**3GPP TSG-CN Meeting #24** **NP-040273**
**02 – 04 June 2004, Seoul, KOREA**

**Source:** **CN5 (OSA)**

**Title:** **7 Rel-6 CRs 29.198 Parts 3, 4-3, 5, 7, 8, 11, 12 OSA API: Remove the <<new>> stereotype from methods which are no longer new**

**Agenda item:** **9.7 (OSA Enhancements [OSA3])**

**Document for:** **APPROVAL**

---

| Doc-1st- | Spec | CR | Rev | Phase | Subject | Cat | Version | Doc-2nd- | Workite |
|---|---|---|---|---|---|---|---|---|---|
| NP-040273 | 29.198-03 | 113 | - | Rel-6 | Remove the <<new>> stereotype from methods which are no longer new | F | 6.0.1 | N5-040284 | OSA3 |
| NP-040273 | 29.198-04-3 | 024 | - | Rel-6 | Remove the <<new>> stereotype from methods which are no longer new | F | 6.1.0 | N5-040285 | OSA3 |
| NP-040273 | 29.198-05 | 050 | - | Rel-6 | Remove the <<new>> stereotype from methods which are no longer new | F | 6.0.1 | N5-040286 | OSA3 |
| NP-040273 | 29.198-07 | 017 | - | Rel-6 | Remove the <<new>> stereotype from methods which are no longer new | F | 6.0.1 | N5-040287 | OSA3 |
| NP-040273 | 29.198-08 | 032 | - | Rel-6 | Remove the <<new>> stereotype from methods which are no longer new | F | 6.0.1 | N5-040288 | OSA3 |
| NP-040273 | 29.198-11 | 028 | - | Rel-6 | Remove the <<new>> stereotype from methods which are no longer new | F | 6.0.1 | N5-040289 | OSA3 |
| NP-040273 | 29.198-12 | 027 | - | Rel-6 | Remove the <<new>> stereotype from methods which are no longer new | F | 6.0.1 | N5-040290 | OSA3 |

CR-Form-v7

# CHANGE REQUEST

⌘ 29.198-03 CR 113 ⌘rev - ⌘ Current version: 6.0.1 ⌘

*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:*** | ⌘ | Remove the <<new>> stereotype from methods which are no longer new |
| ***Source:*** | ⌘ | CN5 Ultan Mulligan, ETSI PTCC |
| ***Work item code:*** | ⌘ | OSA3 |
| ***Date:*** | ⌘ | 14/05/2004 |
| ***Category:*** | ⌘ | F |
| ***Release:*** | ⌘ | REL-6 |

*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 <<new>> stereotype is used to indicate a new method has been added to the specification. However, some methods have been newly introduced more than a year ago, and in Rel-5, yet they still carry the <<new>> stereotype in Rel-6. For these methods, the <<new>> stereotype can be removed from their title and description.<br>Although editorial, this change is being applied throughout the 29.198 Rel-6 specification set, and so, on MCC advice, CRs have been generated to make this process visible. |
| ***Summary of change:*** | ⌘ | Remove the <<new>> stereotype from the documentation of methods which were newly introduced prior to the creation of the Rel-6 specifications, i.e. from methods in the Rel-6 specifications which were newly introduced into Rel-5 at or before the March 2003 plenary. |
| ***Consequences if not approved:*** | ⌘ | The specifications will contain misleading information, indicating certain methods as being newly introduced, when in fact they are not |

| | | | | | |
|---|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 6.3, 7.3.3, 8.3.4 | | | |
| | | **Y** | **N** | | |
| ***Other specs affected:*** | ⌘ | X | | Other core specifications | ⌘ TS 29.198-1, 4-3, 5, 7, 8, 11 and 12 Release 6 |
| | | | X | Test specifications | |
| | | | X | O&M Specifications | |
| ***Other comments:*** | ⌘ | N5-040284 to N5-040290 should be bundled together | | | |

**Change in Clause 6.3**

# 6.3 Interface Classes

## 6.3.1 Trust and Security Management Interface Classes

The Trust and Security Management Interfaces provide:

- the first point of contact for a client to access a Framework provider;
- the authentication methods for the client and Framework provider to perform an authentication protocol;
- the client with the ability to select a service capability feature to make use of;
- the client with a portal to access other Framework interfaces.

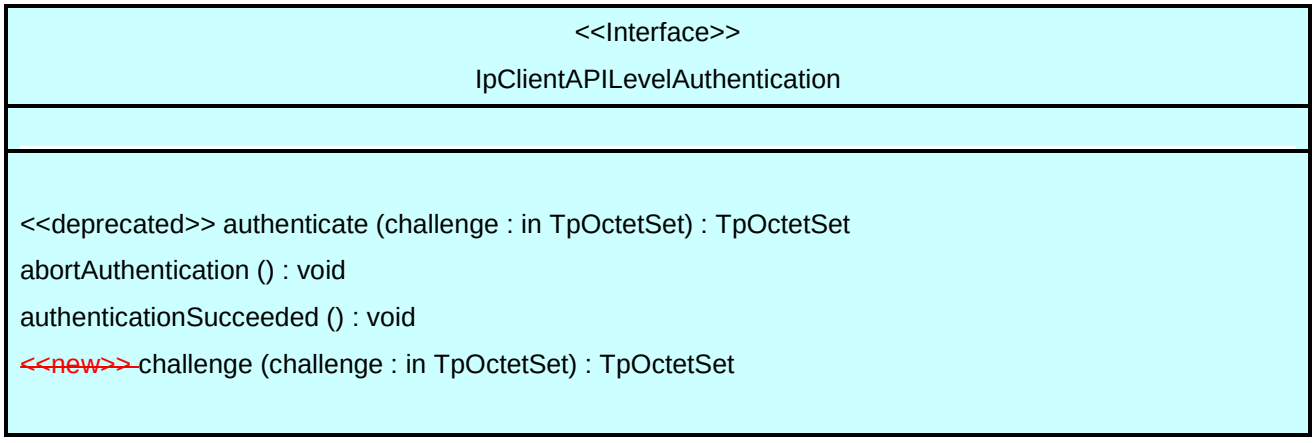
The process by which the client accesses the Framework provider has been separated into 3 stages, each supported by a different Framework interface:

1) Initial Contact with the Framework;
2) Authentication;
3) Access to Framework and Service Capability Features.

### 6.3.1.1 Interface Class IpClientAPILevelAuthentication

Inherits from: IpInterface.

If the IpClientAPILevelAuthentication interface is implemented by a client, authenticate(), challenge(), abortAuthentication() and authenticationSucceeded() methods shall be implemented.




#### 6.3.1.1.1 Method <<deprecated>> authenticate()

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

This method is used by the framework to authenticate the client. The challenge will be encrypted using the mechanism prescribed by selectEncryptionMethod. The client must respond with the correct responses to the challenges presented by the framework. The number of exchanges is dependent on the policies of each side. The authentication of the client is deemed successful when the authenticationSucceeded method is invoked by the Framework.

The invocation of this method may be interleaved with authenticate() calls by the client on the IpAPILevelAuthentication interface. The client shall respond immediately to authentication challenges from the Framework, and not wait until the Framework has responded to any challenge the client may issue.

Returns <response> : This is the response of the client application to the challenge of the framework 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 framework to be responded to by the client. The challenge mechanism used will be in accordance with the IETF PPP Authentication Protocols - Challenge Handshake Authentication Protocol [RFC 1994, August1996]. The challenge will be encrypted with the mechanism prescribed by selectEncryptionMethod().

*Returns*

**`TpOctetSet`**

#### 6.3.1.1.2 Method abortAuthentication()

The framework uses this method to abort the authentication process where the client is authenticating the Framework. This method is invoked if the framework wishes to abort the authentication process before it has been authenticated by the client, (unless the client responded incorrectly to a challenge in which case no further communication with the client should occur.) Calls to this method after the Framework has been authenticated by the client shall not result in an immediate removal of the Framework's authentication (the client may wish to authenticate the Framework again, however).

*Parameters*

No Parameters were identified for this method

#### 6.3.1.1.3 Method authenticationSucceeded()

The Framework uses this method to inform the client of the success of the authentication attempt. The client may invoke requestAccess on the Framework's APILevelAuthentication interface following invocation of this method.

*Parameters*

No Parameters were identified for this method

#### 6.3.1.1.4 Method ~~<<new>>~~ challenge()

This method is used by the framework to authenticate the client. The client must respond with the correct responses to the challenges presented by the framework. The number of exchanges is dependent on the policies of each side. The authentication of the client is deemed successful when the authenticationSucceeded method is invoked by the Framework.

The invocation of this method may be interleaved with challenge() calls by the client on the IpAPILevelAuthentication interface. The client shall respond immediately to authentication challenges from the Framework, and not wait until the Framework has responded to any challenge the client may issue.

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

Returns <response> : This is the response of the client application to the challenge of the framework 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 framework to be responded to by the client. 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.

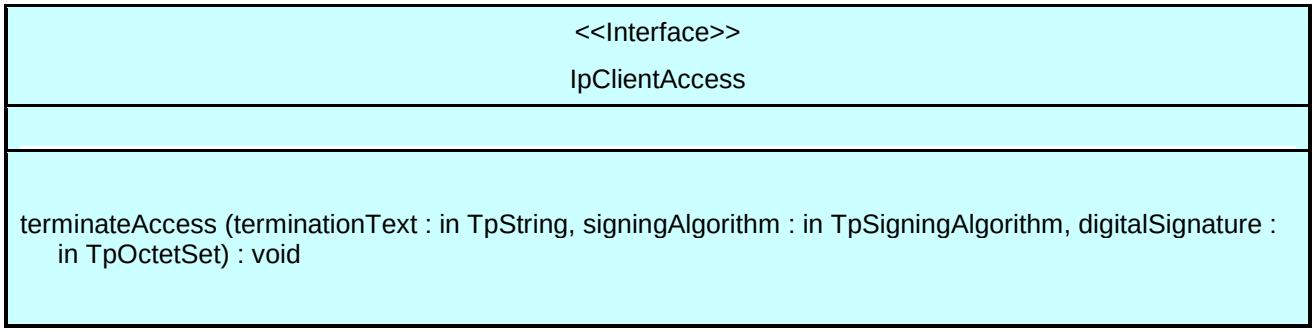
*Returns*

**TpOctetSet**

### 6.3.1.2 Interface Class IpClientAccess

Inherits from: IpInterface.

IpClientAccess interface is offered by the client to the framework to allow it to initiate interactions during the access session. This interface and the terminateAccess() method shall be implemented by a client.




#### 6.3.1.2.1 Method terminateAccess()

The terminateAccess operation is used by the framework to end the client's access session.

After terminateAccess() is invoked, the client will no longer be authenticated with the framework. The client will not be able to use the references to any of the framework interfaces gained during the access session. Any calls to these interfaces will fail. Also, all remaining service instances created by the framework either directly in this access session or on behalf of the client during this access session shall be terminated. 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.

*Parameters*

**terminationText : in TpString**

This is the termination text describes the reason for the termination of the access session.

**signingAlgorithm : in TpSigningAlgorithm**

This is the algorithm used to compute the digital signature. It shall be identical to the one chosen by the framework in response to IpAccess.selectSigningAlgorithm(). If the signingAlgorithm is not the chosen one, is invalid, or unknown to the client, the P_INVALID_SIGNING_ALGORITHM exception will be thrown. The list of possible algorithms is as specified in the TpSigningAlgorithm table. The identifier used in this parameter must correspond to the digestAlgorithm and signatureAlgorithm fields in the SignerInfo field in the digitalSignature (see below).

**digitalSignature : in TpOctetSet**

This contains a CMS (Cryptographic Message Syntax) object (as defined in RFC 2630) with content type Signed-data. The signature is calculated and created as per section 5 of RFC 2630. The content is made of the termination text. The "external signature" construct shall not be used (i.e. the eContent field in the EncapsulatedContentInfo field shall be present and contain the termination text string). The signing-time attribute, as defined in section 11.3 of RFC 2630, shall also be used to provide replay prevention. The framework uses this to confirm its identity to the client. The client can check that the terminationText has been signed by the framework. If a match is made, the access session is terminated, otherwise the P_INVALID_SIGNATURE exception will be thrown.

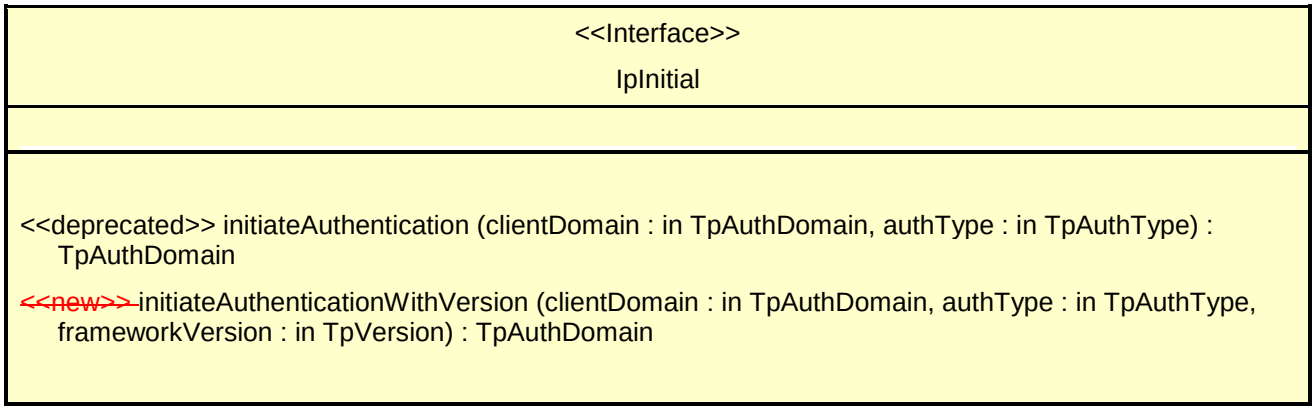
*Raises*

**TpCommonExceptions, P_INVALID_SIGNING_ALGORITHM, P_INVALID_SIGNATURE**

### 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.




#### 6.3.1.3.1 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**

### 6.3.1.3.2 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**

### 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.

| <<Interface>><br>IpAuthentication |
|---|
| |
| requestAccess (accessType : in TpAccessType, clientAccessInterface : in IpInterfaceRef) : IpInterfaceRef |

#### 6.3.1.4.1 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`**

### 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.

<<Interface>>

IpAPILevelAuthentication

<<deprecated>> selectEncryptionMethod (encryptionCaps : in TpEncryptionCapabilityList) : TpEncryptionCapability

<<deprecated>> authenticate (challenge : in TpOctetSet) : TpOctetSet

abortAuthentication () : void

authenticationSucceeded () : void

~~<<new>>~~ selectAuthenticationMechanism (authMechanismList : in TpAuthMechanismList) : TpAuthMechanism

~~<<new>>~~ challenge (challenge : in TpOctetSet) : TpOctetSet

#### 6.3.1.5.1 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**

#### 6.3.1.5.2 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**

#### 6.3.1.5.3 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**

#### 6.3.1.5.4 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**

### 6.3.1.5.5 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**

### 6.3.1.5.6 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`**

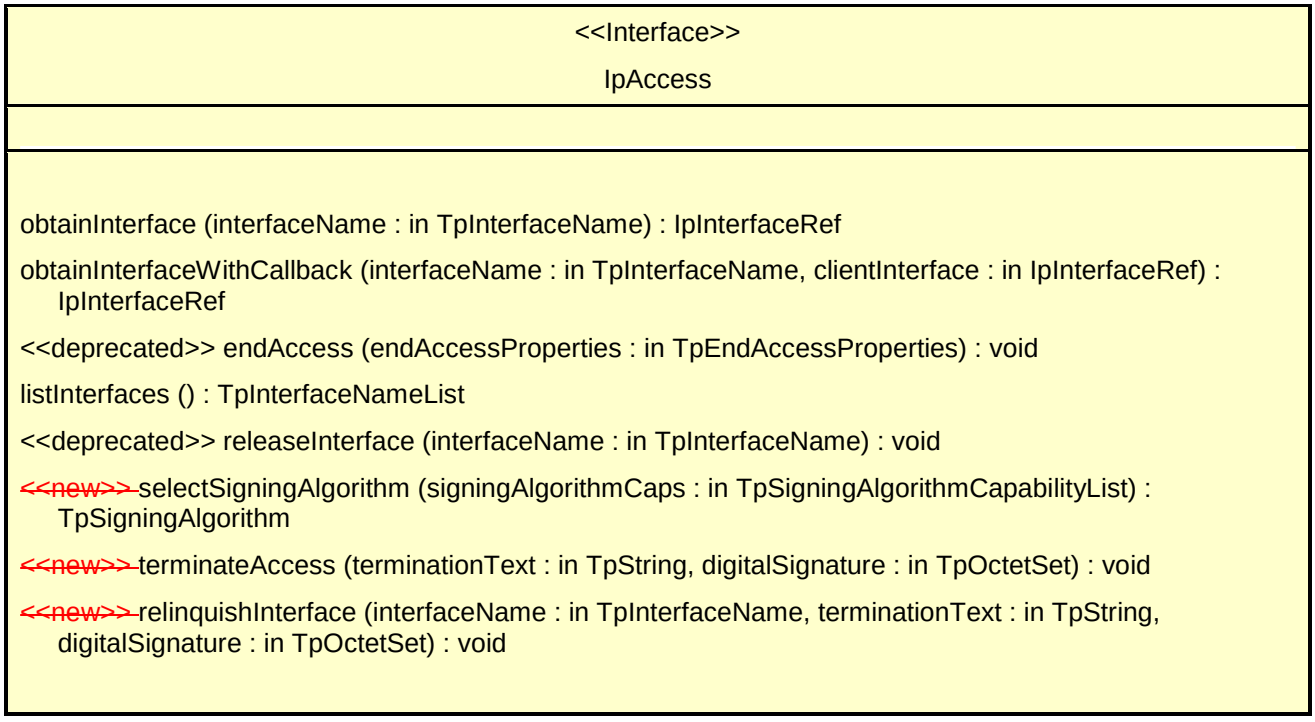
*Raises*

**`TpCommonExceptions, P_ACCESS_DENIED`**

### 6.3.1.6 Interface Class IpAccess

Inherits from: IpInterface.

