3GPP TSG CN Plenary Meeting #22 **NP-030549**
10 - 12 December 2003, Maui, Hawaii, USA

Source: **CN5 (OSA)**

Title: **Rel-5 CRs 29.198-03 OSA API Part 3: Framework**

Agenda item: **8.2**

Document for: **APPROVAL**

Doc-1st-Level	Spec	CR	R	Ph	Subject	Cat	Version-Current	Doc-2nd-Lev	WI
NP-030549	29.198-03	086	-	Rel-5	Correction of the sequence diagram for Fault Management	F	5.4.0	N5-030549	OSA2
NP-030549	29.198-03	087	-	Rel-5	Correction of State Transition Diagram for IpAccess	F	5.4.0	N5-030572	OSA2
NP-030549	29.198-03	088	-	Rel-5	Correction of Correlation Behaviour in Load Management	F	5.4.0	N5-030575	OSA2
NP-030549	29.198-03	089	-	Rel-5	Correction of Correlation Behaviour in Fault Management	F	5.4.0	N5-030632	OSA2
NP-030549	29.198-03	090	-	Rel-5	Correction and Clarification of Framework Access Session Behaviour	F	5.4.0	N5-030633	OSA2

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03** **CR** **086** ⌘ **rev** **-** ⌘ Current version: **5.4.0** ⌘

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

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

Title:	⌘	Correction of the sequence diagram for Fault Management
Source:	⌘	CN5 (Lucent - Musa Unmehopa)
Work item code:	⌘	OSA2
Date:	⌘	31/10/2003
Category:	⌘	**F**
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:	⌘	Correct the sequence diagram for Fault Management, to be in line with method definitions. Per the current sequence diagram 8.1.4.8, the FWK should use appUnavailableInd() on the service after the client calls svcUnavailableInd() on the FWK. When should the FWK use svcUnavailableInd() on the service? Correct behaviour should be the following: the FWK calls svcUnavailableInd() on the service after a client calls svcUnavailableInd() on the FWK. In addition, when the client calls appUnavailableInd() on the FWK, the FWK calls appUnavailableInd() on the service. The sequence diagrams needs to be corrected accordingly.
Summary of change:	⌘	Replace “appUnavailStatusInd()” method with correct “svcUnavailInd()” method.
Consequences if not approved:	⌘	Sequence diagram does not correctly represent the API functionality. Application programmers typically code to the sequence diagrams, resulting in incorrect implementations.

Clauses affected:	⌘	8.1.4.8

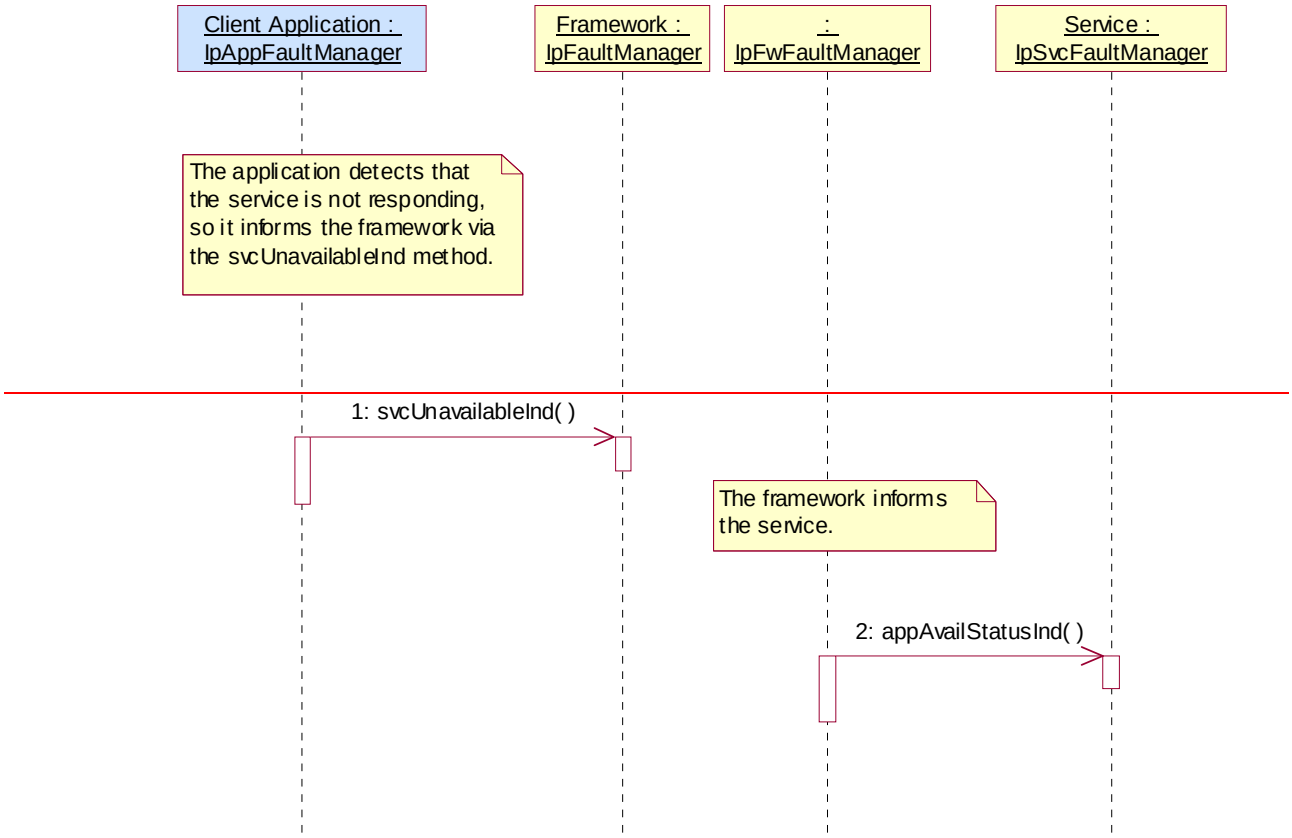
Other specs affected: ⌘	Y	N		⌘
		X	Other core specifications	
		X	Test specifications	
		X	O&M Specifications	

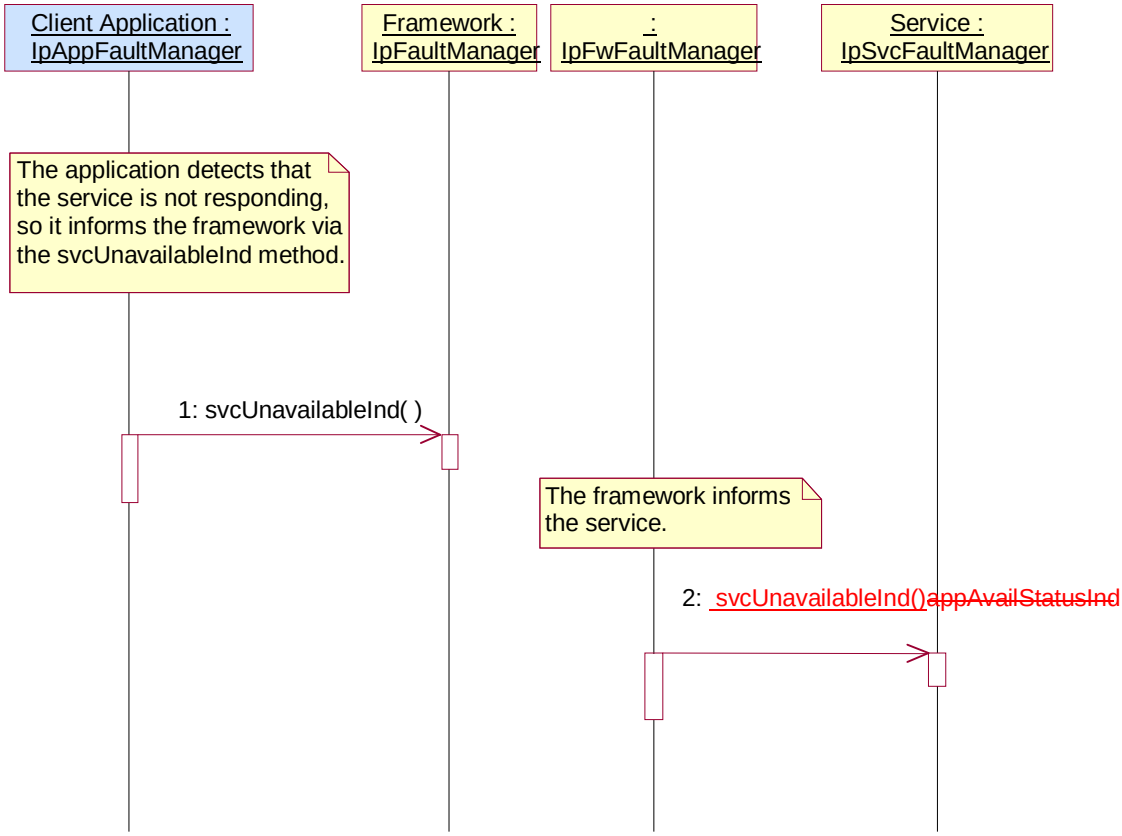
Other comments: ⌘

How to create CRs using this form:

Change in Clause 8.1.4.8

8.1.4.8 Fault Management: Application detects service is unavailable





1: The client application detects that the service instance is currently not available, i.e. the service instance is not responding to the client application in the normal way, so it informs the framework.

2: The framework informs the service instance that the client application was unable to get a response from it and can no longer use the service instance. The service or framework may then decide to carry out an activity test to see whether there is a general problem with the service instance that requires further action.

End of Change in 8.1.4.8
End of Document

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2003	CN_19	NP-030028	076	--	Remove race condition in signServiceAgreement	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	077	--	Change reference to deprecated method "authenticate" in TpAuthMechanism to "challenge"	5.1.0	5.2.0
Jun 2003	CN_20	NP-030237	079	--	Correction to TpEncryptionCapability to correct support for Triple-DES	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	081	--	Correction of the Framework Service Instance Lifecycle Manager Sequence Diagram	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	083	--	Correction of the use of TpDomainID in Framework initiateAuthentication method	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	085	--	Correction to Java Realisation Annex	5.3.0	5.4.0

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-030572
Meeting #25, Bangkok, Thailand, 27 – 31 October 2003

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03** **CR** **087** ⌘ **rev** **-** ⌘ Current version: **5.4.0** ⌘

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

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

Title:	⌘	Correction of State Transition Diagram for IpAccess
Source:	⌘	CN5 (AePONA – Eamonn Murray)
Work item code:	⌘	OSA2
Date:	⌘	31/10/2003
Category:	⌘	**F**
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 method names and location of methods in the Framework Access Session API have previously undergone modification and correction. However the State Transition Diagram for IpAccess has not been updated to reflect these changes in a consistent fashion.
Summary of change:	⌘	Correct the IpAccess State Transition Diagram to align with the specified interface classes and methods.
Consequences if not approved:	⌘	Ambiguous specification may result and implementations will fail to interoperate correctly.

Clauses affected:	⌘	6.4.1.3

	Y	N		
Other specs affected: ⌘		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:

********** Start of Change # 1 **********

6.4.1.3 State Transition Diagrams for IpAccess

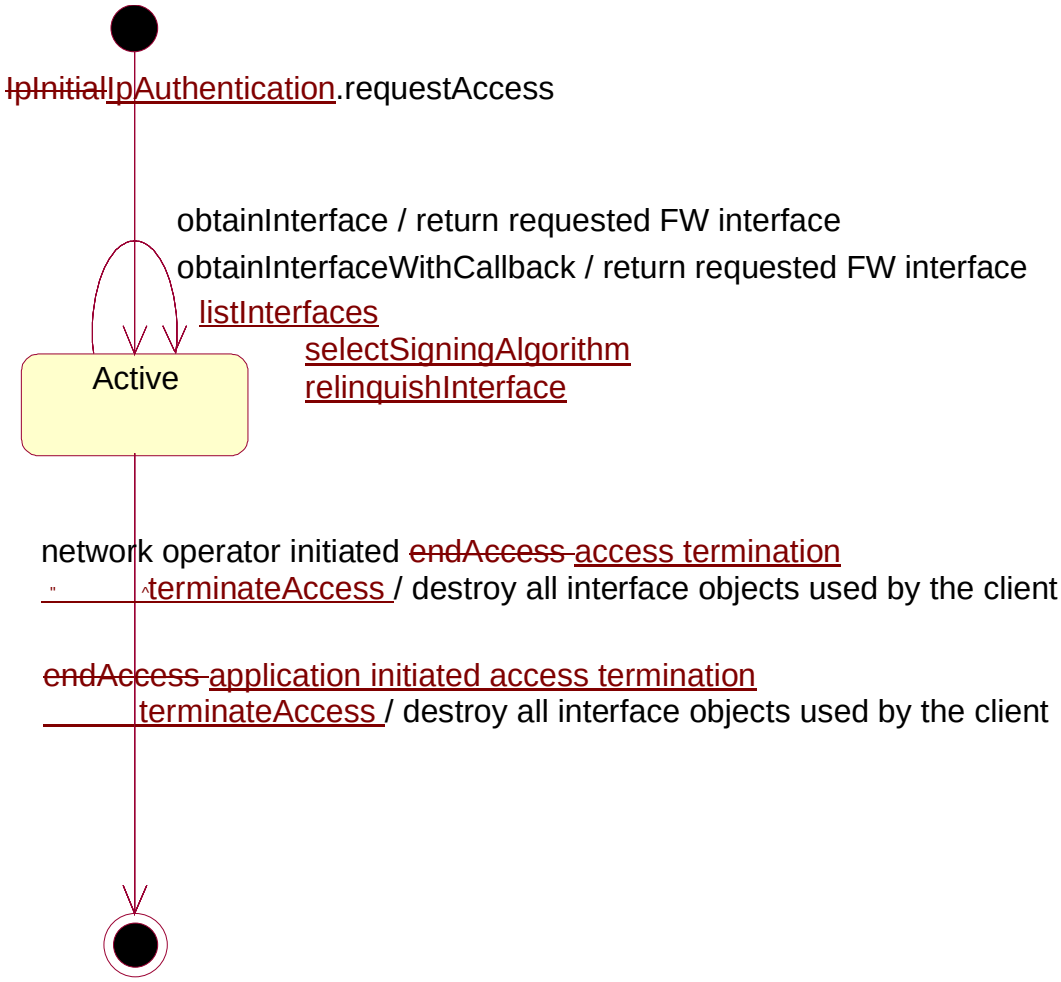


Figure : State Transition Diagram for IpAccess

6.4.1.3.1 Active State

When the client requests access to the Framework on the ~~IpInitial~~ IpAuthentication (IpAPILevelAuthentication) interface, an object implementing the IpAccess interface is created. The client can now request other Framework interfaces, including Service Discovery, Integrity Management, Service Subscription etc., and if at any point these framework interfaces are no longer required, to relinquish these. In addtion the client can select the signing algorithm that shall be used during the access session in cases where a digital signature is required. When the client is no longer interested in using the interfaces it calls the ~~end~~terminateAccess method. This results in the destruction of all interface objects used by the client. In case the network operator decides that the client has no longer access to the interfaces the same will happen.

********** End of Change # 1 **********

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2003	CN_19	NP-030028	076	--	Remove race condition in signServiceAgreement	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	077	--	Change reference to deprecated method "authenticate" in TpAuthMechanism to "challenge"	5.1.0	5.2.0
Jun 2003	CN_20	NP-030237	079	--	Correction to TpEncryptionCapability to correct support for Triple-DES	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	081	--	Correction of the Framework Service Instance Lifecycle Manager Sequence Diagram	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	083	--	Correction of the use of TpDomainID in Framework initiateAuthentication method	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	085	--	Correction to Java Realisation Annex	5.3.0	5.4.0

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-03 CR 088 ⌘rev - ⌘ Current version: 5.4.0 ⌘

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

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

Title:	⌘	Correction of Correlation Behaviour in Load Management
Source:	⌘	CN5 (AePONA – Eamonn Murray)
Work item code:	⌘	OSA2
Date:	⌘	31/10/2003
Category:	⌘	F
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 queryLoadReq/Res/Err methods on the Framework to App and Framework to Svc interfaces are used to ask for and supply load statistics reports on the operation of parties involved in OSA communication. These methods currently use a time period to control the period for which statistics are to be gathered. There is no restriction on invoking multiple requests or indeed requests with overlapping periods in time. In such cases however, there is no mechanism for correlating the responses uniquely with the requests. The res methods include a data type that details a time stamp rather than a unique period, and the err methods use a data type that contains no unique identificiation. (Note the res data type is also capable of returning the error).
Summary of change:	⌘	Correct the correlation between requests and responses by introducing a unique ID that is generated by the requesting entity.
Consequences if not approved:	⌘	The load statistics mechanism of the OSA Fault Management capability cannot be supported.

Clauses affected:	⌘	7.1.2, 7.3.3.7, 7.3.3.8, 8.1.4, 8.3.4.7, 8.3.4.8, 10.4

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.

