3GPP TSG CN Plenary Meeting #22 **NP-030554**
10 - 12 December 2003, Maui, Hawaii, USA

Source:	**CN5 (OSA)**
Title:	**Rel-6 CRs 29.198-03/05 OSA API Part 3/5: Framework / Generic user interaction**
Agenda item:	**9.7**
Document for:	**APPROVAL**

Doc-1st-Level	Spec	CR	R	Ph	Subject	Cat	Version-Current	Doc-2nd-Lev	WI
NP-030554	29.198-03	092	-	Rel-6	Add description for service super and sub types	B	5.4.0	N5-030389	OSA3
NP-030554	29.198-03	093	-	Rel-6	Add support for registration of additional service property types and modes	B	5.4.0	N5-030390	OSA3
NP-030554	29.198-03	094	-	Rel-6	Improve User Interaction message management functions	B	5.4.0	N5-030410	OSA3
NP-030554	29.198-05	043	-	Rel-6	Improve User Interaction message management functions	B	5.4.0	N5-030409	OSA3
NP-030554	29.198-03	095	-	Rel-6	Add new values for TpServiceTypeName for Policy Management	B	5.4.0	N5-030430	OSA3
NP-030554	29.198-03	096	-	Rel-6	Allow for applications to re-obtain the reference to the service manager	B	5.4.0	N5-030431	OSA3
NP-030554	29.198-03	097	-	Rel-6	Add support in OSA to inform applications about new SCSs and their level of Backward compatibility – Align with SA1's 22.127	B	5.4.0	N5-030322	OSA3
NP-030554	29.198-03	098	-	Rel-6	Add “Extended User Status” as service type name - Align with 29.198-06	B	5.4.0	N5-030433	OSA3
NP-030554	29.198-03	099	-	Rel-6	Add P_USER_BINDING to TpServiceTypeName	B	5.4.0	N5-030579	OSA3
NP-030554	29.198-03	100	-	Rel-6	Modify Framework Availability Indication in Fault Management	C	5.4.0	N5-030631	OSA3

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-030389**
Meeting #24, San Francisco, CA, USA, 14 - 18 July 2003

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03 CR 092** ⌘ **rev** **-** ⌘ Current version: **5.4.0** ⌘

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

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

Title:	⌘	Add description for service super and sub types		
Source:	⌘	CN5 (Koen.Schilders@etm.ericsson.se)		
Work item code:	⌘	OSA3	***Date:*** ⌘	18/07/2003
Category:	⌘	**B**	***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 Parlay/OSA framework specification includes the terms super and sub types, but does not specify what is meant with these and how service super and sub types are supposed to work.
Summary of change:	⌘	An explanation is added for service super and sub types. Furthermore, the description of listServiceTypes() is updated to include that this method will return both the super and sub type of a service when only the sub type is registered. The description of serviceTypeName parameter in the discoverService() method is changed to include that an application may both request a service super and sub type.
Consequences if not approved:	⌘	Interoperability problems caused by confusion over what super and sub types are.

Clauses affected:	⌘	7.3.1.1.1, 7.3.1.1.3, 9.1, 9.2				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

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

Introduction

Specification 29.198-03 describes the mechanism of service properties. All service capability servers contain a set of common properties and a set of SCS-specific properties. A service capability server can supply more service properties than defined in the service type. However, the Framework does not know the type of these service properties, and can therefore not do anything with them.

The framework specification already specifies the use of service super and sub types but does not clarify how this mechanism is supposed to work.

Proposed Changes

7.3.1.1.1 Method listServiceTypes()

This operation returns the names of all service super and sub types that are in the repository. The details of the service types can then be obtained using the describeServiceType() method. If a sub type of a service is registered, this method returns, besides the sub type, also the super tyoe.

Returns <listTypes> : The names of the requested service types.

Parameters
No Parameters were identified for this method

Returns

`TpServiceTypeNameList`

Raises

`TpCommonExceptions,P_ACCESS_DENIED`

7.3.1.1.3 Method discoverService()

The discoverService operation is the means by which a client application is able to obtain the service IDs of the services that meet its requirements. The client application passes in a list of desired service properties to describe the service it is looking for, in the form of attribute/value pairs for the service properties. The client application also specifies the maximum number of matched responses it is willing to accept. The framework must not return more matches than the specified maximum, but it is up to the discretion of the Framework implementation to choose to return less than the specified maximum. The discoverService() operation returns a serviceID/Property pair list for those services that match the desired service property list that the client application provided. The service properties returned ~~will~~ form a complete view of what the client application ~~will be able to~~can do with the service, as per the service level agreement. If the framework supports service subscription, the service level agreement will be encapsulated in the subscription properties contained in the contract/profile for the client application, which will be a restriction of the registered properties.

Returns <serviceList> : This parameter gives a list of matching services. Each service is characterised by its service ID and a list of service properties {name and value list} associated with the service.

Parameters

`serviceTypeName : in TpServiceTypeName`

The "serviceTypeName" parameter conveys the required service type. It is key to the central purpose of "service trading". It is the basis for type safe interactions between the service exporters (via registerService) and service importers (via discoverService). By stating a service type, the importer implies the service type and a domain of discourse for talking about properties of service.

· If the string representation of the "type" does not obey the rules for service type identifiers, then the P_ILLEGAL_SERVICE_TYPE exception is raised.

· If the "type" is correct syntactically but is not recognised as a service type within the Framework, then the P_UNKNOWN_SERVICE_TYPE exception is raised.

The framework may return a service of a subtype of the "type" requested. The requestor may also request for a service of a specific subtype. The framework will **not** return the corresponding supertype(s) in this case. ~~A service sub-type can be described by the properties of its supertypes.~~

desiredPropertyList : in TpServicePropertyList

The "desiredPropertyList" parameter is a list of service property {name, mode and value list} tuples that the discovered set of services should satisfy. These properties deal with the non-functional and non-computational aspects of the desired service. The property values in the desired property list must be logically interpreted as "minimum", "maximum", etc. by the framework (due to the absence of a Boolean constraint expression for the specification of the service criterion). It is suggested that, at the time of service registration, each property value be specified as an appropriate range of values, so that desired property values can specify an "enclosing" range of values to help in the selection of desired services.

The desiredPropertyList only contains service properties that are relevant for the application. If an application is not interested in the value of a certain service property, this service property shall not be included in the desiredPropertyList.

P_INVALID_PROPERTY is raised when an application includes an unknown service property name or invalid service property value.

max : in TpInt32

The "max" parameter states the maximum number of services that are to be returned in the "serviceList" result.

Returns

TpServiceList

Raises

TpCommonExceptions,P_ACCESS_DENIED,P_ILLEGAL_SERVICE_TYPE,P_UNKNOWN_SERVICE_TYPE,P_INVALID_PROPERTY

9.1 Service Super and Sub Types

Service Properties are used at service registration to indicate the capabilities of an SCF. They are normally used as an indication for limitations an SCF has. These limitations can come from the way an SCF is implemented or from limitations in the network. The service type of an SCF defines which properties the supplier shall provide at registration of the SCF.

An application uses Service Properties at service discovery to find services that have the required capabilities. The Framework validates the requested properties with the registered properties and provides the application with a list of SCFs that comply to the application's request.

The capabilities of an SCF can be extended by providing service properties in addition to the ones defined in this standard. For this extended SCF, a dedicated sub-type of a service is defined. A sub-type of an SCF shall be fully compatible with the standard SCF, that is, an application shall be able to use the sub type as if it was the standard type. This implies that the interface to the SCF remains unchanged. Also SCF sub types can be further extended. This way a hierarchy of service types can be built with the standard type being the root.

An example of a sub type is a Multy Party Call Control service that allows the application to request a certain quality-of-service level. An additional service property is added for this.

~~9.1~~9.2 Service Property Types

~~The service type defines which properties the supplier of an SCF supplier shall provide when he registers an SCF.~~

At Service Registration the properties of a type shall be interpreted as the set of values that can be supported by the service. If a service type has a certain property (e.g. "`CAN_DO_SOMETHING`"), a service registers with a property value of `{"true", "false"}`. This means that the SCS is able to support Service instances where this property is used or allowed and instances where this property is not used or allowed. This clarifies why sets of values shall be used for the property values instead of primitive types.

At establishment of the Service Level Agreement the property can then be set to the value of the specific agreement. The context of the Service Level Agreement thus restricts the set of property values of the SCS and will thus lead to a sub-set of the service property values. When the correct SCF is instantiated during the discovery and selection procedure (see Note), the Service Properties shall thus be interpreted as the requested property values.

NOTE: This is achieved through the createServiceManager() operation in the Service Instance Lifecycle Manager interface.

All property values are represented by an array of strings. The following table shows all supported service property types.

Service Property type name	Description	Example value (array of strings)	Interpretation of example value
BOOLEAN_SET	set of Booleans	{"FALSE"}	The set of Booleans consisting of the Boolean "false".
INTEGER_SET	set of integers	{"1", "2", "5", "7"}	The set of integers consisting of the integers 1, 2, 5 and 7.
STRING_SET	set of strings	{"Sophia", "Rijen"}	The set of strings consisting of the string “Sophia" and the string "Rijen"
ADDRESSRANGE_SET	set of address ranges	{"123??*", "*.ericsson.se"}	The set of address ranges consisting of ranges 123??* and *.ericsson.se.
INTEGER_INTERVAL	interval of integers	{"5", "100"}	The integers that are between or equal to 5 and 100.
STRING_INTERVAL	interval of strings	{"Rijen", "Sophia"}	The strings that are between or equal to the strings "Rijen" and "Sophia", in lexicographical order.
INTEGER_INTEGER_MAP	map from integers to integers	{"1", "10", "2", "20", "3", "30"}	The map that maps 1 to 10, 2 to 20 and 3 to 30.

The bounds of the string interval and the integer interval types may hold the reserved value "UNBOUNDED". If the left bound of the interval holds the value "UNBOUNDED", the lower bound of the interval is the smallest value supported by the type. If the right bound of the interval holds the value "UNBOUNDED", the upper bound of the interval is the largest value supported by the type.

When an SCF is registerd by the Service Supplier, Service Properties of type BOOLEAN_SET shall not contain an empty set. When a service is discovered by an application, this application shall specify either {TRUE} or {FALSE} as value for service properties of type BOOLEAN_SET.

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-030390**
Meeting #24, San Francisco, CA, USA, 14 - 18 July 2003

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-03**	**CR**	**093**	⌘ **rev**	**-**	⌘ Current version:	**5.4.0** ⌘

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

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

Title:	⌘	Add support for registration of additional service property types and modes
Source:	⌘	CN5 (Koen.Schilders@etm.ericsson.se)
Work item code:	⌘	OSA3
Date:	⌘	18/07/2003
Category:	⌘	**B**
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 Parlay/OSA specification lists the use of service super and sub types. For the super types, the modes and types of the service properties are included. For the sub types this is however not the case. The specification lacks a possibility for a service sub type to indicate the types and modes of its service properties.
Summary of change:	⌘	A new method, registerServiceSubType(), is added to IpServiceDiscovery that supports registration of service sub types. The existing method registerService() is used for registration of service super types only. A new data type is added to hold the modes and types of the service properties registered by a sub type service.
Consequences if not approved:	⌘	Impossible to support extensions to standard service capabilities.

Clauses affected:	⌘	8.1.2, 8.3.1.1, 10.1				
		Y	**N**			
Other specs Affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

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

Introduction

Specification 29.198-03 describes the mechanism of service properties and lists the use of service super and sub types. All service capability servers contain a set of common properties and a set of SCS-specific properties. According the current specification it is not possible for sub type services to specify the types and modes for non-standard service properties when registering at the Framework.

Ericsson proposes to add a new method to the IpFwServiceRegistration interface in 29.198-03 that shall be used for registration of service sub types.

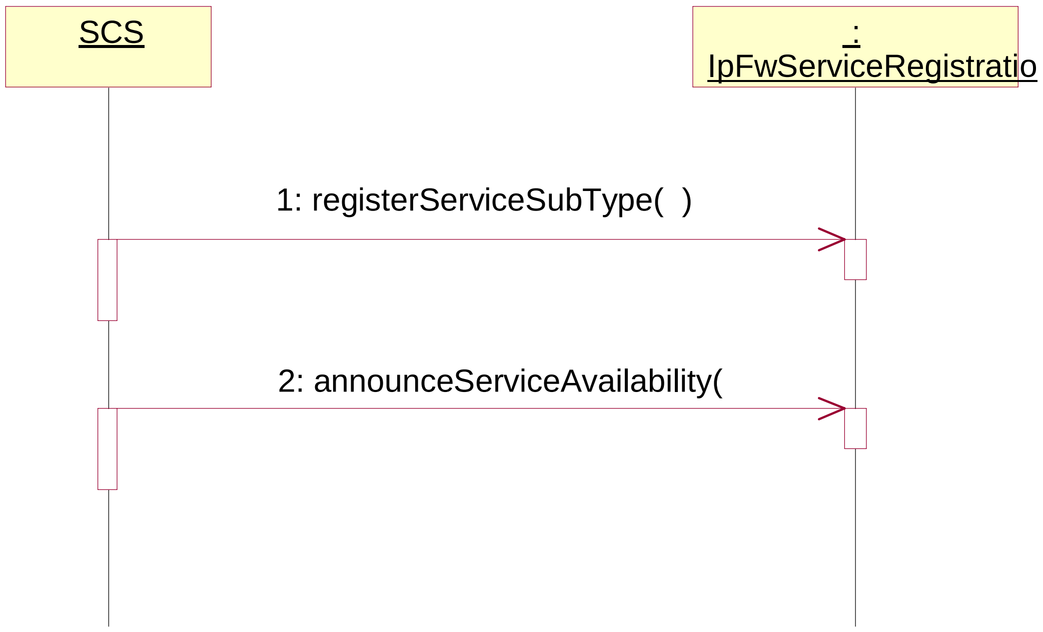
Proposal

IpFwServiceRegistration is extended with a method called registerServiceSubType(). This method allows an extended SCS to register by additionally providing the service property list (name, mode, type, and value).

Proposed Changes

8.1.2.2 New SCF Sub Type Registration

The following figure shows the process of registering a new proprietary Service Capability Feature in the Framework. This SCF is a sub type of the standard SCF.



1: Registration: first step - register service sub type

For sub type registration, besides the values for the standard service properties, the modes, types, and values for the additional service properties must be provided by the SCF.