This interface shall be implemented by a Framework. As a minimum requirement the obtainInterface() and obtainInterfaceWithCallback(), selectSigningAlgorithm() and terminateAccess() methods shall be implemented.




#### 6.3.1.6.1 Method obtainInterface()

This method is used to obtain other framework interfaces. The client uses this method to obtain interface references to other framework interfaces. (The obtainInterfaceWithCallback method should be used if the client is required to supply a callback interface to the framework.)

Returns <fwInterface> : This is the reference to the interface requested.

*Parameters*

**`interfaceName : in TpInterfaceName`**

The name of the framework interface to which a reference to the interface is requested. If the interfaceName is invalid, the framework returns an error code (P_INVALID_INTERFACE_NAME).

*Returns*

**IpInterfaceRef**

*Raises*

**TpCommonExceptions, P_ACCESS_DENIED, P_INVALID_INTERFACE_NAME**

#### 6.3.1.6.2 Method obtainInterfaceWithCallback()

This method is used to obtain other framework interfaces. The client uses this method to obtain interface references to other framework interfaces, when it is required to supply a callback interface to the framework. (The obtainInterface method should be used when no callback interface needs to be supplied.)

Returns <fwInterface> : This is the reference to the interface requested.

*Parameters*

**interfaceName : in TpInterfaceName**

The name of the framework interface to which a reference to the interface is requested. If the interfaceName is invalid, the framework returns an error code (P_INVALID_INTERFACE_NAME).

**clientInterface : in IpInterfaceRef**

This is the reference to the client interface, which is used for callbacks. If a client interface is not needed, then this method should not be used. (The obtainInterface method should be used when no callback interface needs to be supplied.) 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_INTERFACE_NAME, P_INVALID_INTERFACE_TYPE**

#### 6.3.1.6.3 Method <<deprecated>> endAccess()

This method is deprecated and will be removed in a later release. It is replaced with terminateAccess. The endAccess operation is used by the client to request that its access session with the framework is ended. After it is invoked, the client will no longer be authenticated with the framework. The client will not be able to use the references to any of the framework interfaces gained during the access session. Any calls to these interfaces will fail.

*Parameters*

**endAccessProperties : in TpEndAccessProperties**

This is a list of properties that can be used to tell the framework the actions to perform when ending the access session (e.g. existing service sessions may be stopped, or left running). If a property is not recognised by the framework, an error code (P_INVALID_PROPERTY) is returned.

*Raises*

**TpCommonExceptions, P_ACCESS_DENIED, P_INVALID_PROPERTY**

#### 6.3.1.6.4 Method listInterfaces()

The client uses this method to obtain the names of all interfaces supported by the framework. It can then obtain the interfaces it wishes to use using either obtainInterface() or obtainInterfaceWithCallback().

Returns <frameworkInterfaces> : The frameworkInterfaces parameter contains a list of interfaces that the framework makes available.

*Parameters*

No Parameters were identified for this method

*Returns*

**TpInterfaceNameList**

*Raises*

**TpCommonExceptions, P_ACCESS_DENIED**

#### 6.3.1.6.5 Method <<deprecated>> releaseInterface()

This method is deprecated and will be removed in a later release. It is replaced with relinquishInterface. The client uses this method to release a framework interface that was obtained during this access session.

*Parameters*

**interfaceName : in TpInterfaceName**

This is the name of the framework interface which is being released. If the interfaceName is invalid, the framework throws the P_INVALID_INTERFACE_NAME exception. If the interface has not been given to the client during this access session, then the P_TASK_REFUSED exception will be thrown.

*Raises*

**TpCommonExceptions, P_ACCESS_DENIED, P_INVALID_INTERFACE_NAME**

#### 6.3.1.6.6 Method ~~<<new>>~~ selectSigningAlgorithm()

The client uses this method to inform the Framework of the different signing algorithms it supports for use in all cases where digital signatures are required. The Framework will select one of the suggested algorithms. This method shall be the first method invoked by the client on IpAccess. The algorithm chosen as a result of the response to this method remains valid for an instance of IpAccess and until this method is re-invoked by the client. If an algorithm that is acceptable to the framework within the capability of the client cannot be found, the framework throws the P_NO_ACCEPTABLE_SIGNING_ALGORITHM exception.

Returns: selectedAlgorithm. This is the signing algorithm chosen by the Framework. The chosen algorithm shall be taken from the list proposed by the Client.

*Parameters*

**signingAlgorithmCaps : in TpSigningAlgorithmCapabilityList**

The list of signing algorithms supported by the client.

*Returns*

**TpSigningAlgorithm**

*Raises*

**TpCommonExceptions, P_ACCESS_DENIED, P_NO_ACCEPTABLE_SIGNING_ALGORITHM**

### 6.3.1.6.7 Method ~~<<new>>~~ terminateAccess()

The terminateAccess method is used by the client to request that its access session with the framework is ended. After it is invoked, the client will no longer be authenticated with the framework. The client will not be able to use the references to any of the framework interfaces gained during the access session. Any calls to these interfaces will fail. Also, all remaining service instances created by the framework either directly in this access session or on behalf of the client during this access session shall be terminated.

*Parameters*

**terminationText : in TpString**

This is the termination text describes the reason for the termination of the access session.

**digitalSignature : in TpOctetSet**

This contains a CMS (Cryptographic Message Syntax) object (as defined in RFC 2630) with content type Signed-data. The signature is calculated and created as per section 5 of RFC 2630. The content is made of the termination text. The "external signature" construct shall not be used (i.e. the eContent field in the EncapsulatedContentInfo field shall be present and contain the termination text string). The signing-time attribute, as defined in section 11.3 of RFC 2630, shall also be used to provide replay prevention. The client uses this to confirm its identity to the framework. The framework can check that the terminationText has been signed by the client. If a match is made, the access session is terminated, otherwise the P_INVALID_SIGNATURE exception will be thrown.

*Raises*

**TpCommonExceptions, P_INVALID_SIGNATURE**

### 6.3.1.6.8 Method ~~<<new>>~~ relinquishInterface()

The client uses this method to release an instance of a framework interface that was obtained during this access session.

*Parameters*

**interfaceName : in TpInterfaceName**

This is the name of the framework interface which is being released. If the interfaceName is invalid, the framework throws the P_INVALID_INTERFACE_NAME exception. If the interface has not been given to the client during this access session, then the P_TASK_REFUSED exception will be thrown.

**terminationText : in TpString**

This is the termination text describes the reason for the release of the interface. This text is required simply because the digitalSignature parameter requires a terminationText to sign.

**digitalSignature : in TpOctetSet**

This contains a CMS (Cryptographic Message Syntax) object (as defined in RFC 2630) with content type Signed-data. The signature is calculated and created as per section 5 of RFC 2630. The content is made of the termination text. The "external signature" construct shall not be used (i.e. the eContent field in the EncapsulatedContentInfo field shall be present and contain the termination text string). The signing-time attribute, as defined in section 11.3 of RFC 2630, shall also be used to provide replay prevention. The client uses this to confirm its identity to the framework. The framework can check that the terminationText has been signed by the client. If a match is made, the interface is released, otherwise the P_INVALID_SIGNATURE exception will be thrown.

*Raises*

**TpCommonExceptions, P_INVALID_SIGNATURE, P_INVALID_INTERFACE_NAME**

**End of Change in Clause 6.3**

**Start of Change in Clause 7.3.3**

## 7.3.3 Integrity Management Interface Classes

### 7.3.3.1 Interface Class IpAppFaultManager

Inherits from: IpInterface.

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

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

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

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 <<deprecated>> fwFaultRecoveryInd()

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

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 <<deprecated>> fwUnavailableInd()

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

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 SVC_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.

#### 7.3.3.1.16 Method <<new>> fwAvailStatusInd()

The framework invokes this method to inform the client application about the Framework availability status, i.e. that it can no longer use the Framework according to the reason parameter or that the Framework has become available again. The client application may wait for the problem to be solved and just stop the usage of the Framework until the fwAvailStatusInd() is called again with the reason FRAMEWORK_AVAILABLE.

*Parameters*

**reason : in TpFwAvailStatusReason**

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

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

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

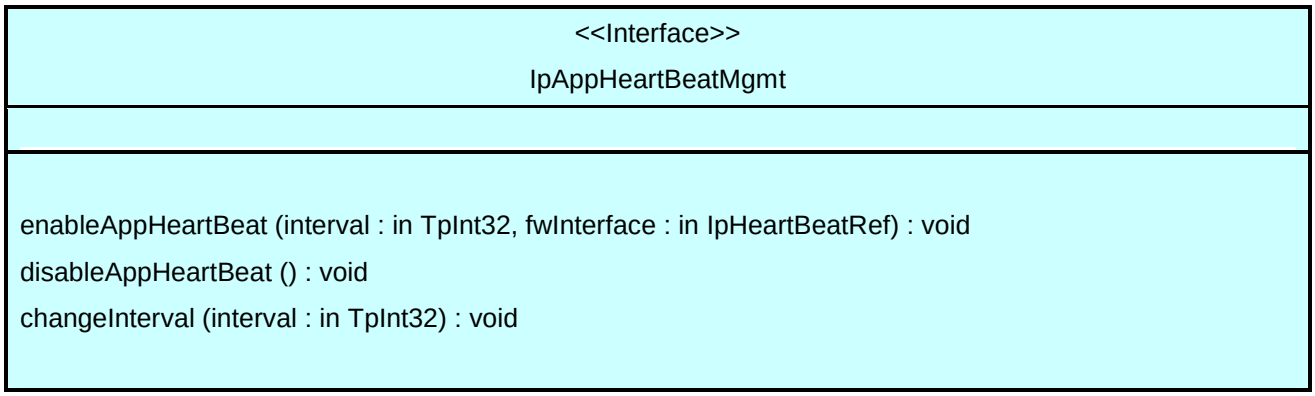
*Raises*

**TpCommonExceptions**

### 7.3.3.3 Interface Class IpAppHeartBeatMgmt

Inherits from: IpInterface.

This interface allows the initialisation of a heartbeat supervision of the client application by the framework.




#### 7.3.3.3.1 Method enableAppHeartBeat()

With this method, the framework instructs the client application to begin sending its heartbeat to the specified interface at the specified interval.

*Parameters*

**`interval : in TpInt32`**

The time interval in milliseconds between the heartbeats.

**`fwInterface : in IpHeartBeatRef`**

This parameter refers to the callback interface the heartbeat is calling.

#### 7.3.3.3.2 Method disableAppHeartBeat()

Instructs the client application to cease the sending of its heartbeat.

*Parameters*

No Parameters were identified for this method

#### 7.3.3.3.3 Method changeInterval()

Allows the administrative change of the heartbeat interval.

*Parameters*

**`interval : in TpInt32`**

The time interval in milliseconds between the heartbeats.

### 7.3.3.4 Interface Class IpAppHeartBeat

Inherits from: IpInterface.

The Heartbeat Application interface is used by the Framework to send the client application its heartbeat.

| <<Interface>><br>IpAppHeartBeat |
|---|
| |
| pulse () : void |

#### 7.3.3.4.1 Method pulse()

The framework uses this method to send its heartbeat to the client application. The application will be expecting a pulse at the end of every interval specified in the parameter to the IpHeartBeatMgmt.enableHeartbeat() method. If the pulse() is not received within the specified interval, then the framework can be deemed to have failed the heartbeat.

*Parameters*

No Parameters were identified for this method

### 7.3.3.5 Interface Class IpHeartBeatMgmt

Inherits from: IpInterface.

This interface allows the initialisation of a heartbeat supervision of the framework by a client application. If the IpHeartBeatMgmt interface is implemented by a Framework, as a minimum enableHeartBeat() and disableHeartBeat() shall be implemented.

| <<Interface>><br>IpHeartBeatMgmt |
|---|
| |
| enableHeartBeat (interval : in TpInt32, appInterface : in IpAppHeartBeatRef) : void<br>disableHeartBeat () : void<br>changeInterval (interval : in TpInt32) : void |

#### 7.3.3.5.1 Method enableHeartBeat()

With this method, the client application instructs the framework to begin sending its heartbeat to the specified interface at the specified interval.

*Parameters*

**`interval : in TpInt32`**

The time interval in milliseconds between the heartbeats.

**`appInterface : in IpAppHeartBeatRef`**

This parameter refers to the callback interface the heartbeat is calling.

*Raises*

**TpCommonExceptions**

#### 7.3.3.5.2 Method disableHeartBeat()

Instructs the framework to cease the sending of its heartbeat.

*Parameters*

No Parameters were identified for this method

*Raises*

**TpCommonExceptions**

#### 7.3.3.5.3 Method changeInterval()

Allows the administrative change of the heartbeat interval.

*Parameters*

**interval : in TpInt32**

The time interval in milliseconds between the heartbeats.

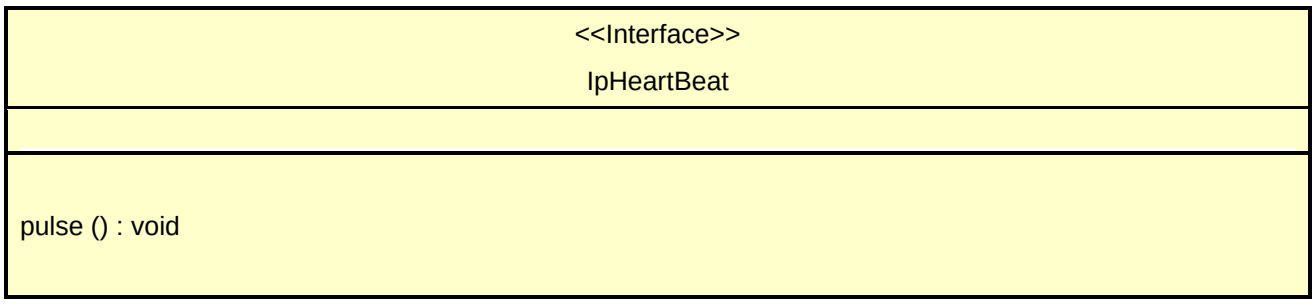
*Raises*

**TpCommonExceptions**

### 7.3.3.6 Interface Class IpHeartBeat

Inherits from: IpInterface.

The Heartbeat Framework interface is used by the client application to send its heartbeat. If a Framework is capable of invoking IpAppHeartBeatMgmt.enableHeartBeat(), it shall implement IpHeartBeat and the pulse() method.