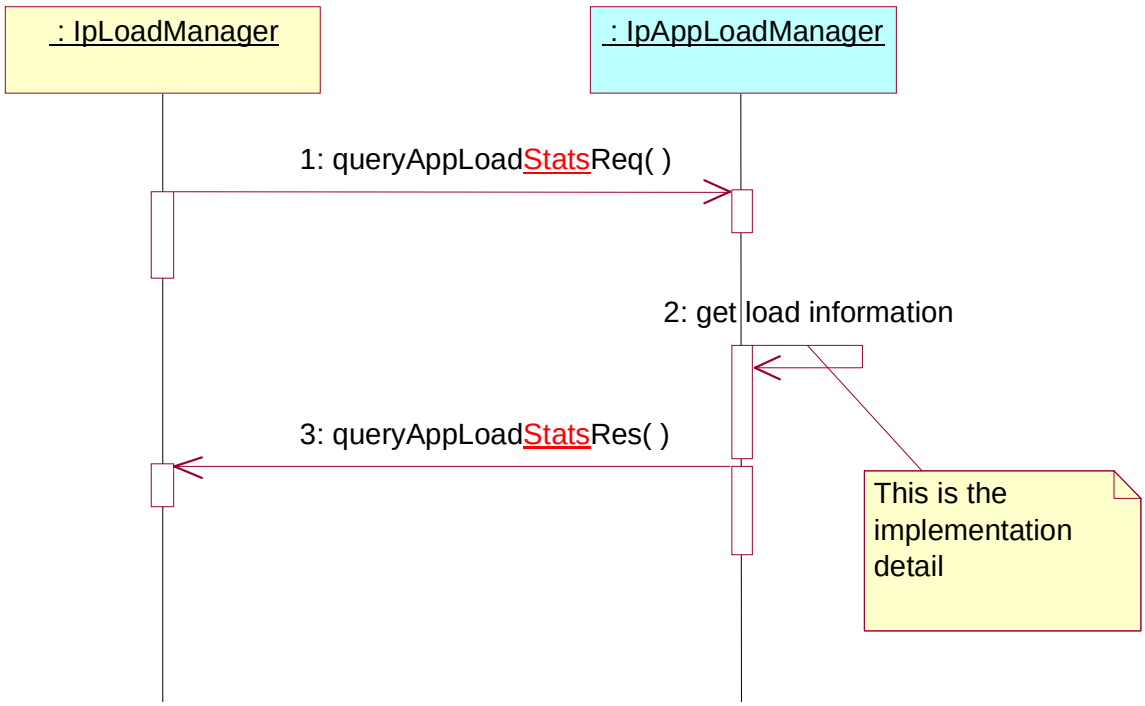
********** Start of Change # 1 **********

7.1.2 Integrity Management Sequence Diagrams

********** CUT **********

7.1.2.2 Load Management: Framework queries load statistics

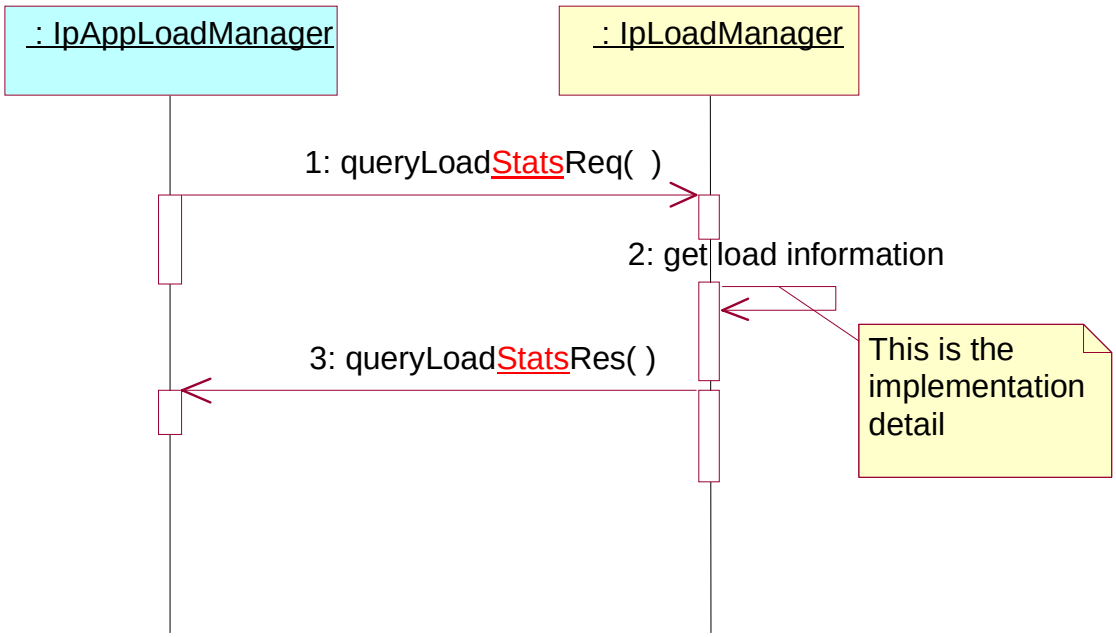
This sequence diagram shows how the framework requests load statistics for an application.



********** CUT **********

7.1.2.5 Load Management: Application queries load statistics

This sequence diagram shows how an application requests load statistics for the framework.



********** End of Change # 1 **********

********** Start of Change # 2 **********

7.3.3.7 Interface Class IpAppLoadManager

Inherits from: IpInterface.

The client application developer supplies the load manager application interface to handle requests, reports and other responses from the framework load manager function. The application supplies the identity of this callback interface at the time it obtains the framework's load manager interface, by use of the obtainInterfaceWithCallback() method on the IpAccess interface.

<<Interface>> IpAppLoadManager
<<deprecated>> queryAppLoadReq (timeInterval : in TpTimeInterval) : void <<deprecated>> queryLoadRes (loadStatistics : in TpLoadStatisticList) : void <<deprecated>> queryLoadErr (loadStatisticsError : in TpLoadStatisticError) : void loadLevelNotification (loadStatistics : in TpLoadStatisticList) : void resumeNotification () : void suspendNotification () : void <<new>> createLoadLevelNotification () : void <<new>> destroyLoadLevelNotification () : void <<new>> queryAppLoadStatsReq (loadStatsReqID : in TpLoadTestID, timeInterval : in TpTimeInterval) : void <<new>> queryLoadStatsRes (loadStatsReqID : in TpLoadTestID, loadStatistics : in TpLoadStatisticList) : void <<new>> queryLoadStatsErr (loadStatsReqID : in TpLoadTestID, loadStatisticsError : in TpLoadStatisticError) : void

7.3.3.7.1 Method <<deprecated>> queryAppLoadReq()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method queryAppLoadStatsReq shall be used instead, using the new identifier to correlate requests and responses.

The framework uses this method to request the application to provide load statistics records for the application.

Parameters

`timeInterval : in TpTimeInterval`

Specifies the time interval for which load statistic records should be reported.

7.3.3.7.2 Method <<deprecated>> queryLoadRes()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method queryLoadStatsRes shall be used instead, using the new identifier to correlate requests and responses.

The framework uses this method to send load statistic records back to the application that requested the information; i.e. in response to an invocation of the queryLoadReq method on the IpLoadManager interface.

Parameters

`loadStatistics : in TpLoadStatisticList`

Specifies the framework-supplied load statistics

7.3.3.7.3 Method <<deprecated>> queryLoadErr()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method queryLoadStatsErr shall be used instead, using the new identifier to correlate requests and errors.

The framework uses this method to return an error response to the application that requested the framework's load statistics information, when the framework is unsuccessful in obtaining any load statistic records; i.e. in response to an invocation of the queryLoadReq method on the IpLoadManager interface.

Parameters

`loadStatisticsError : in TpLoadStatisticError`

Specifies the error code associated with the failed attempt to retrieve the framework's load statistics.

7.3.3.7.4 Method loadLevelNotification()

Upon detecting load condition change, (e.g. load level changing from 0 to 1, 0 to 2, 1 to 0, for the SCFs or framework which have been registered for load level notifications) this method is invoked on the application. In addition this method shall be invoked on the application in order to provide a notification of current load status, when load notifications are first requested, or resumed after suspension.

Parameters

`loadStatistics : in TpLoadStatisticList`

Specifies the framework-supplied load statistics, which include the load level change(s).

7.3.3.7.5 Method resumeNotification()

The framework uses this method to request the application to resume sending it notifications: e.g. after a period of suspension during which the framework handled a temporary overload condition. Upon receipt of this method the client application shall inform the framework of the current load using the reportLoad method on the corresponding IpLoadManager.

Parameters

No Parameters were identified for this method

7.3.3.7.6 Method suspendNotification()

The framework uses this method to request the application to suspend sending it any notifications: e.g. while the framework handles a temporary overload condition.

Parameters

No Parameters were identified for this method

7.3.3.7.7 Method <<new>> createLoadLevelNotification()

The framework uses this method to register to receive notifications of load level changes associated with the application. Upon receipt of this method the client application shall inform the framework of the current load using the reportLoad method on the corresponding IpLoadManager.

Parameters

No Parameters were identified for this method

7.3.3.7.8 Method <<new>> destroyLoadLevelNotification()

The framework uses this method to unregister for notifications of load level changes associated with the application.

Parameters

No Parameters were identified for this method

7.3.3.7.9 Method <<new>> queryAppLoadStatsReq()

The framework uses this method to request the application to provide load statistics records for the application.

Parameters

loadStatsReqID : in TpLoadTestID

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

timeInterval : in TpTimeInterval

Specifies the time interval for which load statistic records should be reported.

7.3.3.7.10 Method <<new>> queryLoadStatsRes()

The framework uses this method to send load statistic records back to the application that requested the information; i.e. in response to an invocation of the queryLoadReq method on the IpLoadManager interface.

Parameters

loadStatsReqID : in TpLoadTestID

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

loadStatistics : in TpLoadStatisticList

Specifies the framework-supplied load statistics

7.3.3.7.11 Method <<new>> queryLoadStatsErr()

The framework uses this method to return an error response to the application that requested the framework's load statistics information, when the framework is unsuccessful in obtaining any load statistic records; i.e. in response to an invocation of the queryLoadReq method on the IpLoadManager interface.

Parameters

loadStatsReqID : in TpLoadTestID

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

loadStatisticsError : in TpLoadStatisticError

Specifies the error code associated with the failed attempt to retrieve the framework's load statistics.

********** End of Change # 2 **********

********** Start of Change # 3 **********

********** End of Change # 3 **********

7.3.3.8 Interface Class IpLoadManager

Inherits from: IpInterface.

The framework API should allow the load to be distributed across multiple machines and across multiple component processes, according to a load management policy. The separation of the load management mechanism and load management policy ensures the flexibility of the load management services. The load management policy identifies what load management rules the framework should follow for the specific client application. It might specify what action the framework should take as the congestion level changes. For example, some real-time critical applications will want to make sure continuous service is maintained, below a given congestion level, at all costs, whereas other services will be satisfied with disconnecting and trying again later if the congestion level rises. Clearly, the load management policy is related to the QoS level to which the application is subscribed. The framework load management function is represented by the IpLoadManager interface. Most methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. To handle responses and reports, the client application developer must implement the IpAppLoadManager interface to provide the callback mechanism. The application supplies the identity of this callback interface at the time it obtains the framework's load manager interface, by use of the obtainInterfaceWithCallback operation on the IpAccess interface.

If the IpLoadManager interface is implemented by a Framework, at least one of the methods shall be implemented as a minimum requirement. If load level notifications are supported, the createLoadLevelNotification() and destroyLoadLevelNotification() methods shall be implemented. If suspendNotification() is implemented, then resumeNotification() shall be implemented also. If a Framework is capable of invoking the IpAppLoadManager.queryAppLoadStatsReq() method, then it shall implement queryAppLoadStatsRes() and queryAppLoadStatsErr() methods in this interface.

<<Interface>> IpLoadManager
reportLoad (loadLevel : in TpLoadLevel) : void <<deprecated>> queryLoadReq (serviceIDs : in TpServiceIDList, timeInterval : in TpTimeInterval) : void <<deprecated>> queryAppLoadRes (loadStatistics : in TpLoadStatisticList) : void <<deprecated>> queryAppLoadErr (loadStatisticsError : in TpLoadStatisticError) : void createLoadLevelNotification (serviceIDs : in TpServiceIDList) : void destroyLoadLevelNotification (serviceIDs : in TpServiceIDList) : void resumeNotification (serviceIDs : in TpServiceIDList) : void suspendNotification (serviceIDs : in TpServiceIDList) : void <<new>> queryLoadStatsReq (loadStatsReqID : in TpLoadTestID, serviceIDs : in TpServiceIDList, timeInterval : in TpTimeInterval) : void <<new>> queryAppLoadStatsRes (loadStatsReqID : in TpLoadTestID, loadStatistics : in TpLoadStatisticList) : void <<new>> queryAppLoadStatsErr (loadStatsReqID : in TpLoadTestID, loadStatisticsError : in TpLoadStatisticError) : void

7.3.3.8.1 Method reportLoad()

The client application uses this method to report its current load level (0,1, or 2) to the framework: e.g. when the load level on the application has changed.

At level 0 load, the application is performing within its load specifications (i.e. it is not congested or overloaded). At level 1 load, the application is overloaded. At level 2 load, the application is severely overloaded. In addition this method shall be called by the application in order to report current load status, when load notifications are first requested, or resumed after suspension.

Parameters

`loadLevel : in TpLoadLevel`

Specifies the application's load level.

Raises

`TpCommonExceptions`

7.3.3.8.2 Method <<deprecated>> queryLoadReq()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method queryLoadStatsReq shall be used instead, using the new identifier to correlate requests and responses.

The client application uses this method to request the framework to provide load statistic records for the framework or for its instances of the individual services. If the application does not have access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID.

Parameters

`serviceIDs : in TpServiceIDList`

Specifies the framework or the services for which load statistics records should be reported. If this parameter is not an empty list, the load statistics records of the client's instances of the specified services are returned, otherwise the load statistics record of the framework is returned.

`timeInterval : in TpTimeInterval`

Specifies the time interval for which load statistics records should be reported.

Raises

`TpCommonExceptions, P_INVALID_SERVICE_ID, P_SERVICE_NOT_ENABLED, P_UNAUTHORISED_PARAMETER_VALUE`

7.3.3.8.3 Method <<deprecated>> queryAppLoadRes()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method queryAppLoadStatsRes shall be used instead, using the new identifier to correlate requests and responses.

The client application uses this method to send load statistic records back to the framework that requested the information; i.e. in response to an invocation of the queryAppLoadReq method on the IpAppLoadManager interface.

Parameters

loadStatistics : in TpLoadStatisticList

Specifies the application-supplied load statistics.

Raises

TpCommonExceptions

7.3.3.8.4 Method <<deprecated>> queryAppLoadErr()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method queryAppLoadStatsErr shall be used instead, using the new identifier to correlate requests and errors.

The client application uses this method to return an error response to the framework that requested the application's load statistics information, when the application is unsuccessful in obtaining any load statistic records; i.e. in response to an invocation of the queryAppLoadReq method on the IpAppLoadManager interface.

Parameters

loadStatisticsError : in TpLoadStatisticError

Specifies the error code associated with the failed attempt to retrieve the application's load statistics.

Raises

TpCommonExceptions

7.3.3.8.5 Method createLoadLevelNotification()

The client application uses this method to register to receive notifications of load level changes associated with either the framework or with its instances of the individual services used by the application. If the application does not have access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID. Upon receipt of this method the framework shall inform the client application of the current framework or service instance load using the loadLevelNotification method on the corresponding IpAppLoadManager.

Parameters

serviceIDs : in TpServiceIDList

Specifies the framework or SCFs to be registered for load control. To register for framework load control, the serviceIDs parameter must be an empty list.

Raises

TpCommonExceptions, P_INVALID_SERVICE_ID, P_UNAUTHORISED_PARAMETER_VALUE

7.3.3.8.6 Method destroyLoadLevelNotification()

The client application uses this method to unregister for notifications of load level changes associated with either the framework or with its instances of the individual services used by the application. If the application does not have

access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID.

Parameters

serviceIDs : in TpServiceIDList

Specifies the framework or the services for which load level changes should no longer be reported. To unregister for framework load control, the serviceIDs parameter must be an empty list.

Raises

TpCommonExceptions, P_INVALID_SERVICE_ID, P_UNAUTHORISED_PARAMETER_VALUE

7.3.3.8.7 Method resumeNotification()

The client application uses this method to request the framework to resume sending it load management notifications associated with either the framework or with its instances of the individual services used by the application; e.g. after a period of suspension during which the application handled a temporary overload condition. If the application does not have access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID. Upon receipt of this method the framework shall inform the client application of the current framework or service instance load using the loadLevelNotification method on the corresponding IpAppLoadManager.

Parameters

serviceIDs : in TpServiceIDList

Specifies the framework or the services for which the sending of notifications of load level changes by the framework should be resumed. To resume notifications for the framework, the serviceIDs parameter must be an empty list.

Raises

TpCommonExceptions, P_INVALID_SERVICE_ID, P_SERVICE_NOT_ENABLED, P_UNAUTHORISED_PARAMETER_VALUE

7.3.3.8.8 Method suspendNotification()

The client application uses this method to request the framework to suspend sending it load management notifications associated with either the framework or with its instances of the individual services used by the application; e.g. while the application handles a temporary overload condition. If the application does not have access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID.

Parameters

serviceIDs : in TpServiceIDList

Specifies the framework or the services for which the sending of notifications by the framework should be suspended. To suspend notifications for the framework, the serviceIDs parameter must be an empty list.

Raises

TpCommonExceptions, P_INVALID_SERVICE_ID, P_SERVICE_NOT_ENABLED, P_UNAUTHORISED_PARAMETER_VALUE

7.3.3.8.9 Method <<new>> queryLoadStatsReq()

The client application uses this method to request the framework to provide load statistic records for the framework or for its instances of the individual services. If the application does not have access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID.

Parameters

loadStatsReqID : in TpLoadTestID

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

serviceIDs : in TpServiceIDList

Specifies the framework or the services for which load statistics records should be reported. If this parameter is not an empty list, the load statistics records of the client's instances of the specified services are returned, otherwise the load statistics record of the framework is returned.

timeInterval : in TpTimeInterval

Specifies the time interval for which load statistics records should be reported.