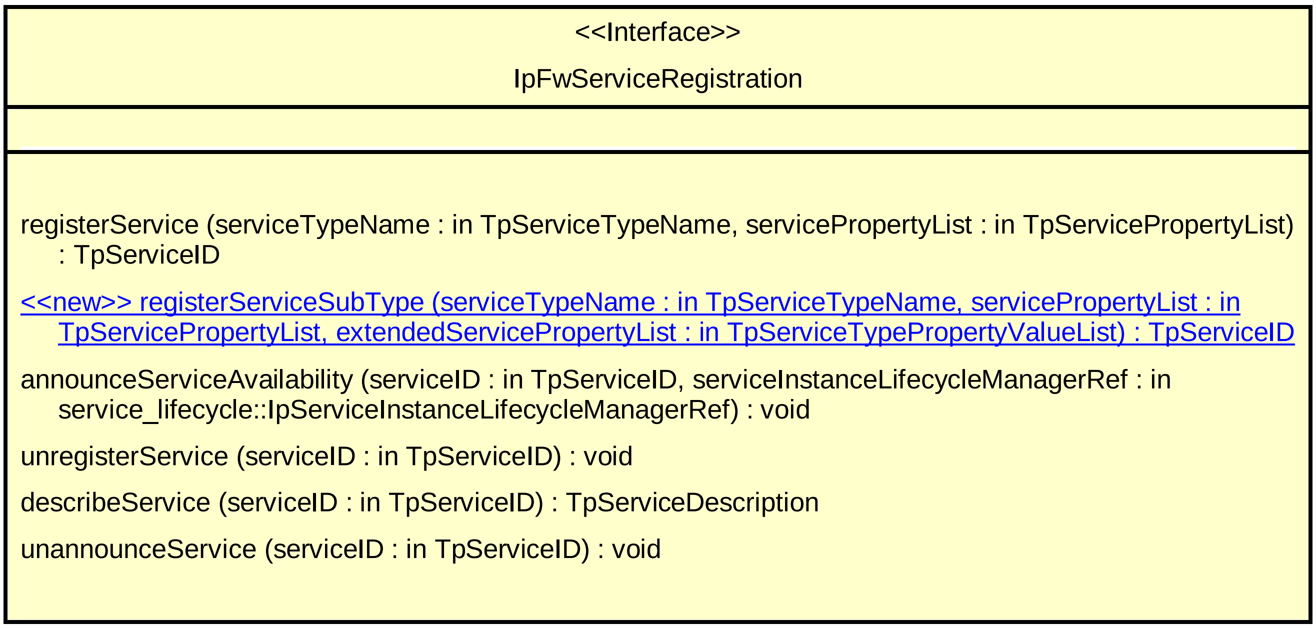
2: Registration: second step - announce service availability

This is identical to announcing availability of super types

8.3.1.1 Interface Class IpFwServiceRegistration

Inherits from: IpInterface.

The Service Registration interface provides the methods used for the registration of network SCFs at the framework. This interface and at least the methods registerService(), announceServiceAvailability(), unregisterService() and unannounceService() shall be implemented by a Framework.



<<Interface>>

IpFwServiceRegistration

registerService (serviceTypeName : in TpServiceTypeName, servicePropertyList : in TpServicePropertyList) : TpServiceID

<<new>> registerServiceSubType (serviceTypeName : in TpServiceTypeName, servicePropertyList : in TpServicePropertyList, extendedServicePropertyList : in TpServiceTypePropertyValueList) : TpServiceID

announceServiceAvailability (serviceID : in TpServiceID, serviceInstanceLifecycleManagerRef : in service_lifecycle::IpServiceInstanceLifecycleManagerRef) : void

unregisterService (serviceID : in TpServiceID) : void

describeService (serviceID : in TpServiceID) : TpServiceDescription

unannounceService (serviceID : in TpServiceID) : void

8.3.1.1.1 Method registerService()

The registerService() operation is the means by which a service is registered in the Framework, for subsequent discovery by the enterprise applications. Registration can only succeed when the Service type of the service is known to the Framework (ServiceType is 'available'). A service-ID is returned to the service supplier when a service is registered in the Framework. When the service is not registered because the ServiceType is 'unavailable', a P_SERVICE_TYPE_UNAVAILABLE is raised. The service-ID is the handle with which the service supplier can identify the registered service when needed (e.g. for withdrawing it). The service-ID is only meaningful in the context of the Framework that generated it.

This method should be used for registration of service super types only. For registering service sub types, the registerServiceSubType() method should be used.

Returns <serviceID> : This is the unique handle that is returned as a result of the successful completion of this operation. The Service Supplier can identify the registered service when attempting to access it via other operations such as unregisterService(), etc. Enterprise client applications are also returned this service-ID when attempting to discover a service of this type.

Parameters

`serviceTypeName : in TpServiceTypeName`

The "serviceTypeName" parameter identifies the service type. If the string representation of the "type" does not obey the rules for identifiers, then a P_ILLEGAL_SERVICE_TYPE exception is raised. If the "type" is correct syntactically but the Framework is able to unambiguously determine that it is not a recognised service type, then a P_UNKNOWN_SERVICE_TYPE exception is raised.

servicePropertyList : in TpServicePropertyList

The "servicePropertyList" parameter is a list of property name and property value pairs. They describe the service being registered. This description typically covers behavioural, non-functional and non-computational aspects of the service. Service properties are marked "mandatory" or "readonly". These property mode attributes have the following semantics:

a. mandatory - a service associated with this service type must provide an appropriate value for this property when registering.

b. readonly - this modifier indicates that the property is optional, but that once given a value, subsequently it may not be modified.

Specifying both modifiers indicates that a value must be provided and that subsequently it may not be modified. Examples of such properties are those which form part of a service agreement and hence cannot be modified by service suppliers during the life time of service.

If the type or the semantics of the type of any of the property values is not the same as the declared type (declared in the service type), then a P_PROPERTY_TYPE_MISMATCH exception is raised. If the "servicePropertyList" parameter omits any property declared in the service type with a mode of mandatory, then a P_MISSING_MANDATORY_PROPERTY exception is raised. If two or more properties with the same property name are included in this parameter, the P_DUPLICATE_PROPERTY_NAME exception is raised.

Returns

TpServiceID

Raises

TpCommonExceptions, P_PROPERTY_TYPE_MISMATCH, P_DUPLICATE_PROPERTY_NAME, P_ILLEGAL_SERVICE_TYPE, P_UNKNOWN_SERVICE_TYPE, P_MISSING_MANDATORY_PROPERTY, P_SERVICE_TYPE_UNAVAILABLE

8.3.1.1.2 Method <<new>> registerServiceSubType()

The registerServiceExtension() operation is the means by which an extended service is registered in the Framework, for subsequent discovery by the enterprise applications. Registration only succeeds if the service type is known to the Framework (ServiceType is 'available'). A service-ID is returned to the service supplier when a service is registered in the Framework. When the service is not registered because the ServiceType is 'unavailable', a P_SERVICE_TYPE_UNAVAILABLE is raised. The service-ID is the handle with which the service supplier can identify the registered service when needed (e.g. for withdrawing it). The service-ID is only meaningful in the context of the Framework that generated it.

This method should be used for registration of service sub types only. For registering service super types, the registerService () method should be used.

Returns <serviceID> : This is the unique handle that is returned as a result of the successful completion of this operation. The Service Supplier can identify the registered service when attempting to access it via other operations such as unregisterService(), etc. Enterprise client applications are also returned this service-ID when attempting to discover a service of this type.

Parameters

serviceTypeName : in TpServiceTypeName

The "serviceTypeName" parameter identifies the service type. If the string representation of the "type" does not obey the rules for identifiers, then a P_ILLEGAL_SERVICE_TYPE exception is raised. If the "type" is correct syntactically but the Framework is able to unambiguously determine that it is not a recognised service type, then a P_UNKNOWN_SERVICE_TYPE exception is raised.

servicePropertyList : in TpServicePropertyList

The "servicePropertyList" parameter is a list of property name and property value pairs corresponding to the service properties applicable to the standard service. They describe the service being registered.

If the type or the semantics of the type of any of the property values is not the same as the declared type (declared in the service type), then a P_PROPERTY_TYPE_MISMATCH exception is raised.

If the "servicePropertyList" parameter omits any property declared in the service type with a mode of mandatory, then a P_MISSING_MANDATORY_PROPERTY exception is raised.

If two or more properties with the same property name are included in this parameter, the P_DUPLICATE_PROPERTY_NAME exception is raised.

`extendedServicePropertyList : in TpServiceTypePropertyValueList`

The "extendedServicePropertyList" parameter is a list of property name, mode, type, and property value tuples corresponding to the service properties applicable to the extended standard service. They describe the service being registered.

If two or more properties with the same property name are included in this parameter, the P_DUPLICATE_PROPERTY_NAME exception is raised.

Returns

`TpServiceID`

Raises

`TpCommonExceptions, P_PROPERTY_TYPE_MISMATCH, P_DUPLICATE_PROPERTY_NAME, P_ILLEGAL_SERVICE_TYPE, P_UNKNOWN_SERVICE_TYPE, P_MISSING_MANDATORY_PROPERTY, P_SERVICE_TYPE_UNAVAILABLE`

10.1.33 TpServiceTypePropertyValue

This data type is a `Sequence of Data Elements` which describes a service property associated with a service. It defines the name and mode of the service property, the service property type (e.g. Boolean, integer), and also value. It is similar to, but distinct from, TpServiceProperty. The latter does not define the modes and types and is used to register values for known service properties only.

Sequence Element Name	Sequence Element Type	Documentation
`ServicePropertyName`	TpServicePropertyName	The name of the service property.
`ServiceTypePropertyMode`	TpServiceTypePropertyMode	The mode of the service property.
`ServicePropertyTypeName`	TpServicePropertyTypeName	The type of the service property.
`ServicePropertyValueList`	TpServicePropertyValueList	The Value-list of the service property.

10.1.34 TpServiceTypePropertyValueList

This data type defines a Numbered Set of Data Elements of type TpServiceTypePropertyValue.

10.1.~~22~~35 TpServicePropertyName

This data type is identical to TpString. It defines a valid SCF property name. The valid service property names are detailed in 10.2 and in the SCF data definitions. Additionally, service property names for proprietary service properties (used for service sub types) are possible.

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03** **CR** **094** ⌘**rev** **-** ⌘ Current version: **5.4.0** ⌘

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

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

Title: ⌘	Improve User Interaction message management functions		
Source: ⌘	CN5 (scottjb@us.ibm.com)		
Work item code: ⌘	OSA3	***Date:*** ⌘	18/07/2003
Category: ⌘	**B**	***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 OSA User Interaction API provides the capability to record and playback messages, but it does not provide a mechanism to retrieve the message content by the application, or provide a mechanism to set the message content by the application. These features are necessary to enable the applications to utilize the content of the messages in a meaningful way for both administration/management and for interaction with enhanced services or users (mid-call). The User Interaction service should be functional for both administrative provisioning of the messages of the gateway, but also functional for application interaction with the user for the purposes of enhanced services and enterprise applications. When recording a message through the IpUICall interface, the application can play it back or delete it, but currently can not retrieve it from the gateway. The application may want to record the user's voice for a credit card authorization or such and then store the recording in its own database. There are many reasons why the application may need to retrieve the waveform data. Clearly, this method is intended for low-frequency usage for performance and bandwidth reasons, however it is still necessary. Additionally, The application also does not have a mechanism to assign a messageID to a new User Interaction message that is provided by the application, thereby adding a message to the provisioned set of messages. Additionally, the currently supported deleteMessageReq() is provided on the IpUICall interface, which can only be used in conjunction with a Call/CallLeg session, however, this is a desireable administrative function that may not be associated with a particular Call/CallLeg session. As with the new methods described above, they may be desireable without a call session. If this functionality is not possible with the infrastructure of a particular switch, it should still be included in the specification for completeness, because from the application perspective this is an important feature, and in time the necessary functions could be integrated with the core network.
Summary of change: ⌘	A new IpUIAdminManager SCF interface is proposed so that it is clear that it is not attached to a Call/CallLeg session or a TpAddress.

		The TpServiceTypeName is updated with the name of the new interface, in accordance with the changes from the corresponding change N5-030409. Changes derived from: ftp://ftp.3gpp.org/specs/2003-06/Rel-5/29_series/
Consequences if not approved:	⌘	The usefulness of the IpUICall interfaces by an application is limited if the application can not retrieve the information provided by the user, or can not dynamically set the messages to be played by the application.

Clauses affected:	⌘	11.1.30				
		Y	**N**			
Other specs affected:	⌘	**X**		Other core specifications	⌘	29.198-05
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘	Related 29.198-05 CR in N5-030409.				

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

11.1.30 TpServiceTypeName

This data type is identical to a TpString, and is defined as a string of characters that uniquely identifies the type of an SCF interface. Other Network operator specific capabilities may also be used, but should be preceded by the string "SP_". The following values are defined.

Character String Value	Description
NULL	An empty (NULL) string indicates no SCF name
P_GENERIC_CALL_CONTROL	The name of the Generic Call Control SCF
P_MULTI_PARTY_CALL_CONTROL	The name of the MultiParty Call Control SCF
P_MULTI_MEDIA_CALL_CONTROL	The name of the MultiMedia Call Control SCF
P_CONFERENCE_CALL_CONTROL	The name of the Conference Call Control SCF
P_USER_INTERACTION	The name of the User Interaction SCFs
P_USER_INTERACTION_ADMIN	The name of the User Interaction Administration SCF
P_TERMINAL_CAPABILITIES	The name of the Terminal Capabilities SCF
P_USER_LOCATION	The name of the User Location SCF
P_USER_LOCATION_CAMEL	The name of the Network User Location SCF
P_USER_LOCATION_EMERGENCY	The name of the User Location Emergency SCF
P_USER_STATUS	The name of the User Status SCF
P_DATA_SESSION_CONTROL	The name of the Data Session Control SCF
P_GENERIC_MESSAGING	The name of the Generic Messaging SCF
P_CONNECTIVITY_MANAGER	The name of the Connectivity Manager SCF
P_CHARGING	The name of the Charging SCF
P_ACCOUNT_MANAGEMENT	The name of the Account Management SCF
P_POLICY_MANAGEMENT	The name of the Policy Management provisioning SCF
P_PAM_ACCESS	The name of PAM presentity SCF
P_PAM_EVENT_MANAGEMENT	The name of PAM watcher SCF
P_PAM_PROVISIONING	The name of PAM provisioning SCF

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-030430**
Meeting #25, Bangkok, Thailand, 27-31 October 2003

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03**	**CR 095**	⌘ **rev**	**-**	⌘ Current version:	**5.4.0** ⌘

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

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

Title: ⌘	Add new values for TpServiceTypeName for Policy Management		
Source: ⌘	CN5 (squtub@lucent.com)		
Work item code: ⌘	OSA3	***Date:*** ⌘	18/07/2003
Category: ⌘	**B**	***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: ⌘	Add new SCF names (for policy provisioning, policy evaluation) to the TpServiceTypeName, so these SCFs are discoverable by applications.
Summary of change: ⌘	Add new SCF names to the TpServiceTypeName.
Consequences if not approved: ⌘	Unable to use the newly introduced Policy Management SCF.

Clauses affected: ⌘	10.1.30			
	Y	**N**		
Other specs affected: ⌘		**X**	Other core specifications	⌘
		X	Test specifications	
		X	O&M Specifications	
Other comments: ⌘				

How to create CRs using this form:

Start of Change

10.1.30 TpServiceTypeName

This data type is identical to a TpString, and is defined as a string of characters that uniquely identifies the type of an SCF interface. Other Network operator specific capabilities may also be used, but should be preceded by the string "SP_". The following values are defined.