#### 7.3.3.6.1 Method pulse()

The client application uses this method to send its heartbeat to the framework. The framework will be expecting a pulse at the end of every interval specified in the parameter to the IpAppHeartBeatMgmt.enableAppHeartbeat() method. If

the pulse() is not received within the specified interval, then the client application can be deemed to have failed the heartbeat.

*Parameters*

No Parameters were identified for this method

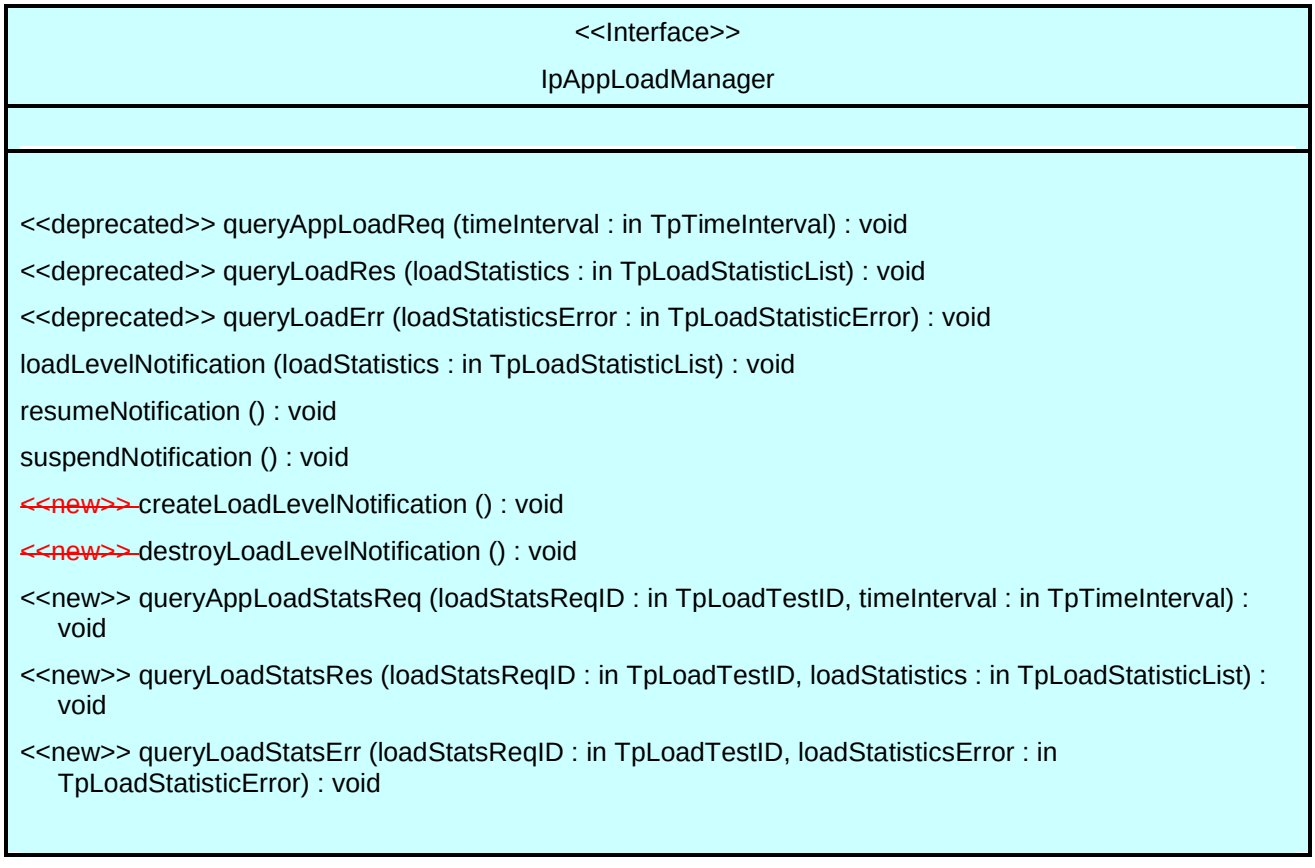
*Raises*

**TpCommonExceptions**

### 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.




#### 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.

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

### 7.3.3.9 Interface Class IpOAM

Inherits from: IpInterface.

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

| <<Interface>> IpOAM |
|---|
| |
| systemDateTimeQuery (clientDateAndTime : in TpDateAndTime) : TpDateAndTime |

#### 7.3.3.9.1 Method systemDateTimeQuery()

This method is used to query the system date and time. The client application passes in its own date and time to the framework. The framework responds with the system date and time.

Returns <systemDateAndTime> : This is the system date and time of the framework.

*Parameters*

**`clientDateAndTime : in TpDateAndTime`**

This is the date and time of the client (application). The error code P_INVALID_DATE_TIME_FORMAT is returned if the format of the parameter is invalid.

*Returns*

**`TpDateAndTime`**

*Raises*

**`TpCommonExceptions, P_INVALID_TIME_AND_DATE_FORMAT`**

### 7.3.3.10 Interface Class IpAppOAM

Inherits from: IpInterface.

The OAM client application interface is used by the Framework to query the application date and time, for synchronisation purposes. This method is invoked by the Framework to interchange the framework and client application date and time.

| <<Interface>> IpAppOAM |
|---|
| |
| systemDateTimeQuery (systemDateAndTime : in TpDateAndTime) : TpDateAndTime |

#### 7.3.3.10.1 Method systemDateTimeQuery()

This method is used to query the system date and time. The framework passes in its own date and time to the application. The application responds with its own date and time.

Returns <clientDateAndTime> : This is the date and time of the client (application).

*Parameters*

**systemDateA ndTime : in TpDateAndTime**

This is the system date and time of the framework.

*Returns*

**TpDateAndTime**

**End of Change in Clause 7.3.3**

**Start of Change in Clause 8.3.4**

## 8.3.4 Integrity Management Interface Classes

### 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.generateFaultStatisticsRecordReq(), it shall implement generateFaultStatisticsRecordRes() and generateFaultStatisticsRecordErr() in this interface. If the Framework is capable of invoking IpSvcFaultManager.generateFaultStatisticsRecordReq(), it shall implement generateFaultStatisticsRecordRes() and generateFaultStatisticsRecordErr() 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>> generateFaultStatsRecordRes (faultStatistics : in TpFaultStatsRecord) : void

<<deprecated>> 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>> 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>> 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`**

### 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>><br>IpSvcFaultManager |
|---|
| |
| activityTestRes (activityTestID : in TpActivityTestID, activityTestResult : in TpActivityTestRes) : void<br>svcActivityTestReq (activityTestID : in TpActivityTestID) : void<br><<deprecated>> fwFaultReportInd (fault : in TpInterfaceFault) : void<br><<deprecated>> fwFaultRecoveryInd (fault : in TpInterfaceFault) : void<br><<deprecated>> fwUnavailableInd (reason : in TpFwUnavailReason) : void<br>svcUnavailableInd () : void<br><<deprecated>> appUnavailableInd () : void<br><<deprecated>> genFaultStatsRecordRes (faultStatistics : in TpFaultStatsRecord, recordSubject : in TpSubjectType) : void<br>activityTestErr (activityTestID : in TpActivityTestID) : void<br><<deprecated>> genFaultStatsRecordErr (faultStatisticsError : in TpFaultStatisticsError, recordSubject : in TpSubjectType) : void<br><<deprecated>> genFaultStatsRecordReq (timePeriod : in TpTimeInterval, serviceIDs : in TpServiceIDList) : void<br><<deprecated>> generateFaultStatsRecordReq (timePeriod : in TpTimeInterval) : void<br>~~<<new>>~~ appAvailStatusInd (reason : in TpAppAvailStatusReason) : void<br><<new>> generateFaultStatisticsRecordRes (faultStatsReqID : in TpFaultReqID, faultStatistics : in TpFaultStatsRecord, recordSubject : in TpSubjectType) : void<br><<new>> generateFaultStatisticsRecordErr (faultStatsReqID : in TpFaultReqID, faultStatisticsError : in TpFaultStatisticsError, recordSubject : in TpSubjectType) : void<br><<new>> generateFaultStatisticsRecordReq (faultStatsReqID : in TpFaultReqID, timePeriod : in TpTimeInterval) : void<br><<new>> fwAvailStatusInd (reason : in TpFwAvailStatusReason) : 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 <<deprecated>> fwFaultReportInd()

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

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 <<deprecated>> fwFaultRecoveryInd()

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

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 <<deprecated>> fwUnavailableInd()

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

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>> 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**

#### 8.3.4.2.17 Method <<new>> fwAvailStatusInd()

The framework invokes this method to inform the service instance about the Framework availability status, i.e. that it can no longer use the Framework according to the reason parameter or that the Framework has become available again. The service instance may wait for the problem to be solved and just stop the usage of the Framework until the fwAvailStatusInd() is called again with the reason FRAMEWORK_AVAILABLE.

*Parameters*

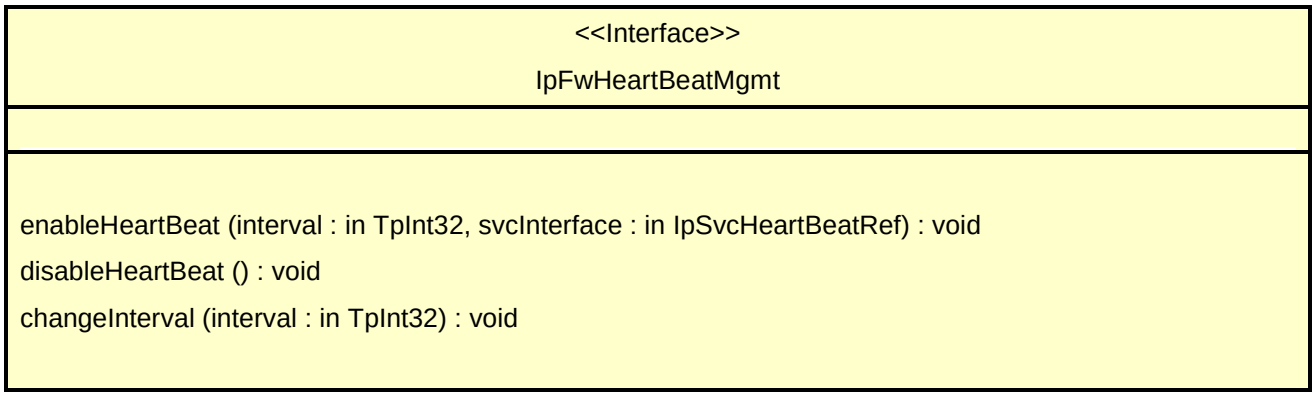
**`reason : in TpFwAvailStatusReason`**

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

### 8.3.4.3 Interface Class IpFwHeartBeatMgmt

Inherits from: IpInterface.

This interface allows the initialisation of a heartbeat supervision of the framework by a service instance. If the IpFwHeartBeatMgmt interface is implemented by a Framework, as a minimum enableHeartBeat() and disableHeartBeat() shall be implemented.




#### 8.3.4.3.1 Method enableHeartBeat()

With this method, the service instance instructs the framework to begin sending its heartbeat to the specified interface at the specified interval.

*Parameters*

**`interval : in TpInt32`**

The time interval in milliseconds between the heartbeats.

**`svcInterface : in IpSvcHeartBeatRef`**

This parameter refers to the callback interface the heartbeat is calling.

*Raises*

**`TpCommonExceptions, P_INVALID_INTERFACE_TYPE`**

#### 8.3.4.3.2 Method disableHeartBeat()

Instructs the framework to cease the sending of its heartbeat.

*Parameters*
No Parameters were identified for this method

*Raises*

**TpCommonExceptions**

#### 8.3.4.3.3 Method changeInterval()

Allows the administrative change of the heartbeat interval.

*Parameters*

**interval : in TpInt32**

The time interval in milliseconds between the heartbeats.

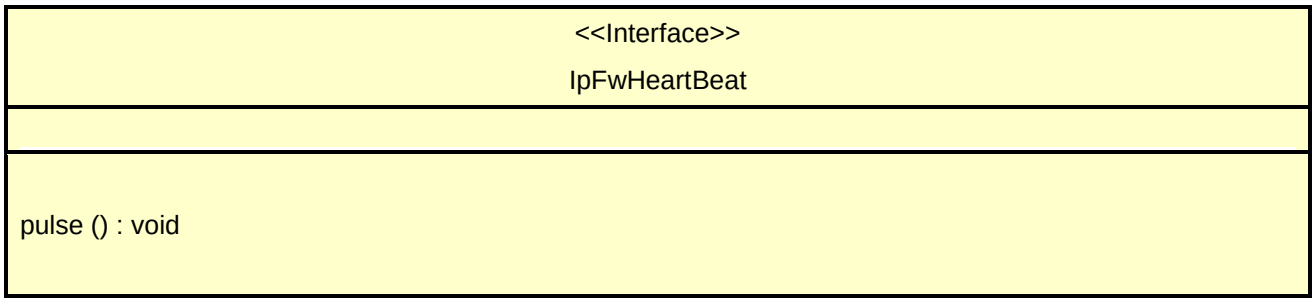
*Raises*

**TpCommonExceptions**

### 8.3.4.4 Interface Class IpFwHeartBeat

Inherits from: IpInterface.

The service side framework heartbeat interface is used by the service instance to send the framework its heartbeat. If a Framework is capable of invoking IpSvcHeartBeatMgmt.enableHeartBeat(), it shall implement IpFwHeartBeat and the pulse() method.




#### 8.3.4.4.1 Method pulse()

The service instance uses this method to send its heartbeat to the framework. The framework will be expecting a pulse at the end of every interval specified in the parameter to the IpSvcHeartBeatMgmt.enableSvcHeartbeat() method. If the pulse() is not received within the specified interval, then the service instance can be deemed to have failed the heartbeat.

*Parameters*
No Parameters were identified for this method

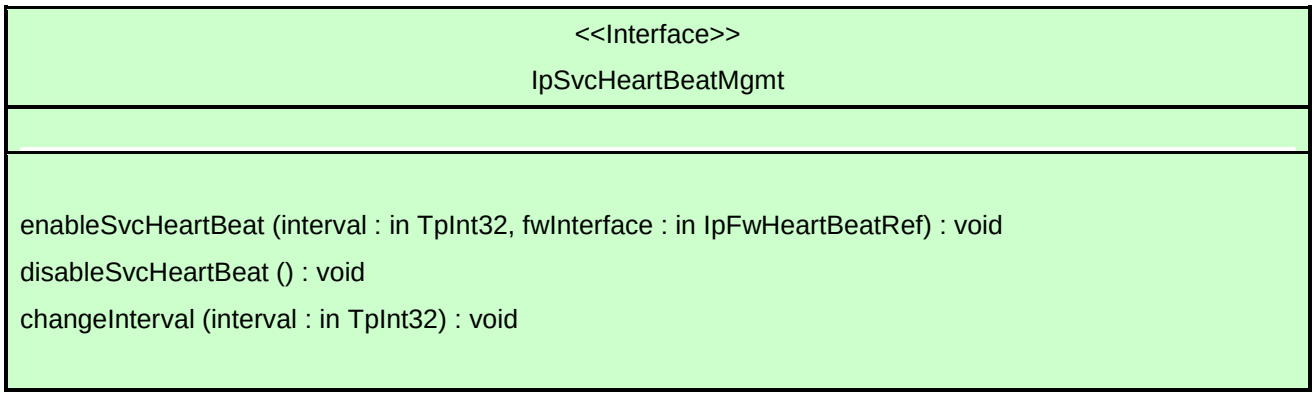
*Raises*

**TpCommonExceptions**

### 8.3.4.5 Interface Class IpSvcHeartBeatMgmt

Inherits from: IpInterface.

This interface allows the initialisation of a heartbeat supervision of the service instance by the framework. If the IpSvcHeartBeatMgmt interface is implemented by a Service, as a minimum enableHeartBeat() and disableHeartBeat() shall be implemented.




#### 8.3.4.5.1 Method enableSvcHeartBeat()

With this method, the framework instructs the service instance to begin sending its heartbeat to the specified interface at the specified interval.

*Parameters*

**interval : in TpInt32**

The time interval in milliseconds between the heartbeats.

**fwInterface : in IpFwHeartBeatRef**

This parameter refers to the callback interface the heartbeat is calling.

