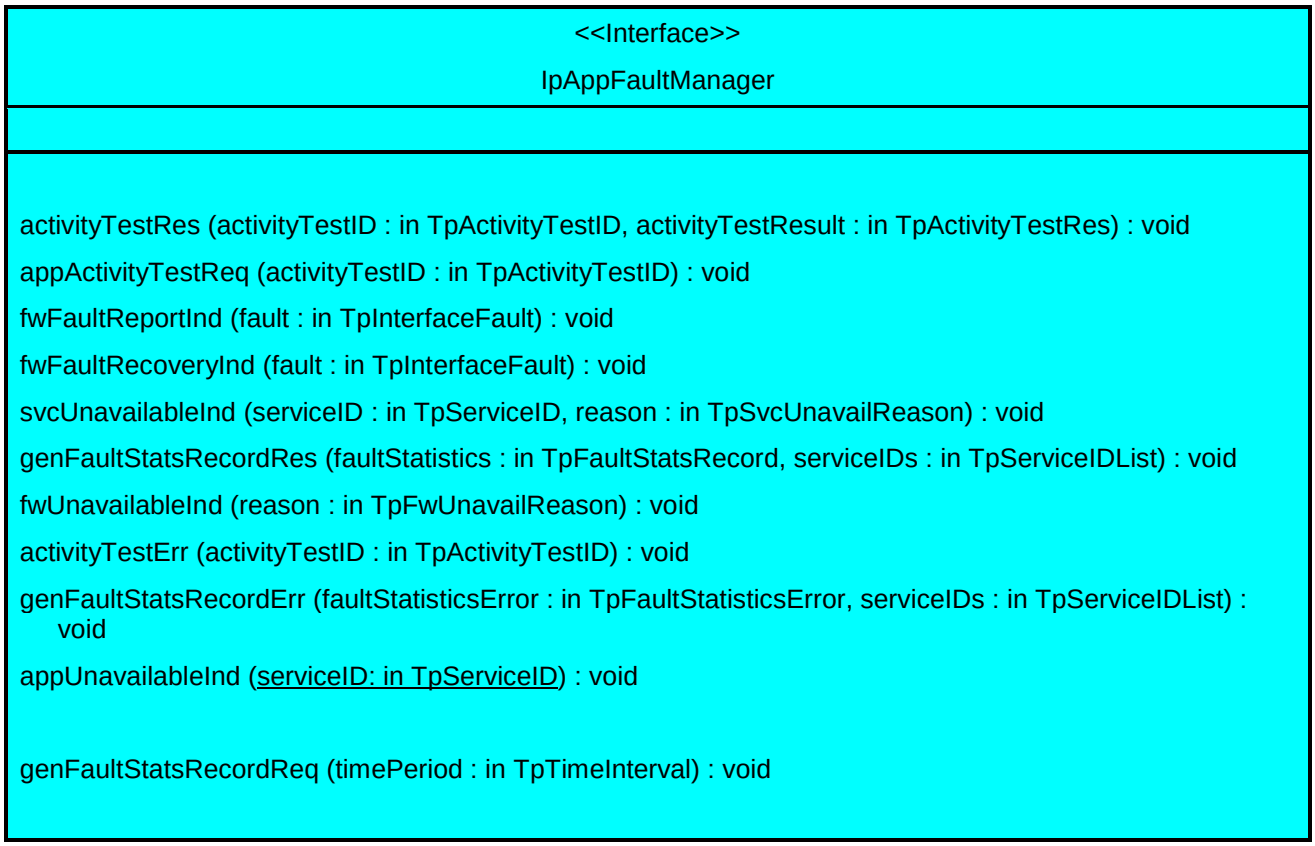
3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

7.3.3 Integrity Management Interface Classes

7.3.3.1 Interface Class IpAppFaultManager

Inherits from: IpInterface.

This interface is used to inform the application of events that affect the integrity of the Framework, Service or Client Application. The Fault Management Framework will invoke methods on the Fault Management Application Interface that is specified when the client application obtains the Fault Management interface: i.e. by use of the obtainInterfaceWithCallback operation on the IpAccess interface



Method

activityTestRes()

The framework uses this method to return the result of a client application-requested activity test.

Parameters

activityTestID : in TpActivityTestID

Used by the client application to correlate this response (when it arrives) with the original request.

activityTestResult : in TpActivityTestRes

The result of the activity test.

Method
appActivityTestReq()

The framework invokes this method to test that the client application is operational. On receipt of this request, the application must carry out a test on itself, to check that it is operating correctly. The application reports the test result by invoking the appActivityTestRes method on the IpFaultManager interface.