Character String Value	Description
NULL	An empty (NULL) string indicates no SCF name
P_GENERIC_CALL_CONTROL	The name of the Generic Call Control SCF
P_MULTI_PARTY_CALL_CONTROL	The name of the MultiParty Call Control SCF
P_MULTI_MEDIA_CALL_CONTROL	The name of the MultiMedia Call Control SCF
P_CONFERENCE_CALL_CONTROL	The name of the Conference Call Control SCF
P_USER_INTERACTION	The name of the User Interaction SCFs
P_TERMINAL_CAPABILITIES	The name of the Terminal Capabilities SCF
P_USER_LOCATION	The name of the User Location SCF
P_USER_LOCATION_CAMEL	The name of the Network User Location SCF
P_USER_LOCATION_EMERGENCY	The name of the User Location Emergency SCF
P_USER_STATUS	The name of the User Status SCF
P_DATA_SESSION_CONTROL	The name of the Data Session Control SCF
P_GENERIC_MESSAGING	The name of the Generic Messaging SCF
P_CONNECTIVITY_MANAGER	The name of the Connectivity Manager SCF
P_CHARGING	The name of the Charging SCF
P_ACCOUNT_MANAGEMENT	The name of the Account Management SCF
P_POLICY_~~MANAGEMENT~~PROVISIONING	The name of the Policy Management provisioning SCF
P_POLICY_EVALUATION	The name of the Policy Management policy evaluation SCF
P_PAM_ACCESS	The name of PAM presentity SCF
P_PAM_EVENT_MANAGEMENT	The name of PAM watcher SCF
P_PAM_PROVISIONING	The name of PAM provisioning SCF

End of Change

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-030431**
Meeting #24, San Francisco, CA, USA, 14 - 18 July 2003

CR-Form-v5

CHANGE REQUEST

⌘ **29.198-03**	**CR** **096**	⌘ **rev**	**-**	⌘ Current version:	**5.4.0** ⌘

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

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

Title:	⌘	Allow for applications to re-obtain the reference to the service manager		
Source:	⌘	CN5 (Erwin.van.Rijssen@ericsson.com)		
Work item code:	⌘	OSA3	***Date:*** ⌘	18/07/2003
Category:	⌘	**B**	***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)*

Reason for change:	⌘	At this moment it is not possible to re-obtain a reference to the service manager of an SCF an application is using. However, in case an application has lost the reference to the Service manager e.g. due to a crash, without the SCS being aware of this, it should be possible for the application to re-obtain a reference to the Service manager.
Summary of change:	⌘	Remove description in method SelectService that exception will be thrown when application invokes the signServiceLevelAgreement method more than once and add text to description of signServiceLevelAgreement that method can be used to re-obtain reference to Service Manager. **This CR was already approved in the CN5#21 Dublin meeting (10/2002) in N5-021150, but its contents have to be implemented on 29.198-03 in Rel-6 and therefore the CR is resubmitted**
Consequences if not approved:	⌘	Applications that loose reference to Service Manager will not be able to use SCF anymore.

Clauses affected:	⌘	7.3.2.2.1, 7.3.2.2.3, 8.3.2.1.1				
		Y	**N**			
Other specs Affected:	⌘		X	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

How to create CRs using this form:

Change in Clause 7.3.2.2.1

7.3.2.2.1 Method signServiceAgreement()

After the framework has called signServiceAgreement() on the application's IpAppServiceAgreementManagement interface, this method is used by the client application to request that the framework sign the service agreement, which allows the client application to use the service. A reference to the service manager interface of the service is returned to the client application. The service manager returned will be configured as per the service level agreement. If the framework uses service subscription, the service level agreement will be encapsulated in the subscription properties contained in the contract/profile for the client application, which will be a restriction of the registered properties. If the client application is not allowed to access the service, then an error code (P_SERVICE_ACCESS_DENIED) is returned. If the client application invokes this method before the processing (i.e. digital signature verification) the reponse of signServiceAgreement() on the application's IpAppServiceAgreementManagement interface completed, a TpCommonExceptions with ExceptionType P_INVALID_STATE may be raised to indicate that this method is currently unable to complete the method due to a race condition. In this case, the TpCommonExceptions with ExceptionType P_INVALID_STATE suggests the application to retry the method invocation after a reasonable amount of time has passed.

There must be only one service instance per client application. Therefore, in case the client attempts to select a service for which it has already signed a service agreement and this service agreement has not been terminated, a reference to the already existing service manager will be returned.

Returns <signatureAndServiceMgr> : This contains the digital signature of the framework for the service agreement, and a reference to the service manager interface of the service.

```
structure TpSignatureAndServiceMgr {
        digitalSignature: TpOctetSet;
          serviceMgrInterface: IpServiceRef;
            };
```

The digitalSignature 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 the agreement text given by the client application. The "external signature" construct shall not be used (i.e. the eContent field in the EncapsulatedContentInfo field shall be present and contain the agreement text string). The signing-time attribute, as defined in section 11.3 of RFC 2630, shall also be used to provide replay prevention.

The serviceMgrInterface is a reference to the service manager interface for the selected service.

Parameters

serviceToken : in TpServiceToken

This is the token returned by the framework in a call to the selectService() method. This token is used to identify the service instance requested by the client application. If the serviceToken is invalid, or has expired, an error code (P_INVALID_SERVICE_TOKEN) is returned.

agreementText : in TpString

This is the agreement text that is to be signed by the framework using the private key of the framework. If the agreementText is invalid, then an error code (P_INVALID_AGREEMENT_TEXT) is returned.

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 framework, an error code (P_INVALID_SIGNING_ALGORITHM) is returned. 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).

Returns

`TpSignatureAndServiceMgr`

Raises

`TpCommonExceptions,P_ACCESS_DENIED,P_INVALID_AGREEMENT_TEXT,P_INVALID_SERVICE_TOKEN,P_INVALID_SIGNING_ALGORITHM,P_SERVICE_ACCESS_DENIED`

End of Change in Clause 7.3.2.2.1

Change in Clause 7.3.2.2.3

7.3.2.2.3 Method selectService()

This method is used by the client application to identify the service that the client application wishes to use. If the client application is not allowed to access the service, then the P_SERVICE_ACCESS_DENIED exception is thrown. ~~The P_SERVICE_ACCESS_DENIED exception is also thrown if the client attempts to select a service for which it has already signed a service agreement for, and therefore obtained an instance of. This is because there must be only one service instance per client application.~~

Returns <serviceToken> : This is a free format text token returned by the framework, which can be signed as part of a service agreement. This will contain operator specific information relating to the service level agreement. The serviceToken has a limited lifetime. If the lifetime of the serviceToken expires, a method accepting the serviceToken will return an error code (P_INVALID_SERVICE_TOKEN). Service Tokens will automatically expire if the client application or framework invokes the endAccess method on the other's corresponding access interface.

Parameters

`serviceID : in TpServiceID`

This identifies the service required. If the serviceID is not recognised by the framework, an error code (P_INVALID_SERVICE_ID) is returned.

Returns

`TpServiceToken`

Raises

`TpCommonExceptions, P_ACCESS_DENIED, P_INVALID_SERVICE_ID, P_SERVICE_ACCESS_DENIED`

End of Change in Clause 7.3.2.2.3

Change in Clause 8.3.2.1.1

8.3.2.1.1 Method createServiceManager()

This method returns a new service manager interface reference for the specified application. The service instance will be configured for the client application using the properties agreed in the service level agreement.

In case there is already a service manager available for the specified application and serviceInstanceID this reference is returned and no new service manager is created.

Returns <serviceManager> : Specifies the service manager interface reference for the specified application ID.

Parameters

`application : in TpClientAppID`

Specifies the application for which the service manager interface is requested.

`serviceProperties : in TpServicePropertyList`

Specifies the service properties and their values that are to be used to configure the service instance. These properties form a part of the service level agreement. An example of these properties is a list of methods that the client application is allowed to invoke on the service interfaces.

`serviceInstanceID : in TpServiceInstanceID`

Specifies the Service Instance ID that the new Service Manager is to be identified by.

Returns

`IpServiceRef`

Raises

`TpCommonExceptions, P_INVALID_PROPERTY`

End of Change in Clause 8.3.2.1.1
End of Document

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-030322**
Meeting #24, San Francisco, CA, USA, 14 - 18 July 2003

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-03**	**CR**	**097**	⌘ **rev**	**-**	⌘	Current version:	**5.4.0** ⌘

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

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

Title: ⌘	Add support in OSA to inform applications about new SCSs and their level of Backward compatibility – Align with SA1's 22.127	
Source: ⌘	CN5 (Erwin.van.Rijssen@ericsson.com)	
Work item code: ⌘	OSA3	***Date:*** ⌘ 18/07/2003
Category: ⌘	**B**	***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: ⌘	Fullfil the requirement in 22.127 on adding support in OSA to inform applications about new SCSs where an application can migrate to.
Summary of change: ⌘	New generic Service Properties and a new event is added to be able to report applications that are currently using a certain SCS that a new SCS to which the applications can migrate to has become available.
Consequences if not approved: ⌘	Mismatch between requirements and the actual API.

Clauses affected: ⌘	8.3.1.1.1, 9.2, 10.2				
	Y	**N**			
Other specs affected: ⌘		**X**	Other core specifications	⌘	
		X	Test specifications		
		X	O&M Specifications		
Other comments: ⌘					

Change in Clause 8.3.1.1.1

8.3.1.1.1 Method registerService()

The registerService() operation is the means by which a service is registered in the Framework, for subsequent discovery by the enterprise applications. Registration can only succeed when the Service type of the service is known to the Framework (ServiceType is 'available'). A service-ID is returned to the service supplier when a service is registered in the Framework. When the service is not registered because the ServiceType is 'unavailable', a P_SERVICE_TYPE_UNAVAILABLE is raised. The service-ID is the handle with which the service supplier can identify the registered service when needed (e.g. for withdrawing it). The service-ID is only meaningful in the context of the Framework that generated it.

Returns <serviceID> : This is the unique handle that is returned as a result of the successful completion of this operation. The Service Supplier can identify the registered service when attempting to access it via other operations such as unregisterService(), etc. Enterprise client applications are also returned this service-ID when attempting to discover a service of this type.

If a service is registered with the property P_COMPATIBLE_WITH_SERVICE in its servicePropertyList, then the Framework shall notify all applications using instances of services identified by this property, using the P_EVENT_FW_MIGRATION_SERVICE_AVAILABLE event, if they have registered for such a notification. If an incorrect combination of properties is included in conjunction with P_COMPATIBLE_WITH_SERVICE, a P_MISSING_MANDATORY_PROPERTY exception is raised.

Parameters

serviceTypeName : in TpServiceTypeName

The "serviceTypeName" parameter identifies the service type. If the string representation of the "type" does not obey the rules for identifiers, then a P_ILLEGAL_SERVICE_TYPE exception is raised. If the "type" is correct syntactically but the Framework is able to unambiguously determine that it is not a recognised service type, then a P_UNKNOWN_SERVICE_TYPE exception is raised.

servicePropertyList : in TpServicePropertyList

The "servicePropertyList" parameter is a list of property name and property value pairs. They describe the service being registered. This description typically covers behavioural, non-functional and non-computational aspects of the service. Service properties are marked "mandatory" or "readonly". These property mode attributes have the following semantics:
a. mandatory - a service associated with this service type must provide an appropriate value for this property when registering.
b. readonly - this modifier indicates that the property is optional, but that once given a value, subsequently it may not be modified.
Specifying both modifiers indicates that a value must be provided and that subsequently it may not be modified. Examples of such properties are those which form part of a service agreement and hence cannot be modified by service suppliers during the life time of service.
If the type or the semantics of the type of any of the property values is not the same as the declared type (declared in the service type), then a P_PROPERTY_TYPE_MISMATCH exception is raised. If the "servicePropertyList" parameter omits any property declared in the service type with a mode of mandatory, then a P_MISSING_MANDATORY_PROPERTY exception is raised. If two or more properties with the same property name are included in this parameter, the P_DUPLICATE_PROPERTY_NAME exception is raised.

Returns

TpServiceID

Raises

TpCommonExceptions, P_PROPERTY_TYPE_MISMATCH, P_DUPLICATE_PROPERTY_NAME, P_ILLEGAL_SERVICE_TYPE, P_UNKNOWN_SERVICE_TYPE, P_MISSING_MANDATORY_PROPERTY, P_SERVICE_TYPE_UNAVAILABLE

End of Change in Clause 8.3.1.1.1

Change in Clause 9.2

9.2 General Service Properties

Each service instance has the following general properties:

- Service Name
- Service Version
- Service Instance ID
- Service Instance Description
- Product Name
- Product Version
- Supported Interfaces
- Operation Set
- Compatible Service
- Backward Compatibility Level
- Migration Required
- Data Migrated
- Migration Date and Time

The following sections describe these general service properties in more detail. The values for the mode are defined in the type TpServiceTypePropertyMode.

...

9.2.9 <<new>> Compatible Service

Property	**Type**	**Mode**	**Description**
P_COMPATIBLE_WITH_SERVICE	STRING_SET	READONLY	Specifies the Set of Services, identified by their ServiceIDs, with which this new service is compatible. This property should at least be accompanied with the properties P_BACKWARD_COMPATIBILITY_LEVEL, P_MIGRATION_REQUIRED. Note that the new Service can be compatible with more than one Service that is currently registered to the Framework. Therefore this Property is a SET, as well as all related properties like Migration Required, Data Migrated, etc. For all these properties the order of the Services shall be identical.

9.2.10 <<new>> Backward Compatibility Level

Property	**Type**	**Mode**	**Description**
P_BACKWARD_COMPATIBILITY_LEVEL	BOOLEAN_SET	READONLY	Specifies if the new service is completely backwards compatible with each service identified in the P_COMPATIBLE_WITH_SERVICE property: Value = TRUE: Service is completely backwards compatible Value = FALSE: SCS is not completely backwards compatible. This property requires the presence of P_COMPATIBLE_WITH_SERVICE property. Note that the new Service can be compatible with more than one Service that is currently registered to the Framework. Therefore this Property is a SET, as well as all related properties. For each service identified in P_COMPATIBLE_WITH_SERVICE, one value of this property shall be present in the value set of this property at service registration. For all these properties the order of the Services shall be identical.

9.2.11 <<new>> Migration Required

Property	Type	Mode	Description
P_MIGRATION_REQUIRED	BOOLEAN_SET	READONLY	Specifies if the new service is replacing the service identified in the P_COMPATIBLE_WITH_SERVICE property: Value = TRUE: new service is replacing the existing one – migration is required before the date/time indicated in P_MIGRATION_DATE_AND_TIME property. Value = FALSE: new service is not replacing the existing one – migration not required, the existing service is retained. This property requires the presence of P_COMPATIBLE_WITH_SERVICE property. If the value set of P_MIGRATION_REQUIRED contains TRUE, P_DATA_MIGRATED and P_MIGRATION_DATE_AND_TIME properties shall also to be present. Note that the new Service can be compatible with more than one Service that is currently registered to the Framework. Therefore this Property is a SET, as well as all related properties. For each service identified in P_COMPATIBLE_WITH_SERVICE, one value of this property shall be present in the value set of this property at service registration. For all these properties the order of the Services shall be identical.