*Raises*

**TpCommonExceptions, P_INVALID_INTERFACE_TYPE**

#### 8.3.4.5.2 Method disableSvcHeartBeat()

Instructs the service instance to cease the sending of its heartbeat.

*Parameters*
No Parameters were identified for this method

*Raises*

**TpCommonExceptions**

#### 8.3.4.5.3 Method changeInterval()

Allows the administrative change of the heartbeat interval.

*Parameters*

**interval : in TpInt32**

The time interval in milliseconds between the heartbeats.

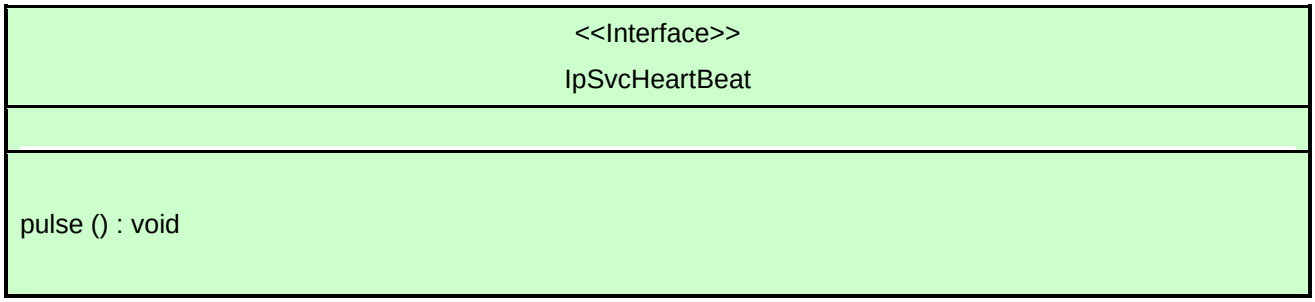
*Raises*

**TpCommonExceptions**

### 8.3.4.6 Interface Class IpSvcHeartBeat

Inherits from: IpInterface.

The service heartbeat interface is used by the framework to send the service instance its heartbeat. If a Service is capable of invoking IpFwHeartBeatMgmt.enableHeartBeat(), it shall implement IpSvcHeartBeat and the pulse() method.




#### 8.3.4.6.1 Method pulse()

The framework uses this method to send its heartbeat to the service instance. The service will be expecting a pulse at the end of every interval specified in the parameter to the IpFwHeartBeatMgmt.enableHeartbeat() method. If the pulse() is not received within the specified interval, then the framework can be deemed to have failed the heartbeat.

*Parameters*
No Parameters were identified for this method

*Raises*

**TpCommonExceptions**

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

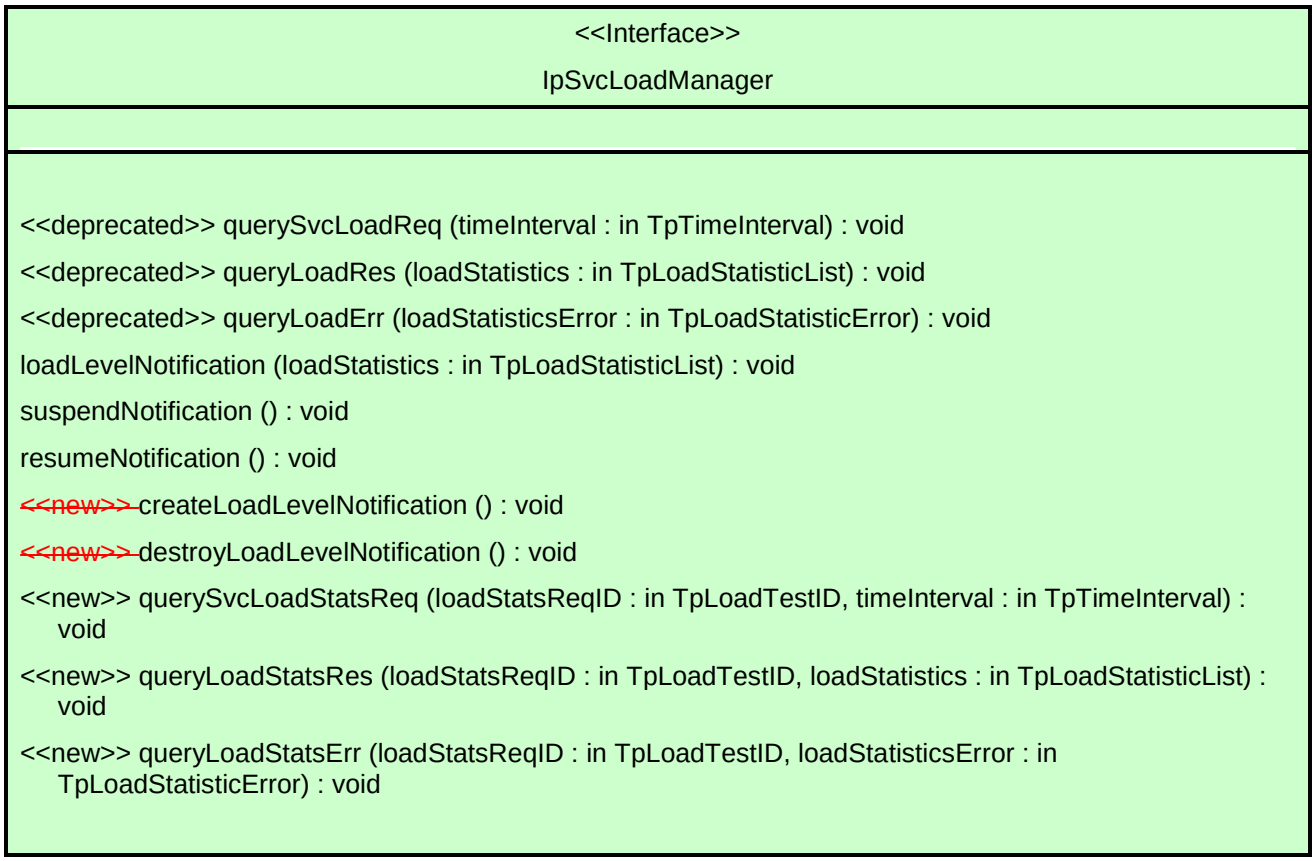
**TpCommonExceptions**

### 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.




#### 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**

#### 8.3.4.8.9 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**

#### 8.3.4.8.10 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**

#### 8.3.4.8.11 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.

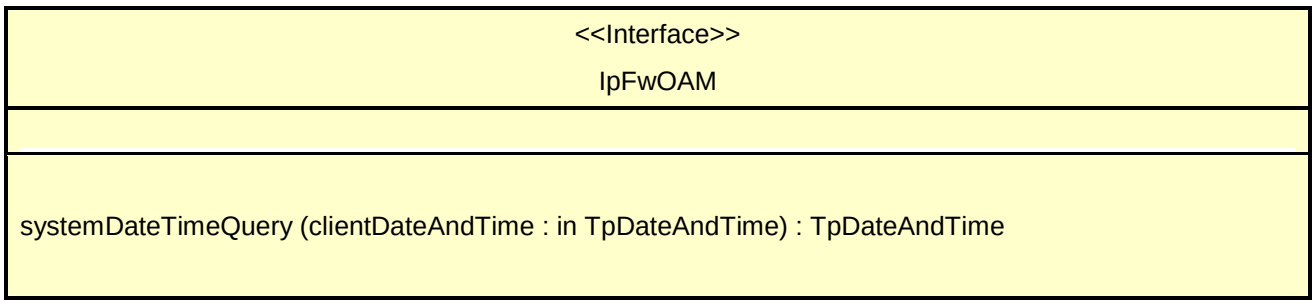
*Raises*

**TpCommonExceptions**

### 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 systemDateTimeQuery()

This method is used to query the system date and time. The client (service) passes in its own date and time to the framework. The framework responds with the system date and time.

Returns <systemDateAndTime> : This is the system date and time of the framework.

*Parameters*

**clientDateAndTime : in TpDateAndTime**

This is the date and time of the client (service). The error code P_INVALID_DATE_TIME_FORMAT is returned if the format of the parameter is invalid.

*Returns*

**TpDateAndTime**

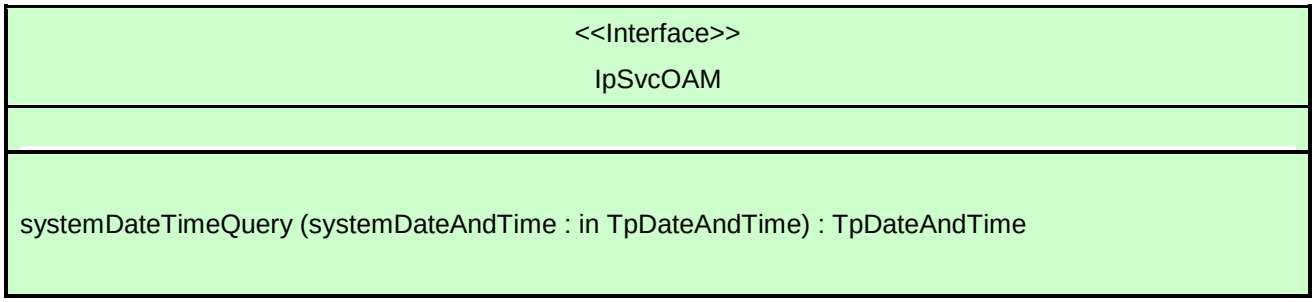
*Raises*

**TpCommonExceptions, P_INVALID_TIME_AND_DATE_FORMAT**

### 8.3.4.10 Interface Class IpSvcOAM

Inherits from: IpInterface.

This interface and the systemDateTimeQuery() method are optional.




#### 8.3.4.10.1 Method systemDateTimeQuery()

This method is used by the framework to send the system date and time to the service. The service responds with its own date and time.

Returns <clientDateAndTime> : This is the date and time of the client (service).

*Parameters*

**systemDateAndTime : in TpDateAndTime**

This is the system date and time of the framework. The error code P_INVALID_DATE_TIME_FORMAT is returned if the format of the parameter is invalid.

*Returns*

**TpDateAndTime**

*Raises*

**TpCommonExceptions, P_INVALID_TIME_AND_DATE_FORMAT**

**End of Change in Clause 8.3.4**

# Annex E (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 |
| Dec 2003 | CN_22 | NP-030549 | 086 | -- | Correction of the sequence diagram for Fault Management | 5.4.0 | 5.5.0 |
| Dec 2003 | CN_22 | NP-030549 | 087 | -- | Correction of State Transition Diagram for IpAccess | 5.4.0 | 5.5.0 |
| Dec 2003 | CN_22 | NP-030549 | 088 | -- | Correction of Correlation Behaviour in Load Management | 5.4.0 | 5.5.0 |
| Dec 2003 | CN_22 | NP-030549 | 089 | -- | Correction of Correlation Behaviour in Fault Management | 5.4.0 | 5.5.0 |
| Dec 2003 | CN_22 | NP-030549 | 090 | -- | Correction and Clarification of Framework Access Session Behaviour | 5.4.0 | 5.5.0 |
| Dec 2003 | CN_22 | NP-030553 | 091 | -- | Add OSA API support for 3GPP2 networks | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 092 | -- | Add description for service super and sub types | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 093 | -- | Add support for registration of additional service property types and modes | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 094 | -- | Improve User Interaction message management functions | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 095 | -- | Add new values for TpServiceTypeName for Policy Management | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 096 | -- | Allow for applications to re-obtain the reference to the service manager | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 097 | -- | Add support in OSA to inform applications about new SCSs and their level of Backward compatibility – Align with SA1's 22.127 | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 098 | -- | Add “Extended User Status” as service type name - Align with 29.198-06 | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 099 | -- | Add P_USER_BINDING to TpServiceTypeName | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 100 | -- | Modify Framework Availability Indication in Fault Management | 5.5.0 | 6.0.0 |
| Feb 2004 | -- | -- | -- | -- | Added Java code attachment 2919803J2EE.zip which was delivered late by outside developers. See Annex C. | 6.0.0 | 6.0.1 |
| | | | | | | | |
| | | | | | | | |

CR-Form-v7

# CHANGE REQUEST

⌘ 29.198-04-3 CR 024 ⌘rev - ⌘ Current version: 6.1.0 ⌘

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

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

| | | |
|---|---|---|
| ***Title:*** | ⌘ | Remove the <<new>> stereotype from methods which are no longer new. |
| ***Source:*** | ⌘ | CN5 Ultan Mulligan, ETSI PTCC |
| ***Work item code:*** | ⌘ | OSA3 |
| ***Date:*** | ⌘ | 14/05/2004 |
| ***Category:*** | ⌘ | F |
| ***Release:*** | ⌘ | REL-6 |

*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 <<new>> stereotype is used to indicate a new method has been added to the specification. However, some methods have been newly introduced more than a year ago, and in Rel-5, yet they still carry the <<new>> stereotype in Rel-6. For these methods, the <<new>> stereotype can be removed from their title and description.<br>Although editorial, this change is being applied throughout the 29.198 Rel-6 specification set, and so, on MCC advice, CRs have been generated to make this process visible. |
| ***Summary of change:*** | ⌘ | Remove the <<new>> stereotype from the documentation of methods which were newly introduced prior to the creation of the Rel-6 specifications, i.e. from methods in the Rel-6 specifications which were newly introduced into Rel-5 at or before the March 2003 plenary. |
| ***Consequences if not approved:*** | ⌘ | The specifications will contain misleading information, indicating certain methods as being newly introduced, when in fact they are not |

| | | | | | |
|---|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 6.1 | | | |
| | | **Y** | **N** | | |
| ***Other specs affected:*** | ⌘ | **X** | | Other core specifications | ⌘ TS 29.198-1, 3, 5, 7, 8, 11 and 12 Release 6 |
| | | | **X** | Test specifications | |
| | | | **X** | O&M Specifications | |
| ***Other comments:*** | ⌘ | N5-040283 to N5-040290 should be bundled together | | | |

**Change in Clause 6.1**

# 6 MultiParty Call Control Service Interface Classes

The Multi-party Call Control service enhances the functionality of the Generic Call Control Service with leg management. It also allows for multi-party calls to be established, i.e., up to a service specific number of legs can be connected simultaneously to the same call.

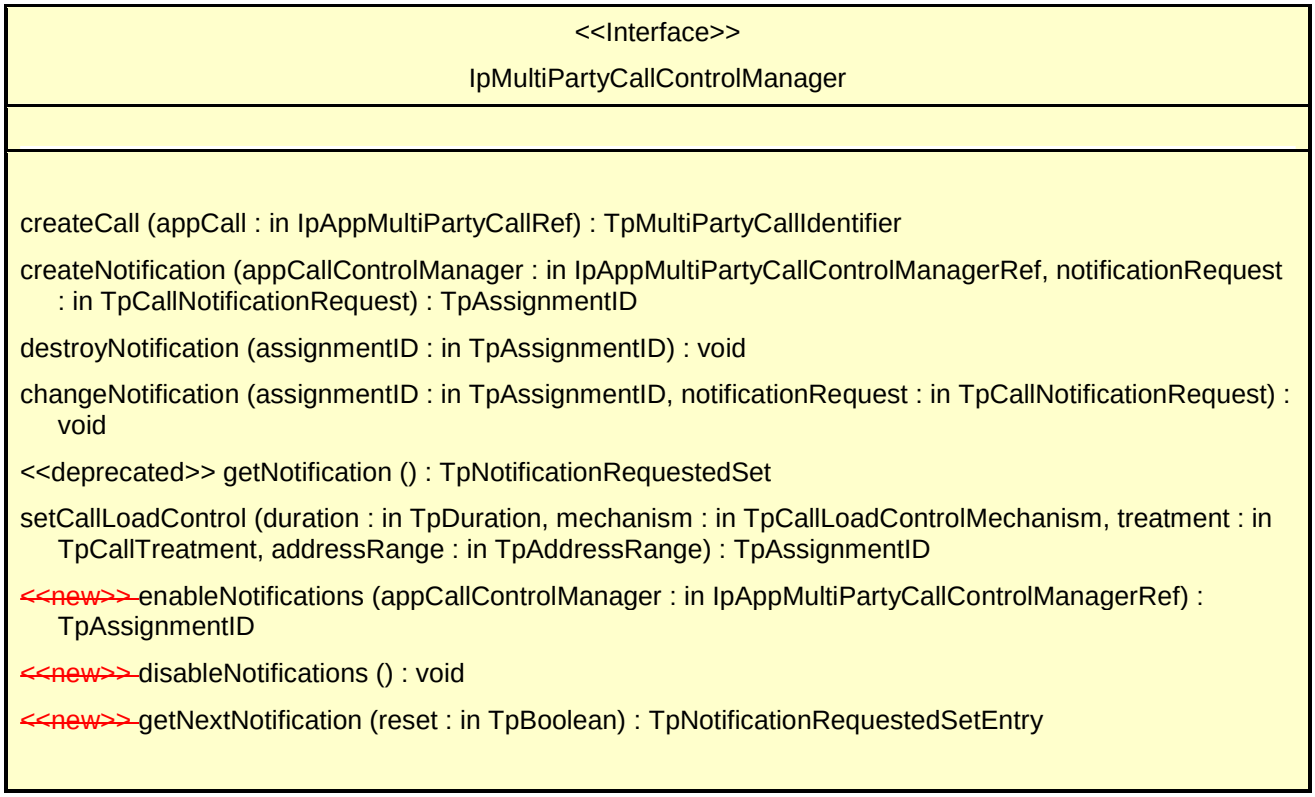
The Multi-party Call Control Service is represented by the IpMultiPartyCallControlManager, IpMultiPartyCall, IpCallLeg interfaces that interface to services provided by the network. Some methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. In this way, the client machine can handle many more calls, than one that uses synchronous message calls. To handle responses and reports, the developer must implement IpAppMultiPartyCallControlManager, IpAppMultiPartyCall and IpAppCallLeg to provide the callback mechanism.