Parameters

activityTestID : in TpActivityTestID

The identifier provided by the framework to correlate the response (when it arrives) with this request.

Method
fwFaultReportInd()

The framework invokes this method to notify the client application of a failure within the framework. The client application must not continue to use the framework until it has recovered (as indicated by a fwFaultRecoveryInd).

Parameters

fault : in TpInterfaceFault

Specifies the fault that has been detected by the framework.

Method
fwFaultRecoveryInd()

The framework invokes this method to notify the client application that a previously reported fault has been rectified. The application may then resume using the framework.

Parameters

fault : in TpInterfaceFault

Specifies the fault from which the framework has recovered.

Method
svcUnavailableInd()

The framework invokes this method to inform the client application that it can no longer use its instance of the indicated service. On receipt of this request, the client application must act to reset its use of the specified service (using the normal mechanisms, such as the discovery and authentication interfaces, to stop use of this service instance and begin use of a different service instance).

Parameters

serviceID : in TpServiceID

Identifies the affected service.

reason : in TpSvcUnavailReason

Identifies the reason why the service is no longer available

Method
genFaultStatsRecordRes()

This method is used by the framework to provide fault statistics to a client application in response to a genFaultStatsRecordReq method invocation on the IpFaultManager interface.

Parameters

faultStatistics : in TpFaultStatsRecord

The fault statistics record.

serviceIDs : in TpServiceIDList

Specifies the framework or services that are included in the general fault statistics record. If the serviceIDs parameter is an empty list, then the fault statistics are for the framework.

Method
fwUnavailableInd()

The framework invokes this method to inform the client application that it is no longer available.

Parameters

reason : in TpFwUnavailReason

Identifies the reason why the framework is no longer available

Method
activityTestErr()

The framework uses this method to indicate that an error occurred during an application-initiated activity test.

Parameters

activityTestID : in TpActivityTestID

Used by the application to correlate this response (when it arrives) with the original request.

Method
genFaultStatsRecordErr()

This method is used by the framework to indicate an error fulfilling the request to provide fault statistics, in response to a genFaultStatsRecordReq method invocation on the IpFaultManager interface.

Parameters

faultStatisticsError : in TpFaultStatisticsError

The fault statistics error.

serviceIDs : in TpServiceIDList

Specifies the framework or services that were included in the general fault statistics record request. If the serviceIDs parameter is an empty list, then the fault statistics were requested for the framework.

Method
appUnavailableInd()

The framework invokes this method to indicate to the application that the service instance has detected that it is not responding. On receipt of this indication, the application must end its current session with the service instance.

Parameters

serviceID : in TpServiceID

Specifies the service for which the indication of unavailability was received.

~~No Parameters were identified for this method~~

Method

genFaultStatsRecordReq()

This method is used by the framework to solicit fault statistics from the client application, for example when the framework was asked for these statistics by a service instance by using the genFaultStatsRecordReq operation on the IpFwFaultManager interface. On receipt of this request, the client application must produce a fault statistics record, for the application during the specified time interval, which is returned to the framework using the genFaultStatsRecordRes operation on the IpFaultManager interface.

Parameters

timePeriod : in TpTimeInterval

The period over which the fault statistics are to be generated. A null value leaves this to the discretion of the client application.

CR-Form-v5

CHANGE REQUEST

⌘ **29.198-03** **CR** **031** ⌘ **rev** **-** ⌘ Current version: **4.4.0** ⌘

*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:	⌘	Adding missing mandatory method (authenticationSucceeded) to sequence flow
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	30/05/2002
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)

Reason for change:	⌘	authenticationSucceeded is mandatory but missing from a sequence diagram
Summary of change:	⌘	Add authenticationSucceeded to 6.1.1.4. It is a mandatory method and is currently missing from the call flow, meaning that readers may get confused about the actual flow of events required for successful authentication.
Consequences if not approved:	⌘	A misleading specification leading to possible implementation interoperability issues.