9.2.12 <<new>> Data Migrated

Property	Type	Mode	Description
P_DATA_MIGRATED	BOOLEAN_SET	READONLY	Indicates if the data (e.g. notifications) from the existing service identified in the P_COMPATIBLE_WITH_SERVICE property is also available in this Service. Value = TRUE: all data is migrated Value = FALSE: no data is migrated This property requires the presence of P_COMPATIBLE_WITH_SERVICE and the P_MIGRATION_REQUIRED properties. Note that the new Service can be compatible with more than one Service that is currently registered to the Framework. Therefore this Property is a SET, as well as all related properties. For each service identified in P_COMPATIBLE_WITH_SERVICE, one value of this property shall be present in the value set of this property at service registration. For all these properties the order of the Services shall be identical.

9.2.13 <<new>> Migration Date And Time

Property	**Type**	**Mode**	**Description**
P_MIGRATION_DATE_AND_TIME	STRING_SET	READONLY	This property contains the date and time, in the format described in TpDateAndTime, by which point applications shall have migrated from existing services to this new service. Migration to the new service requires the application to terminate the existing service agreement, and sign a new one. Failure to do this by the migration date and time indicated in this property may result in the service agreement being terminated by the Framework, since the service supplier may choose to unregister the service following this date and time. Only one value of TpDateAndTime is permitted to be present in this property at service registration. This property requires the presence of P_COMPATIBLE_WITH_SERVICE, P_MIGRATION_REQUIRED and P_DATA_MIGRATED properties. Note that the new Service can be compatible with more than one Service that is currently registered to the Framework. Therefore this Property is a SET, as well as all related properties. For each service identified in P_COMPATIBLE_WITH_SERVICE, one value of this property shall be present in the value set of this property at service registration. For all these properties the order of the Services shall be identical. For those services for which migration is not required (P_MIGRATION_REQUIRED set to FALSE), the corresponding value of this property shall be ignored.

End of Change in Clause 9.2

Change in Clause 10.2

10.2 Event Notification Data Definitions

10.2.1 TpFwEventName

Defines the name of event being notified.

Name	Value	Description
P_EVENT_FW_NAME_UNDEFINED	0	Undefined
P_EVENT_FW_SERVICE_AVAILABLE	1	Notification of SCS(s) available
P_EVENT_FW_SERVICE_UNAVAILABLE	2	Notification of SCS(s) becoming unavailable
P_EVENT_FW_MIGRATION_SERVICE_AVAILABLE	3	Notification of a backwards compatible SCS becoming available, to which the application can migrate.

10.2.2 TpFwEventCriteria

Defines the `Tagged Choice of Data Elements` that specify the criteria for an event notification to be generated.

	Tag Element Type	
	TpFwEventName	

Tag Element Value	Choice Element Type	Choice Element Name
P_EVENT_FW_NAME_UNDEFINED	TpString	EventNameUndefined
P_EVENT_FW_ SERVICE_AVAILABLE	TpServiceTypeNameList	ServiceTypeNameList
P_EVENT_FW_SERVICE_UNAVAILABLE	TpServiceTypeNameList	UnavailableServiceTypeNameList
P_EVENT_FW_MIGRATION_SERVICE_AVAILABLE	TpServiceTypeNameList	CompatibleServiceTypeNameList

10.2.3 TpFwEventInfo

Defines the `Tagged Choice of Data Elements` that specify the information returned to the application in an event notification.

	Tag Element Type	
	TpFwEventName	

Tag Element Value	Choice Element Type	Choice Element Name
P_EVENT_FW_NAME_UNDEFINED	TpString	EventNameUndefined
P_EVENT_FW_ SERVICE_AVAILABLE	TpServiceIDList	ServiceIDList
P_EVENT_FW_SERVICE_UNAVAILABLE	TpServiceIDList	UnavailableServiceIDList
P_EVENT_FW_MIGRATION_SERVICE_AVAILABLE	TpFWMigrationServiceAvailableInfo	MigrationServiceAvailableList

10.2.4 <<new>> TpFwMigrationServiceAvailableInfo

Defines the information to be supplied when an SCS becomes available

Sequence Element Name	Sequence Element Type	Documentation
ServiceType	TpServiceTypeName	Type of SCS that has become available
ServiceID	TpServiceID	ID of the SCS that has become available
CompatibleServiceID	TpServiceID	ID of the SCS with which this new SCS is compatible with.
BackwardCompatibilityLevel	TpBoolean	Specifies if the new SCS is completely backwards compatible with the currently used SCS. Value = TRUE: SCS is completley backwards compatible Value = FALSE: SCS is not completely backwards compatible. Contact the Framework operator for more information.on how to migrate.
MigrationRequired	TpBoolean	Specifies if the new SCS is replacing the existing SCS Value = TRUE: new SCS is replacing the existing one - migration is required before the date/time indicated in MigrationDateAndTime field Value = FALSE: new SCS is not replacing the existing one, but is provided in addition. All migration to the new SCS, whether required or not, shall involve the application terminating the existing service agreement and signing a new one.
DataMigrated	TpBoolean	Indicates whether all the data the application set in the previous SCS (e.g. notifications) is also available in the new SCS. Value = FALSE : the new SCS has not obtained all data (e.g. notifications) related to the old SCS and the application needs to reset all the previous data. Value = TRUE: the new SCS has obtained data (e.g. notifications) related to the old SCS, the application can use the new SCS without resetting data.
MigrationDataAndTime	TpDataAndTime	Indicates the date and time before which applications shall have migrated from existing the existing SCS to this new SCS. Migration to the new SCS requires the application to terminate the existing service agreement, and sign a new one. Failure to do this by the migration date and time indicated in this field may result in the service agreement being terminated by the Framework, since the service supplier may choose to unregister the service following this date and time. The value of this parameter, if present, shall be ignored if MigrationRequired is set to FALSE
MigrationAdditionalInfo	TpMigrationAdditionalInfoSet	Contains additional migration information. This is initially provided to permit addition of information in later releases without impacting backwards compatibiltiy.

10.2.5 <<new>> TpMigrationAdditionalInfo

Defines the Tagged Choice of Data Elements that specify additional migration-related information.

	Tag Element Type	
	TpMigrationAdditionalInfoType	

Tag Element Value	Choice Element Type	Choice Element Name
P_MIGRATION_INFO_UNDEFINED	NULL	MigrationInfoUndefined

10.2.6 <<new>> TpMigrationAdditionalInfoType

Defines the type of migration-related additional information.

Name	Value	Description
P_MIGRATION_INFO_UNDEFINED	0	Undefined

10.2.7 <<new>> TpMigrationAdditionalInfoSet

Defines a Numbered Set of Data Elements of TpMigrationAdditionalInfo.

End of Change in Clause 10.2
End of Document

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-030433**
Meeting #24, San Francisco, CA, USA, 14 - 18 July 2003

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03** **CR** **098** ⌘ **rev** **-** ⌘ Current version: **5.4.0** ⌘

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

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

Title:	⌘	Add “Extended User Status” as service type name - Align with 29.198-06		
Source:	⌘	CN5 (Erwin.van.Rijssen@ericsson.com)		
Work item code:	⌘	OSA3	***Date:*** ⌘	18/07/2003
Category:	⌘	**B**	***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:	⌘	Extension of User Status results in new service type name
Summary of change:	⌘	Add “Extended User Status” as service type name in 29.198-03 (Framework)
Consequences if not approved:	⌘	Not aligned with 29.198-06 where ExtendedUserStatus has been added

Clauses affected:	⌘	10.1.30				
		Y	**N**			
Other specs Affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

How to create CRs using this form:

Change in Clause 10.1.30

10.1.30 TpServiceTypeName

This data type is identical to a TpString, and is defined as a string of characters that uniquely identifies the type of an SCF interface. Other Network operator specific capabilities may also be used, but should be preceded by the string "SP_". The following values are defined.

Character String Value	Description
NULL	An empty (NULL) string indicates no SCF name
P_GENERIC_CALL_CONTROL	The name of the Generic Call Control SCF
P_MULTI_PARTY_CALL_CONTROL	The name of the MultiParty Call Control SCF
P_MULTI_MEDIA_CALL_CONTROL	The name of the MultiMedia Call Control SCF
P_CONFERENCE_CALL_CONTROL	The name of the Conference Call Control SCF
P_USER_INTERACTION	The name of the User Interaction SCFs
P_TERMINAL_CAPABILITIES	The name of the Terminal Capabilities SCF
P_USER_LOCATION	The name of the User Location SCF
P_USER_LOCATION_CAMEL	The name of the Network User Location SCF
P_USER_LOCATION_EMERGENCY	The name of the User Location Emergency SCF
P_USER_STATUS	The name of the User Status SCF
P_EXTENDED_USER_STATUS	The name of Extended User Status SCF
P_DATA_SESSION_CONTROL	The name of the Data Session Control SCF
P_GENERIC_MESSAGING	The name of the Generic Messaging SCF
P_CONNECTIVITY_MANAGER	The name of the Connectivity Manager SCF
P_CHARGING	The name of the Charging SCF
P_ACCOUNT_MANAGEMENT	The name of the Account Management SCF
P_POLICY_MANAGEMENT	The name of the Policy Management provisioning SCF
P_PAM_ACCESS	The name of PAM presentity SCF
P_PAM_EVENT_MANAGEMENT	The name of PAM watcher SCF
P_PAM_PROVISIONING	The name of PAM provisioning SCF

End of Change in Clause 10.1.30
End of Document

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) N5-030579
Meeting #25, Bangkok, THAILAND, 27 - 31 October 2003

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-03**	**CR** **099**	⌘ **rev**	**-**	⌘ Current version:	**5.4.0** ⌘

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

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

Title:	⌘	Add P_USER_BINDING to TpServiceTypeName		
Source:	⌘	CN5 (Telcordia & NTT)		
Work item code:	⌘	OSA3	***Date:*** ⌘	17/10/2003
Category:	⌘	**B**	***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 Framework lists all SCFs part of OSA/Parlay. The User Binding SCF is new and needs to be listed.
Summary of change:	⌘	Added P_USER_BINDING to TpServiceTypeName
Consequences if not approved:	⌘	The Parlay/OSA specifications will not be alligned.

Clauses affected:	⌘	10.1.30				
		Y	**N**			
Other specs affected:	⌘		**X**	Other core specifications	⌘	
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

How to create CRs using this form:

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

Change in Clause 10.1.3

10.1.30 TpServiceTypeName

This data type is identical to a TpString, and is defined as a string of characters that uniquely identifies the type of an SCF interface. Other Network operator specific capabilities may also be used, but should be preceded by the string "SP_". The following values are defined.

Character String Value	Description
NULL	An empty (NULL) string indicates no SCF name
P_GENERIC_CALL_CONTROL	The name of the Generic Call Control SCF
P_MULTI_PARTY_CALL_CONTROL	The name of the MultiParty Call Control SCF
P_MULTI_MEDIA_CALL_CONTROL	The name of the MultiMedia Call Control SCF
P_CONFERENCE_CALL_CONTROL	The name of the Conference Call Control SCF
P_USER_INTERACTION	The name of the User Interaction SCFs
P_TERMINAL_CAPABILITIES	The name of the Terminal Capabilities SCF
P_USER_BINDING	The name of the User Binding SCF
P_USER_LOCATION	The name of the User Location SCF
P_USER_LOCATION_CAMEL	The name of the Network User Location SCF
P_USER_LOCATION_EMERGENCY	The name of the User Location Emergency SCF
P_USER_STATUS	The name of the User Status SCF
P_DATA_SESSION_CONTROL	The name of the Data Session Control SCF
P_GENERIC_MESSAGING	The name of the Generic Messaging SCF
P_CONNECTIVITY_MANAGER	The name of the Connectivity Manager SCF
P_CHARGING	The name of the Charging SCF
P_ACCOUNT_MANAGEMENT	The name of the Account Management SCF
P_POLICY_MANAGEMENT	The name of the Policy Management provisioning SCF
P_PAM_ACCESS	The name of PAM presentity SCF
P_PAM_EVENT_MANAGEMENT	The name of PAM watcher SCF
P_PAM_PROVISIONING	The name of PAM provisioning SCF

End of Change in Clause 12.3
End of Document

Annex D (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2003	CN_19	NP-030028	071	--	Correction of status of methods to interfaces in clause 6.3	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	075	--	Adding the appAvailStatusInd() and svcAvailStatusInd() methods	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	076	--	Remove race condition in signServiceAgreement	5.1.0	5.2.0
Mar 2003	CN_19	NP-030028	077	--	Change reference to deprecated method "authenticate" in TpAuthMechanism to "challenge"	5.1.0	5.2.0
Jun 2003	CN_20	NP-030237	079	--	Correction to TpEncryptionCapability to correct support for Triple-DES	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	081	--	Correction of the Framework Service Instance Lifecycle Manager Sequence Diagram	5.2.0	5.3.0
Jun 2003	CN_20	NP-030237	083	--	Correction of the use of TpDomainID in Framework initiateAuthentication method	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	085	--	Correction to Java Realisation Annex	5.3.0	5.4.0

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-030631**
Meeting #25, Bangkok, Thailand, 27 – 31 October 2003

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-03**	**CR**	**100**	⌘ **rev**	**-**	⌘	Current version:	**5.4.0** ⌘

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

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

Title: ⌘	Modify Framework Availability Indication in Fault Management		
Source: ⌘	CN5 (AePONA – Eamonn Murray)		
Work item code: ⌘	OSA3	***Date:*** ⌘	31/10/2003
Category: ⌘	**C**	***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 Fault Management Interfaces have been revised to replace a svcUnavailableInd with a svcAvailStatusInd. This has been done to ensure that when a service becomes available again that an indication can be provided. The equivalent behaviour cannot be supported for the Framework itself, therefore though it is possible for the framework to indicate that it is no longer available, it is not possible for the framework to indicate when it becomes available again. The Framework does include a fault report and recovery mechanism, however this represents only a subset of the functionality supported by the availability indication, excluding indication of overload conditions and software upgrade.
Summary of change: ⌘	Deprecate the current fwUnavailableInd, fwFaultReportInd and fwFaultRecoveryInd methods from the existing Fault Management interfaces and replace with a fwAvailStatusInd.
Consequences if not approved: ⌘	OSA Fault management functionality for the Framework is not aligned with the fault management capability of other SCFs. The Framework functionality is therefore incomplete.