## 6.1 Interface Class IpMultiPartyCallControlManager

Inherits from: IpService

This interface is the 'service manager' interface for the Multi-party Call Control Service. The multi-party call control manager interface provides the management functions to the multi-party call control service. The application programmer can use this interface to provide overload control functionality, create call objects and to enable or disable call-related event notifications. The action table associated with the STD shows in what state the IpMultiPartyCallControlManager must be if a method can successfully complete. In other words, if the IpMultiPartyCallControlManager is in another state the method will throw an exception immediately.

This interface shall be implemented by a Multi Party Call Control SCF. As a minimum requirement either the createCall() method shall be implemented, or the createNotification() and destroyNotification() methods shall be implemented, or the enableNotifications() and disableNotifications() methods shall be implemented.



### 6.1.1 Method createCall()

This method is used to create a new call object. An IpAppMultiPartyCallControlManager should already have been passed to the IpMultiPartyCallControlManager,

otherwise the call control will not be able to report a callAborted() to the application (the application should invoke setCallback() if it wishes to ensure this).

Returns callReference: Specifies the interface reference and sessionID of the call created.

*Parameters*

**appCall : in IpAppMultiPartyCallRef**

Specifies the application interface for callbacks from the call created.

*Returns*

**TpMultiPartyCallIdentifier**

*Raises*

**TpCommonExceptions, P_INVALID_INTERFACE_TYPE**

### 6.1.2 Method createNotification()

This method is used to enable call notifications so that events can be sent to the application. This is the first step an application has to do to get initial notifications of calls happening in the network. When such an event happens, the application will be informed by reportNotification(). In case the application is interested in other events during the context of a particular call session it has to use the createAndRouteCallLegReq() method on the call object or the eventReportReq() method on the call leg object. The application will get access to the call object when it receives the reportNotification(). (Note that createNotification() is not applicable if the call is setup by the application).

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

If some application already requested notifications with criteria that overlap the specified criteria or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA. The criteria are said to overlap when it leads to more than one application controlling the call or session at the same point in time during call or session processing.

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

If a notification is requested by an application with an event type that is mutually exclusive compared to existing requested event types, then there is no need to check against the rest of the criteria for overlap. An example could be one application that trigger on "user busy" together with another application that trigger on "answer" - both requests should be allowed as only one can occur on the same call or session.

The overlap criteria have been defined to prevent multiple points of control, leading to possible interaction problems in networks that have no multi service support. Notice that dynamic aspects cannot be taken into account in the overlap criteria check. Therefore where dynamic event arming from an application causes a persistent control relationship it can prevent other applications to be invoked in the case single point of application control applies in the network.

However, the criteria check for overlap may as a network option be overruled by Multi Service networks allowing more services or applications to gain control of the same call or session at the same point in time. Refer to Call Control Common Definitions subpart of this specification (TS 29.198-4-1) for further details on application control over a call or session.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The

gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the createNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().

Returns assignmentID: Specifies the ID assigned by the call control manager interface for this newly-enabled event notification.

*Parameters*

**appCallControlManager : in IpAppMultiPartyCallControlManagerRef**

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

**notificationRequest : in TpCallNotificationRequest**

Specifies the event specific criteria used by the application to define the event required. Only events that meet these criteria are reported. Examples of events are "incoming call attempt reported by network", "answer", "no answer", "busy". Individual addresses or address ranges may be specified for destination and/or origination.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE, P_INVALID_EVENT_TYPE**

### 6.1.3 Method destroyNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

Specifies the assignment ID given by the multi party call control manager interface when the previous createNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the exception P_INVALID_ASSIGNMENTID will be raised. If two callbacks have been registered under this assignment ID both of them will be disabled.

*Raises*

**TpCommonExceptions, P_INVALID_ASSIGNMENT_ID**

### 6.1.4 Method changeNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

Specifies the ID assigned by the multi party call control manager interface for the event notification. If two callbacks have been registered under this assignment ID both of them will be changed.

**notificationRequest : in TpCallNotificationRequest**

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

*Raises*

**TpCommonExceptions, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE**

### 6.1.5 Method <<deprecated>> getNotification()

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

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

Returns notificationsRequested: Specifies the notifications that have been requested by the application. An empty set is returned when no notifications exist.

*Parameters*

No Parameters were identified for this method

*Returns*

**TpNotificationRequestedSet**

*Raises*

**TpCommonExceptions**

### 6.1.6 Method setCallLoadControl()

This method imposes or removes load control on calls made to a particular address range within the call control service. The address matching mechanism is similar as defined for TpCallEventCriteria.

Returns assignmentID: Specifies the assignmentID assigned by the gateway to this request. This assignmentID can be used to correlate the callOverloadEncountered and callOverloadCeased methods with the request.

*Parameters*

**duration : in TpDuration**

Specifies the duration for which the load control should be set.

A duration of 0 indicates that the load control should be removed.

A duration of -1 indicates an infinite duration (i.e., until disabled by the application)

A duration of -2 indicates the network default duration.

**mechanism : in TpCallLoadControlMechanism**

Specifies the load control mechanism to use (for example, admit one call per interval), and any necessary parameters, such as the call admission rate. The contents of this parameter are ignored if the load control duration is set to zero.

**treatment : in TpCallTreatment**

Specifies the treatment of calls that are not admitted. The contents of this parameter are ignored if the load control duration is set to zero.

**addressRange : in TpAddressRange**

Specifies the address or address range to which the overload control should be applied or removed.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_INVALID_ADDRESS, P_UNSUPPORTED_ADDRESS_PLAN**

### 6.1.7 Method ~~<<new>>~~ enableNotifications()

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

If the same application requests to enable notifications for a second time with a different IpAppMultiPartyCallControlManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used.

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

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

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

*Parameters*

**appCallControlManager : in IpAppMultiPartyCallControlManagerRef**

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

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions**

### 6.1.8 Method ~~<<new>>~~ disableNotifications()

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

*Parameters*

No Parameters were identified for this method

*Raises*

**TpCommonExceptions**

### 6.1.9 Method ~~<<new>>~~ getNextNotification()

This method is used by the application to query the event criteria set with createNotification or changeNotification. Since a lot of data can potentially be returned (which might cause problem in the middleware), this method must be used in an iterative way. Each method invocation may return part of the total set of notifications if the set is too large to return it at once. The reset parameter permits the application to indicate whether an invocation to getNextNotification is requesting more notifications from the total set of notifications or is requesting that the total set of notifications shall be returned from the beginning.

Returns notificationRequestedSetEntry: The set of notifications and an indication whether all off the notifications have been obtained or if more notifications are available that have not yet been obtained by the application. If no notifications exist, an empty set is returned and the final indication shall be set to TRUE.

Note that the (maximum) number of items provided to the application is determined by the gateway.

*Parameters*

**reset : in TpBoolean**

TRUE: indicates that the application is intended to obtain the set of notifications starting at the beginning.

FALSE: indicates that the application requests the next set of notifications that have not (yet) been obtained since the last call to this method with this parameter set to TRUE.

The first time this method is invoked, reset shall be set to TRUE. Following the receipt of a final indication in TpNotificationRequestedSetEntry, for the next call to this method reset shall be set to TRUE. P_TASK_REFUSED may be thrown if these conditions are not met.

*Returns*

**TpNotificationRequestedSetEntry**

*Raises*

**TpCommonExceptions**

**End of Change in Clause 6.1**

# Annex E (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 | 1.0.0 |
| June 2001 | CN_12 | NP-010327 | -- | -- | Approved at TSG CN#12 and placed under Change Control | 2.0.0 | 4.0.0 |
| Sep 2001 | CN_13 | NP-010467 | 001 | -- | Changing references to JAIN | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 002 | -- | Correction of text descriptions for methods enableCallNotification and createNotification | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 003 | -- | Specify the behaviour when a call leg times out | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 004 | -- | Removal of Faulty state in MPCCS Call State Transition Diagram and method callFaultDetected in MPCCS in OSA R4 | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 005 | -- | Missing TpCallAppInfoSet description in OSA R4 | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 006 | -- | Redirecting a call leg vs. creating a call leg clarification in OSA R4 | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 007 | -- | Introduction of MPCC Originating and Terminating Call Leg STDs for IpCallLeg | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 008 | -- | Corrections to SetChargePlan() Addition of PartyToCharge parmeter | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 009 | -- | Corrections to SetChargePlan() | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 010 | -- | Remove distinction between final- and intermediate-report | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 011 | -- | Inclusion of TpMediaType | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 012 | -- | Corrections to GCC STD | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 013 | -- | Introduction of sequence diagrams for MPCC services | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 014 | -- | The use of the REDIRECT event needs to be illustrated | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 015 | -- | Corrections to SetCallChargePlan() | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 016 | -- | Add one additional error indication | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 017 | -- | Corrections to Call Control – GCCS Exception handling | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010467 | 018 | -- | Corrections to Call Control – Errors in Exceptions | 4.0.0 | 4.1.0 |
| Dec 2001 | CN_14 | NP-010597 | 019 | -- | Replace Out Parameters with Return Types | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 020 | -- | Removal of time based charging property | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 021 | -- | Make attachMedia() and detachMedia() asynchronous | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 022 | -- | Correction of treatment datatype in superviseReq on call leg | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 023 | -- | Corrections to Call Control Data Types | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 024 | -- | Correction to Call Control (CC) | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 025 | -- | Amend the Generic Call Control introductory part | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 026 | -- | Correction in TpCallEventType | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 027 | -- | Addition of missing description of RouteErr() | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 028 | -- | Misleading description of createAndRouteCallLegErr() | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010597 | 029 | -- | Correction to values of TpCallNotificationType, TpCallLoadControlMechanismType | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010695 | 030 | -- | Correction of method getLastRedirectionAddress | 4.1.0 | 4.2.0 |
| Mar 2002 | CN_15 | NP-020106 | 031 | -- | Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID() | 4.2.0 | 4.3.0 |
| Mar 2002 | CN_15 | NP-020106 | 032 | -- | Correction of Event Subscription/Notification Data Type | 4.2.0 | 4.3.0 |
| Mar 2002 | CN_15 | NP-020106 | 033 | -- | Correction of parameter name in IpCallLeg.routeReq() and in IpCallLeg.setAdviceOfCharge() | 4.2.0 | 4.3.0 |
| Mar 2002 | CN_15 | NP-020106 | 034 | -- | Clarification of ambiguous Event handling rules | 4.2.0 | 4.3.0 |
| Jun 2002 | CN_16 | NP-020180 | 035 | -- | Correction to TpCallChargePlan | 4.3.0 | 4.4.0 |
| Jun 2002 | CN_16 | NP-020180 | 036 | -- | Correction to CAMEL Service Property values | 4.3.0 | 4.4.0 |
| Jun 2002 | CN_16 | NP-020181 | 037 | - | Addition of support for Java API technology realisation | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020182 | 038 | - | Addition of support for WSDL realisation | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 039 | - | Addition of support for Emergency Telecommunications Service | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020183 | 040 | - | Addition of support for Network Controlled Notifications MPCC | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 041 | - | Changes to getNotification() | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 042 | - | Addition of P_UNSUPPORTED_MEDIA release cause to TpReleaseCause | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 043 | - | Addition of CAMEL Phase 4 Service Property values | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 044 | - | Addition of indication whether SCS supports initially multiple routeReqs in parallel | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 045 | - | Explicit exception for continueProcessing when not in interrupted mode | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 046 | - | Indication needed that supervision will be ended when call or callLeg is deassigned | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 047 | - | Clarify ambiguous Supervision duration | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 048 | - | Detach/Attach request illegal during pending Attach/Detach request | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 049 | - | Correction of Multi-Party Call Control properties | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 050 | - | Correcting the sequence diagram descriptions in GCC and MPCC | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 051 | - | Correcting erroneous description of UI behaviour in call control | 4.4.0 | 5.0.0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Jun 2002 | CN_16 | NP-020187 | 052 | - | Correcting the descriptions of sequence diagrams that don't match the diagram | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 053 | - | Correcting erroneous references to GCC in MPCC | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 054 | - | Addition of the Multi-media APIs to Call control SCF (29.198-4) | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020187 | 055 | - | Updating Clause 4 for Release 5 | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020188 | 056 | - | Spliting of 29.198-04 into 4 separate TSs (sub-parts) | 4.4.0 | 5.0.0 |
| Sep 2002 | CN_17 | NP-020431 | 001 | | 29.198-04-3 Correction of error in Call Forward on Busy sequence diagram | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020431 | 002 | | Correct inconsistencies in IpCallLeg state transition diagrams | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020431 | 003 | | Clarification of the overlapping criteria definition and eventType mapping to IN TDPs | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020431 | 004 | | Add support for Carrier selection | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020431 | 005 | | Correction on use of NULL in Call Control API | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020395 | 006 | | Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications | 5.0.0 | 5.1.0 |
| Mar 2003 | CN_19 | NP-030031 | 007 | -- | Correction of status of MPCC methods | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030031 | 008 | -- | Inconsistent description of use of secondary callback | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030020 | 009 | -- | Correction to TpReleaseCauseSet in Multi Party Call Control IDL | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030130 | 010 | -- | Correction of definition of the P_MAX_CALLLEGS_PER_CALL | 5.1.0 | 5.2.0 |
| Jun 2003 | CN_20 | NP-030238 | 011 | -- | Correction of the description for callEventNotify & reportNotification | 5.2.0 | 5.3.0 |
| Jun 2003 | CN_20 | NP-030305 | 012 | 1 | Unclear overlap criteria for rejection of createNotification | 5.3.0 | 6.0.0 |
| Jun 2003 | CN_20 | NP-030247 | 013 | -- | Add support for advanced subscriber presentation | 5.3.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030550 | 017 | -- | Correction of description of TpNotificationRequestedSetEntry | 6.0.0 | 6.1.0 |
| Dec 2003 | CN_22 | NP-030553 | 019 | -- | Add OSA API support for 3GPP2 networks | 6.0.0 | 6.1.0 |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

CR-Form-v7

# CHANGE REQUEST

⌘ **29.198-05** **CR** **050** ⌘ **rev** **-** ⌘ Current version: **6.0.1** ⌘

*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:*** | ⌘ | Remove the <<new>> stereotype from methods which are no longer new |
| ***Source:*** | ⌘ | CN5 Ultan Mulligan, ETSI PTCC |
| ***Work item code:*** | ⌘ | OSA3 |
| ***Date:*** | ⌘ | 14/05/2004 |
| ***Category:*** | ⌘ | **F** |
| ***Release:*** | ⌘ | *REL-6* |

*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 <<new>> stereotype is used to indicate a new method has been added to the specification. However, some methods have been newly introduced more than a year ago, and in Rel-5, yet they still carry the <<new>> stereotype in Rel-6. For these methods, the <<new>> stereotype can be removed from their title and description.<br>Although editorial, this change is being applied throughout the 29.198 Rel-6 specification set, and so, on MCC advice, CRs have been generated to make this process visible. |
| ***Summary of change:*** | ⌘ | Remove the <<new>> stereotype from the documentation of methods which were newly introduced prior to the creation of the Rel-6 specifications, i.e. from methods in the Rel-6 specifications which were newly introduced into Rel-5 at or before the March 2003 plenary. |
| ***Consequences if not approved:*** | ⌘ | The specifications will contain misleading information, indicating certain methods as being newly introduced, when in fact they are not |

| | | | | | |
|---|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 8.1 | | | |
| ***Other specs affected:*** | ⌘ | **Y** / **N** | Other core specifications | ⌘ | 29.198, 3, 4-3, 7, 8, 11 and 12 Release 6 |
| | | **X** / | Other core specifications | | |
| | | / **X** | Test specifications | | |
| | | / **X** | O&M Specifications | | |
| ***Other comments:*** | ⌘ | N5-040284 to N5-040290 should be bundled together | | | |

**Change in Clause 8.1**

## 8.1.1 Interface Class IpUIManager

Inherits from: IpService.