Raises

TpCommonExceptions, P_INVALID_SERVICE_ID, P_SERVICE_NOT_ENABLED, P_UNAUTHORISED_PARAMETER_VALUE

7.3.3.8.10 Method <<new>> queryAppLoadStatsRes()

The client application uses this method to send load statistic records back to the framework that requested the information; i.e. in response to an invocation of the queryAppLoadStatsReq method on the IpAppLoadManager interface.

Parameters

loadStatsReqID : in TpLoadTestID

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

loadStatistics : in TpLoadStatisticList

Specifies the application-supplied load statistics.

Raises

TpCommonExceptions

7.3.3.8.11 Method <<new>> queryAppLoadStatsErr()

The client application uses this method to return an error response to the framework that requested the application's load statistics information, when the application is unsuccessful in obtaining any load statistic records; i.e. in response to an invocation of the queryAppLoadStatsReq method on the IpAppLoadManager interface.

Parameters

loadStatsReqID : in TpLoadTestID

Used by the framework to correlate this error (when it arrives) with the original request.

loadStatisticsError : in TpLoadStatisticError

Specifies the error code associated with the failed attempt to retrieve the application's load statistics.

Raises

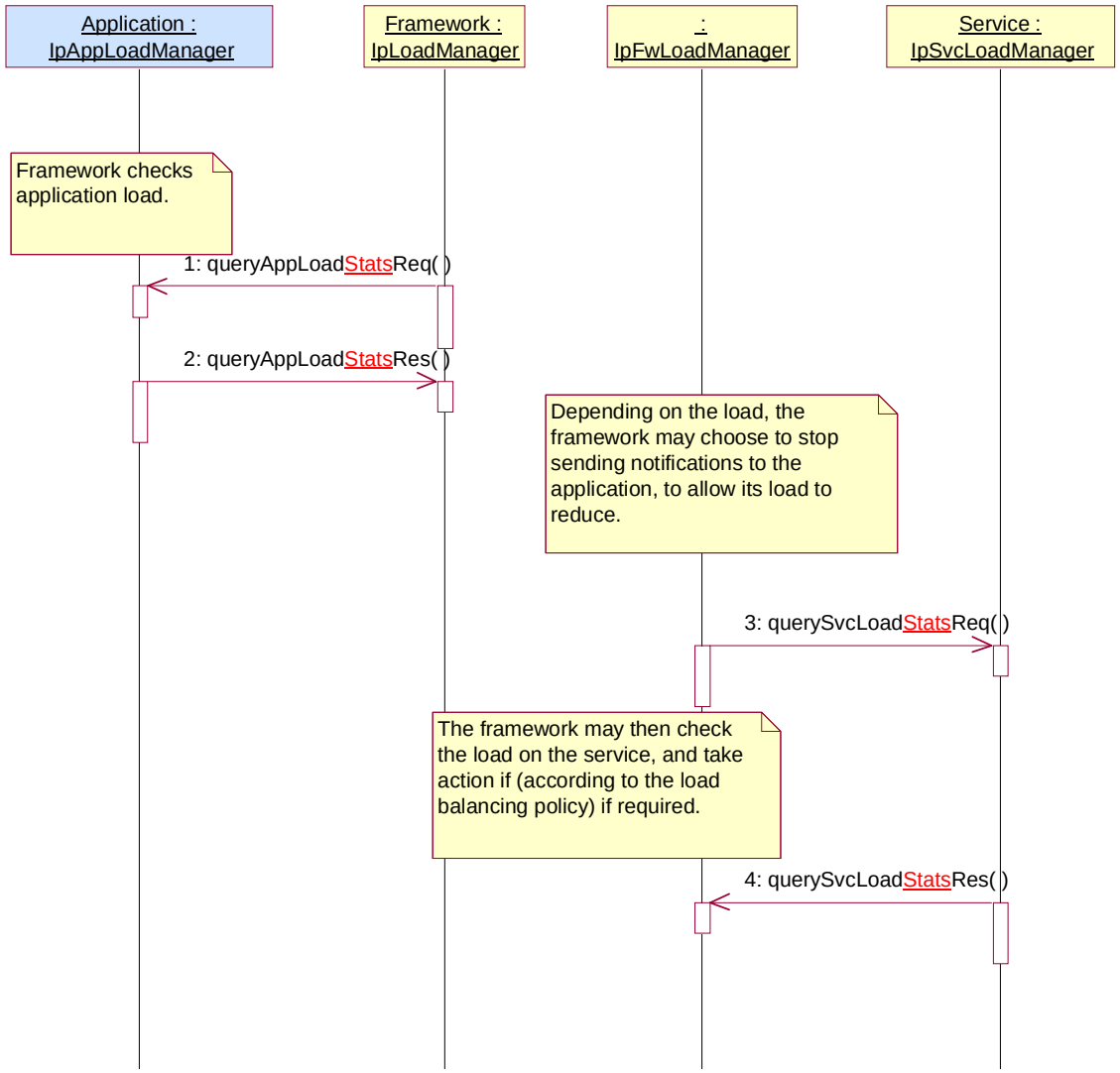
TpCommonExceptions

********** Start of Change # 4 **********

8.1.4 Integrity Management Sequence Diagrams

********** CUT **********

8.1.4.3 Load Management: Client and Service Load Balancing



********** End of Change # 4 **********

********** Start of Change # 5 **********

8.3.4.7 Interface Class IpFwLoadManager

Inherits from: IpInterface.

The framework API should allow the load to be distributed across multiple machines and across multiple component processes, according to a load management policy. The separation of the load management mechanism and load management policy ensures the flexibility of the load management services. The load management policy identifies what load management rules the framework should follow for the specific service. It might specify what action the framework should take as the congestion level changes. For example, some real-time critical applications will want to make sure continuous service is maintained, below a given congestion level, at all costs, whereas other services will be satisfied with disconnecting and trying again later if the congestion level rises. Clearly, the load management policy is related to the QoS level to which the application is subscribed. The framework load management function is represented by the IpFwLoadManager interface. To handle responses and reports, the service developer must implement the IpSvcLoadManager interface to provide the callback mechanism.

If the IpFwLoadManager interface is implemented by a Framework, at least one of the methods shall be implemented as a minimum requirement. If load level notifications are supported, the createLoadLevelNotification()

and destroyLoadLevelNotification() methods shall be implemented. If suspendNotification() is implemented, then resumeNotification() shall be implemented also. If a Framework is capable of invoking the IpSvcLoadManager.querySvcLoadStatsReq() method, then it shall implement querySvcLoadStatsRes() and querySvcLoadStatsErr() methods in this interface.

<<Interface>> IpFwLoadManager
reportLoad (loadLevel : in TpLoadLevel) : void <<deprecated>> queryLoadReq (querySubject : in TpSubjectType, timeInterval : in TpTimeInterval) : void <<deprecated>> querySvcLoadRes (loadStatistics : in TpLoadStatisticList) : void <<deprecated>> querySvcLoadErr (loadStatisticError : in TpLoadStatisticError) : void createLoadLevelNotification (notificationSubject : in TpSubjectType) : void destroyLoadLevelNotification (notificationSubject : in TpSubjectType) : void suspendNotification (notificationSubject : in TpSubjectType) : void resumeNotification (notificationSubject : in TpSubjectType) : void <<new>> queryLoadStatsReq (loadStatsReqID : in TpLoadTestID, querySubject : in TpSubjectType, timeInterval : in TpTimeInterval) : void <<new>> querySvcLoadStatsRes (loadStatsReqID : in TpLoadTestID, loadStatistics : in TpLoadStatisticList) : void <<new>> querySvcLoadStatsErr (loadStatsReqID : in TpLoadTestID, loadStatisticError : in TpLoadStatisticError) : void

8.3.4.7.1 Method reportLoad()

The service instance uses this method to report its current load level (0,1, or 2) to the framework: e.g. when the load level on the service instance has changed.

At level 0 load, the service instance is performing within its load specifications (i.e. it is not congested or overloaded). At level 1 load, the service instance is overloaded. At level 2 load, the service instance is severely overloaded. In addition this method shall be called by the service instance in order to report current load status, when load notifications are first requested, or resumed after suspension.

Parameters

`loadLevel : in TpLoadLevel`

Specifies the service instance's load level.

Raises

`TpCommonExceptions`

8.3.4.7.2 Method <<deprecated>> queryLoadReq()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method queryLoadStatsReq shall be used instead, using the new identifier to correlate requests and responses.

The service instance uses this method to request the framework to provide load statistics records for the framework or for the application that uses the service instance.

Parameters

querySubject : in TpSubjectType

Specifies the entity (framework or application) for which load statistics records should be reported.

timeInterval : in TpTimeInterval

Specifies the time interval for which load statistics records should be reported.

Raises

TpCommonExceptions

8.3.4.7.3 Method <<deprecated>> querySvcLoadRes()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method querySvcLoadStatsRes shall be used instead, using the new identifier to correlate requests and responses.

The service instance uses this method to send load statistic records back to the framework that requested the information; i.e. in response to an invocation of the querySvcLoadReq method on the IpSvcLoadManager interface.

Parameters

loadStatistics : in TpLoadStatisticList

Specifies the service-supplied load statistics.

Raises

TpCommonExceptions

8.3.4.7.4 Method <<deprecated>> querySvcLoadErr()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method querySvcLoadStatsErr shall be used instead, using the new identifier to correlate requests and errors.

The service instance uses this method to return an error response to the framework that requested the service instance's load statistics information, when the service instance is unsuccessful in obtaining any load statistic records; i.e. in response to an invocation of the querySvcLoadReq method on the IpSvcLoadManager interface.

Parameters

loadStatisticError : in TpLoadStatisticError

Specifies the error code associated with the failed attempt to retrieve the service instance's load statistics.

Raises

TpCommonExceptions

8.3.4.7.5 Method createLoadLevelNotification()

The service instance uses this method to register to receive notifications of load level changes associated with the framework or with the application that uses the service instance. Upon receipt of this method the framework shall inform the service instance of the current framework or application load using the loadLevelNotification method on the corresponding IpSvcLoadManager.

Parameters

notificationSubject : in TpSubjectType

Specifies the entity (framework or application) for which load level changes should be reported.

Raises

TpCommonExceptions

8.3.4.7.6 Method destroyLoadLevelNotification()

The service instance uses this method to unregister for notifications of load level changes associated with the framework or with the application that uses the service instance.

Parameters

notificationSubject : in TpSubjectType

Specifies the entity (framework or application) for which load level changes should no longer be reported.

Raises

TpCommonExceptions

8.3.4.7.7 Method suspendNotification()

The service instance uses this method to request the framework to suspend sending it notifications associated with the framework or with the application that uses the service instance; e.g. while the service instance handles a temporary overload condition.

Parameters

notificationSubject : in TpSubjectType

Specifies the entity (framework or application) for which the sending of notifications by the framework should be suspended.

Raises

TpCommonExceptions

8.3.4.7.8 Method resumeNotification()

The service instance uses this method to request the framework to resume sending it notifications associated with the framework or with the application that uses the service instance; e.g. after a period of suspension during which the service instance handled a temporary overload condition. Upon receipt of this method the framework shall inform the service instance of the current framework or application load using the loadLevelNotification method on the corresponding IpSvcLoadManager.

Parameters

notificationSubject : in TpSubjectType

Specifies the entity (framework or application) for which the sending of notifications of load level changes by the framework should be resumed.

Raises

TpCommonExceptions

8.3.4.7.9 Method <<new>> queryLoadStatsReq()

The service instance uses this method to request the framework to provide load statistics records for the framework or for the application that uses the service instance.

Parameters

loadStatsReqID : in TpLoadTestID

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

querySubject : in TpSubjectType

Specifies the entity (framework or application) for which load statistics records should be reported.

timeInterval : in TpTimeInterval

Specifies the time interval for which load statistics records should be reported.

Raises

TpCommonExceptions

8.3.4.7.10 Method <<new>> querySvcLoadStatsRes()

The service instance uses this method to send load statistic records back to the framework that requested the information; i.e. in response to an invocation of the querySvcLoadStatsReq method on the IpSvcLoadManager interface.

Parameters

loadStatsReqID : in TpLoadTestID

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

loadStatistics : in TpLoadStatisticList

Specifies the service-supplied load statistics.

Raises

TpCommonExceptions

8.3.4.7.11 Method <<new>> querySvcLoadStatsErr()

The service instance uses this method to return an error response to the framework that requested the service instance's load statistics information, when the service instance is unsuccessful in obtaining any load statistic records; i.e. in response to an invocation of the querySvcLoadStatsReq method on the IpSvcLoadManager interface.

Parameters

loadStatsReqID : in TpLoadTestID

Used by the framework to correlate this error (when it arrives) with the original request.

loadStatisticError : in TpLoadStatisticError

Specifies the error code associated with the failed attempt to retrieve the service instance's load statistics.

Raises

TpCommonExceptions

********** End of Change # 5 **********

********** Start of Change # 6 **********

8.3.4.8 Interface Class IpSvcLoadManager

Inherits from: IpInterface.

The service developer supplies the load manager service interface to handle requests, reports and other responses from the framework load manager function. The service instance supplies the identity of its callback interface at the time it obtains the framework's load manager interface, by use of the obtainInterfaceWithCallback() method on the IpAccess interface.

If the IpSvcLoadManager interface is implemented by a Service, at least one of the methods shall be implemented as a minimum requirement. If load level notifications are supported, then loadLevelNotification() shall be implemented. If a the Service is capable of invoking the IpFwLoadManager.queryLoadStatsReq() method, then it shall implement queryLoadStatsRes() and queryLoadStatsErr() methods in this interface.

<<Interface>> IpSvcLoadManager
<<deprecated>> querySvcLoadReq (timeInterval : in TpTimeInterval) : void <<deprecated>> queryLoadRes (loadStatistics : in TpLoadStatisticList) : void <<deprecated>> queryLoadErr (loadStatisticsError : in TpLoadStatisticError) : void loadLevelNotification (loadStatistics : in TpLoadStatisticList) : void suspendNotification () : void resumeNotification () : void <<new>> createLoadLevelNotification () : void <<new>> destroyLoadLevelNotification () : void <<new>> querySvcLoadReq (loadStatsReqID : in TpLoadTestID, timeInterval : in TpTimeInterval) : void <<new>> queryLoadRes (loadStatsReqID : in TpLoadTestID, loadStatistics : in TpLoadStatisticList) : void <<new>> queryLoadErr (loadStatsReqID : in TpLoadTestID, loadStatisticsError : in TpLoadStatisticError) : void

8.3.4.8.1 Method <<deprecated>> querySvcLoadReq()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method querySvcLoadStatsReq shall be used instead, using the new identifier to correlate requests and responses.

The framework uses this method to request the service instance to provide its load statistic records.

Parameters

`timeInterval : in TpTimeInterval`

Specifies the time interval for which load statistic records should be reported.

Raises

`TpCommonExceptions`

8.3.4.8.2 Method <<deprecated>> queryLoadRes()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method queryLoadStatsRes shall be used instead, using the new identifier to correlate requests and responses.

The framework uses this method to send load statistic records back to the service instance that requested the information; i.e. in response to an invocation of the queryLoadReq method on the IpFwLoadManager interface.

Parameters

`loadStatistics : in TpLoadStatisticList`

Specifies the framework-supplied load statistics

Raises

TpCommonExceptions

8.3.4.8.3 Method <<deprecated>> queryLoadErr()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method queryLoadStatsErr shall be used instead, using the new identifier to correlate requests and errors.

The framework uses this method to return an error response to the service that requested the framework's load statistics information, when the framework is unsuccessful in obtaining any load statistic records; i.e. in response to an invocation of the queryLoadReq method on the IpFwLoadManager interface.

Parameters

loadStatisticsError : in TpLoadStatisticError

Specifies the error code associated with the failed attempt to retrieve the framework's load statistics.

Raises

TpCommonExceptions

8.3.4.8.4 Method loadLevelNotification()

Upon detecting load condition change, (e.g. load level changing from 0 to 1, 0 to 2, 1 to 0, for the application or framework which has been registered for load level notifications) this method is invoked on the SCF. In addition this method shall be invoked on the SCF in order to provide a notification of current load status, when load notifications are first requested, or resumed after suspension.

Parameters

loadStatistics : in TpLoadStatisticList

Specifies the framework-supplied load statistics, which include the load level change(s).

Raises

TpCommonExceptions

8.3.4.8.5 Method suspendNotification()

The framework uses this method to request the service instance to suspend sending it any notifications: e.g. while the framework handles a temporary overload condition.

Parameters

No Parameters were identified for this method

Raises

TpCommonExceptions

8.3.4.8.6 Method resumeNotification()

The framework uses this method to request the service instance to resume sending it notifications: e.g. after a period of suspension during which the framework handled a temporary overload condition. Upon receipt of this method the service instance shall inform the framework of the current load using the reportLoad method on the corresponding IpFwLoadManager.

Parameters
No Parameters were identified for this method

Raises

TpCommonExceptions

8.3.4.8.7 Method <<new>> createLoadLevelNotification()

The framework uses this method to register to receive notifications of load level changes associated with the service instance. Upon receipt of this method the service instance shall inform the framework of the current load using the reportLoad method on the corresponding IpFwLoadManager.

Parameters
No Parameters were identified for this method

Raises

TpCommonExceptions

8.3.4.8.8 Method <<new>> destroyLoadLevelNotification()

The framework uses this method to unregister for notifications of load level changes associated with the service instance.

Parameters
No Parameters were identified for this method

Raises

TpCommonExceptions

8.3.4.9 Interface Class IpFwOAM

Inherits from: IpInterface.

The OAM interface is used to query the system date and time. The service and the framework can synchronise the date and time to a certain extent. Accurate time synchronisation is outside the scope of this API. This interface and the systemDateTimeQuery() method are optional.