Clauses affected: ⌘	7.2, 7.3.3.1, 7.4.3.4, 8.2, 8.3.4.2, 8.4.4.2, 10.4			
	Y	**N**		
Other specs affected: ⌘		**X**	Other core specifications	⌘
		X	Test specifications	
		X	O&M Specifications	
Other comments: ⌘				

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

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

7.2 Class Diagrams

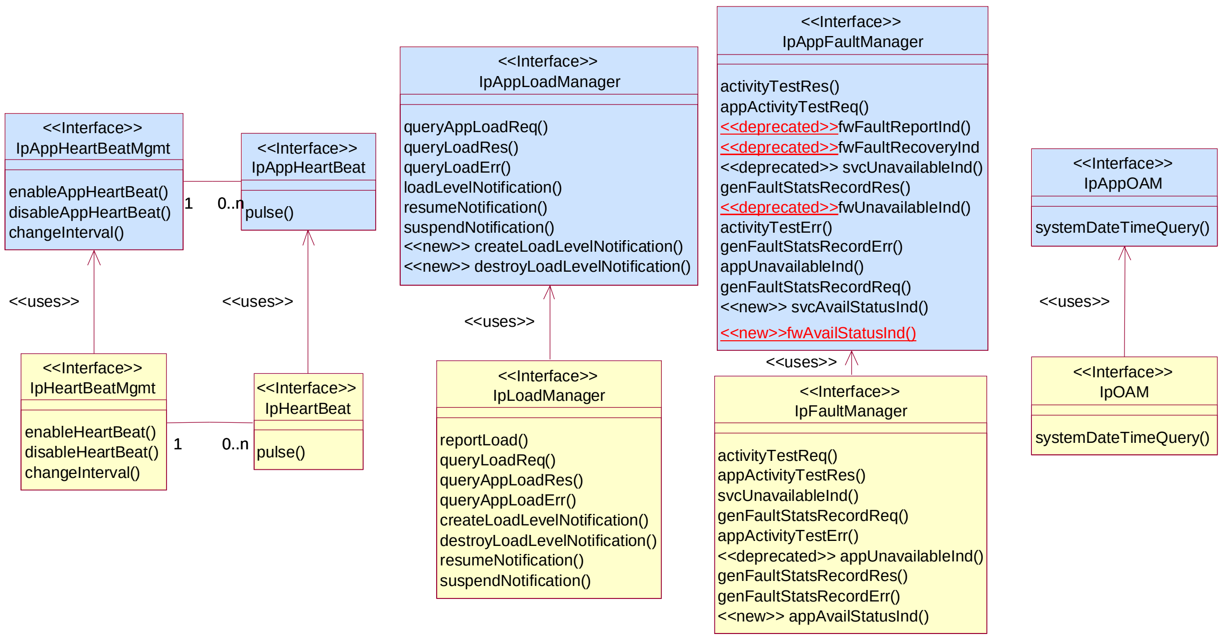


Figure: Integrity Management Package Overview

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

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

7.3.3.1 Interface Class IpAppFaultManager

Inherits from: IpInterface.

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

<<Interface>>

IpAppFaultManager

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

appActivityTestReq (activityTestID : in TpActivityTestID) : void

<<deprecated>>fwFaultReportInd (fault : in TpInterfaceFault) : void

<<deprecated>>fwFaultRecoveryInd (fault : in TpInterfaceFault) : void

<<deprecated>> svcUnavailableInd (serviceID : in TpServiceID, reason : in TpSvcUnavailReason) : void

genFaultStatsRecordRes (faultStatistics : in TpFaultStatsRecord, serviceIDs : in TpServiceIDList) : void

<<deprecated>>fwUnavailableInd (reason : in TpFwUnavailReason) : void

activityTestErr (activityTestID : in TpActivityTestID) : void

genFaultStatsRecordErr (faultStatisticsError : in TpFaultStatisticsError, serviceIDs : in TpServiceIDList) : void

appUnavailableInd (serviceID : in TpServiceID) : void

genFaultStatsRecordReq (timePeriod : in TpTimeInterval) : void

<<new>> svcAvailStatusInd (serviceID : in TpServiceID, reason : in TpSvcAvailStatusReason) : void

<<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 genFaultStatsRecordRes()

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

Parameters

`faultStatistics : in TpFaultStatsRecord`

The fault statistics record.

`serviceIDs : in TpServiceIDList`

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

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 genFaultStatsRecordErr()

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

Parameters

`faultStatisticsError : in TpFaultStatisticsError`

The fault statistics error.

`serviceIDs : in TpServiceIDList`

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

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 genFaultStatsRecordReq()

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

Parameters

`timePeriod : in TpTimeInterval`

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

7.3.3.1.12 Method <<new>> svcAvailStatusInd()

The framework invokes this method to inform the client application about the Service instance availability status, i.e. that it can no longer use its instance of the indicated service according to the reason parameter but as well information when the Service Instance becomes available again. On receipt of this request, the client application either acts to reset its use of the specified service (using the normal mechanisms, such as the discovery and authentication interfaces, to stop use of this service instance and begin use of a different service instance). The client application can also wait for the problem to be solved and just stop the usage of the service instance until the svcAvailStatusInd() is called again with the reason ~~SERVICE~~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>> 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.

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

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

7.4.3.4 State Transition Diagrams for IpFaultManager

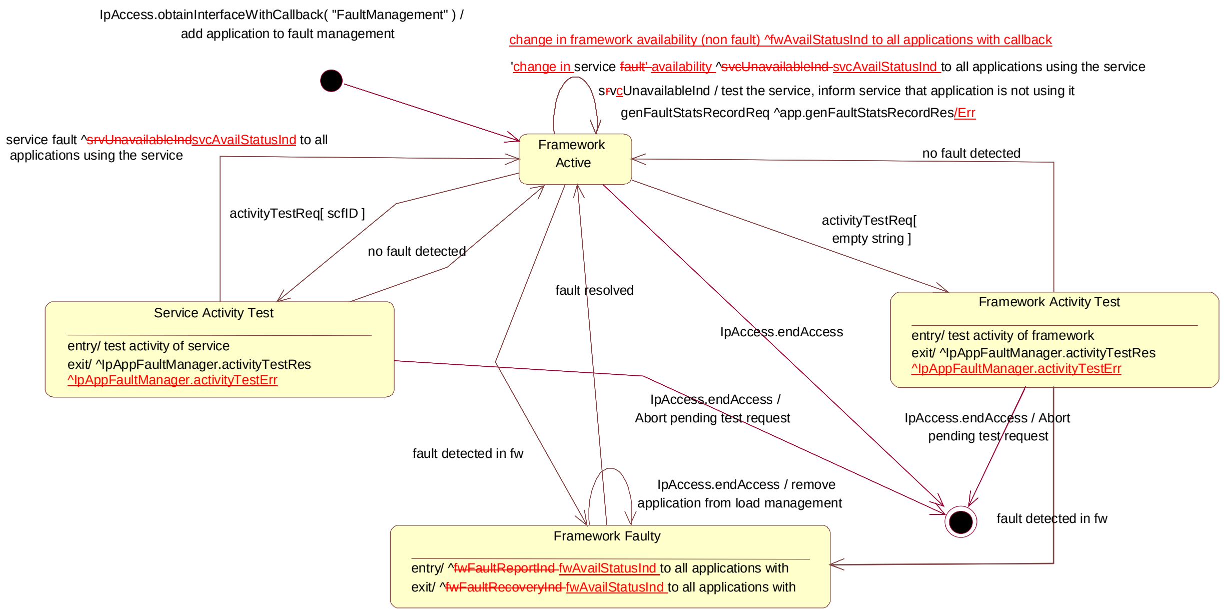

IpAccess.obtainInterfaceWithCallback("FaultManagement") / add application to fault management
change in framework availability (non fault) ^fwAvailStatusInd to all applications with callback
'change in service ~~fault'~~ availability ^~~svcUnavailableInd~~ svcAvailStatusInd to all applications using the service
s~~r~~vcUnavailableInd / test the service, inform service that application is not using it
genFaultStatsRecordReq ^app.genFaultStatsRecordRes/Err
service fault ^~~srvUnavailableInd~~svcAvailStatusInd to all applications using the service
Framework Active
no fault detected
activityTestReq[scfID]
no fault detected
activityTestReq[empty string]
fault resolved
IpAccess.endAccess
Service Activity Test
entry/ test activity of service
exit/ ^IpAppFaultManager.activityTestRes
^IpAppFaultManager.activityTestErr
Framework Activity Test
entry/ test activity of framework
exit/ ^IpAppFaultManager.activityTestRes
^IpAppFaultManager.activityTestErr
IpAccess.endAccess / Abort pending test request
IpAccess.endAccess / Abort pending test request
fault detected in fw
IpAccess.endAccess / remove application from load management
fault detected in fw
Framework Faulty
entry/ ^~~fwFaultReportInd~~ fwAvailStatusInd to all applications with
exit/ ^~~fwFaultRecoveryInd~~ fwAvailStatusInd to all applications with

Figure : State Transition Diagram for IpFaultManager

7.4.3.4.1 Framework Active State

This is the normal state of the framework, which is fully functional and able to handle requests from both applications and services capability features.

7.4.3.4.2 Framework Faulty State

In this state, the framework has detected an internal problem with itself such that application and services capability features cannot communicate with it anymore; attempts to invoke any methods that belong to any SCFs of the framework return an error. If the framework ever recovers, applications with fault management callbacks will be notified via a ~~fwFaultRecoveryInd~~ fwAvailStatusInd message.

7.4.3.4.3 Framework Activity Test State

In this state, the framework is performing self-diagnostic test. If a problem is diagnosed, all applications with fault management callbacks are notified through a ~~fwFaultReportInd~~ fwAvailStatusInd message.

7.4.3.4.4 Service Activity Test State

In this state, the framework is performing a test on one service capability feature. If the SCF is faulty, applications with fault management callbacks are notified accordingly through a ~~svcUnavailableInd~~ svcAvailStatusInd message.

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

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

8.2 Class Diagrams

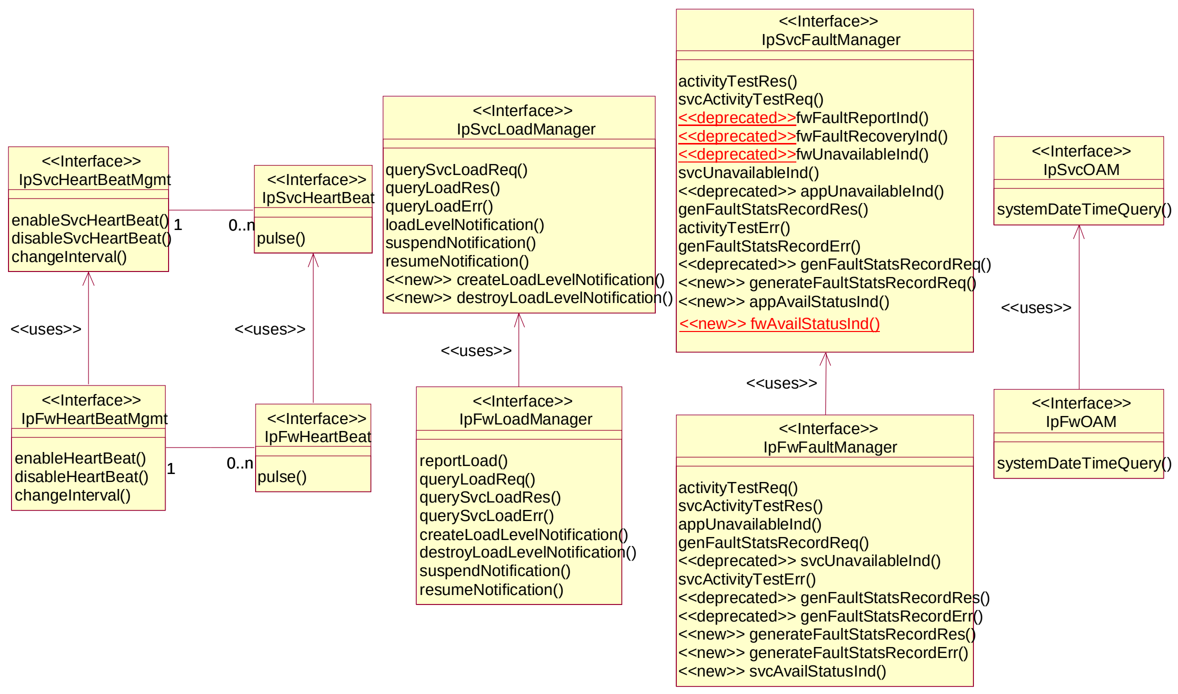


Figure: Integrity Management Package Overview

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

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

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.genFaultStatsRecordReq(), it shall implement genFaultStatsRecordRes() and genFaultStatsRecordErr() in this interface.

<<Interface>>

IpSvcFaultManager

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

svcActivityTestReq (activityTestID : in TpActivityTestID) : void

<<deprecated>>fwFaultReportInd (fault : in TpInterfaceFault) : void

<<deprecated>>fwFaultRecoveryInd (fault : in TpInterfaceFault) : void

<<deprecated>>fwUnavailableInd (reason : in TpFwUnavailReason) : void

svcUnavailableInd () : void

<<deprecated>> appUnavailableInd () : void

genFaultStatsRecordRes (faultStatistics : in TpFaultStatsRecord, recordSubject : in TpSubjectType) : void

activityTestErr (activityTestID : in TpActivityTestID) : void

genFaultStatsRecordErr (faultStatisticsError : in TpFaultStatisticsError, recordSubject : in TpSubjectType) : void

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

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

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

<<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 genFaultStatsRecordRes()

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 genFaultStatsRecordErr()

This method is used by the framework to indicate an error fulfilling the request to provide fault statistics, in response to a genFaultStatsRecordReq method invocation on the 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 <<new>> generateFaultStatsRecordReq()

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>> 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.

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

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

8.4.4.2 State Transition Diagrams for IpFWFaultManager

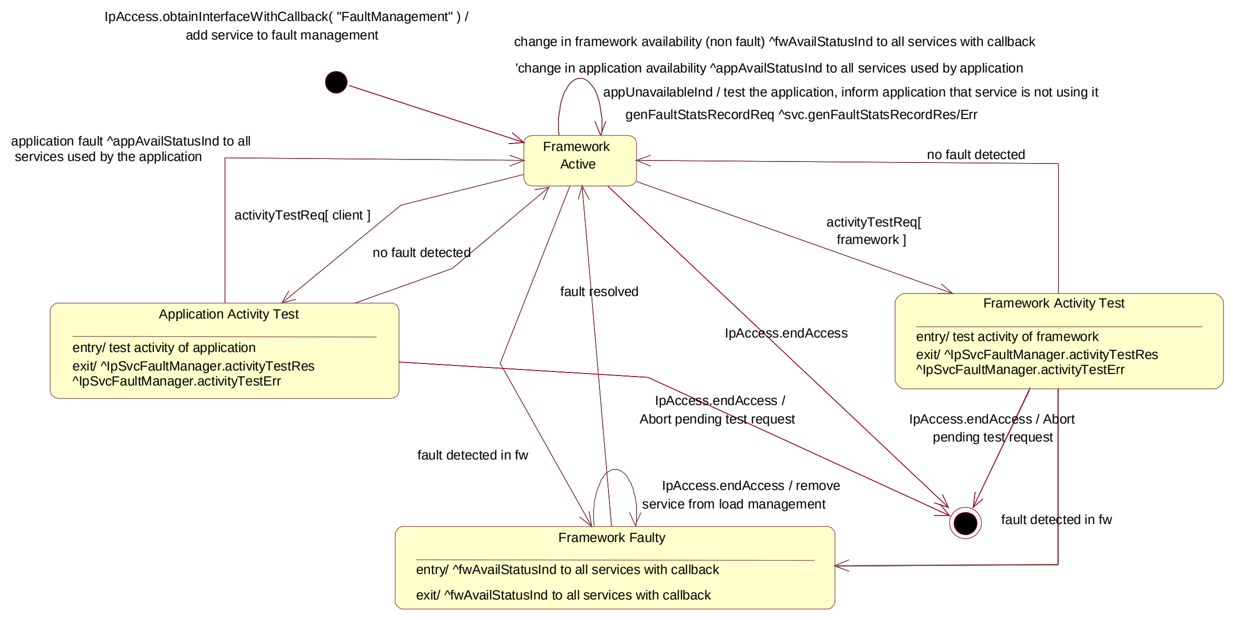

IpAccess.obtainInterfaceWithCallback("FaultManagement") / add service to fault management
change in framework availability (non fault) ^fwAvailStatusInd to all services with callback
'change in application availability ^appAvailStatusInd to all services used by application
appUnavailableInd / test the application, inform application that service is not using it
genFaultStatsRecordReq ^svc.genFaultStatsRecordRes/Err
application fault ^appAvailStatusInd to all services used by the application
Framework Active
no fault detected
activityTestReq[client]
no fault detected
activityTestReq[framework]
fault resolved
Application Activity Test
entry/ test activity of application
exit/ ^IpSvcFaultManager.activityTestRes ^IpSvcFaultManager.activityTestErr
IpAccess.endAccess
Framework Activity Test
entry/ test activity of framework
exit/ ^IpSvcFaultManager.activityTestRes ^IpSvcFaultManager.activityTestErr
IpAccess.endAccess / Abort pending test request
IpAccess.endAccess / Abort pending test request
fault detected in fw
IpAccess.endAccess / remove service from load management
fault detected in fw
Framework Faulty
entry/ ^fwAvailStatusInd to all services with callback
exit/ ^fwAvailStatusInd to all services with callback

Figure : State Transition Diagram for IpFWFaultManager

8.4.4.2.1 Framework Active State

This is the normal state of the framework, which is fully functional and able to handle requests from both applications and service capability features.