This interface is the 'service manager' interface for the Generic User Interaction Service and provides the management functions to the Generic User Interaction Service.

This interface shall be implemented by a Generic User Interaction SCF. The createUI() method, or the createUICall() method, or both the createNotification() and destroyNotification methods, or both the enableNotifications() and disableNotifications() methods shall be implemented as a minimum requirement.

| <<Interface>><br>IpUIManager |
|---|
| |
| createUI (appUI : in IpAppUIRef, userAddress : in TpAddress) : TpUIIdentifier<br>createUICall (appUI : in IpAppUICallRef, uiTargetObject : in TpUITargetObject) : TpUICallIdentifier<br>createNotification (appUIManager : in IpAppUIManagerRef, eventCriteria : in TpUIEventCriteria) : TpAssignmentID<br>destroyNotification (assignmentID : in TpAssignmentID) : void<br>changeNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpUIEventCriteria) : void<br>getNotification () : TpUIEventCriteriaResultSet<br>~~<<new>>~~ enableNotifications (appUIManager : in IpAppUIManagerRef) : TpAssignmentID<br>~~<<new>>~~ disableNotifications () : void |

### 8.1.1.1 Method createUI()

This method is used to create a new user interaction object for non-call related purposes

Results: userInteraction

Specifies the interface and sessionID of the user interaction created.

*Parameters*

**appUI : in IpAppUIRef**

Specifies the application interface for callbacks from the user interaction created.

**userAddress : in TpAddress**

Indicates the end-user with whom to interact.

*Returns*

**TpUIIdentifier**

*Raises*

**TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_INTERFACE_TYPE**

### 8.1.1.2 Method createUICall()

This method is used to create a new user interaction object for call related purposes.

The user interaction can take place to the specified party or to all parties in a call. Note that for certain implementation user interaction can only be performed towards the controlling call party, which shall be the only party in the call.

Returns: userInteraction

Specifies the interface and sessionID of the user interaction created.

*Parameters*

**appUI : in IpAppUICallRef**

Specifies the application interface for callbacks from the user interaction created.

**uiTargetObject : in TpUITargetObject**

Specifies the object on which to perform the user interaction. This can either be a Call, Multi-party Call or call leg object.

*Returns*

**TpUICallIdentifier**

*Raises*

**TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_INTERFACE_TYPE**

### 8.1.1.3 Method createNotification()

This method is used by the application to install specified notification criteria, for which the reporting is implicitly activated. If some application already requested notifications with criteria that overlap the specified criteria, or the specified criteria overlap with criteria already present in the network (when provisioned from within the network), the request is refused with P_INVALID_CRITERIA.

The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used and the same servicecode is used.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used.

Returns: assignmentID

Specifies the ID assigned by the generic user interaction manager interface for this newly installed notification criteria.

*Parameters*

**appUIManager : in IpAppUIManagerRef**

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

**eventCriteria : in TpUIEventCriteria**

Specifies the event specific criteria used by the application to define the event required, like user address and service code.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_INVALID_CRITERIA, P_INVALID_INTERFACE_TYPE**

### 8.1.1.4 Method destroyNotification()

This method is used by the application to destroy previously installed notification criteria via the createNotification method.

*Parameters*

**assignmentID : in TpAssignmentID**

Specifies the assignment ID given by the generic user interaction manager interface when the previous createNotification() was called. If the assignment ID does not correspond to one of the valid assignment IDs, the framework will return the error code P_INVALID_ASSIGNMENT_ID.

*Raises*

**TpCommonExceptions, P_INVALID_ASSIGNMENT_ID**

### 8.1.1.5 Method changeNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

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

**eventCriteria : in TpUIEventCriteria**

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

*Raises*

**TpCommonExceptions, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA**

### 8.1.1.6 Method getNotification()

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

Returns: eventCriteria

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

*Parameters*
No Parameters were identified for this method

*Returns*

**TpUIEventCriteriaResultSet**

*Raises*

**TpCommonExceptions**

### 8.1.1.7 Method ~~<<new>>~~ enableNotifications()

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

If the same application requests to enable notifications for a second time with a different IpAppUIManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used.

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

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

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

*Parameters*

**appUIManager : in IpAppUIManagerRef**

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

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions**

### 8.1.1.8 Method ~~<<new>>~~ disableNotifications()

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

*Parameters*
No Parameters were identified for this method

*Raises*

**TpCommonExceptions**

## 8.1.2 Interface Class IpAppUIManager

Inherits from: IpInterface.

The Generic User Interaction Service manager application interface provides the application callback functions to the Generic User Interaction Service.

| <<Interface>><br>IpAppUIManager |
|---|
| |
| userInteractionAborted (userInteraction : in TpUIIdentifier) : void<br><<deprecated>> reportNotification (userInteraction : in TpUIIdentifier, eventInfo : in TpUIEventInfo, assignmentID : in TpAssignmentID) : IpAppUIRef<br>userInteractionNotificationInterrupted () : void<br>userInteractionNotificationContinued () : void<br>~~<<new>>~~ reportEventNotification (userInteraction : in TpUIIdentifier, eventNotificationInfo : in TpUIEventNotificationInfo, assignmentID : in TpAssignmentID) : IpAppUIRef |

### 8.1.2.1 Method userInteractionAborted()

This method indicates to the application that the User Interaction service instance has terminated or closed abnormally. No further communication will be possible between the User Interaction service instance and application.

*Parameters*

**userInteraction : in TpUIIdentifier**

Specifies the interface and sessionID of the user interaction service that has terminated.

### 8.1.2.2 Method <<deprecated>> reportNotification()

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

This method notifies the application of an occurred network event which matches the criteria installed by the createNotification method.

Returns: appUI

Specifies a reference to the application interface, which implements the callback interface for the new user interaction.

If the application has previously explicitly passed a reference to the IpAppUI interface using a setCallbackWithSessionID() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallbackWithSessionID().

*Parameters*

**`userInteraction : in TpUIIdentifier`**

Specifies the reference to the interface and the sessionID to which the notification relates.

**`eventInfo : in TpUIEventInfo`**

Specifies data associated with this event.

**`assignmentID : in TpAssignmentID`**

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

*Returns*

**`IpAppUIRef`**

### 8.1.2.3 Method userInteractionNotificationInterrupted()

This method indicates to the application that all event notifications have been temporarily interrupted (for example, due to faults detected). Note that more permanent failures are reported via the Framework (integrity management).

*Parameters*

No Parameters were identified for this method

### 8.1.2.4 Method userInteractionNotificationContinued()

This method indicates to the application that event notifications will again be possible.

*Parameters*

No Parameters were identified for this method

### 8.1.2.5 Method ~~<<new>>~~ reportEventNotification()

This method notifies the application of an occurred network event which matches the criteria installed by the createNotification method.

Returns: appUI

Specifies a reference to the application interface, which implements the callback interface for the new user interaction.

If the application has previously explicitly passed a reference to the IpAppUI interface using a setCallbackWithSessionID() invocation, this parameter may be null, or if supplied must be the same as that provided during the setCallbackWithSessionID().

*Parameters*

**`userInteraction : in TpUIIdentifier`**

Specifies the reference to the interface and the sessionID to which the notification relates.

**`eventNotificationInfo : in TpUIEventNotificationInfo`**

Specifies data associated with this event.

**`assignmentID : in TpAssignmentID`**

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

*Returns*

**`IpAppUIRef`**

**End of Change in Clause 8.1**

# Annex E (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 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010468 | 002 | -- | Changing references to JAIN | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010598 | 003 | -- | Replace Out Parameters with Return Types | 4.2.0 | 4.3.0 |
| Dec 2001 | CN_14 | NP-010598 | 004 | -- | Correction of description of sendInfoRes() | 4.2.0 | 4.3.0 |
| Dec 2001 | CN_14 | NP-010598 | 005 | -- | Correction to handling of deassign on related object | 4.2.0 | 4.3.0 |
| Dec 2001 | CN_14 | NP-010598 | 006 | -- | Correction to Exceptions Raised in UI | 4.2.0 | 4.3.0 |
| Dec 2001 | CN_14 | NP-010598 | 007 | -- | Correction to values of TpUIInfoType | 4.2.0 | 4.3.0 |
| Mar 2002 | CN_15 | NP-020107 | 008 | -- | Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID() | 4.3.0 | 4.4.0 |
| Jun 2002 | CN_16 | NP-020181 | 009 | -- | Addition of support for Java API technology realisation | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020189 | 010 | -- | Improve the vague description of P_ID_NOT_FOUND | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020182 | 011 | -- | Addition of support for WSDL realisation | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020189 | 012 | -- | Detach call leg before playing announcement or collecting digits | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020189 | 013 | -- | Delete P_INVALID_CRITERIA from sendInfoAndCollectReq() | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020183 | 014 | -- | Addition of Support for Network Controlled Notifications UI | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020189 | 015 | -- | Correcting erroneous description of UI behaviour in call control | 4.4.0 | 5.0.0 |
| Sep 2002 | CN_17 | NP-020432 | 018 | -- | Add text to clarify requirements on support of methods | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020432 | 019 | -- | Correction on use of NULL in User Interaction API | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020432 | 020 | -- | Correction to TpUIInfo data type to support binary data for SMS services | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020395 | 021 | | Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications | 5.0.0 | 5.1.0 |
| Mar 2003 | CN_19 | NP-030021 | 023 | -- | Correction to User Interaction Prepaid Sequence Diagrams | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030021 | 025 | -- | Correction to getNotification to remove P_INVALID_CRITERIA exception | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030021 | 028 | -- | Addition of status of methods to User Interaction interfaces | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030021 | 031 | -- | Corrections to User Interaction | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030021 | 033 | -- | Correction of User Interaction Event Notification to support non text encodings | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030033 | 029 | -- | Inconsistent description of use of secondary callback | 5.1.0 | 5.2.0 |
| Jun 2003 | CN_20 | NP-030238 | 035 | -- | Correction of the description for callEventNotify & reportNotification | 5.2.0 | 5.3.0 |
| Jun 2003 | CN_20 | NP-030244 | 036 | -- | Clarify IpUI sendInfoReq() | 5.2.0 | 5.3.0 |
| Jun 2003 | CN_20 | NP-030244 | 037 | -- | Update TpUIInfo for consistency with GMS capabilities | 5.2.0 | 5.3.0 |
| Jun 2003 | CN_20 | NP-030299 | 038 | 1 | Specifying the origin of a GUI message | 5.2.0 | 5.3.0 |
| Sep 2003 | CN_21 | NP-030352 | 039 | -- | Correction to Java Realisation Annex | 5.3.0 | 5.4.0 |
| Dec 2003 | CN_22 | NP-030545 | 041 | -- | Correction to UI service responseRequested logic | 5.4.0 | 5.5.0 |
| Dec 2003 | CN_22 | NP-030553 | 042 | -- | Add OSA API support for 3GPP2 networks | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030554 | 043 | -- | Improve User Interaction message management functions | 5.5.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030555 | 044 | -- | Add speech recognition/synthesis capability to the Generic User Interaction | 5.5.0 | 6.0.0 |
| Feb 2004 | -- | -- | -- | -- | Added Java code attachment 2919805J2EE.zip which was delivered late by outside developers. See Annex C. | 6.0.0 | 6.0.1 |
| | | | | | | | |
| | | | | | | | |

CR-Form-v7

# CHANGE REQUEST

⌘ 29.198-07 CR 017 ⌘rev - ⌘ Current version: 6.0.1 ⌘

*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:*** | ⌘ | Remove the <<new>> stereotype from methods which are no longer new |
| ***Source:*** | ⌘ | CN5 Ultan Mulligan, ETSI PTCC |
| ***Work item code:*** | ⌘ | OSA3 |
| ***Date:*** | ⌘ | 14/05/2004 |
| ***Category:*** | ⌘ | F |
| ***Release:*** | ⌘ | REL-6 |

*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 <<new>> stereotype is used to indicate a new method has been added to the specification. However, some methods have been newly introduced more than a year ago, and in Rel-5, yet they still carry the <<new>> stereotype in Rel-6. For these methods, the <<new>> stereotype can be removed from their title and description.<br>Although editorial, this change is being applied throughout the 29.198 Rel-6 specification set, and so, on MCC advice, CRs have been generated to make this process visible. |
| ***Summary of change:*** | ⌘ | Remove the <<new>> stereotype from the documentation of methods which were newly introduced prior to the creation of the Rel-6 specifications, i.e. from methods in the Rel-6 specifications which were newly introduced into Rel-5 at or before the March 2003 plenary. |
| ***Consequences if not approved:*** | ⌘ | The specifications will contain misleading information, indicating certain methods as being newly introduced, when in fact they are not |

| | | | | | |
|---|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 8.2 | | | |
| | | **Y** | **N** | | |
| ***Other specs affected:*** | ⌘ | **X** | | Other core specifications | TS 29.198-3, 4-3, 5, 8, 11 and 12 Release 6 |
| | | | **X** | Test specifications | |
| | | | **X** | O&M Specifications | |
| ***Other comments:*** | ⌘ | N5-040284 to N5-040290 should be bundled together | | | |

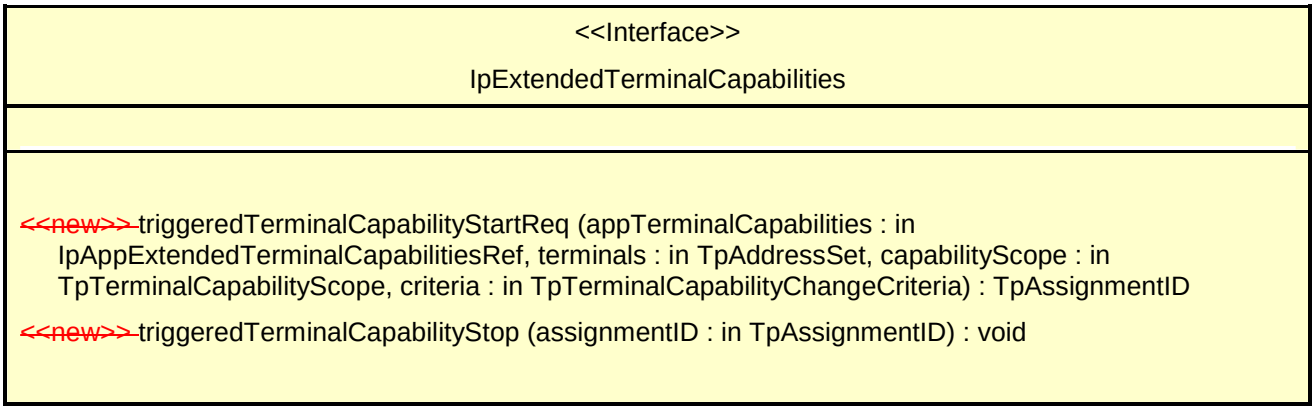
**Change in Clause 8.2**

# 8.2 Interface Class IpExtendedTerminalCapabilities

Inherits from: IpTerminalCapabilities.