8.3.4.9.1 Method <<new>> querySvcLoadStatsReq()

The framework uses this method to request the service instance to provide its load statistic records.

Parameters

loadStatsReqID : in TpLoadTestID

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

timeInterval : in TpTimeInterval

Specifies the time interval for which load statistic records should be reported.

Raises

TpCommonExceptions

8.3.4.9.2 Method <<new>> queryLoadStatsRes()

The framework uses this method to send load statistic records back to the service instance that requested the information; i.e. in response to an invocation of the queryLoadReq method on the IpFwLoadManager interface.

Parameters

loadStatsReqID : in TpLoadTestID

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

loadStatistics : in TpLoadStatisticList

Specifies the framework-supplied load statistics

Raises

TpCommonExceptions

8.3.4.9.3 Method <<new>> queryLoadStatsErr()

The framework uses this method to return an error response to the service that requested the framework's load statistics information, when the framework is unsuccessful in obtaining any load statistic records; i.e. in response to an invocation of the queryLoadReq method on the IpFwLoadManager interface.

Parameters

loadStatsReqID : in TpLoadTestID

Used by the service instance to correlate this error (when it arrives) with the original request.

loadStatisticsError : in TpLoadStatisticError

Specifies the error code associated with the failed attempt to retrieve the framework's load statistics.

Raises

TpCommonExceptions

********** End of Change # 6 **********

********** Start of Change # 7 **********

10.4 Integrity Management Data Definitions

10.4.1 TpActivityTestRes

This type is identical to TpString and is an implementation specific result. The values in this data type are "Available" or "Unavailable".

10.4.2 TpFaultStatsRecord

This defines the set of records to be returned giving fault information for the requested time period.

Sequence Element Name	Sequence Element Type
Period	TpTimeInterval
FaultStatsSet	TpFaultStatsSet

10.4.3 TpFaultStats

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

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

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

10.4.4 TpFaultStatisticsError

Defines the error code associated with a failed attempt to retrieve any fault statistics information.

Name	Value	Description
P_FAULT_INFO_ERROR_UNDEFINED	0	Undefined error
P_FAULT_INFO_UNAVAILABLE	1	Fault statistics unavailable

10.4.5 TpFaultStatsSet

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

10.4.6 TpActivityTestID

This data type is identical to a TpInt32, and is used as a token to match activity test requests with their results..

10.4.7 TpInterfaceFault

Defines the cause of the interface fault detected.

Name	Value	Description
INTERFACE_FAULT_UNDEFINED	0	Undefined
INTERFACE_FAULT_LOCAL_FAILURE	1	A fault in the local API software or hardware has been detected
INTERFACE_FAULT_GATEWAY_FAILURE	2	A fault in the gateway API software or hardware has been detected
INTERFACE_FAULT_PROTOCOL_ERROR	3	An error in the protocol used on the client-gateway link has been detected

10.4.8 TpSvcUnavailReason

Defines the reason why a SCF is unavailable.

Name	Value	Description
SERVICE_UNAVAILABLE_UNDEFINED	0	Undefined
SERVICE_UNAVAILABLE_LOCAL_FAILURE	1	The Local API software or hardware has failed
SERVICE_UNAVAILABLE_GATEWAY_FAILURE	2	The gateway API software or hardware has failed
SERVICE_UNAVAILABLE_OVERLOADED	3	The SCF is fully overloaded
SERVICE_UNAVAILABLE_CLOSED	4	The SCF has closed itself (e.g. to protect from fraud or malicious attack)

10.4.9 TpFwUnavailReason

Defines the reason why the Framework is unavailable.

Name	Value	Description
FW_UNAVAILABLE_UNDEFINED	0	Undefined
FW_UNAVAILABLE_LOCAL_FAILURE	1	The Local API software or hardware has failed
FW_UNAVAILABLE_GATEWAY_FAILURE	2	The gateway API software or hardware has failed
FW_UNAVAILABLE_OVERLOADED	3	The Framework is fully overloaded
FW_UNAVAILABLE_CLOSED	4	The Framework has closed itself (e.g. to protect from fraud or malicious attack)
FW_UNAVAILABLE_PROTOCOL_FAILURE	5	The protocol used on the client-gateway link has failed

10.4.10 TpFaultTestID

This data type is identical to a TpInt32, and is used as a token to match fault statistics requests with their results.

~~10.4.10~~10.4.11 TpLoadLevel

Defines the Sequence of Data Elements that specify load level values.

Name	Value	Description
LOAD_LEVEL_NORMAL	0	Normal load
LOAD_LEVEL_OVERLOAD	1	Overload
LOAD_LEVEL_SEVERE_OVERLOAD	2	Severe Overload

~~10.4.11~~10.4.12 TpLoadThreshold

Defines the Sequence of Data Elements that specify the load threshold value. The actual load threshold value is application and SCF dependent, so is their relationship with load level.

Sequence Element Name	Sequence Element Type
LoadThreshold	TpFloat

********** End of Change # 7 **********

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Jun 2003	CN_20	NP-030237	079	--	Correction to TpEncryptionCapability to correct support for Triple-DES	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	081	--	Correction of the Framework Service Instance Lifecycle Manager Sequence Diagram	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	083	--	Correction of the use of TpDomainID in Framework initiateAuthentication method	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	085	--	Correction to Java Realisation Annex	5.3.0	5.4.0

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-030632
Meeting #25, Bangkok, Thailand, 27 – 31 October 2003

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03** **CR** **089** ⌘ **rev** **-** ⌘ Current version: **5.4.0** ⌘

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

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

Title:	⌘	Correction of Correlation Behaviour in Fault Management
Source:	⌘	CN5 (AePONA – Eamonn Murray)
Work item code:	⌘	OSA2
Date:	⌘	31/10/2003
Category:	⌘	**F**
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 genFaultStatsReq/Res/Err methods on the Framework to App and Framework to Svc interfaces are used to ask for and supply fault statistics reports on the operation of parties involved in OSA communication. These methods currently use a time period to control the period for which statistics are to be gathered. There is no restriction on invoking multiple requests or indeed requests with overlapping periods in time. In such cases however, there is no mechanism for correlating the responses uniquely with the requests. Although the res methods include a time period that could be used to match with the corresponding Req, the Err methods do not support any suitable identification. In addition, in the case of applications that request fault statistics for a list of services, there is no correlation between the fault statistics returned and the service in question.
Summary of change:	⌘	Correct the correlation between requests and responses by introducing a unique ID that is generated by the requesting entity. In addition clarify the ordering of information returned to applications when a list of services is used.
Consequences if not approved:	⌘	The fault statistics mechanism of the OSA Fault Management capability cannot be supported.

Clauses affected:	⌘	7.3.3.1, 7.3.3.2, 8.3.4.1, 8.3.4.2, 10.4

	Y	N		
Other specs affected: ⌘		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.

********** Start of Change # 1 **********

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

<<Interface>> IpAppFaultManager
activityTestRes (activityTestID : in TpActivityTestID, activityTestResult : in TpActivityTestRes) : void appActivityTestReq (activityTestID : in TpActivityTestID) : void fwFaultReportInd (fault : in TpInterfaceFault) : void fwFaultRecoveryInd (fault : in TpInterfaceFault) : void <<deprecated>> svcUnavailableInd (serviceID : in TpServiceID, reason : in TpSvcUnavailReason) : void <<deprecated>> genFaultStatsRecordRes (faultStatistics : in TpFaultStatsRecord, serviceIDs : in TpServiceIDList) : void fwUnavailableInd (reason : in TpFwUnavailReason) : void activityTestErr (activityTestID : in TpActivityTestID) : void <<deprecated>> genFaultStatsRecordErr (faultStatisticsError : in TpFaultStatisticsError, serviceIDs : in TpServiceIDList) : void appUnavailableInd (serviceID : in TpServiceID) : void <<deprecated>> genFaultStatsRecordReq (timePeriod : in TpTimeInterval) : void <<new>> svcAvailStatusInd (serviceID : in TpServiceID, reason : in TpSvcAvailStatusReason) : void <<new>> generateFaultStatisticsRecordRes (faultStatsReqID : in TpFaultReqID, faultStatistics : in TpFaultStatsRecord, serviceIDs : in TpServiceIDList) : void <<new>> generateFaultStatisticsRecordErr (faultStatsReqID : in TpFaultReqID, faultStatistics : in TpFaultStatsErrorList, serviceIDs : in TpServiceIDList) : void <<new>> generateFaultStatisticsRecordReq (faultStatsReqID : in TpFaultReqID, faultStatistics : in TpFaultStatsRecord) : void

7.3.3.1.1 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.

7.3.3.1.2 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.

7.3.3.1.3 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.

7.3.3.1.4 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.

7.3.3.1.5 Method <<deprecated>> svcUnavailableInd()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method svcAvailStatusInd shall be used instead, using the new type of reason parameter to inform the Application the reason why the Service is unavailable and also when the Service becomes available again.

The framework invokes this method to inform the client application that it may experience difficulties using its instance of the indicated service.

Parameters

serviceID : in TpServiceID

Identifies the affected service.

reason : in TpSvcUnavailReason

Identifies the reason why the service is no longer available

7.3.3.1.6 Method <<deprecated>> genFaultStatsRecordRes()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordRes shall be used instead, using the new identifier to correlate requests and responses.

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.

7.3.3.1.7 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

7.3.3.1.8 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.

7.3.3.1.9 Method <<deprecated>> genFaultStatsRecordErr()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordErr shall be used instead, using the new identifier to correlate requests and errors.

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.

7.3.3.1.10 Method appUnavailableInd()

The framework invokes this method to indicate to the application that the service instance has detected that it is not responding.

Parameters

serviceID : in TpServiceID

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

7.3.3.1.11 Method <<deprecated>> genFaultStatsRecordReq()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordReq shall be used instead, using the new identifier to correlate requests and responses.

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. Supplying both a start time and stop time as empty strings leaves the time period to the discretion of the client application.

7.3.3.1.12 Method <<new>> svcAvailStatusInd()

The framework invokes this method to inform the client application about the Service instance availability status, i.e. that it can no longer use its instance of the indicated service according to the reason parameter but as well information when the Service Instance becomes available again. On receipt of this request, the client application either acts 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). The client application can also wait for the problem to be solved and just stop the usage of the service instance until the svcAvailStatusInd() is called again with the reason SERVICE_AVAILABLE.

Parameters

serviceID : in TpServiceID

Identifies the affected service.

reason : in TpSvcAvailStatusReason

Identifies the reason why the service is no longer available or that it has become available again.

7.3.3.1.13 Method <<new>> generateFaultStatisticsRecordRes()

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

Parameters

faultStatsReqID : in TpFaultReqID

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

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.

In the case where a list of services is present, this is an ordered list in which the location of the service in this list corresponds to the location of the related fault statistics in the TpFaultStatsRecord returned.

7.3.3.1.14 Method <<new>> generateFaultStatisticsRecordErr()

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

Parameters

faultStatsReqID : in TpFaultReqID

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

faultStatistics : in TpFaultStatsErrorList

The list of fault statistics errors returned.

serviceIDs : in TpServiceIDList

Specifies the framework or services that are included in the list of fault statistics errors returned. If the serviceIDs parameter is an empty list, then the fault statistics error relates to the framework.

In the case where a list of services is present, this is an ordered list in which the location of the service in this list corresponds to the location of the related fault statistics error in the TpFaultStatsErrorList returned.

7.3.3.1.15 Method <<new>> generateFaultStatisticsRecordReq()

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 generateFaultStatisticsRecordReq 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 generateFaultStatisticsRecordRes operation on the IpFaultManager interface.

Parameters

faultStatsReqID : in TpFaultReqID

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

timePeriod : in TpTimeInterval

The period over which the fault statistics are to be generated. Supplying both a start time and stop time as empty strings leaves the time period to the discretion of the client application.

********** End of Change # 1 **********

********** Start of Change # 2 **********

7.3.3.2 Interface Class IpFaultManager

Inherits from: IpInterface.

This interface is used by the application to inform the framework of events that affect the integrity of the framework and services, and to request information about the integrity of the system. The fault manager operations do not exchange callback interfaces as it is assumed that the client application supplies its Fault Management callback interface at the time it obtains the Framework's Fault Management interface, by use of the obtainInterfaceWithCallback operation on the IpAccess interface.

If the IpFaultManager interface is implemented by a Framework, at least one of these methods shall be implemented. If the Framework is capable of invoking the IpAppFaultManager.appActivityTestReq() method, it shall implement appActivityTestRes() and appActivityTestErr() in this interface. If the Framework is capable of invoking IpAppFaultManager.generate~~FaultStats~~FaultStatisticsRecordReq(), it shall implement generate~~FaultStats~~FaultStatisticsRecordRes() and generate~~FaultStats~~FaultStatisticsRecordErr() in this interface.

<<Interface>> IpFaultManager
activityTestReq (activityTestID : in TpActivityTestID, svcID : in TpServiceID) : void appActivityTestRes (activityTestID : in TpActivityTestID, activityTestResult : in TpActivityTestRes) : void svcUnavailableInd (serviceID : in TpServiceID) : void <<deprecated>> genFaultStatsRecordReq (timePeriod : in TpTimeInterval, serviceIDs : in TpServiceIDList) : void appActivityTestErr (activityTestID : in TpActivityTestID) : void <<deprecated>> appUnavailableInd (serviceID : in TpServiceID) : void <<deprecated>> genFaultStatsRecordRes (faultStatistics : in TpFaultStatsRecord) : void <<deprecated>> genFaultStatsRecordErr (faultStatisticsError : in TpFaultStatisticsError) : void <<new>> appAvailStatusInd (reason : in TpAppAvailStatusReason) : void <<new>> generateFaultStatisticsRecordReq (faultStatsReqID : in TpFaultReqID, timePeriod : in TpTimeInterval, serviceIDs : in TpServiceIDList) : void <<new>> generateFaultStatisticsRecordRes (faultStatsReqID : in TpFaultReqID, faultStatistics : in TpFaultStatsRecord) : void <<new>> generateFaultStatisticsRecordErr (faultStatsReqID : in TpFaultReqID, faultStatisticsError : in TpFaultStatisticsError) : void

7.3.3.2.1 Method activityTestReq()

The application invokes this method to test that the framework or its instance of a service is operational. On receipt of this request, the framework must carry out a test on itself or on the client's instance of the specified service, to check that it is operating correctly. The framework reports the test result by invoking the activityTestRes method on the IpAppFaultManager interface. If the application does not have access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID.

For security reasons the client application has access to the service ID rather than the service instance ID. However, as there is a one to one relationship between the client application and a service, i.e. there is only one service instance of

the specified service per client application, it is the obligation of the framework to determine the service instance ID from the service ID.

Parameters

activityTestID : in TpActivityTestID

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

svcID : in TpServiceID

Identifies either the framework or a service for testing. The framework is designated by an empty string.

Raises

TpCommonExceptions,P_INVALID_SERVICE_ID, P_UNAUTHORISED_PARAMETER_VALUE

7.3.3.2.2 Method appActivityTestRes()

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

Parameters

activityTestID : in TpActivityTestID

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

activityTestResult : in TpActivityTestRes

The result of the activity test.

Raises

TpCommonExceptions, P_INVALID_ACTIVITY_TEST_ID

7.3.3.2.3 Method svcUnavailableInd()

This method is used by the client application to inform the framework that it can no longer use its instance of the indicated service (either due to a failure in the client application or in the service instance itself). On receipt of this request, the framework should take the appropriate corrective action.

Parameters

serviceID : in TpServiceID

Identifies the service that the application can no longer use.

Raises

TpCommonExceptions ,P_INVALID_SERVICE_ID, P_UNAUTHORISED_PARAMETER_VALUE

7.3.3.2.4 Method <<deprecated>> genFaultStatsRecordReq()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordReq shall be used instead, using the new identifier to correlate requests and responses.

This method is used by the application to solicit fault statistics from the framework. On receipt of this request the framework must produce a fault statistics record, for either the framework or for the client's instances of the specified services during the specified time interval, which is returned to the client application using the genFaultStatsRecordRes operation on the IpAppFaultManager interface. If the application does not have access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID.

Parameters

`timePeriod : in TpTimeInterval`

The period over which the fault statistics are to be generated. Supplying both a start time and stop time as empty strings leaves the time period to the discretion of the framework.

`serviceIDs : in TpServiceIDList`

Specifies either the framework or services to be included in the general fault statistics record. If this parameter is not an empty list, the fault statistics records of the client's instances of the specified services are returned, otherwise the fault statistics record of the framework is returned.

Raises

`TpCommonExceptions, P_INVALID_SERVICE_ID, P_UNAUTHORISED_PARAMETER_VALUE`

7.3.3.2.5 Method appActivityTestErr()

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

Parameters

`activityTestID : in TpActivityTestID`

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

Raises

`TpCommonExceptions, P_INVALID_ACTIVITY_TEST_ID`

7.3.3.2.6 Method <<deprecated>> appUnavailableInd()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. Applications can indicate they no longer use a particular service instance using IpServiceAgreementManagement.terminateServiceAgreement(). Applications can indicate a fault with a particular service instance using IpFaultManager.svcUnavailableInd().

This method is used by the application to inform the framework that it is ceasing its use of the service instance. This may a result of the application detecting a failure. The framework assumes that the session between this client application and service instance is to be closed and updates its own records appropriately as well as attempting to inform the service instance and/or its administrator.

Parameters

serviceID : in TpServiceID

Identifies the affected application.

Raises

TpCommonExceptions

7.3.3.2.7 Method <<deprecated>> genFaultStatsRecordRes()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordRes shall be used instead, using the new identifier to correlate requests and responses.

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

Parameters

faultStatistics : in TpFaultStatsRecord

The fault statistics record.