8.4.4.2.2 Framework Faulty State

In this state, the framework has detected an internal problem with itself such that application and service capability features cannot communicate with it anymore; attempts to invoke any methods that belong to any SCFs of the framework return an error. If the framework ever recovers, services with fault management callbacks will be notified via a fwAvailStatusInd message.

8.4.4.2.3 Framework Activity Test State

In this state, the framework is performing a self-diagnostic test. If a problem is diagnosed, all services with fault management callbacks are notified through an fwAvailStatusInd message.

8.4.4.2.4 Application Activity Test State

In this state, the framework is performing a test on one client application. If the application is faulty, services that are used by the application and that have provided fault management callbacks are notified accordingly through an appAvailStatusInd message.

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

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

10.4 Integrity Management Data Definitions

10.4.1 TpActivityTestRes

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

10.4.2 TpFaultStatsRecord

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

Sequence Element Name	Sequence Element Type
Period	TpTimeInterval
FaultStatsSet	TpFaultStatsSet

10.4.3 TpFaultStats

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

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

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

10.4.4 TpFaultStatisticsError

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

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

10.4.5 TpFaultStatsSet

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

10.4.6 TpActivityTestID

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

10.4.7 TpInterfaceFault

Defines the cause of the interface fault detected.

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

10.4.8 TpSvcUnavailReason

Defines the reason why a SCF is unavailable.

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

10.4.9 TpFwUnavailReason

Defines the reason why the Framework is unavailable.

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

10.4.10 TpLoadLevel

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

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

10.4.11 TpLoadThreshold

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

Sequence Element Name	Sequence Element Type
LoadThreshold	TpFloat

10.4.12 TpLoadInitVal

Defines the Sequence of Data Elements that specify the pair of load level and associated load threshold value.

Sequence Element Name	Sequence Element Type
LoadLevel	TpLoadLevel
LoadThreshold	TpLoadThreshold

10.4.13 TpLoadPolicy

Defines the load balancing policy.

Sequence Element Name	Sequence Element Type
LoadPolicy	TpString

10.4.14 TpLoadStatistic

Defines the Sequence of Data Elements that represents a load statistic record for a specific entity (i.e. Framework, service or application) at a specific date and time.

Sequence Element Name	Sequence Element Type
LoadStatisticEntityID	TpLoadStatisticEntityID
TimeStamp	TpDateAndTime
LoadStatisticInfo	TpLoadStatisticInfo

10.4.15 TpLoadStatisticList

Defines a Numbered List of Data Elements of type TpLoadStatistic.

10.4.16 TpLoadStatisticData

Defines the Sequence of Data Elements that represents load statistic information

Sequence Element Name	Sequence Element Type
LoadValue (see Note)	TpFloat
LoadLevel	TpLoadLevel
NOTE: LoadValue is expressed as a percentage.	

10.4.17 TpLoadStatisticEntityID

Defines the Tagged Choice of Data Elements that specify the type of entity (i.e. service, application or Framework) providing load statistics.

	Tag Element Type	
	TpLoadStatisticEntityType	

Tag Element Value	Choice Element Type	Choice Element Name
P_LOAD_STATISTICS_FW_TYPE	TpFwID	FrameworkID
P_LOAD_STATISTICS_SVC_TYPE	TpServiceID	ServiceID
P_LOAD_STATISTICS_APP_TYPE	TpClientAppID	ClientAppID

10.4.18 TpLoadStatisticEntityType

Defines the type of entity (i.e. service, application or Framework) supplying load statistics.

Name	Value	Description
P_LOAD_STATISTICS_FW_TYPE	0	Framework-type load statistics
P_LOAD_STATISTICS_SVC_TYPE	1	Service-type load statistics
P_LOAD_STATISTICS_APP_TYPE	2	Application-type load statistics

10.4.19 TpLoadStatisticInfo

Defines the Tagged Choice of Data Elements that specify the type of load statistic information (i.e. valid or invalid).

	Tag Element Type	
	TpLoadStatisticInfoType	

Tag Element Value	Choice Element Type	Choice Element Name
P_LOAD_STATISTICS_VALID	TpLoadStatisticData	LoadStatisticData
P_LOAD_STATISTICS_INVALID	TpLoadStatisticError	LoadStatisticError

10.4.20 TpLoadStatisticInfoType

Defines the type of load statistic information (i.e. valid or invalid).

Name	Value	Description
P_LOAD_STATISTICS_VALID	0	Valid load statistics
P_LOAD_STATISTICS_INVALID	1	Invalid load statistics

10.4.21 TpLoadStatisticError

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

Name	Value	Description
P_LOAD_INFO_ERROR_UNDEFINED	0	Undefined error
P_LOAD_INFO_UNAVAILABLE	1	Load statistics unavailable

10.4.22 TpSvcAvailStatusReason

Defines the reason detailing the change in status of Service availability~~Defines the reason why a SCF is unavailable~~.

Name	Value	Description
SVC_UNAVAILABLE_UNDEFINED	0	Undefined
SVC_UNAVAILABLE_LOCAL_FAILURE	1	The Local API software or hardware has failed. Normally take longer time to correct
SVC_UNAVAILABLE_GATEWAY_FAILURE	2	The gateway API software or hardware has failed Normally take longer time to correct
SVC_UNAVAILABLE_OVERLOADED	3	The SCF is fully overloaded Normally a temporary problem
SVC_UNAVAILABLE_CLOSED	4	The SCF has closed itself (e.g. to protect from fraud or malicious attack) Normally take longer time to correct
SVC_UNAVAILABLE_NO_RESPONSE	5	The Framework has detected that the service has failed: e.g. non-response from an activity test, failure to return heartbeats
SVC_UNAVAILABLE_SW_UPGRADE	6	The Service is unavailable due to SW upgrade or other similar maintenance Normally a temporary problem
SVC_AVAILABLE	7	The Service has become available again

10.4.23 TpAppAvailStatusReason

Defines the reason detailing the change in status of Application availability~~Defines the reason why the Application is unavailable~~.

Name	Value	Description
APP_UNAVAILABLE_UNDEFINED	0	Undefined
APP_UNAVAILABLE_LOCAL_FAILURE	1	A local failure in the Application has been detected Normally take longer time to correct
APP_UNAVAILABLE_REMOTE_FAILURE	2	A remote failure to the application has been detected, e.g. a database is not working Normally take longer time to correct
APP_UNAVAILABLE_OVERLOADED	3	The Application is fully overloaded Often a temporary problem
APP_UNAVAILABLE_CLOSED	4	The Application has closed itself (e.g. to protect from fraud or malicious attack) Normally take longer time to correct
APP_UNAVAILABLE_NO_RESPONSE	5	The Framework has detected that the application has failed: e.g. non-response from an activity test, failure to return heartbeats
APP_UNAVAILABLE_SW_UPGRADE	6	The Application is unavailable due to SW upgrade or other similar maintenance Often a temporary problem
APP_AVAILABLE	7	The Application has become available

10.4.24 TpFwAvailStatusReason

Defines the reason detailing the change in status of Framework availability.

Name	Value	Description
FRAMEWORK_UNAVAILABLE_UNDEFINED	0	Undefined
FRAMEWORK_UNAVAILABLE_LOCAL_FAILURE	1	A local failure in the Framework has been detected Normally take longer time to correct
FRAMEWORK_UNAVAILABLE_REMOTE_FAILURE	2	A remote failure to the Framework has been detected, e.g. a database is not working Normally take longer time to correct
FRAMEWORK_UNAVAILABLE_OVERLOADED	3	The Framework is fully overloaded Often a temporary problem
FRAMEWORK_UNAVAILABLE_CLOSED	4	The Framework has closed itself (e.g. to protect from fraud or malicious attack) Normally take longer time to correct
FRAMEWORK_UNAVAILABLE_PROTOCOL_FAILURE	5	The Framework has detected that the protocol used between client and framework has failed
FRAMEWORK_UNAVAILABLE_SW_UPGRADE	6	The Framework is unavailable due to SW upgrade or other similar maintenance Often a temporary problem
FRAMEWORK_AVAILABLE	7	The Framework has become available

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

Annex D (informative): Change history

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

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-030409**
Meeting #24, San Francisco, CA, USA, 14 - 18 July 2003

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-05 CR 043** ⌘ **rev** **-** ⌘ Current version: **5.4.0** ⌘

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

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

Title:	⌘	Improve User Interaction message management functions		
Source:	⌘	CN5 (scottjb@us.ibm.com)		
Work item code:	⌘	OSA3	***Date:*** ⌘	18/07/2003
Category:	⌘	**B**	***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 OSA User Interaction API provides the capability to record and playback messages, but it does not provide a mechanism to retrieve the message content by the application, or provide a mechanism to set the message content by the application. These features are necessary to enable the applications to utilize the content of the messages in a meaningful way for both administration/management and for interaction with enhanced services or users (mid-call).

The User Interaction service should be functional for both administrative provisioning of the messages of the gateway, but also functional for application interaction with the user for the purposes of enhanced services and enterprise applications.

When recording a message through the IpUICall interface, the application can play it back or delete it, but currently can not retrieve it from the gateway. The application may want to record the user's voice for a credit card authorization or such and then store the recording in its own database. There are many reasons why the application may need to retrieve the waveform data. Clearly, this method is intended for low-frequency usage for performance and bandwidth reasons, however it is still necessary.

Additionally, The application also does not have a mechanism to assign a messageID to a new User Interaction message that is provided by the application, thereby adding a message to the provisioned set of messages.

Additionally, the currently supported deleteMessageReq() is provided on the IpUICall interface, which can only be used in conjunction with a Call/CallLeg session, however, this is a desireable administrative function that may not be associated with a particular Call/CallLeg session. As with the new methods described above, they may be desireable without a call session.

If this functionality is not possible with the infrastructure of a particular switch, it should still be included in the specification for completeness, because from the application perspective this is an important feature, and in time the necessary functions could be integrated with the core network.

Summary of change: ⌘

New methods are necessary to allow the application to set and retrieve the recorded audio data that is used for a message. These methods allow the application to retrieve the content of a message recorded by a user, or set the content of a message that can be played by the application (without having the administrator of the gateway have to customize the configuration.) A new IpUIAdminManager SCF interface is proposed so that it is clear that it is not attached to a Call/CallLeg session or a TpAddress.

The following new methods are proposed for IpUIAdminManager:

```
    TpAssignmentID getMessageReq (TpSessionID uiSessionID,
                                  TpInt32 messageID);

    TpAssignmentID putMessageReq (TpSessionID uiSessionID,
                                  TpUIInfo msg);

    TpAssignmentID deleteMessageReq (TpSessionID uiSessionID,
                                  TpInt32 messageID);
```

Along with their responses in IpAppUIAdminManager:

```
    Void getMessageRes (TpSessionID uiSessionID, TpAssignmentID assignID,
                        TpUIInfo info);

    Void getMessageErr (TpSessionID uiSessionId, TpAssignmentID assignID,
                        TpUIError err);

    Void putMessageRes (TpSessionID uiSessionID, TpAssignmentID assignID,
                        TpInt32 messageID);

    Void putMessageErr (TpSessionID uiSessionId, TpAssignmentID assignID,
                        TpUIError err);

    Void deleteMessageRes (TpSessionID uiSessionID,
                           TpAssignmentID assignID);

    Void deleteMessageErr (TpSessionID uiSessionId, TpAssignmentID assignID,
                        TpUIError err);
```

The User Interaction service will utilize the application context to ensure that one application does interfere with the messages of another application, such as playing, retrieving or deleting them. Also, the application can not delete the shared messages that are pre-provisioned on the OSA Gateway, but can play or retrieve them.

The IpUICall interface is enhanced with getMessageReq() to allow call-based retrieval of recorded data. Either the IpUICall or IpUIAdminManager can manage the messages.

This is also a correction because the recordMessageReq() processing is not useful in the current design because the application can not access the recorded data.

See the associated change in N5-030410.

Changes derived from:
ftp://ftp.3gpp.org/specs/2003-06/Rel-5/29_series/

Consequences if not approved:	⌘	The usefulness of the IpUICall interfaces by an application is limited if the application can not retrieve the information provided by the user, or can not dynamically set the messages to be played by the application.

Clauses affected:	⌘	4,5,6,8				
		Y	**N**			
Other specs affected:	⌘	**X**		Other core specifications	⌘	Rel-6 29.198-03
			X	Test specifications		
			X	O&M Specifications		

Other comments:	⌘	Related Rel-6 29.198-03 CR in N5-030410.

How to create CRs using this form:

Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm.

4 Generic and Call User Interaction and Administration SCF

4.1 Generic and Call User Interaction SCF

The Generic User Interaction service capability feature is used by applications to interact with end users. It consists of three interfaces:

1) User Interaction Manager, containing management functions for User Interaction related issues;

2) Generic User Interaction, containing methods to interact with an end-user.

3) Call User Interaction, containing methods to interact with an end-user engaged in a call.

The Generic User Interaction service capability feature is described in terms of the methods in the Generic User Interaction interfaces.

The following table gives an overview of the Generic User Interaction methods and to which interfaces these methods belong.