This interface can be used as an extended version of terminal capability monitoring. The application programmer can use this interface to request terminal capability reports that are triggered by their changes. Note that the underlying mechanisms for this network feature are currently not fully standardised.

This interface, or IpTerminalCapabilities, shall be implemented by a Terminal Capabilities SCF as a minimum requirement. The triggeredTerminalCapabilityStartReq() and triggeredTerminalCapabilityStop() methods shall be implemented as a minimum requirement. An implementation of IpExtendedTerminalCapabilities is not required to implement the minimum mandatory methods of IpTerminalCapabilities.




## 8.2.1 Method ~~<<new>>~~ triggeredTerminalCapabilityStartReq()

Request for terminal capability reports when the capabilities change or when the application obviously does not have the current terminal capability information when this method is invoked.

Returns: assignmentID

Specifies the assignment ID of the triggered terminal capability reporting request.

*Parameters*

**`appTerminalCapabilities : in IpAppExtendedTerminalCapabilitiesRef`**

Specifies the application interface for callbacks.

**`terminals : in TpAddressSet`**

Specifies the terminal(s) for which the capabilities shall be reported. TpAddress fields have the following use:

- Plan: Used to indicate the numbering plan
- AddrString: Used to indicate the subscriber address
- Name: Used to indicate the terminal identity. May be applied also together with AddrString to indicate subscriber's particular terminal. The precise format is not defined.
- Presentation: No defined use
- Screening: No defined use
- SubAddressString: No defined use

Hence it is possible to indicate the subscriber and/or the terminal identification. This terminal addressing is implementation specific e.g. subscriber identification may not always be sufficient information to get the capabilities of the terminal.

**capabilityScope : in TpTerminalCapabilityScope**

Specifies the scope of the capabilities that the application is interested in. The contents are implementation specific. One possibility is to use the CC/PP definitions as in TpTerminalCapabilities.

**criteria : in TpTerminalCapabilityChangeCriteria**

Specifies the trigger conditions for the reports e.g. software or hardware update.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_INFORMATION_NOT_AVAILABLE, P_INVALID_INTERFACE_TYPE, P_INVALID_CRITERIA, P_INVALID_TERMINAL_ID**

### 8.2.2 Method ~~<<new>>~~ triggeredTerminalCapabilityStop()

Stop reporting for terminal capability changes that were started by triggeredTerminalCapabilityStartReq().

*Parameters*

**assignmentID : in TpAssignmentID**

Specifies the assignment ID for the task to be stopped.

*Raises*

**TpCommonExceptions, P_INVALID_ASSIGNMENT_ID**

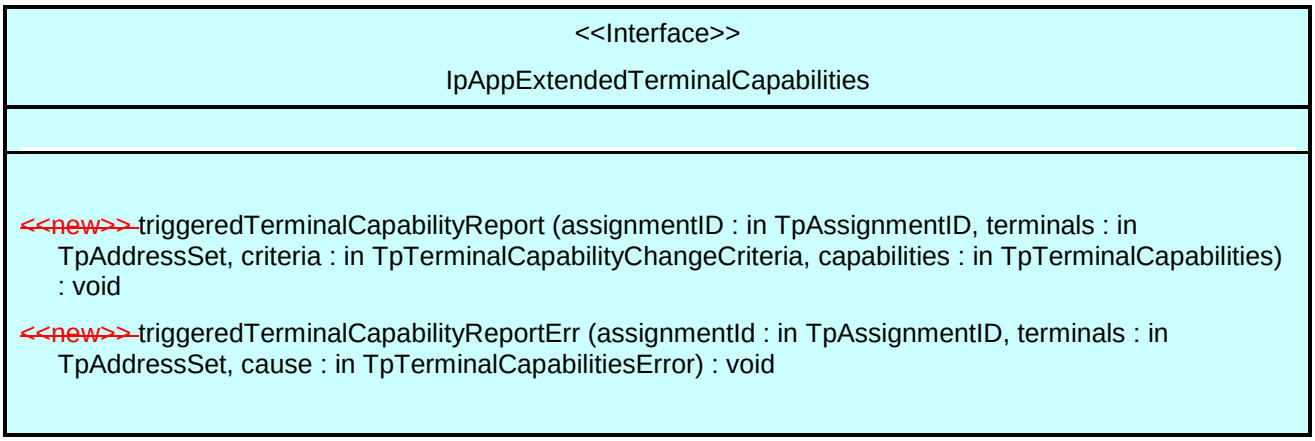
**End of change in Clause 8.2**

**Change in Clause 8.3**

## 8.3 Interface Class IpAppExtendedTerminalCapabilities

Inherits from: IpInterface.

IpAppExtendedTerminalCapabilities interface is used to send triggered terminal capability reports. It is implemented by the client application developer.




### 8.3.1 Method <<new>> triggeredTerminalCapabilityReport()

This terminal capability report is issued when the capabilities of the terminal have changed in the way specified by the criteria parameter in the previously invoked triggeredTerminalCapabilityStartReq () method.

*Parameters*

**`assignmentID : in TpAssignmentID`**

Specifies the assignment ID of the report.

**`terminals : in TpAddressSet`**

Specifies the terminal(s) either by subscriber or terminal ID or both as described for the triggeredTerminalCapabilityStartReq () method.

**`criteria : in TpTerminalCapabilityChangeCriteria`**

Specifies the criteria that caused the report to be sent.

**`capabilities : in TpTerminalCapabilities`**

Specifies the capabilities of the terminal. The network may override some capabilities that have been indicated by the terminal itself due to network policies or other restrictions or modifications in the supported capabilities.

### 8.3.2 Method <<new>> triggeredTerminalCapabilityReportErr()

This method indicates that the requested reporting has failed. Note that errors may concern the whole assignment or just some terminals. In the former case no terminals are specified.

*Parameters*

**`assignmentId : in TpAssignmentID`**

Specifies the assignment ID.

**`terminals : in TpAddressSet`**

Specifies the terminal(s) either by subscriber or terminal ID or both as described for the triggeredTerminalCapabilityStartReq () method.

**`cause : in TpTerminalCapabilitiesError`**

Specifies the error that led to the failure.

**End of change in Clause 8.3**

# Annex E (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 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010470 | 002 | -- | Changing references to JAIN | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010600 | 003 | -- | Replace Out Parameters with Return Types | 4.2.0 | 4.3.0 |
| Mar 2002 | CN_15 | NP-020109 | 004 | -- | Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID() | 4.3.0 | 4.4.0 |
| Mar 2002 | CN_15 | NP-020113 | 005 | -- | Addition of terminal capability change notifications | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020182 | 006 | -- | Addition of support for WSDL realisation | 5.0.0 | 5.1.0 |
| Sep 2002 | CN-17 | NP-020434 | 007 | -- | Add text to clarify requirements on support of methods | 5.1.0 | 5.2.0 |
| Sep 2002 | CN-17 | NP-020395 | 008 | -- | Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030023 | 011 | -- | Addition of status of methods to Terminal Capabilities interfaces | 5.2.0 | 5.3.0 |
| Mar 2003 | CN_19 | NP-030023 | 013 | -- | Correction to TpTerminalCapabilities in Terminal Capabilities | 5.2.0 | 5.3.0 |
| Sep 2003 | CN_21 | NP-030352 | 014 | -- | Correction to Java Realisation Annex | 5.3.0 | 5.4.0 |
| Dec 2003 | CN_22 | NP-030553 | 015 | -- | Add OSA API support for 3GPP2 networks | 5.4.0 | 6.0.0 |
| Feb 2004 | -- | -- | -- | -- | Added Java code attachment 2919807J2EE.zip which was delivered late by outside developers. See Annex C. | 6.0.0 | 6.0.1 |
| | | | | | | | |

CR-Form-v7

# CHANGE REQUEST

⌘ **29.198-08** **CR** **032** ⌘ **rev** **-** ⌘ Current version: **6.0.1** ⌘

*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:*** | ⌘ | Remove the <<new>> stereotype from methods which are no longer new |
| ***Source:*** | ⌘ | CN5 Ultan Mulligan, ETSI PTCC |
| ***Work item code:*** | ⌘ | OSA3 |
| ***Date:*** | ⌘ | 14/05/2004 |
| ***Category:*** | ⌘ | **F** |
| ***Release:*** | ⌘ | *REL-6* |

*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 <<new>> stereotype is used to indicate a new method has been added to the specification. However, some methods have been newly introduced more than a year ago, and in Rel-5, yet they still carry the <<new>> stereotype in Rel-6. For these methods, the <<new>> stereotype can be removed from their title and description.<br>Although editorial, this change is being applied throughout the 29.198 Rel-6 specification set, and so, on MCC advice, CRs have been generated to make this process visible. |
| ***Summary of change:*** | ⌘ | Remove the <<new>> stereotype from the documentation of methods which were newly introduced prior to the creation of the Rel-6 specifications, i.e. from methods in the Rel-6 specifications which were newly introduced into Rel-5 at or before the March 2003 plenary. |
| ***Consequences if not approved:*** | ⌘ | The specifications will contain misleading information, indicating certain methods as being newly introduced, when in fact they are not |

| | | | | | |
|---|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 8.4 | | | |
| | | **Y** | **N** | | |
| ***Other specs affected:*** | ⌘ | **X** | | Other core specifications | TS 29.198-1, 3, 4-3, 5, 7, 11 and 12 Release 6 |
| | | | **X** | Test specifications | |
| | | | **X** | O&M Specifications | |
| ***Other comments:*** | ⌘ | N5-040284 to N5-040290 should be bundled together | | | |

**Change in Clause 8.4**

# 8.4 Interface Class IpDataSessionControlManager

Inherits from: IpService.

This interface is the 'SCF manager' interface for Data Session Control. This interface shall be implemented by a Data Session Control SCF. As a minimum requirement, the createNotifications() and destroyNotification(), or the enableNotifications() and disableNotifications() methods shall be implemented.

| <<Interface>><br>IpDataSessionControlManager |
|---|
| |
| <<deprecated>> createNotification (appDataSessionControlManager : in IpAppDataSessionControlManagerRef, eventCriteria : in TpDataSessionEventCriteria) : TpAssignmentID<br>destroyNotification (assignmentID : in TpAssignmentID) : void<br>changeNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpDataSessionEventCriteria) : void<br><<deprecated>> getNotification () : TpDataSessionEventCriteria<br>~~<<new>>~~ enableNotifications (appDataSessionControlManager : in IpAppDataSessionControlManagerRef) : TpAssignmentID<br>~~<<new>>~~ disableNotifications () : void<br>~~<<new>>~~ getNotifications () : TpDataSessionEventCriteriaResultSet<br>~~<<new>>~~ createNotifications (appDataSessionControlManager : in IpAppDataSessionControlManagerRef, eventCriteria : in TpDataSessionEventCriteria) : TpAssignmentID |

## 8.4.1 Method <<deprecated>> createNotification()

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

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

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

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

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

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used.

In case the createNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the callback that has been registered by setCallback().

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

*Parameters*

**appDataSessionControlManager : in IpAppDataSessionControlManagerRef**

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

**eventCriteria : in TpDataSessionEventCriteria**

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

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE**

## 8.4.2 Method destroyNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

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

*Raises*

**TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID**

## 8.4.3 Method changeNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

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

**eventCriteria : in TpDataSessionEventCriteria**

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

*Raises*

**TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE**

## 8.4.4 Method <<deprecated>> getNotification()

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

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

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

*Parameters*

No Parameters were identified for this method

*Returns*

**TpDataSessionEventCriteria**

*Raises*

**TpCommonExceptions, P_INVALID_NETWORK_STATE**

## 8.4.5 Method ~~<<new>>~~ enableNotifications()

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

If the same application requests to enable notifications for a second time with a different IpAppDataSessionControlManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used.

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

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

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

*Parameters*

**appDataSessionControlManager : in IpAppDataSessionControlManagerRef**

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

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions**

### 8.4.6 Method ~~<<new>>~~ disableNotifications()

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

*Parameters*

No Parameters were identified for this method

*Raises*

**TpCommonExceptions**

### 8.4.7 Method ~~<<new>>~~ getNotifications()

This method replaces getNotification().

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

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

*Parameters*

No Parameters were identified for this method

*Returns*

**TpDataSessionEventCriteriaResultSet**

*Raises*

**TpCommonExceptions, P_INVALID_NETWORK_STATE**

### 8.4.8 Method ~~<<new>>~~ createNotifications()

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

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

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

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

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

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

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

*Parameters*

**appDataSessionControlManager : in IpAppDataSessionControlManagerRef**

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

**eventCriteria : in TpDataSessionEventCriteria**

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

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE, P_INVALID_INTERFACE_TYPE**

**End of change in Clause 8.4**

# Annex E (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 | 1.0.0 |
| Jun 2001 | CN_12 | NP-010330 | 001 | -- | Corrections to OSA API Rel4 | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010471 | 002 | -- | Changing references to JAIN | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010601 | 003 | -- | Replace Out Parameters with Return Types | 4.2.0 | 4.3.0 |
| Dec 2001 | CN_14 | NP-010601 | 004 | -- | Corrections and alignment additions to the Data Session Control SCF | 4.2.0 | 4.3.0 |
| Mar 2002 | CN_15 | NP-020110 | 005 | -- | Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID() | 4.3.0 | 4.4.0 |
| Jun 2002 | CN_16 | NP-020182 | 006 | -- | Addition of support for WSDL realisation | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020183 | 007 | -- | Addition of Support for Network Controlled Notifications DSC | 4.4.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020192 | 008 | -- | Adding missing text concerning the activity timer and criteria overlap | 4.4.0 | 5.0.0 |
| Sep 2002 | CN_17 | NP-020435 | 011 | | Remove duplicate exception from IpDataSessionControlManager.createNotification() | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020435 | 012 | | Remove P_SERVICE_INFORMATION_MISSING and P_SERVICE_FAULT_ENCOUNTERED exceptions from_DataSessionControl methods. | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020435 | 013 | | Introduce new method getNotifications to correct the result type of IpDataSessionControlManager.getNotification() to permit retreival of all created notifications. | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020435 | 014 | | Add P_INVALID_INTERFACE_TYPE exception to IpDataSessionControlManager.createNotification(), resulting in new createNotifications() method | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020435 | 015 | | Add text to clarify requirements on support of methods | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020435 | 016 | | Correction on use of NULL in Data Session Control API | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020395 | 017 | | Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications | 5.0.0 | 5.1.0 |
| Mar 2003 | CN_19 | NP-030024 | 019 | -- | Addition of status of methods to Data Session Control interfaces | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030024 | 021 | -- | Corrections to data types in Data Session Control | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030034 | 022 | -- | Inconsistent description of use of secondary callback | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030034 | 023 | -- | Promotion of TpDataSessionQosClass data type definition to the Common Data Types | 5.1.0 | 5.2.0 |
| Jun 2003 | CN_20 | NP-030238 | 025 | -- | Correction of the description for callEventNotify & reportNotification | 5.2.0 | 5.3.0 |
| Sep 2003 | CN_21 | NP-030352 | 026 | -- | Correction to Java Realisation Annex | 5.3.0 | 5.4.0 |
| Dec 2003 | CN_22 | NP-030553 | 027 | -- | Add OSA API support for 3GPP2 networks | 5.4.0 | 6.0.0 |
| Feb 2004 | -- | -- | -- | -- | Added Java code attachment 2919808J2EE.zip which was delivered late by outside developers. See Annex C. | 6.0.0 | 6.0.1 |
| | | | | | | | |

CR-Form-v7

# CHANGE REQUEST

⌘ 29.198-11 CR 028 ⌘rev - ⌘ Current version: 6.0.1 ⌘

*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:*** | ⌘ | Remove the <<new>> stereotype from methods which are no longer new |
| ***Source:*** | ⌘ | CN5 Ultan Mulligan, ETSI PTCC |
| ***Work item code:*** | ⌘ | OSA3 |
| ***Date:*** | ⌘ | 14/05/2004 |
| ***Category:*** | ⌘ | F |
| ***Release:*** | ⌘ | REL-6 |

*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 <<new>> stereotype is used to indicate a new method has been added to the specification. However, some methods have been newly introduced more than a year ago, and in Rel-5, yet they still carry the <<new>> stereotype in Rel-6. For these methods, the <<new>> stereotype can be removed from their title and description.<br>Although editorial, this change is being applied throughout the 29.198 Rel-6 specification set, and so, on MCC advice, CRs have been generated to make this process visible. |
| ***Summary of change:*** | ⌘ | Remove the <<new>> stereotype from the documentation of methods which were newly introduced prior to the creation of the Rel-6 specifications, i.e. from methods in the Rel-6 specifications which were newly introduced into Rel-5 at or before the March 2003 plenary. |
| ***Consequences if not approved:*** | ⌘ | The specifications will contain misleading information, indicating certain methods as being newly introduced, when in fact they are not |

| | | | | | |
|---|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 8.1 | | | |
| | | **Y** | **N** | | |
| ***Other specs affected:*** | ⌘ | X | | Other core specifications | ⌘ TS 29.198-1, 3, 4-3, 5, 7, 8 and 12 Release 6 |
| | | | X | Test specifications | |
| | | | X | O&M Specifications | |
| ***Other comments:*** | ⌘ | N5-040284 to N5-040290 should be bundled together | | | |

**Change in Clause 8.1**

# 8.1 Interface Class IpAccountManager

Inherits from: IpService.