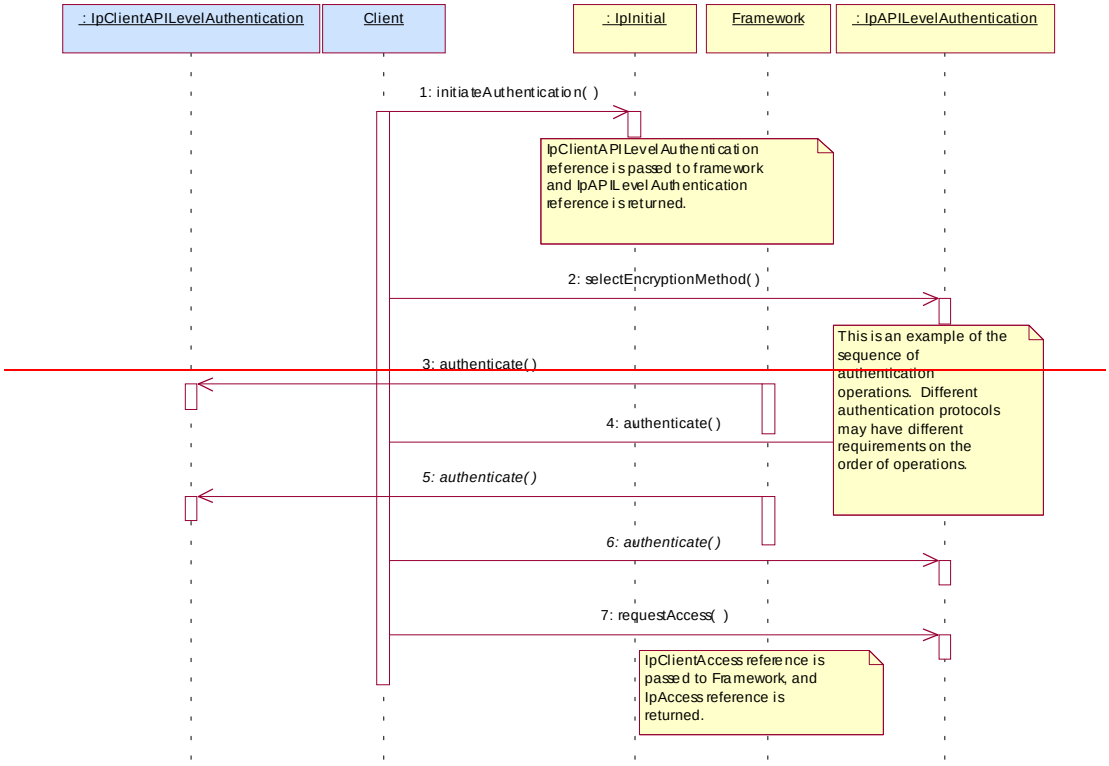
Clauses affected:	⌘	6.1.1.4
Other specs affected:	⌘	☐ Other core specifications ⌘ ☐ Test specifications ☐ O&M Specifications
Other comments:	⌘	

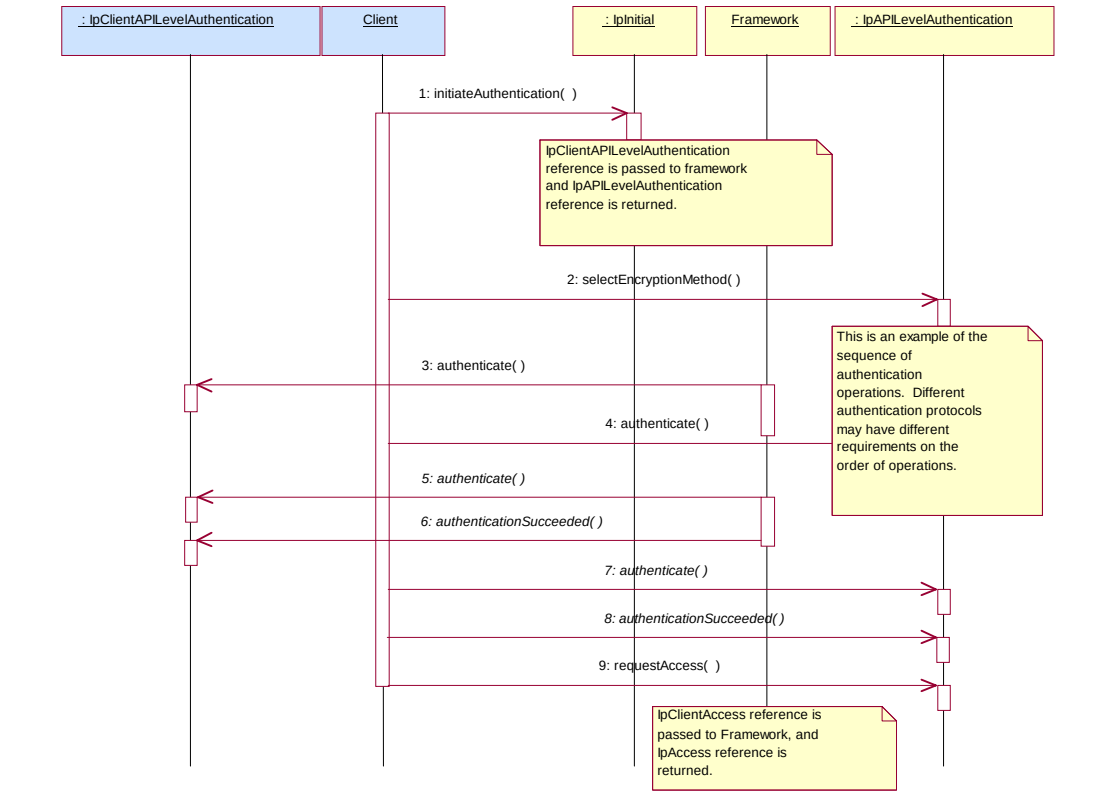
How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at: http://www.3gpp.org/3G_Specs/CRs.htm. Below is a brief summary:

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.

6.1.1.4 API Level Authentication

This sequence diagram illustrates the two-way mechanism by which the client and the framework mutually authenticate one another.
The OSA API supports multiple authentication techniques. The procedure used to select an appropriate technique for a given situation is described below. The authentication mechanisms may be supported by cryptographic processes to provide confidentiality, and by digital signatures to ensure integrity. The inclusion of cryptographic processes and digital signatures in the authentication procedure depends on the type of authentication technique selected. In some cases strong authentication may need to be enforced by the Framework to prevent misuse of resources. In addition it may be necessary to define the minimum encryption key length that can be used to ensure a high degree of confidentiality.
The client must authenticate with the Framework before it is able to use any of the other interfaces supported by the Framework. Invocations on other interfaces will fail until authentication has been successfully completed.
1) The client calls initiateAuthentication on the OSA Framework Initial interface. This allows the client to specify the type of authentication process. This authentication process may be specific to the provider, or the implementation technology used. The initiateAuthentication method can be used to specify the specific process, (e.g. CORBA security). OSA defines a generic authentication interface (API Level Authentication), which can be used to perform the authentication process. The initiateAuthentication method allows the client to pass a reference to its own authentication interface to the Framework, and receive a reference to the authentication interface preferred by the client, in return. In this case the API Level Authentication interface.
2) The client invokes the selectEncryptionMethod on the Framework's API Level Authentication interface. This includes the encryption capabilities of the client. The framework then chooses an encryption method based on the encryption capabilities of the client and the Framework. If the client is capable of handling more than one encryption method, then the Framework chooses one option, defined in the prescribedMethod parameter. In some instances, the encryption capability of the client may not fulfil the demands of the Framework, in which case, the authentication will fail.
3) The application and Framework interact to authenticate each other. For an authentication method of P_OSA_AUTHENTICATION, this procedure consists of a number of challenge/ response exchanges. This authentication protocol is performed using the authenticate method on the API Level Authentication interface. P_OSA_AUTHENTICATION is based on CHAP, which is primarily a one-way protocol. Mutual authentication is achieved by the framework invoking the authenticate method on the client's APILevelAuthentication interface.
Note that at any point during the access session, either side can request re-authentication. Re-authentication does not have to be mutual.


: IpClientAPILevelAuthentication
Client
: IpInitial
Framework
: IpAPILevelAuthentication
1: initiateAuthentication( )
IpClientAPILevelAuthentication reference is passed to framework and IpAPILevelAuthentication reference is returned.
2: selectEncryptionMethod( )
This is an example of the sequence of authentication operations. Different authentication protocols may have different requirements on the order of operations.
3: authenticate( )
4: authenticate( )
5: authenticate( )
6: authenticate( )
7: requestAccess( )
IpClientAccess reference is passed to Framework, and IpAccess reference is returned.


: IpClientAPILevelAuthentication
Client
: IpInitial
Framework
: IpAPILevelAuthentication
1: initiateAuthentication( )
IpClientAPILevelAuthentication reference is passed to framework and IpAPILevelAuthentication reference is returned.
2: selectEncryptionMethod( )
This is an example of the sequence of authentication operations. Different authentication protocols may have different requirements on the order of operations.
3: authenticate( )
4: authenticate( )
5: authenticate( )
6: authenticationSucceeded( )
7: authenticate( )
8: authenticationSucceeded( )
9: requestAccess( )
IpClientAccess reference is passed to Framework, and IpAccess reference is returned.