Raises

TpCommonExceptions

7.3.3.2.8 Method <<deprecated>> genFaultStatsRecordErr()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordErr shall be used instead, using the new identifier to correlate requests and errors.

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

Parameters

faultStatisticsError : in TpFaultStatisticsError

The fault statistics error.

Raises

TpCommonExceptions

7.3.3.2.9 Method <<new>> appAvailStatusInd()

This method is used by the application to inform the framework of its availability status. If the Application has detected a failure it uses one of the APP_UNAVAILABLE reason types to indicate the problem and that it is ceasing its use of all of its subscribed service instances. When the Application is working again it shall call this method again with the APP_AVAILABLE reason to inform the Framework that it is working properly again. The Framework shall also attempt to inform all of the service instances used by the specific application and/or its administrator about the problem.

Parameters

reason : in TpAppAvailStatusReason

Identifies the reason why the application is no longer available. APP_AVAILABLE is used to inform the Framework and the Service that the Application is available again.

Raises

TpCommonExceptions

7.3.3.2.10 Method <<new>> generateFaultStatisticsRecordReq()

This method is used by the application to solicit fault statistics from the framework. On receipt of this request the framework must produce a fault statistics record, for either the framework or for the client's instances of the specified services during the specified time interval, which is returned to the client application using the generateFaultStatisticsRecordRes operation on the IpAppFaultManager interface. If the application does not have access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID.

Parameters

faultStatsReqID : in TpFaultReqID

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

timePeriod : in TpTimeInterval

The period over which the fault statistics are to be generated. Supplying both a start time and stop time as empty strings leaves the time period to the discretion of the framework.

serviceIDs : in TpServiceIDList

Specifies either the framework or services to be included in the general fault statistics record. If this parameter is not an empty list, the fault statistics records of the client's instances of the specified services are returned, otherwise the fault statistics record of the framework is returned.

Raises

TpCommonExceptions, P_INVALID_SERVICE_ID, P_UNAUTHORISED_PARAMETER_VALUE

7.3.3.2.11 Method <<new>> generateFaultStatisticsRecordRes()

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

Parameters

faultStatsReqID : in TpFaultReqID

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

faultStatistics : in TpFaultStatsRecord

The fault statistics record.

Raises

TpCommonExceptions

7.3.3.2.12 Method <<new>> generateFaultStatisticsRecordErr()

This method is used by the client application to indicate an error fulfilling the request to provide fault statistics, in response to a generateFaultStatisticsRecordReq method invocation on the IpAppFaultManager interface.

Parameters

faultStatsReqID : in TpFaultReqID

Used by the framework to correlate this error (when it arrives) with the original request.

faultStatisticsError : in TpFaultStatisticsError

The fault statistics error.

Raises

TpCommonExceptions

********** End of Change # 2 **********

********** Start of Change # 3 **********

8.3.4.1 Interface Class IpFwFaultManager

Inherits from: IpInterface.

This interface is used by the service instance to inform the framework of events which affect the integrity of the API, and request fault management status information from the framework. The fault manager operations do not exchange callback interfaces as it is assumed that the service instance has supplied its Fault Management callback interface at the time it obtains the Framework's Fault Management interface, by use of the obtainInterfaceWithCallback operation on the IpAccess interface.

If the IpFwFaultManager interface is implemented by a Framework, at least one of these methods shall be implemented. If the Framework is capable of invoking the IpSvcFaultManager.svcActivityTestReq() method, it shall implement svcActivityTestRes() and svcActivityTestErr() in this interface. If the Framework is capable of invoking IpSvcFaultManager.generate~~FaultStats~~FaultStatisticsRecordReq(), it shall implement generate~~FaultStats~~FaultStatisticsRecordRes() and generate~~FaultStats~~FaultStatisticsRecordErr() in this interface. If the Framework is capable of invoking IpSvcFaultManager.generate~~FaultStats~~FaultStatisticsRecordReq(), it shall implement generate~~FaultStats~~FaultStatisticsRecordRes() and generate~~FaultStats~~FaultStatisticsRecordErr() in this interface.

<<Interface>>

IpFwFaultManager

activityTestReq (activityTestID : in TpActivityTestID, testSubject : in TpSubjectType) : void

svcActivityTestRes (activityTestID : in TpActivityTestID, activityTestResult : in TpActivityTestRes) : void

appUnavailableInd () : void

<<deprecated>> genFaultStatsRecordReq (timePeriod : in TpTimeInterval, recordSubject : in TpSubjectType) : void

<<deprecated>> svcUnavailableInd (reason : in TpSvcUnavailReason) : void

svcActivityTestErr (activityTestID : in TpActivityTestID) : void

<<deprecated>> genFaultStatsRecordRes (faultStatistics : in TpFaultStatsRecord, serviceIDs : in TpServiceIDList) : void

<<deprecated>> genFaultStatsRecordErr (faultStatisticsError : in TpFaultStatisticsError, serviceIDs : in TpServiceIDList) : void

<<deprecated>> ~~<<new>>~~ generateFaultStatsRecordRes (faultStatistics : in TpFaultStatsRecord) : void

<<deprecated>> ~~<<new>>~~ generateFaultStatsRecordErr (faultStatisticsError : in TpFaultStatisticsError) : void

<<new>> svcAvailStatusInd (reason : in TpSvcAvailStatusReason) : void

<<new>> generateFaultStatisticsRecordReq (faultStatsReqID : in TpFaultReqID, timePeriod : in TpTimeInterval, recordSubject : in TpSubjectType) : void

<<new>> generateFaultStatisticsRecordRes (faultStatsReqID : in TpFaultReqID, faultStatistics : in TpFaultStatsRecord) : void

<<new>> generateFaultStatisticsRecordErr (faultStatsReqID : in TpFaultReqID, faultStatisticsError : in TpFaultStatisticsError) : void

8.3.4.1.1 Method activityTestReq()

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

Parameters

`activityTestID : in TpActivityTestID`

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

`testSubject : in TpSubjectType`

Identifies the subject for testing (framework or client application).

Raises

`TpCommonExceptions`

8.3.4.1.2 Method svcActivityTestRes()

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

Parameters

`activityTestID : in TpActivityTestID`

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

`activityTestResult : in TpActivityTestRes`

The result of the activity test.

Raises

`TpCommonExceptions,P_INVALID_ACTIVITY_TEST_ID`

8.3.4.1.3 Method appUnavailableInd()

This method is used by the service instance to inform the framework that the client application is not responding. On receipt of this indication, the framework must act to inform the client application.

Parameters

No Parameters were identified for this method

Raises

`TpCommonExceptions`

8.3.4.1.4 Method <<deprecated>> genFaultStatsRecordReq()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordReq shall be used instead, using the new identifier to correlate requests and responses.

This method is used by the service instance to solicit fault statistics from the framework. On receipt of this request, the framework must produce a fault statistics record, for the framework or for the application during the specified time interval, which is returned to the service instance using the genFaultStatsRecordRes operation on the IpSvcFaultManager interface.

Parameters

`timePeriod : in TpTimeInterval`

The period over which the fault statistics are to be generated. Supplying both a start time and stop time as empty strings leaves the time period to the discretion of the framework.

`recordSubject : in TpSubjectType`

Specifies the subject to be included in the general fault statistics record (framework or application).

Raises

`TpCommonExceptions`

8.3.4.1.5 Method <<deprecated>> svcUnavailableInd()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method svcAvailStatusInd() shall be used instead, using the new and updated reason parameter to inform the Framework the reason why the Service has become unavailable and also when the Service instance becomes available again.

This method is used by the service instance to inform the framework that it is about to become unavailable for use. The framework should inform the client application that is currently using this service instance that it is unavailable for use (via the svcUnavailableInd method on the IpAppFaultManager interface).

Parameters

`reason : in TpSvcUnavailReason`

Identifies the reason for the service instance's unavailability.

Raises

`TpCommonExceptions`

8.3.4.1.6 Method svcActivityTestErr()

The service instance uses this method to indicate that an error occurred during a framework-requested activity test.

Parameters

`activityTestID : in TpActivityTestID`

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

Raises

`TpCommonExceptions, P_INVALID_ACTIVITY_TEST_ID`

8.3.4.1.7 Method <<deprecated>> genFaultStatsRecordRes()

This method is deprecated and will be removed in a later release. It cannot be used as described, since the serviceIDs parameter has no meaning. It is replaced with generateFaultStatsRecordRes().

This method is used by the service to provide fault statistics to the framework in response to a genFaultStatsRecordReq method invocation on the IpSvcFaultManager interface.

Parameters

`faultStatistics : in TpFaultStatsRecord`

The fault statistics record.

`serviceIDs : in TpServiceIDList`

Specifies the services that are included in the general fault statistics record. The serviceIDs parameter is not allowed to be an empty list.

Raises

TpCommonExceptions

8.3.4.1.8 Method <<deprecated>> genFaultStatsRecordErr()

This method is deprecated and will be removed in a later release. It cannot be used as described, since the serviceIDs parameter has no meaning. It is replaced with generateFaultStatsRecordErr().

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

Parameters

faultStatisticsError : in TpFaultStatisticsError

The fault statistics error.

serviceIDs : in TpServiceIDList

Specifies the services that were included in the general fault statistics record request. The serviceIDs parameter is not allowed to be an empty list.

Raises

TpCommonExceptions

8.3.4.1.9 Method <<deprecated>> ~~<<new>>~~ generateFaultStatsRecordRes()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordRes shall be used instead, using the new identifier to correlate requests and responses.

This method is used by the service to provide fault statistics to the framework in response to a genFaultStatsRecordReq method invocation on the IpSvcFaultManager interface.

Parameters

faultStatistics : in TpFaultStatsRecord

The fault statistics record.

Raises

TpCommonExceptions

8.3.4.1.10 Method <<deprecated>> ~~<<new>>~~ generateFaultStatsRecordErr()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordErr shall be used instead, using the new identifier to correlate requests and errors.

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

Parameters

faultStatisticsError : in TpFaultStatisticsError

The fault statistics error.

Raises

TpCommonExceptions

8.3.4.1.11 Method <<new>> svcAvailStatusInd()

This method is used by the service instance to inform the framework that it is about to become unavailable for use according to the provided reason and as well to inform the Framework when the Service instance becomes available again. The framework should inform the client applications that are currently using this service instance that it is unavailable and as well when it becomes available again for use (via the svcAvailStatusInd method on the IpAppFaultManager interface).

Parameters

reason : in TpSvcAvailStatusReason

Identifies the reason for the service instance's unavailability and also the reason SERVICE_AVAILABLE to be used to inform the Framework when the Service instance becomes available again.

Raises

TpCommonExceptions

8.3.4.1.12 Method <<new>> generateFaultStatisticsRecordReq()

This method is used by the service instance to solicit fault statistics from the framework. On receipt of this request, the framework must produce a fault statistics record, for the framework or for the application during the specified time interval, which is returned to the service instance using the generateFaultStatisticsRecordRes operation on the IpSvcFaultManager interface.

Parameters

faultStatsReqID : in TpFaultReqID

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

timePeriod : in TpTimeInterval

The period over which the fault statistics are to be generated. Supplying both a start time and stop time as empty strings leaves the time period to the discretion of the framework.

recordSubject : in TpSubjectType

Specifies the subject to be included in the general fault statistics record (framework or application).

Raises

TpCommonExceptions

8.3.4.1.13 Method <<new>> generateFaultStatisticsRecordRes()

This method is used by the service to provide fault statistics to the framework in response to a generateFaultStatisticsRecordReq method invocation on the IpSvcFaultManager interface.

Parameters

faultStatsReqID : in TpFaultReqID

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

faultStatistics : in TpFaultStatsRecord

The fault statistics record.

Raises

TpCommonExceptions

8.3.4.1.14 Method <<new>> generateFaultStatisticsRecordErr()

This method is used by the service to indicate an error fulfilling the request to provide fault statistics, in response to a generateFaultStatisticsRecordReq method invocation on the IpSvcFaultManager interface.

Parameters

faultStatsReqID : in TpFaultReqID

Used by the framework to correlate this error (when it arrives) with the original request.

faultStatisticsError : in TpFaultStatisticsError

The fault statistics error.

Raises

TpCommonExceptions

********** End of Change # 3 **********

********** Start of Change # 4 **********

8.3.4.2 Interface Class IpSvcFaultManager

Inherits from: IpInterface.

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

If the IpSvcFaultManager interface is implemented by a Service, at least one of these methods shall be implemented. If the Service is capable of invoking the IpFwFaultManager.activityTestReq() method, it shall implement activityTestRes() and activityTestErr() in this interface. If the Service is capable of invoking

IpFwFaultManager.generateFaultStatisticsRecordReq(), it shall implement generateFaultStatisticsRecordRes() and generateFaultStatisticsRecordErr() in this interface.

<<Interface>>

IpSvcFaultManager

activityTestRes (activityTestID : in TpActivityTestID, activityTestResult : in TpActivityTestRes) : void

svcActivityTestReq (activityTestID : in TpActivityTestID) : void

fwFaultReportInd (fault : in TpInterfaceFault) : void

fwFaultRecoveryInd (fault : in TpInterfaceFault) : void

fwUnavailableInd (reason : in TpFwUnavailReason) : void

svcUnavailableInd () : void

<<deprecated>> appUnavailableInd () : void

<<deprecated>> genFaultStatsRecordRes (faultStatistics : in TpFaultStatsRecord, recordSubject : in TpSubjectType) : void

activityTestErr (activityTestID : in TpActivityTestID) : void

<<deprecated>> genFaultStatsRecordErr (faultStatisticsError : in TpFaultStatisticsError, recordSubject : in TpSubjectType) : void

<<deprecated>> genFaultStatsRecordReq (timePeriod : in TpTimeInterval, serviceIDs : in TpServiceIDList) : void

<<deprecated>> ~~<<new>>~~ generateFaultStatsRecordReq (timePeriod : in TpTimeInterval) : void

<<new>> appAvailStatusInd (reason : in TpAppAvailStatusReason) : void

<<new>> generateFaultStatisticsRecordRes (faultStatsReqID : in TpFaultReqID, faultStatistics : in TpFaultStatsRecord, recordSubject : in TpSubjectType) : void

<<new>> generateFaultStatisticsRecordErr (faultStatsReqID : in TpFaultReqID, faultStatisticsError : in TpFaultStatisticsError, recordSubject : in TpSubjectType) : void

<<new>> generateFaultStatisticsRecordReq (faultStatsReqID : in TpFaultReqID, timePeriod : in TpTimeInterval) : void

8.3.4.2.1 Method activityTestRes()

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

Parameters

`activityTestID : in TpActivityTestID`

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

`activityTestResult : in TpActivityTestRes`

The result of the activity test.

Raises

TpCommonExceptions,P_INVALID_ACTIVITY_TEST_ID

8.3.4.2.2 Method svcActivityTestReq()

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

Parameters

activityTestID : in TpActivityTestID

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

Raises

TpCommonExceptions

8.3.4.2.3 Method fwFaultReportInd()

The framework invokes this method to notify the service instance of a failure within the framework. The service instance 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.

Raises

TpCommonExceptions

8.3.4.2.4 Method fwFaultRecoveryInd()

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

Parameters

fault : in TpInterfaceFault

Specifies the fault from which the framework has recovered.

Raises

TpCommonExceptions

8.3.4.2.5 Method fwUnavailableInd()

The framework invokes this method to inform the service instance that it is no longer available.

Parameters

`reason : in TpFwUnavailReason`

Identifies the reason why the framework is no longer available

Raises

`TpCommonExceptions`

8.3.4.2.6 Method svcUnavailableInd()

The framework invokes this method to inform the service instance that the client application has reported that it can no longer use the service instance.

Parameters

No Parameters were identified for this method

Raises

`TpCommonExceptions`

8.3.4.2.7 Method <<deprecated>> appUnavailableInd()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method appAvailStatusInd shall be used instead, using the new reason parameter to inform the Service the reason why the Application is unavailable and also when the application becomes available again.

The framework invokes this method to inform the service instance that the framework may have detected that the application has failed: e.g. non-response from an activity test, failure to return heartbeats.

Parameters

No Parameters were identified for this method

Raises

`TpCommonExceptions`

8.3.4.2.8 Method <<deprecated>> genFaultStatsRecordRes()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordRes shall be used instead, using the new identifier to correlate requests and responses.

This method is used by the framework to provide fault statistics to a service instance in response to a genFaultStatsRecordReq method invocation on the IpFwFaultManager interface.

Parameters

faultStatistics : in TpFaultStatsRecord

The fault statistics record.

recordSubject : in TpSubjectType

Specifies the entity (framework or application) whose fault statistics record has been provided.

Raises

TpCommonExceptions

8.3.4.2.9 Method activityTestErr()

The framework uses this method to indicate that an error occurred during a service-requested activity test.

Parameters

activityTestID : in TpActivityTestID

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

Raises

TpCommonExceptions, P_INVALID_ACTIVITY_TEST_ID

8.3.4.2.10 Method <<deprecated>> genFaultStatsRecordErr()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordErr shall be used instead, using the new identifier to correlate requests and errors.

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 IpFwFaultManager interface.

Parameters

faultStatisticsError : in TpFaultStatisticsError

The fault statistics error.

recordSubject : in TpSubjectType

Specifies the entity (framework or application) whose fault statistics record was requested.

Raises

TpCommonExceptions

8.3.4.2.11 Method <<deprecated>> genFaultStatsRecordReq()

This method is deprecated and will be removed in a later release. It cannot be used as described, since the serviceIDs parameter has no meaning. It is replaced with generateFaultStatsRecordReq().