Table 1: Overview of Generic User Interaction interfaces and their methods

User Interaction Manager	Generic User Interaction
createUI	sendInfoReq
createUICall	sendInfoRes
createNotification	sendInfoErr
destroyUINotification	sendInfoAndCollectReq
reportNotification	sendInfoAndCollectRes
userInteractionAborted	sendInfoAndCollectErr
userInteractionNotificationInterrupted	Release
userInteractionNotificationContinued	userInteractionFaultDetected
changeNotification	setOriginatingAddress
getNotification	getOriginatingAddress
enableNotifications	
disableNotifications	

The following table gives an overview of the Call User Interaction methods and to which interfaces these methods belong.

Table 2: Overview of Call User Interaction interfaces and their methods

User Interaction Manager	Call User Interaction
As defined for the Generic User Interaction SCF	Inherits from Generic User Interaction and adds:
	recordMessageReq
	recordMessageRes
	recordMessageErr
	deleteMessageReq
	deleteMessageRes
	deleteMessageErr
	abortActionReq
	abortActionRes
	abortActionErr
	getMessageReq
	getMessageRes
	getMessageErr

The IpUI Interface provides functions to send information to, or gather information from the user, i.e. this interface allows applications to send SMS and USSD messages. An application can use this interface independently of other

SCFs. The IpUICall Interface provides functions to send information to, or gather information from the user (or call party) attached to a call.

4.2 Generic User Interaction Administration SCF

The Generic User Interaction Administration service capability feature is used by application to interact with the service to manage the user announcement and recorded messages. It consists of one interface:

1) User Interaction Administration Manager, containing message management functions for User Interaction.

Table 3: Overview of Call User Interaction Administration interfaces and their methods

User Interaction Administration Manager
getMessageReq
putMessageReq
deleteMessageReq

4.3 Generic User Interaction SCF Design Aspects

The following clauses describe each aspect of the Generic User Interaction Service and Generic User Interaction Administration Service Capability Feature (SCF).

The order is as follows:

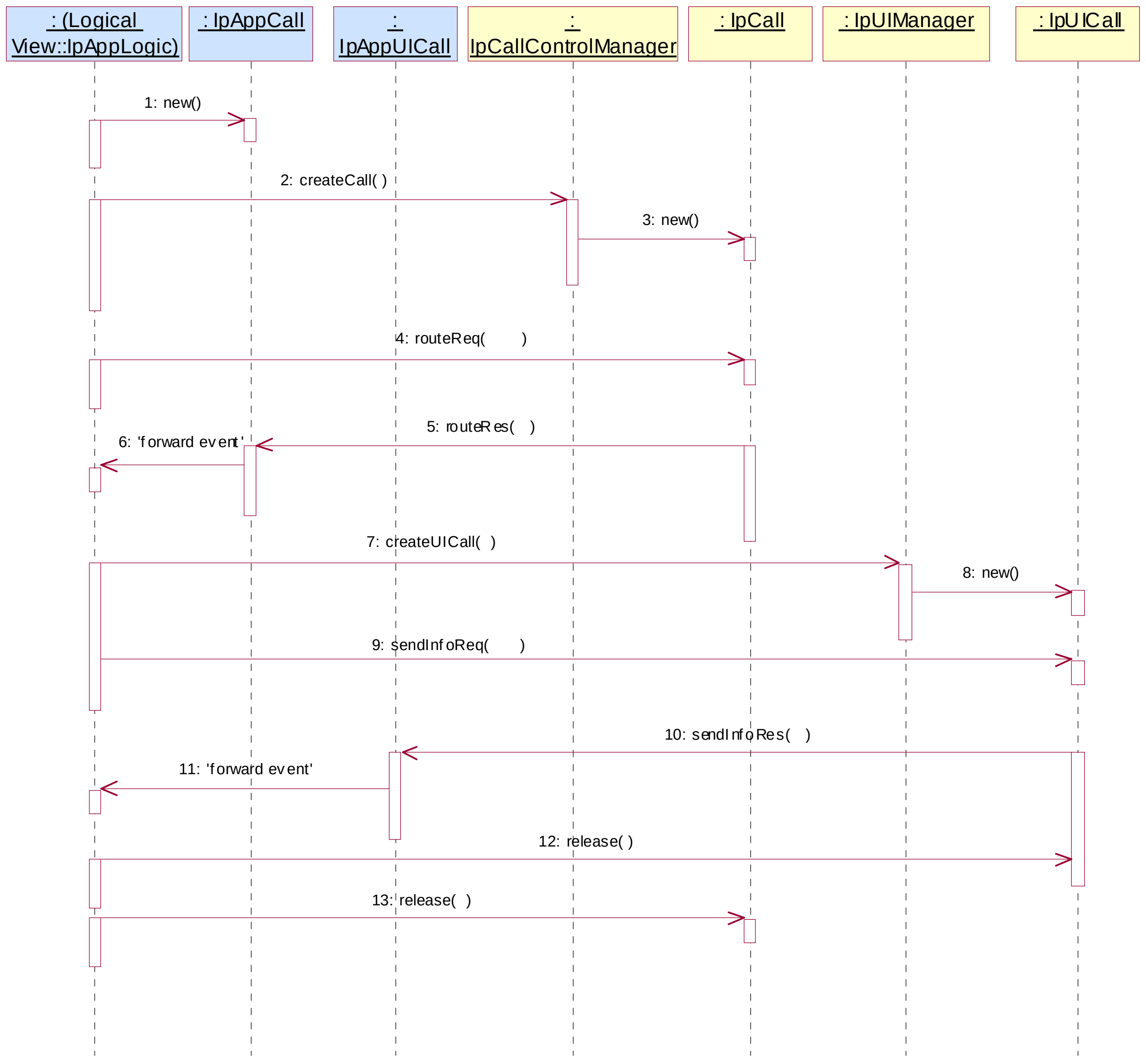
- The Sequence diagrams give the reader a practical idea of how each of the SCFs is implemented.
- The Class relationships clause show how each of the interfaces applicable to the SCF, relate to one another.
- The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part. This clause also includes Call User interaction.
- The State Transition Diagrams (STD) show the transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.
- The Data Definitions clause show a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part ES 202 915-2.

5 Sequence Diagrams

5.1 Generic and Call User Interaction Sequence Diagrams

5.1.1 Alarm Call

The following sequence diagram shows a 'reminder message', in the form of an alarm, being delivered to a customer as a result of a trigger from an application. Typically, the application would be set to trigger at a certain time, however, the application could also trigger on events.



1: This message is used to create an object implementing the IpAppCall interface.

2: This message requests the object implementing the IpCallControlManager interface to create an object implementing the IpCall interface.

3: Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) are met it is created.

4: This message instructs the object implementing the IpCall interface to route the call to the customer destined to receive the 'reminder message'

5: This message passes the result of the call being answered to its callback object.

6: This message is used to forward the previous message to the IpAppLogic.

7: The application requests a new UICall object that is associated with the call object.

8: Assuming all criteria are met, a new UICall object is created by the service.

9: This message instructs the object implementing the IpUICall interface to send the alarm to the customer's call.

10: When the announcement ends this is reported to the call back interface.

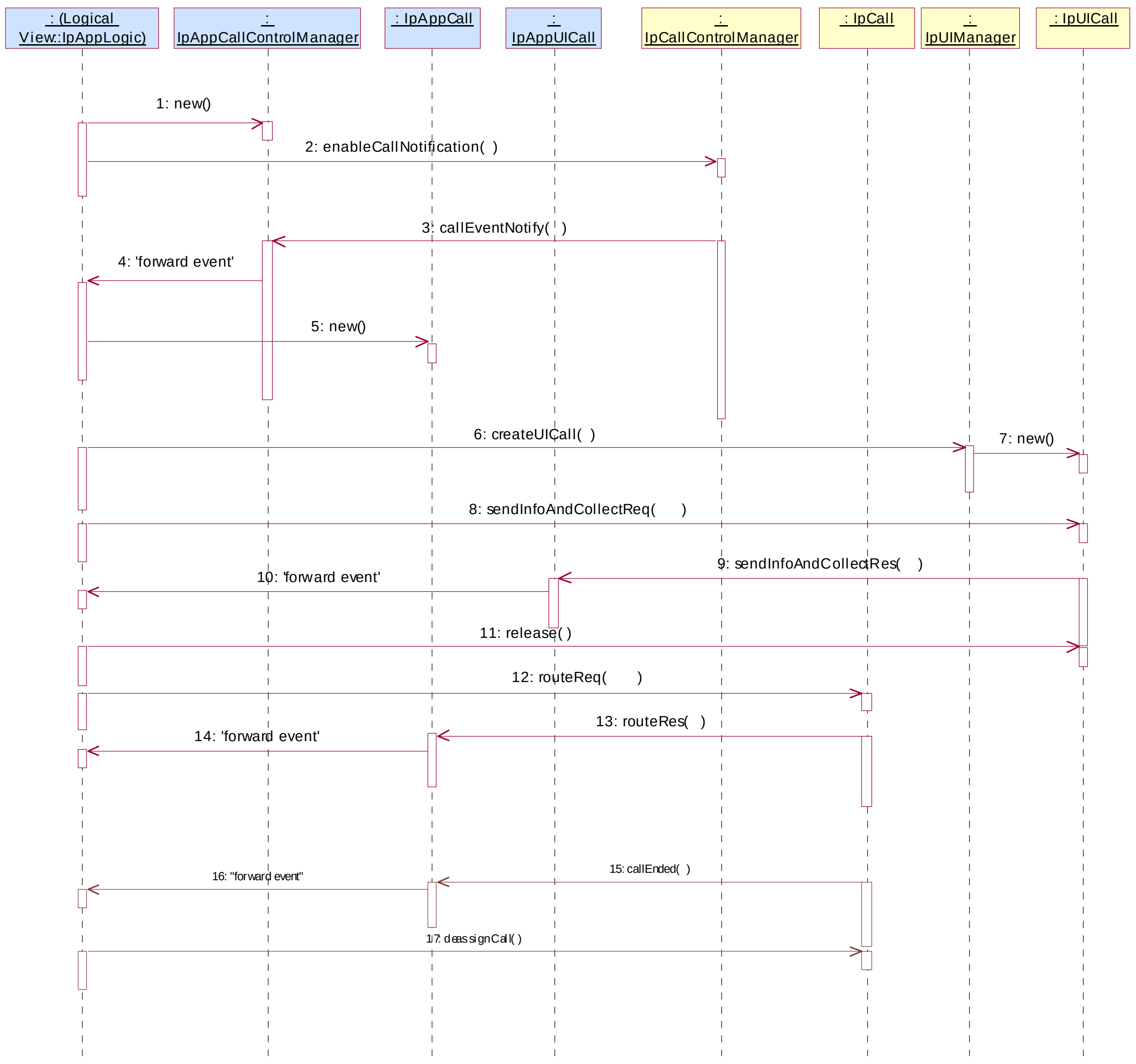
11: The event is forwarded to the application logic.

12: The application releases the UICall object, since no further announcements are required. Alternatively, the application could have indicated P_FINAL_REQUEST in the sendInfoReq in which case the UICall object would have been implicitly released after the announcement was played.

13: The application releases the call and all associated parties.

5.1.2 Call Barring 1

The following sequence diagram shows a call barring service, initiated as a result of a prearranged event being received by the call control service. Before the call is routed to the destination number, the calling party is asked for a PIN code. The code is accepted and the call is routed to the original called party.



1: This message is used by the application to create an object implementing the IpAppCallControlManager interface.

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a call barring service, it is likely that all new call events destined for a particular address or address range prompted for a password before the call is allowed to progress. When a new call, that matches the event criteria set, arrives, a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) are met, other messages (not shown) are used to create the call and associated call leg object.

3: This message is used to pass the new call event to the object implementing the IpAppCallControlManager interface.

4: This message is used to forward the previous message to the IpAppLogic.

5: This message is used by the application to create an object implementing the IpAppCall interface. The reference to this object is passed back to the object implementing the IpCallControlManager using the return parameter of the callEventNotify.

6: This message is used to create a new UICall object. The reference to the call object is given when creating the UICall.

7: Provided all the criteria are fulfilled, a new UICall object is created.

8: The call barring service dialogue is invoked.

9: The result of the dialogue, which in this case is the PIN code, is returned to its callback object.

10: This message is used to forward the previous message to the IpAppLogic.

11: This message releases the UICall object.

12: Assuming the correct PIN is entered, the call is forward routed to the destination party.

13: This message passes the result of the call being answered to its callback object.

14: This message is used to forward the previous message to the IpAppLogic

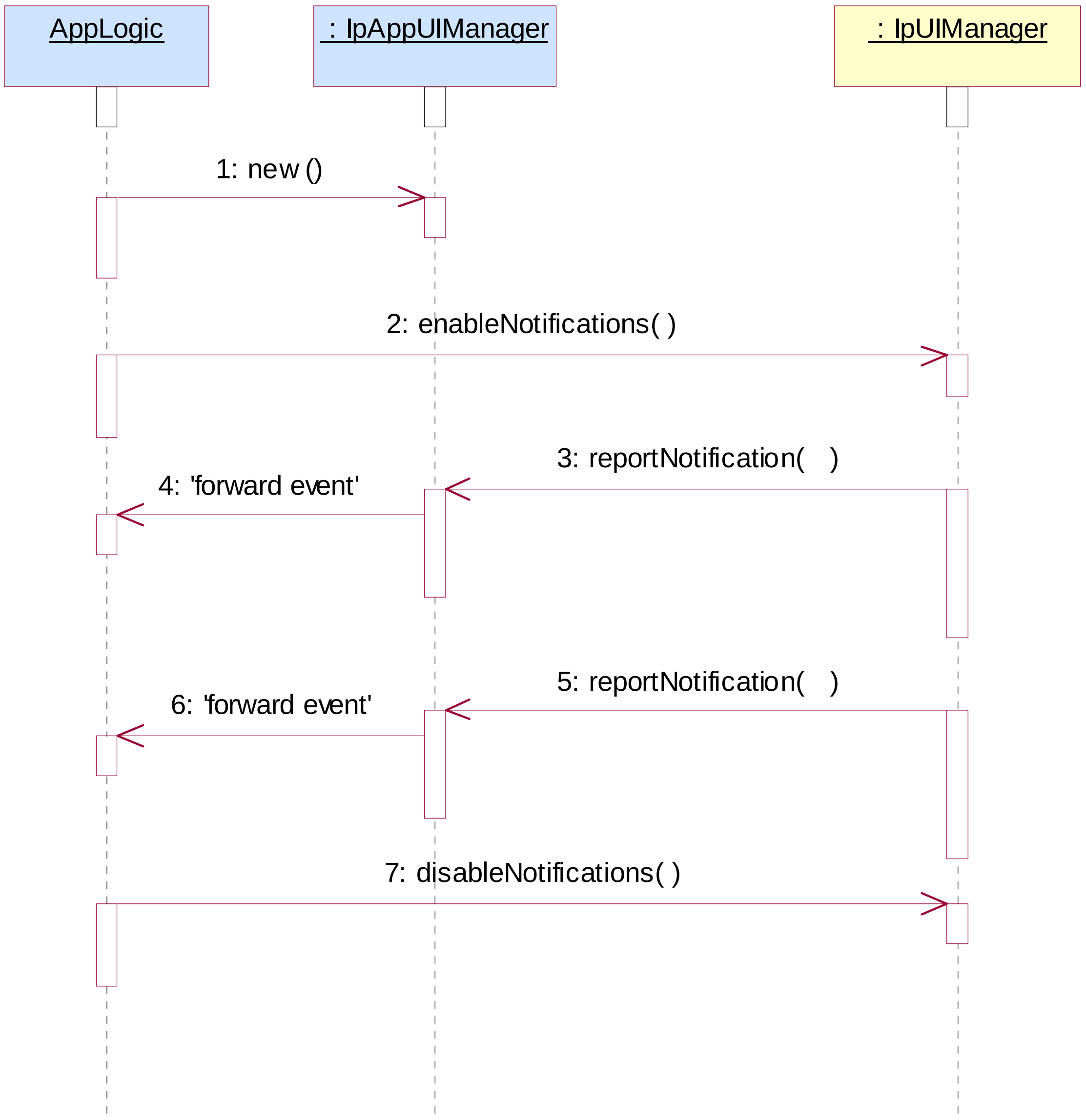
15: When the call is terminated in the network, the application will receive a notification. This notification will always be received when the call is terminated by the network in a normal way, the application does not have to request this event explicitly.