The account manager interface provides methods for monitoring accounts. Applications can use this interface to enable or disable charging-related event notifications and to query account balances.

This interface shall be implemented by an Account Management SCF. The queryBalanceReq() method, or the retrieveTransactionHistoryReq() method, or both the createNotification() and destroyNotification methods, or both the enableNotifications and disableNotifications methods shall be implemented as a minimum requirement.

<<Interface>>

IpAccountManager

createNotification (appAccountManager : in IpAppAccountManagerRef, chargingEventCriteria : in TpChargingEventCriteria) : TpAssignmentID

destroyNotification (assignmentId : in TpAssignmentID) : void

queryBalanceReq (users : in TpAddressSet) : TpAssignmentID

changeNotification (assignmentID : in TpAssignmentID, eventCriteria : in TpChargingEventCriteria) : void

getNotification () : TpChargingEventCriteriaResultSet

retrieveTransactionHistoryReq (user : in TpAddress, transactionInterval : in TpTimeInterval) : TpAssignmentID

~~<<new>>~~ enableNotifications (appAccountManager : in IpAppAccountManagerRef) : TpAssignmentID

~~<<new>>~~ disableNotifications () : void

## 8.1.1 Method createNotification()

This method is used by the application to enable charging event notifications to be sent to the application.

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

Returns assignmentId : Specifies the ID assigned by the account management object for this newly enabled event notification.

*Parameters*

**`appAccountManager : in IpAppAccountManagerRef`**

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

**`chargingEventCriteria : in TpChargingEventCriteria`**

Specifies the event specific criteria used by the application to define the charging event required. Individual addresses or address ranges may be specified for subscriber accounts. Example of events are "charging" and "recharging".

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_INVALID_ADDRESS, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE, P_UNKNOWN_SUBSCRIBER**

### 8.1.2 Method destroyNotification()

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

*Parameters*

**assignmentId : in TpAssignmentID**

Specifies the assignment ID that was given by the account management object when the application enabled the charging notification.

*Raises*

**TpCommonExceptions, P_INVALID_ASSIGNMENT_ID**

### 8.1.3 Method queryBalanceReq()

This method is used by the application to query the balance of an account for one or several users.

Returns queryId : Specifies the ID of the balance query request.

*Parameters*

**users : in TpAddressSet**

Specifies the user(s) for which the balance is queried.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION**

### 8.1.4 Method changeNotification()

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

*Parameters*

**assignmentID : in TpAssignmentID**

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

**eventCriteria : in TpChargingEventCriteria**

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

*Raises*

**TpCommonExceptions, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE, P_UNKNOWN_SUBSCRIBER, P_INVALID_ADDRESS**

### 8.1.5 Method getNotification()

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

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

*Parameters*

No Parameters were identified for this method

*Returns*

**TpChargingEventCriteriaResultSet**

*Raises*

**TpCommonExceptions**

### 8.1.6 Method retrieveTransactionHistoryReq()

This asynchronous method is used by the application to retrieve a transaction history of a subscriber's account. The history is a set of Detailed Records.

Returns retrievalID : Specifies the retrieval ID of the transaction history retrieval request.

*Parameters*

**user : in TpAddress**

Specifies the subscriber for whose account the transaction history is to be retrieved.

**transactionInterval : in TpTimeInterval**

Specifies the time interval for which the application history is to be retrieved.

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions, P_UNKNOWN_SUBSCRIBER, P_UNAUTHORIZED_APPLICATION, P_INVALID_TIME_AND_DATE_FORMAT**

### 8.1.7 Method ~~<<new>>~~ enableNotifications()

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

If the same application requests to enable notifications for a second time with a different IpAppAccountManager reference (i.e. without first disabling them), the second callback will be treated as an additional callback. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used.

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

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

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

*Parameters*

**appAccountManager : in IpAppAccountManagerRef**

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

*Returns*

**TpAssignmentID**

*Raises*

**TpCommonExceptions**

### 8.1.8 Method ~~<<new>>~~ disableNotifications()

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

*Parameters*

No Parameters were identified for this method

*Raises*

**TpCommonExceptions**

**End of change in Clause 8.1**

# Annex E (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 | 1.0.0 |
| Jun 2001 | CN_12 | NP-010327 | -- | -- | Approved at TSG CN#12 and placed under Change Control | 2.0.0 | 4.0.0 |
| Sep 2001 | CN_13 | NP-010472 | 001 | -- | Changing references to JAIN | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010472 | 002 | -- | Missing exceptions for enabling and changing the notifications | 4.0.0 | 4.1.0 |
| Dec 2001 | CN_14 | NP-010602 | 003 | -- | Replace Out Parameters with Return Types | 4.1.0 | 4.2.0 |
| Dec 2001 | CN_14 | NP-010602 | 004 | -- | Replace erroneous use of incorrect data type TpSessionID by TpAssignmentID in Account Management interface | 4.1.0 | 4.2.0 |
| Mar 2002 | CN_15 | NP-020111 | 005 | -- | Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID() | 4.2.0 | 4.3.0 |
| Mar 2002 | CN_15 | NP-020111 | 006 | -- | Correction of parameter name in IpAccountManager.createNotification() | 4.2.0 | 4.3.0 |
| Mar 2002 | CN_15 | NP-020111 | 007 | -- | Correction of result parameter of getNotification, set in stead of single result | 4.2.0 | 4.3.0 |
| Jun 2002 | CN_16 | NP-020193 | 008 | -- | Change to new Service Property P_MAX_ADDRESSES_PER_QUERY for Account Management | 4.3.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020182 | 009 | -- | Addition of support for WSDL realisation | 4.3.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020183 | 010 | -- | Addition of Support for Network Controlled Notifications AM | 4.3.0 | 5.0.0 |
| Sep 2002 | CN_17 | NP-020436 | 011 | -- | Correction of IpAccountManager STD to permit multiple notifications | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020436 | 012 | -- | Add text to clarify requirements on support of methods | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020436 | 013 | -- | Add missing callback interface for notifications in Account Management | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020395 | 014 | -- | Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications | 5.0.0 | 5.1.0 |
| Mar 2003 | CN_19 | NP-030025 | 016 | -- | Correction to TpChargingEventCriteria in Account Management | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030025 | 018 | -- | Addition of status of methods to Account Management interfaces | 5.1.0 | 5.2.0 |
| Mar 2003 | CN_19 | NP-030035 | 019 | -- | Inconsistent description of use of secondary callback | 5.1.0 | 5.2.0 |
| Sep 2003 | CN_21 | NP-030352 | 020 | -- | Correction to Java Realisation Annex | 5.2.0 | 5.3.0 |
| Dec 2003 | CN_22 | NP-030556 | 021 | -- | Add methods for balanceUpdate(), voucherUpdate() and getCreditExpiryDate() to OSA Account Management | 5.3.0 | 6.0.0 |
| Dec 2003 | CN_22 | NP-030553 | 022 | -- | Add OSA API support for 3GPP2 networks | 5.3.0 | 6.0.0 |
| Feb 2004 | -- | -- | -- | -- | Added Java code attachment 2919811J2EE.zip which was delivered late by outside developers. See Annex C. | 6.0.0 | 6.0.1 |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

CR-Form-v7

# CHANGE REQUEST

⌘ 29.198-12 CR 027 ⌘rev - ⌘ Current version: 6.0.1 ⌘

*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:*** | ⌘ | Remove the <<new>> stereotype from methods which are no longer new |
| ***Source:*** | ⌘ | CN5 Ultan Mulligan, ETSI PTCC |
| ***Work item code:*** | ⌘ | OSA3 |
| ***Date:*** | ⌘ | 14/05/2004 |
| ***Category:*** | ⌘ | F |
| ***Release:*** | ⌘ | REL-6 |

*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 <<new>> stereotype is used to indicate a new method has been added to the specification. However, some methods have been newly introduced more than a year ago, and in Rel-5, yet they still carry the <<new>> stereotype in Rel-6. For these methods, the <<new>> stereotype can be removed from their title and description.<br>Although editorial, this change is being applied throughout the 29.198 Rel-6 specification set, and so, on MCC advice, CRs have been generated to make this process visible. |
| ***Summary of change:*** | ⌘ | Remove the <<new>> stereotype from the documentation of methods which were newly introduced prior to the creation of the Rel-6 specifications, i.e. from methods in the Rel-6 specifications which were newly introduced into Rel-5 at or before the March 2003 plenary. |
| ***Consequences if not approved:*** | ⌘ | The specifications will contain misleading information, indicating certain methods as being newly introduced, when in fact they are not |

| | | | | | |
|---|---|---|---|---|---|
| ***Clauses affected:*** | ⌘ | 8.1 | | | |
| | | **Y** | **N** | | |
| ***Other specs affected:*** | ⌘ | **X** | | Other core specifications | ⌘ TS 29.198-3, 4-3, 5, 7, 8 and 11 Release 6 |
| | | | **X** | Test specifications | |
| | | | **X** | O&M Specifications | |
| ***Other comments:*** | ⌘ | N5-040284 to N5-040290 should be bundled together | | | |

**Change in Clause 8.1**

# 8.1 Interface Class IpChargingManager

Inherits from: IpService.

This interface is the 'service manager' interface for the Charging Service. The Charging manager interface provides management functions to the charging service. The application programmer can use this interface to start charging sessions.

This interface shall be implemented by a Charging SCF. As a minimum requirement, at least one of createChargingSession() or createSplitChargingSession() shall be implemented.

<<Interface>>

IpChargingManager

createChargingSession (appChargingSession : in IpAppChargingSessionRef, sessionDescription : in TpString, merchantAccount : in TpMerchantAccountID, user : in TpAddress, correlationID : in TpCorrelationID) : TpChargingSessionID

~~<<new>>~~ createSplitChargingSession (appChargingSession : in IpAppChargingSessionRef, sessionDescription : in TpString, merchantAccount : in TpMerchantAccountID, users : in TpAddressSet, correlationID : in TpCorrelationID) : TpChargingSessionID

## 8.1.1 Method createChargingSession()

This method creates an instance of the IpChargingSession interface to handle the charging events related to the specified user and to the application invoking this method. An IpAppChargingManager should already have been passed to the IpChargingManager, otherwise the charging manager will not be able to report a sessionAborted() to the application (the application should invoke setCallback() if it wishes to ensure this).

Returns chargingSession: Defines the session.

*Parameters*

**appChargingSession : in IpAppChargingSessionRef**

Callback interface for the session in the application.

**sessionDescription : in TpString**

Descriptive text for informational purposes.

**merchantAccount : in TpMerchantAccountID**

Identifies the account of the party providing the application to be used.

**user : in TpAddress**

Specifies the user that is using the application. This may or may not be the user that will be charged. The Charging service will determine the charged user. When this method is invoked the Charging service shall determine if charging is allowed for this application for this subscriber. An exception shall be thrown if this type of charging is not allowed.

**correlationID : in TpCorrelationID**

This value can be used to correlate the charging to network activity.

*Returns*

**TpChargingSessionID**

*Raises*

**TpCommonExceptions, P_INVALID_USER, P_INVALID_ACCOUNT**

### 8.1.2 Method ~~<<new>>~~ createSplitChargingSession()

This method creates an instance of the IpChargingSession interface to handle the charging events related to the specified users and to the application invoking this method. This method differs from createChargingSession() in that it allows to specify multiple users to be charged. The SCS implementation is responsible to figure out how later reserve and charge operations are split among these subscribers. The algorithm may be selected and controlled e.g. through the chargingParameter argument in the respective methods. The algorithms provided and the details how they interpret any parameters are vendor specific.

Returns chargingSession: Defines the session.

*Parameters*

**appChargingSession : in IpAppChargingSessionRef**

Callback interface for the session in the application.

**sessionDescription : in TpString**

Descriptive text for informational purposes.

**merchantAccount : in TpMerchantAccountID**

Identifies the account of the party providing the application to be used.

**users : in TpAddressSet**

Specifies the users that are involved in using the application. This could be all users in a multi-party application (conference call, multi-user-game).

**correlationID : in TpCorrelationID**

This value can be used to correlate the charging to network activity.

*Returns*

**TpChargingSessionID**

*Raises*

**TpCommonExceptions, P_INVALID_USER, P_INVALID_ACCOUNT**

**End of change in Clause 8.1**

# Annex E (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 | 1.0.0 |
| Jun 2001 | CN_12 | NP-010328 | -- | -- | | 2.0.0 | 4.0.0 |
| Sep 2001 | CN_13 | NP-010473 | 001 | -- | Changing references to JAIN | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010473 | 002 | -- | Error corrections charging | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010473 | 003 | -- | Changed semantics of closeReservation parameter | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010473 | 004 | -- | Missing errors in definition of (credit/debit)(Amoun/Unit)Err | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010473 | 005 | -- | Clarification of Unit Reservation | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010473 | 006 | -- | Improving correlation request and response for applications | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010473 | 007 | -- | Remove the P_CHS_PARAM_RESULT value from the TpChargingParameterID type | 4.0.0 | 4.1.0 |
| Sep 2001 | CN_13 | NP-010473 | 008 | -- | Align the order of parameters for similar methods | 4.0.0 | 4.1.0 |
| Dec 2001 | CN_14 | NP-010603 | 009 | -- | Replace Out Parameters with Return Types | 4.1.0 | 4.2.0 |
| Mar 2002 | CN_15 | NP-020112 | 010 | -- | Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID() | 4.2.0 | 4.3.0 |
| Mar 2002 | CN_15 | NP-020112 | 011 | -- | Correction of parameter name in IpAppChargingSession.extendLifeTimeRes() | 4.2.0 | 4.3.0 |
| Jun 2002 | CN_16 | NP-020194 | 012 | -- | Clarify the use of setCallback with charging | 4.3.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020194 | 013 | -- | Adding Service Properties for the Content Based Charging API | 4.3.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020194 | 014 | -- | Addition of support for interactive authorization of payments ("User Confirmation") | 4.3.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020194 | 015 | -- | Addition of support for Split Charging feature | 4.3.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020181 | 016 | -- | Addition of support for Java API technology realisation | 4.3.0 | 5.0.0 |
| Jun 2002 | CN_16 | NP-020182 | 017 | -- | Addition of support for WSDL realisation | 4.3.0 | 5.0.0 |
| Sep 2002 | CN_17 | NP-020437 | 018 | | Add text to clarify requirements on support of methods | 5.0.0 | 5.1.0 |
| Sep 2002 | CN_17 | NP-020395 | 019 | | Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications | 5.0.0 | 5.1.0 |
| Mar 2003 | CN_19 | NP-030026 | 021 | -- | Addition of status of methods to Charging interfaces | 5.1.0 | 5.2.0 |
| Sep 2003 | CN_21 | NP-030352 | 022 | -- | Correction to Java Realisation Annex | 5.2.0 | 5.3.0 |
| Dec 2003 | CN_22 | NP-030546 | 024 | - | Correcting charging State Transition when reservation closed | 5.3.0 | 5.4.0 |
| Dec 2003 | CN_22 | NP-030553 | 025 | - | Add OSA API support for 3GPP2 networks | 5.4.0 | 6.0.0 |
| Feb 2004 | -- | -- | -- | -- | Added Java code attachment 2919812J2EE.zip; which was delivered late by outside developers. See Annex C. | 6.0.0 | 6.0.1 |
| | | | | | | | |