This method is used by the framework to solicit fault statistics from the service, for example when the framework was asked for these statistics by the client application using the genFaultStatsRecordReq operation on the IpFaultManager interface. On receipt of this request the service must produce a fault statistics record, for either the framework or for the client's instances of the specified services during the specified time interval, which is returned to the framework using the genFaultStatsRecordRes operation on the IpFwFaultManager interface. If the framework does not have access to a service instance with the specified serviceID, the P_UNAUTHORISED_PARAMETER_VALUE exception shall be thrown. The extraInformation field of the exception shall contain the corresponding serviceID.

Parameters

timePeriod : in TpTimeInterval

The period over which the fault statistics are to be generated. Supplying both a start time and stop time as empty strings leaves the time period to the discretion of the service.

serviceIDs : in TpServiceIDList

Specifies the services to be included in the general fault statistics record. This parameter is not allowed to be an empty list.

Raises

TpCommonExceptions, P_INVALID_SERVICE_ID, P_UNAUTHORISED_PARAMETER_VALUE

8.3.4.2.12 Method <<deprecated>> ~~<<new>>~~ generateFaultStatsRecordReq()

This method is deprecated and will be removed in a later release. It is strongly recommended not to implement this method. The new method generateFaultStatisticsRecordReq shall be used instead, using the new identifier to correlate requests and responses.

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

Parameters

timePeriod : in TpTimeInterval

The period over which the fault statistics are to be generated. Supplying both a start time and stop time as empty strings leaves the time period to the discretion of the service.

Raises

TpCommonExceptions

8.3.4.2.13 Method <<new>> appAvailStatusInd()

The framework invokes this method to inform the service instance that the client application is no longer available using different reasons for the unavailability. This may be a result of the application reporting a failure. Alternatively, the framework may have detected that the application has failed: e.g. non-response from an activity test, failure to return heartbeats, using the reason APP_UNAVAILABLE_NO_RESPONSE. When the application becomes available again the reason APP_AVAILABLE shall be used to inform the Service about that.

Parameters

reason : in TpAppAvailStatusReason

Identifies the reason why the application is no longer available. APP_AVAILABLE is used to inform the Service that the Application is available again.

Raises

TpCommonExceptions

8.3.4.2.14 Method <<new>> generateFaultStatisticsRecordRes()

This method is used by the framework to provide fault statistics to a service instance in response to a generateFaultStatisticsRecordReq method invocation on the IpFwFaultManager interface.

Parameters

faultStatsReqID : in TpFaultReqID

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

faultStatistics : in TpFaultStatsRecord

The fault statistics record.

recordSubject : in TpSubjectType

Specifies the entity (framework or application) whose fault statistics record has been provided.

Raises

TpCommonExceptions

8.3.4.2.15 Method <<new>> generateFaultStatisticsRecordErr()

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

Parameters

faultStatsReqID : in TpFaultReqID

Used by the service instance to correlate this error (when it arrives) with the original request.

faultStatisticsError : in TpFaultStatisticsError

The fault statistics error.

recordSubject : in TpSubjectType

Specifies the entity (framework or application) whose fault statistics record was requested.

Raises

TpCommonExceptions

8.3.4.2.16 Method <<new>> generateFaultStatisticsRecordReq()

This method is used by the framework to solicit fault statistics from the service instance, for example when the framework was asked for these statistics by the client application using the generateFaultStatisticsRecordReq operation

on the IpFaultManager interface. On receipt of this request the service instance must produce a fault statistics record during the specified time interval, which is returned to the framework using the generateFaultStatisticsRecordRes operation on the IpFwFaultManager interface.

Parameters

faultStatsReqID : in TpFaultReqID

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

timePeriod : in TpTimeInterval

The period over which the fault statistics are to be generated. Supplying both a start time and stop time as empty strings leaves the time period to the discretion of the service.

Raises

TpCommonExceptions

********** End of Change # 4 **********

********** Start of Change # 5 **********

10.4 Integrity Management Data Definitions

10.4.1 TpActivityTestRes

This type is identical to TpString and is an implementation specific result. The values in this data type are "Available" or "Unavailable".

10.4.2 TpFaultStatsRecord

This defines the set of records to be returned giving fault information for the requested time period.

Sequence Element Name	Sequence Element Type
Period	TpTimeInterval
FaultStatsSet	TpFaultStatsSet

10.4.3 TpFaultStats

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

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

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

10.4.4 TpFaultStatsErrorList

Defines a Numbered List of Data Elements of type TpFaultStatisticsError.

~~10.4.4~~10.4.5 TpFaultStatisticsError

Defines the error code associated with a failed attempt to retrieve any fault statistics information.

Name	Value	Description
P_FAULT_INFO_ERROR_UNDEFINED	0	Undefined error
P_FAULT_INFO_UNAVAILABLE	1	Fault statistics unavailable

~~10.4.5~~10.4.6 TpFaultStatsSet

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

~~10.4.6~~10.4.7 TpActivityTestID

This data type is identical to a TpInt32, and is used as a token to match activity test requests with their results..

10.4.8 TpFaultReqID

This data type is identical to a TpInt32, and is used as a token to match fault statistics requests with their results.

********** End of Change # 5 **********

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	--	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	4.0.0
Jun 2001	CN_12	NP-010330	001	--	Corrections to OSA API Rel4	4.0.0	4.0.1
Sep 2001	CN_13	NP-010466	002	--	Changing references to JAIN	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	003	--	Update to the definitions of method svcUnavailableInd	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	004	--	Only one subject per method invocation for fault and load management	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	005	--	Fault management is missing some *Err methods	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	006	--	Method balance on Fault management interfaces	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	007	--	Change "TpString" into "TpOctetSets" in authentication and access	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	008	--	Replacement of register/unregisterLoadController	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	009	--	Redundant Framework Heartbeat Mechanism	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	010	--	Add a releaseInterface() method to IpAccess	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	011	--	Removal of serviceID from queryAppLoadReq()	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	012	--	Addition of listInterfaces() method	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	013	--	Introduction and use of new Service Instance ID	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	014	--	P_UNAUTHORISED_PARAMETER_VALUE thrown if non-accessible serviceID is provided	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	015	--	Introduction of Service Instance Lifecycle Management	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	016	--	Add support for multi-vendorship	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	017	--	Removal of P_SERVICE_ACCESS_TYPE	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	018	--	Confusing meaning of prescribedMethod	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	019	--	A client should only have one instance of a given service	4.1.0	4.2.0
Sep 2001	CN_13	NP-010466	020	--	Some methods on the IpApp interfaces should throw exceptions	4.1.0	4.2.0
Dec 2001	CN_14	NP-010596	021	--	Replace Out Parameters with Return Types	4.2.0	4.3.0
Dec 2001	CN_14	NP-010596	022	--	Correctionto Framework (FW)	4.2.0	4.3.0
Mar 2002	CN_15	NP-020105	023	--	Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID()	4.3.0	4.4.0
Mar 2002	CN_15	NP-020105	024	--	Replace erroneous mention of P_OSA_ACCESS by the correct value P_OSA_AUTHENTICATION	4.3.0	4.4.0
Mar 2002	CN_15	NP-020105	025	--	Add missing inheritance in service agreement management interfaces	4.3.0	4.4.0
Mar 2002	CN_15	NP-020105	026	--	Include Operation Set as part of General Service Properties	4.3.0	4.4.0
Mar 2002	CN_15	NP-020105	027	--	Improved description of activityTestReq with respect to ServiceInstanceID	4.3.0	4.4.0
Mar 2002	CN_15	NP-020105	028	--	OSA Framework - Generate statistics records on behalf of another entity using genFaultStatsRecordReq	4.3.0	4.4.0
Mar 2002	CN_15	NP-020105	029	--	Update the interface names for alignment between 3GPP and ETSI/Parlay	4.3.0	4.4.0
Jun 2002	CN_16	NP-020179	030	--	Solving the problem in the OSA Framework with method appUnavailableInd() in a scenario with multiple service sessions per access session	4.4.0	4.5.0
Jun 2002	CN_16	NP-020179	031	--	Adding missing mandatory method (authenticationSucceeded) to sequence flow	4.4.0	4.5.0
Jun 2002	CN_16	NP-020186	032	--	Remove redundant data type definition TpServiceSpecString	4.5.0	5.0.0
Jun 2002	CN_16	NP-020181	033	--	Addition of support for Java API technology realisation	4.5.0	5.0.0
Jun 2002	CN_16	NP-020182	035	--	Addition of support for WSDL realisation	4.5.0	5.0.0
Jun 2002	CN_16	NP-020186	036	--	Clarify semantics of service properties of type BOOLEAN_SET	4.5.0	5.0.0
Jun 2002	CN_16	NP-020186	037	--	Addition of version management support to the Framework (29.198-03) in run-time	4.5.0	5.0.0
Jun 2002	CN_16	NP-020186	038	--	Enhancements on subscription management error information	4.5.0	5.0.0
Jun 2002	CN_16	NP-020186	039	--	Delete conflicting description of P_APPLICATION_NOT_ACTIVATED	4.5.0	5.0.0
Jun 2002	CN_16	NP-020186	040	--	Note added for P_SERVICE_INSTANCE Choice Element Name	4.5.0	5.0.0
Jun 2002	CN_16	NP-020186	041	--	Correcting the method descriptions for abortAuthentication and for initiateAuthentication	4.5.0	5.0.0
Jun 2002	CN_16	NP-020186	042	--	Correcting the description of heartbeat failure	4.5.0	5.0.0
Jun 2002	CN_16	NP-020186	043	--	Correcting erroneous FW<->Service instance sequence diagrams	4.5.0	5.0.0
Jun 2002	CN_16	NP-020186	044	--	Correcting the scope of TpFwID, which currently is giving it false limitations	4.5.0	5.0.0
Sep 2002	CN_17	NP-020428	046		Correction to description of TpServicePropertyTypeName	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	047		Remove undefined exception in registerService	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	048		Remove ServiceIDs from IpFwFaultManager.genFaultStatsRecordReq()	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	049		Correct appUnavailableInd and related methods	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	050		Remove unusable exception from IpFaultManager.appActivityTestRes()	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	051		Clarify the sequence of events in signing the service agreement	5.0.0	5.1.0

Sep 2002	CN_17	NP-020428	052		Correct use of electronic signatures	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	053		Addition of Sequence Diagrams for terminateAccess	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	054		Add indication what part of service agreement must be signed	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	055		Add text to clarify requirements on support of methods	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	056		Introduce types and modes for generic properties	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	057		Correction on use of NULL in Framework API	5.0.0	5.1.0
Sep 2002	CN_17	NP-020428	058		Add Negotiation of Authentication Mechanism for OSA level Authentication	5.0.0	5.1.0
Sep 2002	CN_17	NP-020395	058		Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications	5.0.0	5.1.0
Mar 2003	CN_19	NP-030019	063	-	Correction to Initial Access Sequence Diagram	5.1.0	5.2.0
Mar 2003	CN_19	NP-030019	065	-	Enable creation/destruction of load level notifications at the request of Framework	5.1.0	5.2.0
Mar 2003	CN_19	NP-030019	067	-	Correction of Sequence for Framework – Service load management	5.1.0	5.2.0
Mar 2003	CN_19	NP-030019	074	-	Add Initial Load Notification report for Framework Integrity Management Load Notification model	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	068	--	Correction to Application's requirements for supporting methods	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	069	--	Correction of status of methods to interfaces in clause 7.3	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	070	--	Correction of status of methods to interfaces in clause 8.3	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	071	--	Correction of status of methods to interfaces in clause 6.3	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	075	--	Adding the appAvailStatusInd() and svcAvailStatusInd() methods	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	076	--	Remove race condition in signServiceAgreement	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	077	--	Change reference to deprecated method "authenticate" in TpAuthMechanism to "challenge"	5.1.0	5.2.0
Jun 2003	CN_20	NP-030237	079	--	Correction to TpEncryptionCapability to correct support for Triple-DES	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	081	--	Correction of the Framework Service Instance Lifecycle Manager Sequence Diagram	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	083	--	Correction of the use of TpDomainID in Framework initiateAuthentication method	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	085	--	Correction to Java Realisation Annex	5.3.0	5.4.0

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-030633
Meeting #25, Bangkok, Thailand, 27 – 31 October 2003

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03** **CR** **090** ⌘ **rev** **-** ⌘ Current version: **5.4.0** ⌘

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

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

Title:	⌘	Correction and Clarification of Framework Access Session Behaviour
Source:	⌘	CN5 (AePONA – Eamonn Murray)
Work item code:	⌘	OSA2
Date:	⌘	31/10/2003
Category:	⌘	**F**
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:	⌘	Section 6 of the Framework specification contains a number of clauses in which the definition of the Framework Access session and intended behaviour is either misleading or lacks sufficient clarity regarding the behaviour intended and the possible uses of the Access session.
Summary of change:	⌘	Introduce additional clarifying text and correct misleading statements or references.
Consequences if not approved:	⌘	Ambiguity around the intended use of the Framework Access session shall result and give rise to interoperabilty and incompatibility problems for vendors and implementors.

Clauses affected:	⌘	6.1, 6.3.1.3, 6.3.1.4, 6.3.1.5

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:

********** Start of Change # 1 **********

6.1 Sequence Diagrams

6.1.1 Trust and Security Management Sequence Diagrams

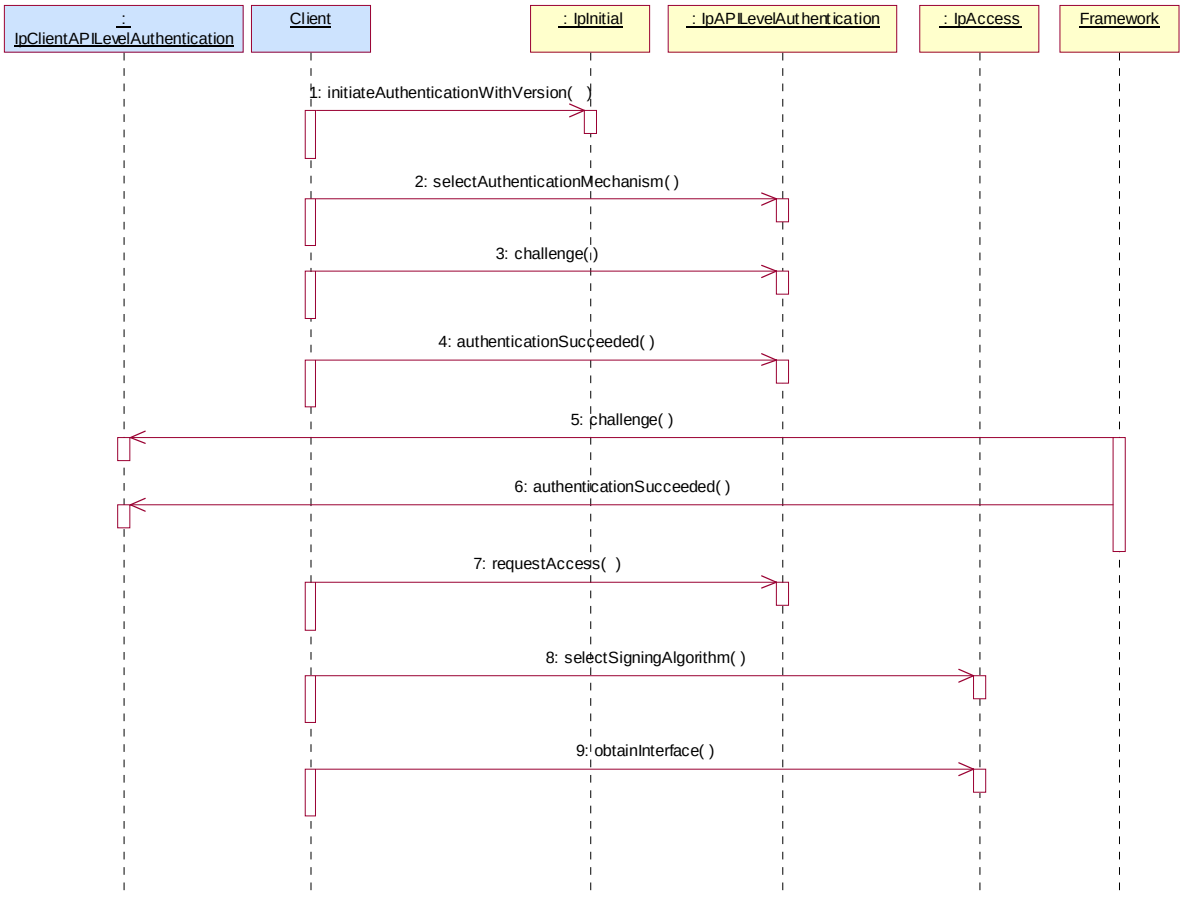
6.1.1.1 Initial Access

The following figure shows a client accessing the OSA Framework for the first time.

Before being authorized to use the OSA SCFs, the client must first of all authenticate itself with the Framework. For this purpose the client needs a reference to the Initial Contact interfaces for the Framework; this may be obtained through a URL, a Naming or Trading Service or an equivalent service, a stringified object reference, etc. At this stage, the client has no guarantee that this is a Framework interface reference, but it is to initiate the authentication process with the Framework. The Initial Contact interface supports ~~only~~ the initiateAuthenticationWithVersion and the deprecated initiateAuthentication methods to allow the authentication process to take place.

Once the client has been authenticated by the Framework, it can gain access to other framework interfaces and SCFs. This is done by invoking the requestAccess method, by which the client requests a certain type of access SCF.

Independently, the client could decide to authenticate the Framework, before deciding to continue using the interfaces provided by the Framework.



1: Initiate Authentication

The client invokes initiateAuthenticationWithVersion on the Framework's "public" (initial contact) interface to initiate the authentication process. It provides in turn a reference to its own authentication interface. The Framework returns a reference to its authentication interface.

2: Select Authentication Mechanism

The client invokes selectAuthenticationMechanism on the Framework's API Level Authentication interface, identifying the authentication algorithm it supports for use with CHAP authentication. The Framework prescribes the method to be used. OSA authentication is based on CHAP, which prescribes the MD5 hashing algorithm as the minimum to be supported. Note however that the framework need not accept this algorithm.

3: The client authenticates the Framework, issuing a challenge in the challenge() method.

4: The client provides an indication if authentication succeeded.

5: The Framework authenticates the client. The sequence diagram illustrates one of a series of one or more invocations of the challenge method on the client's API Level Authentication interface. In each invocation, the Framework supplies a challenge and the client returns the correct response. The Framework could authenticate the client before the client authenticates the Framework, or afterwards, or the two authentication processes could be interleaved. However, the client shall respond immediately to any challenge issued by the Framework, as the Framework might not respond to any challenge issued by the client until the Framework has successfully authenticated the client.

6: The Framework provides an indication if authentication succeeded.

7: Request Access

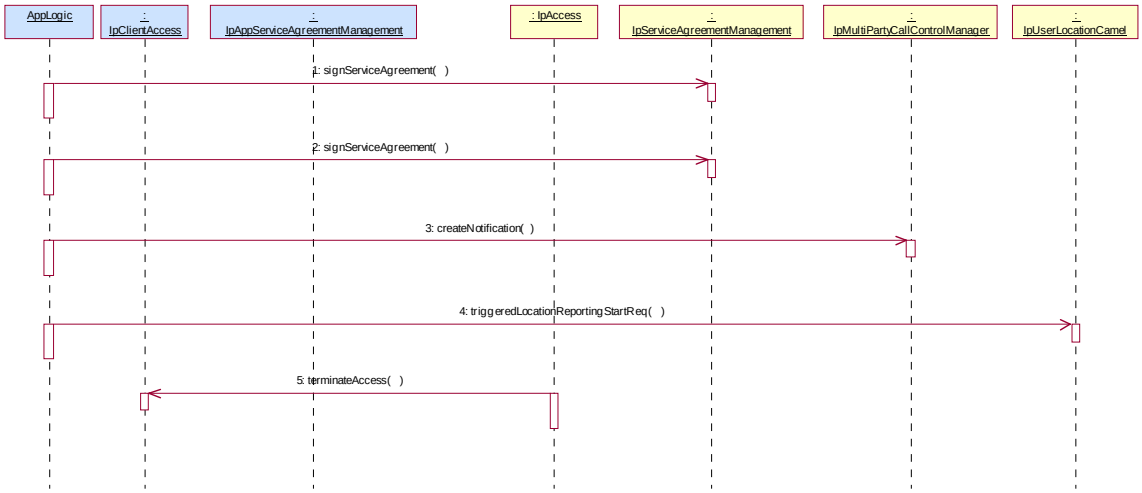
Upon successful authentication of the client by the Framework, the client is permitted to invoke requestAccess on the Framework's API Level Authentication interface, providing in turn a reference to its own access interface. The Framework returns a reference to ~~its~~ a framework ~~access~~ Access interface that is unique for this client. The success or failure of the client's authentication of the Framework does not affect the client's right to invoke requestAccess.

8: The client and framework negotiate the signing algorithm to be used for any signed exchanges.

9: The client invokes obtainInterface or obtainInterfaceWithCallback on the framework's Access interface. This is used to obtain a reference to a framework interface that supports the required framework functionality, such as service discovery, integrity management, service subscription etc. ~~to obtain a reference to its service discovery interface.~~

6.1.1.2 Framework Terminates Access

This sequence shows how a Framework could terminate an application's use of the Framework and of all service instances. This type of termination is unusual, but possible with the terminateAccess method. Note that if at any point the framework's level of confidence in the identity of the client becomes too low, perhaps due to re-authentication failing, the framework should terminate all outstanding service agreements for that client, and should take steps to terminate the client's access session WITHOUT invoking terminateAccess() on the client. This follows a generally accepted security model where the framework has decided that it can no longer trust the client and will therefore sever ALL contact with it.



1: Following successful authentication and service discovery, the client initiates the service agreement signing process (not shown). This is completed when the client invokes signServiceAgreement on the Framework's IpServiceAgreementManagement interface, and a reference to an instance of a service manager interface is returned.

2: The client (application) had initiated service agreement signing process for a second service agreement (not shown), and when the client signs this second service agreement, a reference to an instance of another service manager, for another service type, is returned.

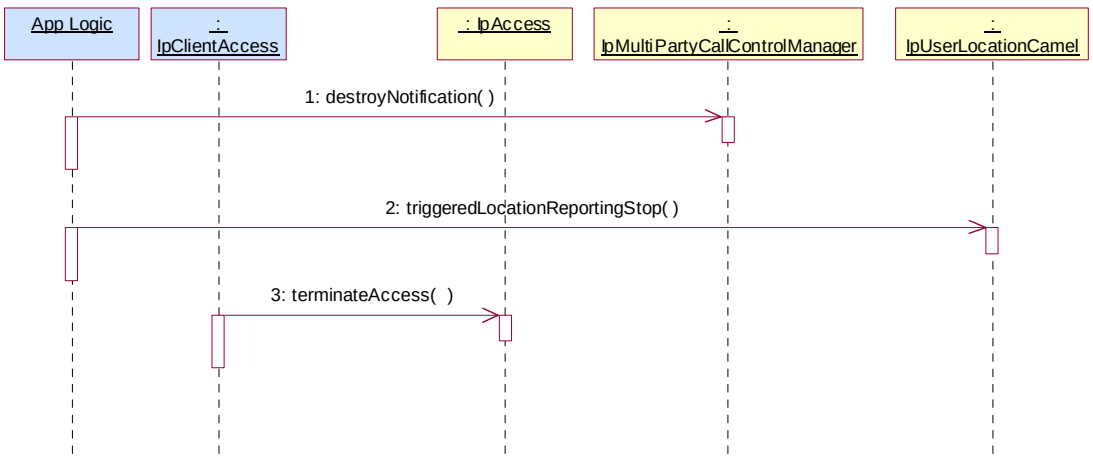
3: The application starts to use the new service manager interface.

4: The application starts to use the other new service manager interface.

5: The framework decides to terminate the application's access session, and to terminate all its service agreements. This is an unusual and drastic step, but could be e.g. due to violation or expiry of the application's service agreements, or some problem within the framework itself. The framework will also destroy each of the service managers the application was using (not shown). The application is now no longer authenticated with the framework, and all Framework and service interfaces it was using are destroyed.

6.1.1.3 Application Terminates Access

This sequence shows how an application could terminate its use of the Framework and of all service instances. This type of termination is unusual, but possible with the terminateAccess method.



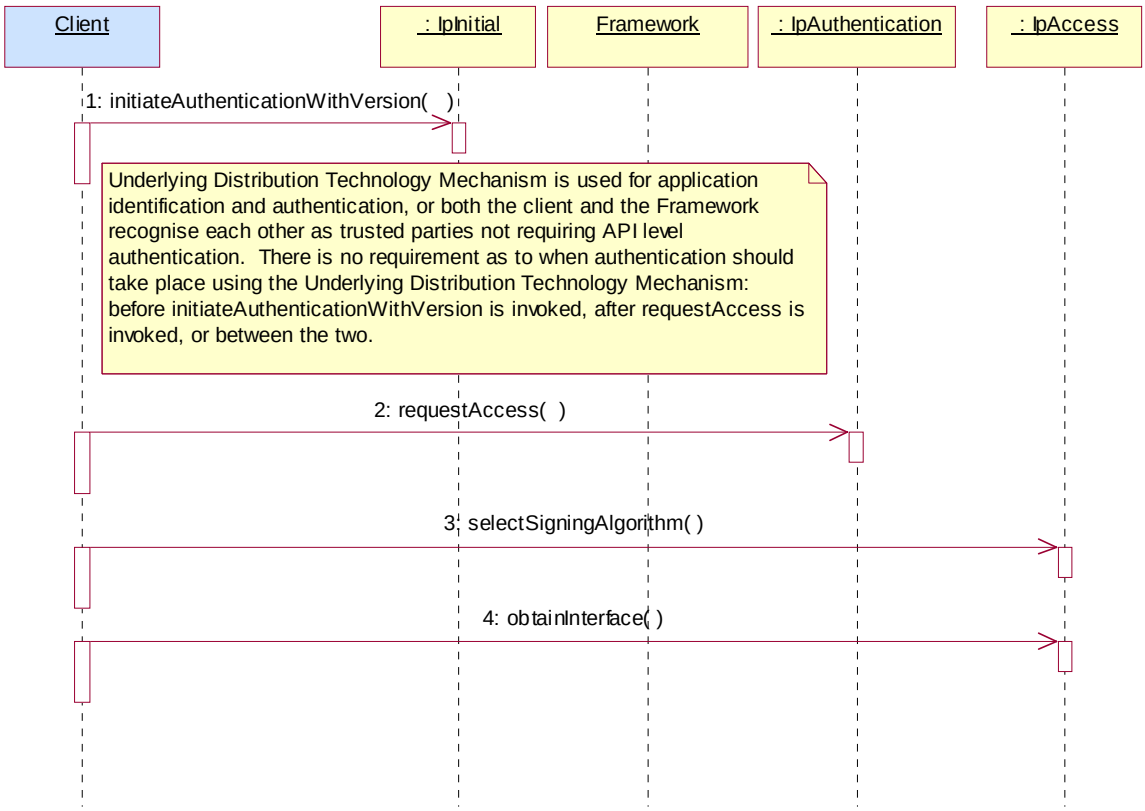
1: The application terminates its use of the multi-party call control service manager ~~instances~~ in a controlled manner.

2: The application ceases to use the user location camel SCF .

3: The application decides to terminate its access session and all its service agreements in one go. The framework will also destroy each of the service managers the application was using (not shown). The application is now no longer authenticated with the framework, and all Framework and service interfaces it was using are destroyed. The application could have terminated its service agreements one by one, by invoking terminateServiceAgreement on the Framework's IpServiceAgreementManager interface, and then invoked terminateAccess on the Framework's IpAccess interface, which would have been a more controlled shutdown.

6.1.1.4 Non-API level Authentication

The following figure shows a client accessing the OSA Framework for the first time. The client and the framework have mutually authenticated one another using an underlying distribution technology mechanism, or the client and the framework recognise each other as a trusted party, not requiring authentication.



1: The client calls initiateAuthenticationWithVersion on the OSA Framework Initial interface. This allows the client to specify the type of authentication process. In this case, the client selects to use the underlying distribution technology mechanism for identification and authentication. What that mechanism is, if it even exists, is outside the scope of the API.

2: The client invokes the requestAccess method on the Framework's Authentication interface. This returns a reference to the framework Access interface that is unique for the client.

3: If the authentication was successful, the client and the framework can negotiate, on the framework's Access interface, the signing algorithm to be used for any signed exchanges.

4: The client can now invoke obtainInterface or obtainInterfaceWithCallback on the framework's Access interface. This is used to obtain a reference to a framework interface such as service discovery, integrity management, service subscription etc. ~~to obtain a reference to its service discovery interface.~~

6.1.1.5 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.

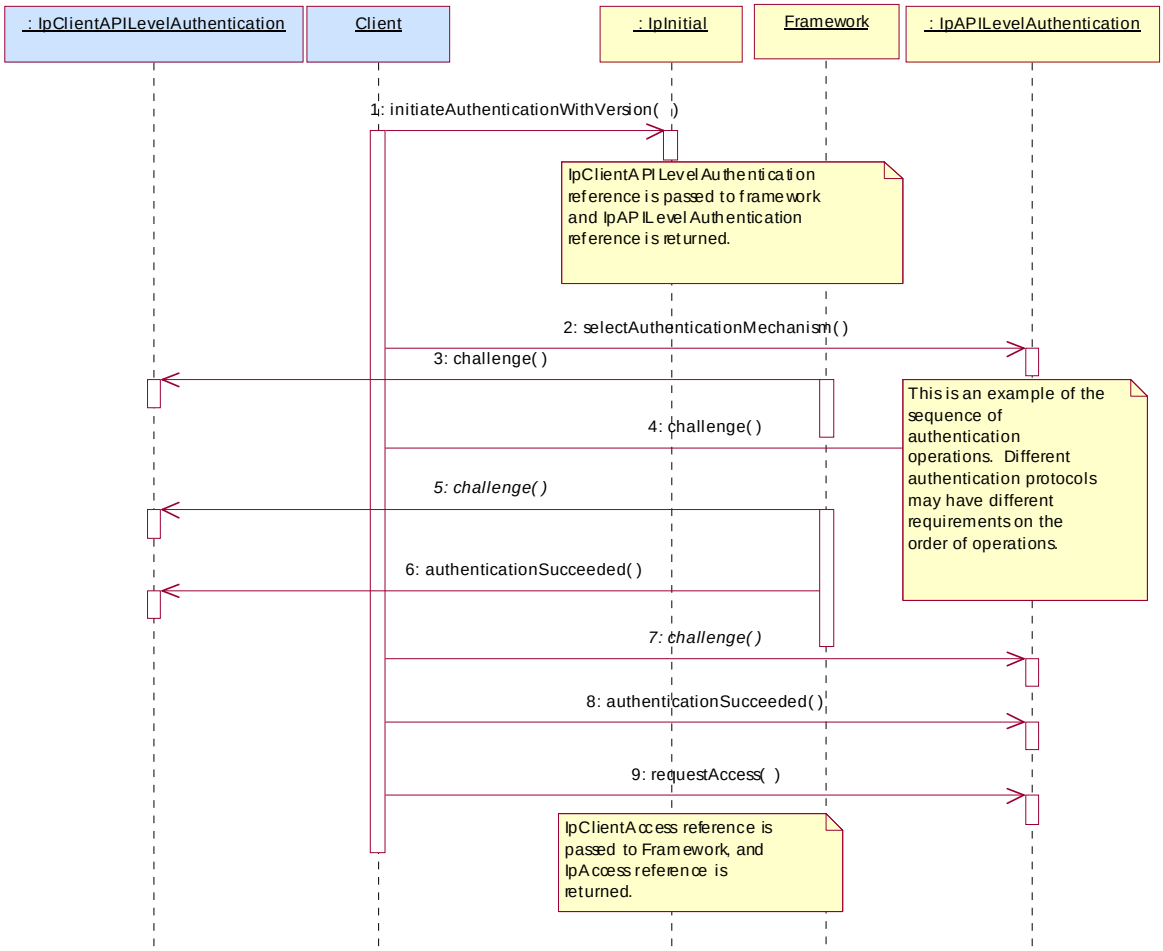
The client must authenticate with the Framework before it is able to use any of the other interfaces supported by the Framework. Invocations on other interfaces will fail until authentication has been successfully completed.

1) The client calls initiateAuthenticationWithVersion 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 initiateAuthenticationWithVersion 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 initiateAuthenticationWithVersion 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 selectAuthenticationMechanism on the Framework's API Level Authentication interface. This includes the authentication algorithms supported by the client. The framework then chooses a mechanism based on the capabilities of the client and the Framework. If the client is capable of handling more than one mechanism, then the Framework chooses one option, defined in the prescribedMethod parameter. In some instances, the authentication mechanism of the client may not fulfil the demands of the Framework, in which case, the authentication will fail, for example CHAP prescribes the MD5 hashing algorithm as the minimum to be supported, however the framework need not accept this algorithm.

3) The application and Framework interact to authenticate each other by using the challenge method. 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~~ challenge method on the API Level Authentication interface. P_OSA_AUTHENTICATION is based on CHAP, which is primarily a one-way protocol. There are in fact two authentication processes: authentication of the client performed by the Framework , and authentication of the Framework performed by the client. Mutual authentication is achieved by both these processes terminating successfully. Mutual authentication may not necessarily be required, i.e. it could be that a client may not need to authenticate the Framework. There is also no required order for the execution of these two authentication processes, however, the client shall respond immediately to any challenge issued by the Framework, as the Framework might not respond to any challenge issued by the client until the Framework has successfully authenticated the client.

Note that at any point during the access session, either side can request re-authentication of the other side.



********** End of Change # 1 **********

********** Start of Change # 2 **********

6.3.1.3 Interface Class IpInitial

Inherits from: IpInterface.

The Initial Framework interface is used by the client to initiate the authentication with the Framework. This interface shall be implemented by a Framework. The initiateAuthentication() and the initiateAuthenticationWithVersion() methods shall be implemented.

<<Interface>> IpInitial
<<deprecated>> initiateAuthentication (clientDomain : in TpAuthDomain, authType : in TpAuthType) : TpAuthDomain <<new>> initiateAuthenticationWithVersion (clientDomain : in TpAuthDomain, authType : in TpAuthType, frameworkVersion : in TpVersion) : TpAuthDomain

6.3.1.3.4 Method <<deprecated>> initiateAuthentication()

This method is deprecated in this version, this means that it will be supported until the next major release of the present document.

This method is invoked by the client to start the process of authentication with the framework, and request the use of a specific authentication method.

Returns <fwDomain> : This provides the client with a framework identifier, and a reference to call the authentication interface of the framework.

```
structure TpAuthDomain {
    domainID:       TpDomainID;
    authInterface:  IpInterfaceRef;
        };
```

The domainID parameter is an identifier for the framework (i.e. TpFwID). It is used to identify the framework to the client.

The authInterface parameter is a reference to the authentication interface of the framework. The type of this interface is defined by the authType parameter. The client uses this interface to authenticate with the framework.

Parameters

`clientDomain : in TpAuthDomain`

This identifies the client domain to the framework, and provides a reference to the domain's authentication interface.

```
structure TpAuthDomain {
    domainID:       TpDomainID;
    authInterface:  IpInterfaceRef;
        };
```

The domainID parameter is an identifier either for a client application (i.e. TpClientAppID) or for an enterprise operator (i.e. TpEntOpID), or for an instance of a service for which a client application has signed a service agreement (i.e. TpServiceInstanceID), or for a service supplier (i.e. TpServiceSupplierID). It is used to identify the client domain to the framework, (see authenticate() on IpAPILevelAuthentication). If the framework does not recognise the domainID, the framework returns an error code (P_INVALID_DOMAIN_ID).

The authInterface parameter is a reference to call the authentication interface of the client. The type of this interface is defined by the authType parameter. If the interface reference is not of the correct type, the framework returns an error code (P_INVALID_INTERFACE_TYPE).

`authType : in TpAuthType`

This identifies the type of authentication mechanism requested by the client. It provides operators and clients with the opportunity to use an alternative to the API level Authentication interface, e.g. an implementation specific authentication mechanism like CORBA Security, using the IpAuthentication interface, or Operator specific Authentication interfaces. OSA API level Authentication is the default authentication mechanism (P_OSA_AUTHENTICATION). If P_OSA_AUTHENTICATION is selected, then the clientDomain and fwDomain

authInterface parameters are references to interfaces of type Ip(Client)APILevelAuthentication. If P_AUTHENTICATION is selected, the fwDomain authInterface parameter references to interfaces of type IpAuthentication which is used when an underlying distribution technology authentication mechanism is used.

Returns

TpAuthDomain

Raises

TpCommonExceptions, P_INVALID_DOMAIN_ID, P_INVALID_INTERFACE_TYPE, P_INVALID_AUTH_TYPE

6.3.1.3.5 Method <<new>> initiateAuthenticationWithVersion()

This method is invoked by the client to start the process of authentication with the framework, and request the use of a specific authentication method using the new method with support for backward compatibility in the framework. The returned fwDomain authInterface will be selected to match the proposed version from the Client in the Framework response. If the Framework cannot work with the proposed framework version the framework returns an error code (P_INVALID_VERSION).

Returns <fwDomain> : This provides the client with a framework identifier, and a reference to call the authentication interface of the framework.

```
structure TpAuthDomain {
    domainID:       TpDomainID;
    authInterface:  IpInterfaceRef;
        };
```

The domainID parameter is an identifier for the framework (i.e. TpFwID). It is used to identify the framework to the client.

The authInterface parameter is a reference to the authentication interface of the framework that is unique for the requesting client. The type of this interface is defined by the authType parameter. The client uses this interface to authenticate with the framework.

Note, there are no identifiers used in the authentication interface to correlate requests and responses, therefore the authentication interface may not be shared amongst multiple clients.

Parameters

clientDomain : in TpAuthDomain

This identifies the client domain to the framework, and provides a reference to the domain's authentication interface.

```
structure TpAuthDomain {
    domainID:       TpDomainID;
    authInterface:  IpInterfaceRef;
        };
```

The domainID parameter is an identifier either for a client application (i.e. TpClientAppID) or for an enterprise operator (i.e. TpEntOpID), or for an instance of a service for which a client application has signed a service agreement (i.e. TpServiceInstanceID), or for a service supplier (i.e. TpServiceSupplierID). It is used to identify the client domain to the framework, (see challenge() on IpAPILevelAuthentication). If the framework does not recognise the domainID, the framework returns an error code (P_INVALID_DOMAIN_ID).

The authInterface parameter is a reference to call the authentication interface of the client. The type of this interface is defined by the authType parameter. If the interface reference is not of the correct type, the framework returns an error code (P_INVALID_INTERFACE_TYPE).

authType : in TpAuthType

This identifies the type of authentication mechanism requested by the client. It provides operators and clients with the opportunity to use an alternative to the API level Authentication interface, e.g. an implementation specific

authentication mechanism like CORBA Security, using the IpAuthentication interface, or Operator specific Authentication interfaces. OSA API level Authentication is the default authentication mechanism (P_OSA_AUTHENTICATION). If P_OSA_AUTHENTICATION is selected, then the clientDomain and fwDomain authInterface parameters are references to interfaces of type Ip(Client)APILevelAuthentication. If P_AUTHENTICATION is selected, the fwDomain authInterface parameter references to interfaces of type IpAuthentication that is used when an underlying distribution technology authentication mechanism is used.

`frameworkVersion : in TpVersion`

This identifies the version of the Framework implemented in the client. The TpVersion is a String containing the version number. Valid version numbers are defined in the respective framework specification.

Returns

`TpAuthDomain`

Raises

`TpCommonExceptions, P_INVALID_DOMAIN_ID, P_INVALID_INTERFACE_TYPE, P_INVALID_AUTH_TYPE, P_INVALID_VERSION`

********** End of Change # 2 **********

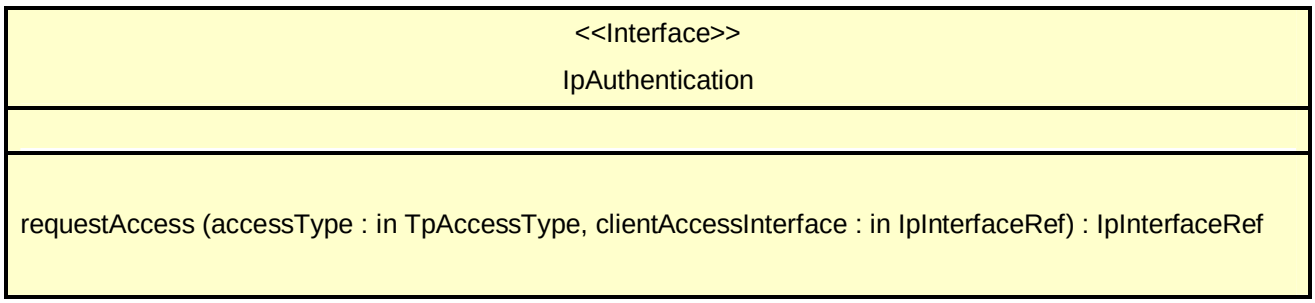
********** Start of Change # 3 **********

6.3.1.4 Interface Class IpAuthentication

Inherits from: IpInterface.

The Authentication Framework interface is used by client to request access to other interfaces supported by the Framework. The authentication process should in this case be done with some underlying distribution technology authentication mechanism, e.g. CORBA Security.

At least one of IpAuthentication or IpAPILevelAuthentication interfaces shall be implemented by a Framework as a minimum requirement. The requestAccess() method shall be implemented in each.



6.3.1.4.4 Method requestAccess()

Once the client has been authenticated by the framework, the client may invoke the requestAccess operation on the IpAuthentication or IpAPILevelAuthentication interface. This allows the client to request the type of access they require. If they request P_OSA_ACCESS, then a reference to the IpAccess interface is returned. (Operators can define their own access interfaces to satisfy client requirements for different types of access.)

If this method is called before the client has been successfully authenticated, then the request fails, and an error code (P_ACCESS_DENIED) is returned.

This method may be invoked by the client immediately on IpAuthentication, when API Level authentication is not being used, since there is no indication to the client at API level that it is authenticated with the Framework.

Returns <fwAccessInterface> : This provides the reference for the client to call the access interface of the framework. The access reference provided is unique to the requesting client.

Parameters

accessType : in TpAccessType

This identifies the type of access interface requested by the client. If the framework does not provide the type of access identified by accessType, then an error code (P_INVALID_ACCESS_TYPE) is returned.

clientAccessInterface : in IpInterfaceRef

This provides the reference for the framework to call the access interface of the client. If the interface reference is not of the correct type, the framework returns an error code (P_INVALID_INTERFACE_TYPE).

Returns

IpInterfaceRef

Raises

TpCommonExceptions, P_ACCESS_DENIED, P_INVALID_ACCESS_TYPE, P_INVALID_INTERFACE_TYPE

********** End of Change # 3 **********

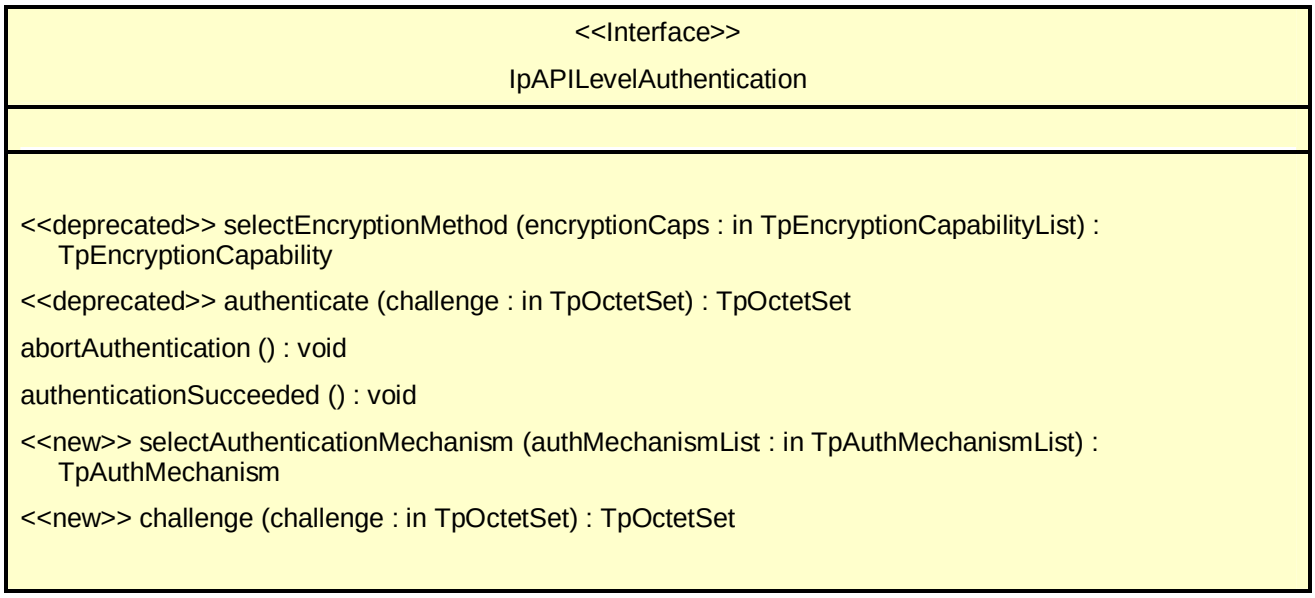
********** Start of Change # 4 **********

6.3.1.5 Interface Class IpAPILevelAuthentication

Inherits from: IpAuthentication.

The API Level Authentication Framework interface is used by the client to authenticate the Framework. It is also used to initiate the authentication process.

If the IpAPILevelAuthentication interface is implemented by a Framework, then selectEncryptionMethod(), selectAuthenticationMechanism(), authenticate(), challenge(), abortAuthentication() and authenticationSucceeded () shall be implemented. IpAPILevelAuthentication inherits the requirements of IpAuthentication, therefore requestAccess() shall be implemented.



6.3.1.5.4 Method <<deprecated>> selectEncryptionMethod()

This method is deprecated and replaced by selectAuthenticationMechanism(). It shall only be used when the IpAPILevelAuthentication interface is obtained by using the deprecated method initiateAuthentication() instead of initiateAuthenticationWithVersion() on the IpInitial interface. This method will be removed in a later release.

The client uses this method to initiate the authentication process. The framework returns its preferred mechanism. This should be within capability of the client. If a mechanism that is acceptable to the framework within the capability of the client cannot be found, the framework throws the P_NO_ACCEPTABLE_ENCRYPTION_CAPABILITY exception. Once the framework has returned its preferred mechanism, it will wait for a predefined unit of time before invoking the client's authenticate() method (the wait is to ensure that the client can initialise any resources necessary to use the prescribed encryption method).

Returns <prescribedMethod> : This is returned by the framework to indicate the mechanism preferred by the framework for the encryption process. If the value of the prescribedMethod returned by the framework is not understood by the client, it is considered a catastrophic error and the client must abort.

Parameters

encryptionCaps : in TpEncryptionCapabilityList

This is the means by which the encryption mechanisms supported by the client are conveyed to the framework.

Returns

TpEncryptionCapability

Raises

TpCommonExceptions, P_ACCESS_DENIED,
P_NO_ACCEPTABLE_ENCRYPTION_CAPABILITY

6.3.1.5.5 Method <<deprecated>> authenticate()

This method is deprecated and replaced by challenge(). It shall only be used when the IpAPILevelAuthentication interface is obtained by using the deprecated method initiateAuthentication() instead of initiateAuthenticationWithVersion() on the IpInitial interface. This method will be removed in a later release.

This method is used by the client to authenticate the framework. The challenge will be encrypted using the mechanism prescribed by selectEncryptionMethod. The framework must respond with the correct responses to the challenges presented by the client. The domainID received in the initiateAuthentication() can be used by the framework to reference the correct public key for the client (the key management system is currently outside of the scope of the OSA APIs). The number of exchanges is dependent on the policies of each side. The authentication of the framework is deemed successful when the authenticationSucceeded method is invoked by the client.

The invocation of this method may be interleaved with authenticate() calls by the framework on the client's APILevelAuthentication interface.

Returns <response> : This is the response of the framework to the challenge of the client in the current sequence. The response will be based on the challenge data, decrypted with the mechanism prescribed by selectEncryptionMethod().

Parameters

challenge : in TpOctetSet

The challenge presented by the client to be responded to by the framework. The challenge mechanism used will be in accordance with the IETF PPP Authentication Protocols - Challenge Handshake Authentication Protocol (RFC 1994). The challenge will be encrypted with the mechanism prescribed by selectEncryptionMethod().

Returns

TpOctetSet

Raises

TpCommonExceptions, P_ACCESS_DENIED

6.3.1.5.6 Method abortAuthentication()

The client uses this method to abort the authentication process where the framework is authenticating the client. This method is invoked if the client no longer wishes to continue the authentication process, (unless the framework responded incorrectly to a challenge in which case no further communication with the framework should occur.) If this method has been invoked before the client has been authenticated by the Framework, calls to the requestAccess operation on IpAPILevelAuthentication will return an error code (P_ACCESS_DENIED), until the client has been properly authenticated. If this method is invoked after the client has been authenticated by the Framework, it shall not result in the immediate removal of the client's authentication. (The Framework may wish to authenticate the client again, however).

Parameters

No Parameters were identified for this method

Raises

TpCommonExceptions,P_ACCESS_DENIED

6.3.1.5.7 Method authenticationSucceeded()

The client uses this method to inform the framework of the success of the authentication attempt. Calls to this method have no impact on the client's rights to call requestAccess(), which depend exclusively on the framework's successful authentication of the client.

Parameters

No Parameters were identified for this method

Raises

TpCommonExceptions, P_ACCESS_DENIED

6.3.1.5.8 Method <<new>> selectAuthenticationMechanism()

The client uses this method to inform the Framework of the different authentication mechanisms it supports as part of API level Authentication. The Framework will select one of the suggested authentication mechanisms and that mechanism shall be used for authentication by both Framework and Client. The authentication mechanism chosen as a result of the response to this method remains valid for an instance of IpAPILevelAuthentication and until this method is re-invoked by the client. If a mechanism that is acceptable to the framework within the capability of the client cannot be found, the framework throws the P_NO_ACCEPTABLE_AUTHENTICATION_MECHANISM exception.

This method shall only be used when the IpAPILevelAuthentication interface is obtained by using initiateAuthenticationWithVersion() on the IpInitial interface.

Returns: selectedMechanism. This is the authentication mechanism chosen by the Framework. The chosen mechanism shall be taken from the list of mechanisms proposed by the Client.

Parameters

authMechanismList : in TpAuthMechanismList

The list of authentication mechanisms supported by the client.

Returns

TpAuthMechanism

Raises

TpCommonExceptions, P_ACCESS_DENIED,
P_NO_ACCEPTABLE_AUTHENTICATION_MECHANISM

6.3.1.5.9 Method <<new>> challenge()

This method is used by the client to authenticate the framework. The framework must respond with the correct responses to the challenges presented by the client. The domainID received in the initiateAuthenticationWithVersion() can be used by the framework to reference the correct public key for the client (the key management system is currently outside of the scope of the OSA APIs). The number of exchanges is dependent on the policies of each side. The authentication of the framework is deemed successful when the authenticationSucceeded method is invoked by the client.

The invocation of this method may be interleaved with challenge() calls by the framework on the client's APILevelAuthentication interface.

This method shall only be used when the IpAPILevelAuthentication interface is obtained by using initiateAuthenticationWithVersion() on the IpInitial interface.

Returns <response> : This is the response of the framework to the challenge of the client in the current sequence. The formatting of this parameter shall be according to section 4.1 of RFC 1994. A complete CHAP Response packet shall be used to carry the response string. The Response packet shall make the contents of this returned parameter. The Name field of the CHAP Response packet shall be present but not contain any useful value.

Parameters

challenge : in TpOctetSet

The challenge presented by the client to be responded to by the framework. The challenge format used will be in accordance with the IETF PPP Authentication Protocols - Challenge Handshake Authentication Protocol (RFC 1994).

The formatting of the challenge value shall be according to section 4.1 of RFC 1994. A complete CHAP Request packet shall be used to carry the challenge value. The Name field of the CHAP Request packet shall be present but not contain any useful value.

Returns

TpOctetSet

Raises

TpCommonExceptions, P_ACCESS_DENIED

********** End of Change # 4 **********

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Jun 2003	CN_20	NP-030237	079	--	Correction to TpEncryptionCapability to correct support for Triple-DES	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	081	--	Correction of the Framework Service Instance Lifecycle Manager Sequence Diagram	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	083	--	Correction of the use of TpDomainID in Framework initiateAuthentication method	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	085	--	Correction to Java Realisation Annex	5.3.0	5.4.0