16: The event is forwarded to the application.

17: The application must free the call related resources in the gateway by calling deassignCall.

5.1.3 Network Controlled Notifications

The following sequence diagram shows how an application can receive notifications that have not been created by the application, but are provisioned from within the network.



1: The application is started. The application creates a new IpAppUIManager to handle callbacks.

2: The enableNotifications method is invoked on the IpUIManager interface to indicate that the application is ready to receive notifications that are created in the network. For illustrative purposes we assume notifications of type "B" are created in the network.

3: When a network created trigger occurs the application is notified on the callback interface.

4: The event is forwarded to the application.

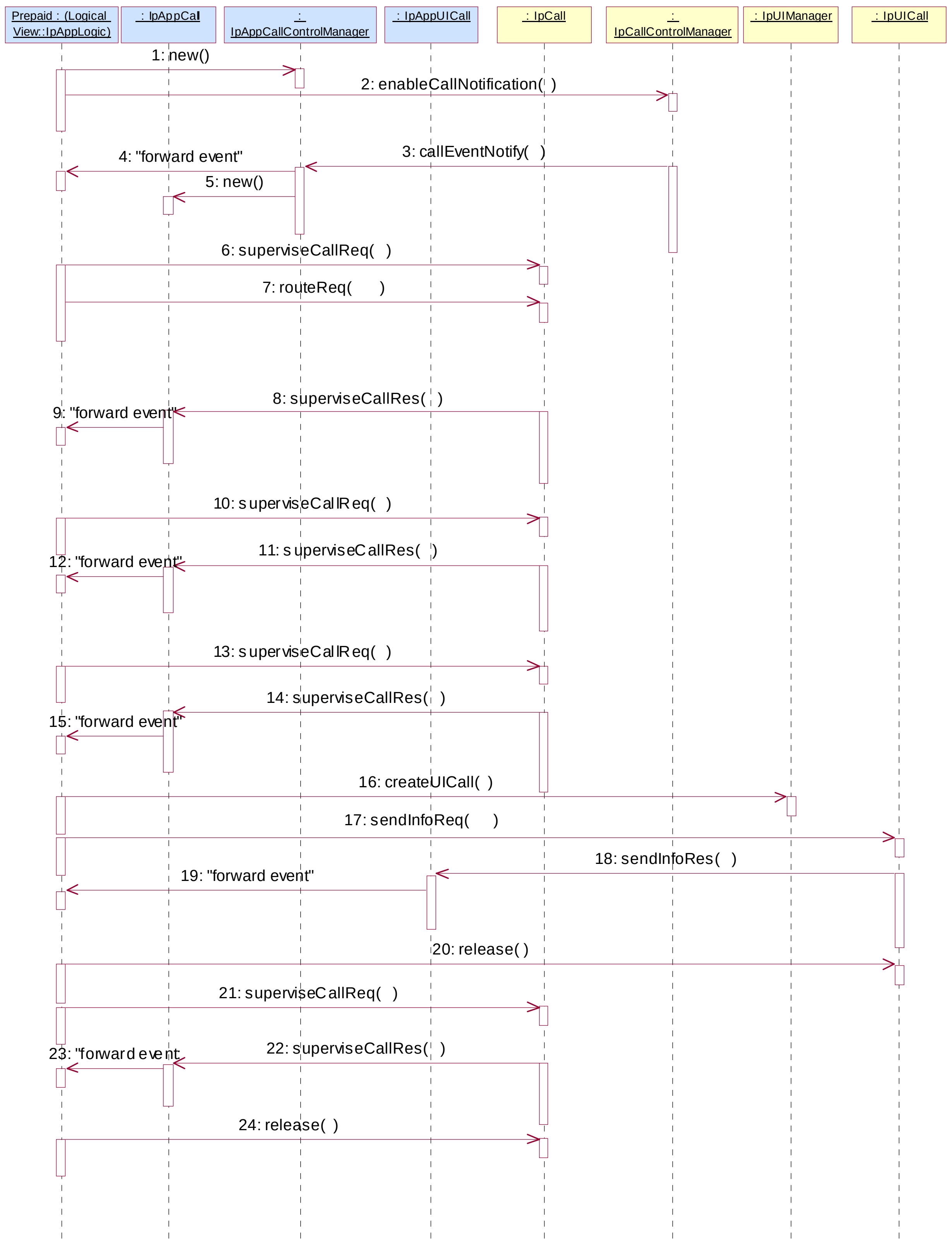
5: When a network created trigger occurs the application is notified on the callback interface.

6: The event is forwarded to the application.

7: When the application does not want to receive notifications created in the network anymore, it invokes disableNotifications on the IpMultiPartyCallConrolManager interface. From now on the gateway will not send any notifications to the application that are created in the network.

5.1.4 Prepaid

This sequence shows a Pre-paid application. The subscriber is using a pre-paid card or credit card to pay for the call. The application each time allows a certain timeslice for the call. After the timeslice, a new timeslice can be started or the application can terminate the call. In the following sequence the end-user will received an announcement before his final timeslice.



1: This message is used by the application to create an object implementing the IpAppCallControlManager interface.

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a pre-paid service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) are met, other messages (not shown) are used to create the call and associated call leg object.

3: The incoming call triggers the Pre-Paid Application (PPA).

4: The message is forwarded to the application.

5: A new object on the application side for the Generic Call object is created

6: The Pre-Paid Application (PPA) requests to supervise the call. The application will be informed after the period indicated in the message. This period is related to the credits left on the account of the pre-paid subscriber.

7: Before continuation of the call, PPA sends all charging information, a possible tariff switch time and the call duration supervision period, towards the GW which forwards it to the network.

8: At the end of each supervision period the application is informed and a new period is started.

9: The message is forwarded to the application.

10: The Pre-Paid Application (PPA) requests to supervise the call for another call duration.

11: At the end of each supervision period the application is informed and a new period is started.

12: The message is forwarded to the application.

13: The Pre-Paid Application (PPA) requests to supervise the call for another call duration. When the timer expires it will indicate that the user is almost out of credit.

14: When the user is almost out of credit the application is informed.

15: The message is forwarded to the application.

16: The application decides to play an announcement to the parties in this call. A new UICall object is created and associated with the call.

17: An announcement is played informing the user about the near-expiration of his credit limit.

18: When the announcement is completed the application is informed.

19: The message is forwarded to the application.

20: The application releases the UICall object.

21: The user does not terminate so the application terminates the call after the next supervision period.

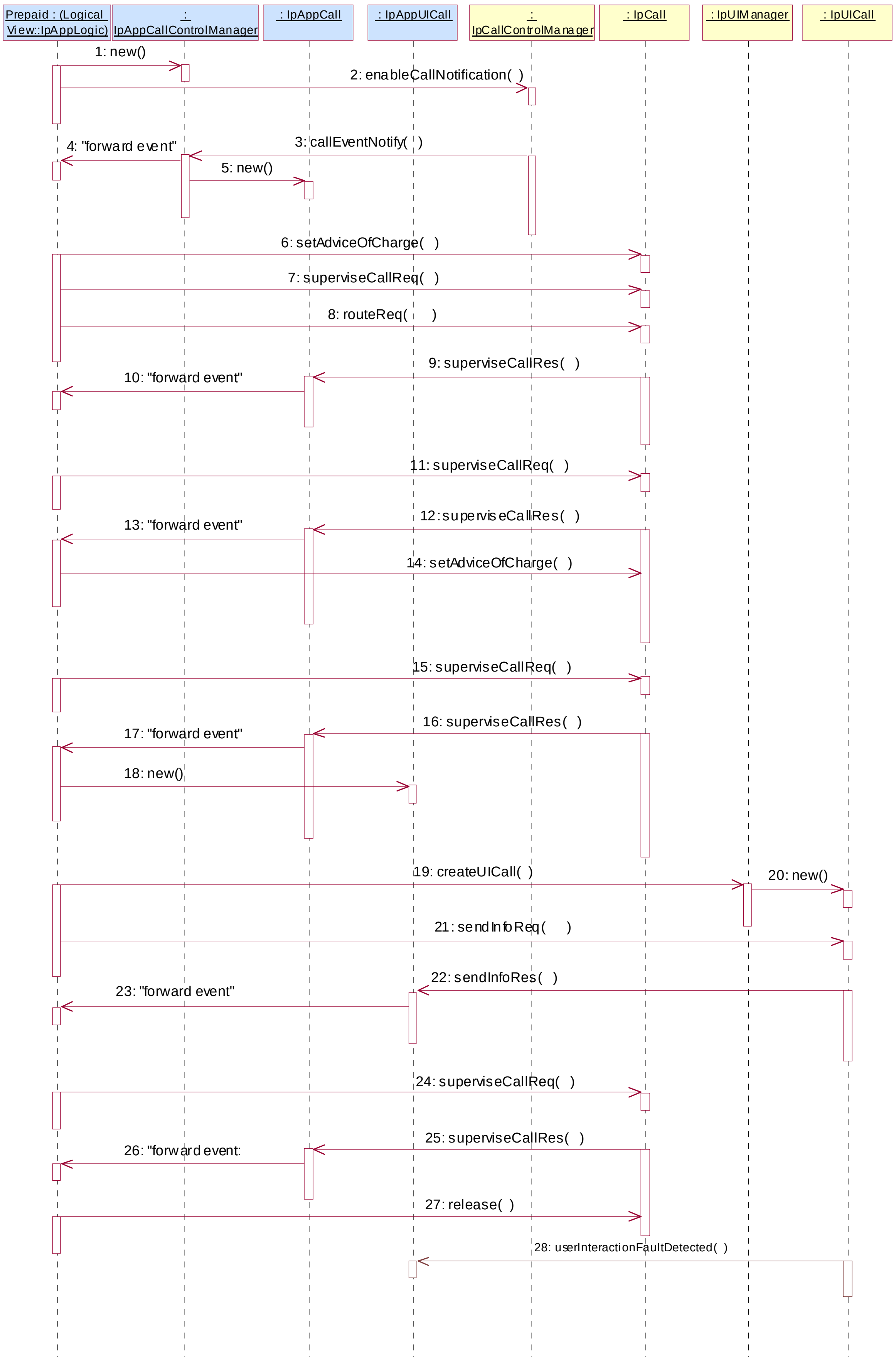
22: The supervision period ends

23: The event is forwarded to the logic.

24: The application terminates the call. Since the user interaction is already explicitly terminated no userInteractionFaultDetected is sent to the application.

5.1.5 Pre-Paid with Advice of Charge (AoC)

This sequence shows a Pre-paid application that uses the Advice of Charge feature. The application will send the charging information before the actual call setup and when during the call the charging changes new information is sent in order to update the end-user. Note that the Advice of Charge feature requires an application in the end-user terminal to display the charges for the call, depending on the information received from the application.


Prepaid : (Logical View::IpAppLogic)
: IpAppCallControlManager
: IpAppCall
: IpAppUICall
: IpCallControlManager
: IpCall
: IpUIManager
: IpUICall
1: new()
2: enableCallNotification( )
3: callEventNotify( )
4: "forward event"
5: new()
6: setAdviceOfCharge( )
7: superviseCallReq( )
8: routeReq( )
9: superviseCallRes( )
10: "forward event"
11: superviseCallReq( )
12: superviseCallRes( )
13: "forward event"
14: setAdviceOfCharge( )
15: superviseCallReq( )
16: superviseCallRes( )
17: "forward event"
18: new()
19: createUICall( )
20: new()
21: sendInfoReq( )
22: sendInfoRes( )
23: "forward event"
24: superviseCallReq( )
25: superviseCallRes( )
26: "forward event:
27: release( )
28: userInteractionFaultDetected( )

1: This message is used by the application to create an object implementing the IpAppCallControlManager interface.

2: This message is sent by the application to enable notifications on new call events. As this sequence diagram depicts a pre-paid service, it is likely that only new call events within a certain address range will be enabled. When a new call, that matches the event criteria, arrives a message (not shown) is directed to the object implementing the IpCallControlManager. Assuming that the criteria for creating an object implementing the IpCall interface (e.g. load control values not exceeded) are met, other messages (not shown) are used to create the call and associated call leg object.

3: The incoming call triggers the Pre-Paid Application (PPA).

4: The message is forwarded to the application.

5: A new object on the application side for the Call object is created

6: The Pre-Paid Application (PPA) sends the AoC information (e.g. the tariff switch time). (it shall be noted the PPA contains ALL the tariff information and knows how to charge the user).

During this call sequence 2 tariff changes take place. The call starts with tariff 1, and at the tariff switch time (e.g., 18:00 hours) switches to tariff 2. The application is not informed about this (but the end-user is!)

7: The Pre-Paid Application (PPA) requests to supervise the call. The application will be informed after the period indicated in the message. This period is related to the credits left on the account of the pre-paid subscriber.

8: The application requests to route the call to the destination address.

9: At the end of each supervision period the application is informed and a new period is started.

10: The message is forwarded to the application.

11: The Pre-Paid Application (PPA) requests to supervise the call for another call duration.

12: At the end of each supervision period the application is informed and a new period is started.

13: The message is forwarded to the application.

14: Before the next tariff switch (e.g., 19:00 hours) the application sends a new AOC with the tariff switch time. Again, at the tariff switch time, the network will send AoC information to the end-user.

15: The Pre-Paid Application (PPA) requests to supervise the call for another call duration. When the timer expires it will indicate that the user is almost out of credit.

16: When the user is almost out of credit the application is informed.

17: The message is forwarded to the application.

18: The application creates a new call back interface for the User interaction messages.

19: A new UI Call object that will handle playing of the announcement needs to be created

20: The Gateway creates a new UI call object that will handle playing of the announcement.

21: With this message the announcement is played to the parties in the call.

22: The user indicates that the call should continue.

23: The message is forwarded to the application.

24: The user does not terminate so the application terminates the call after the next supervision period.

25: The user is out of credit and the application is informed.

26: The message is forwarded to the application.

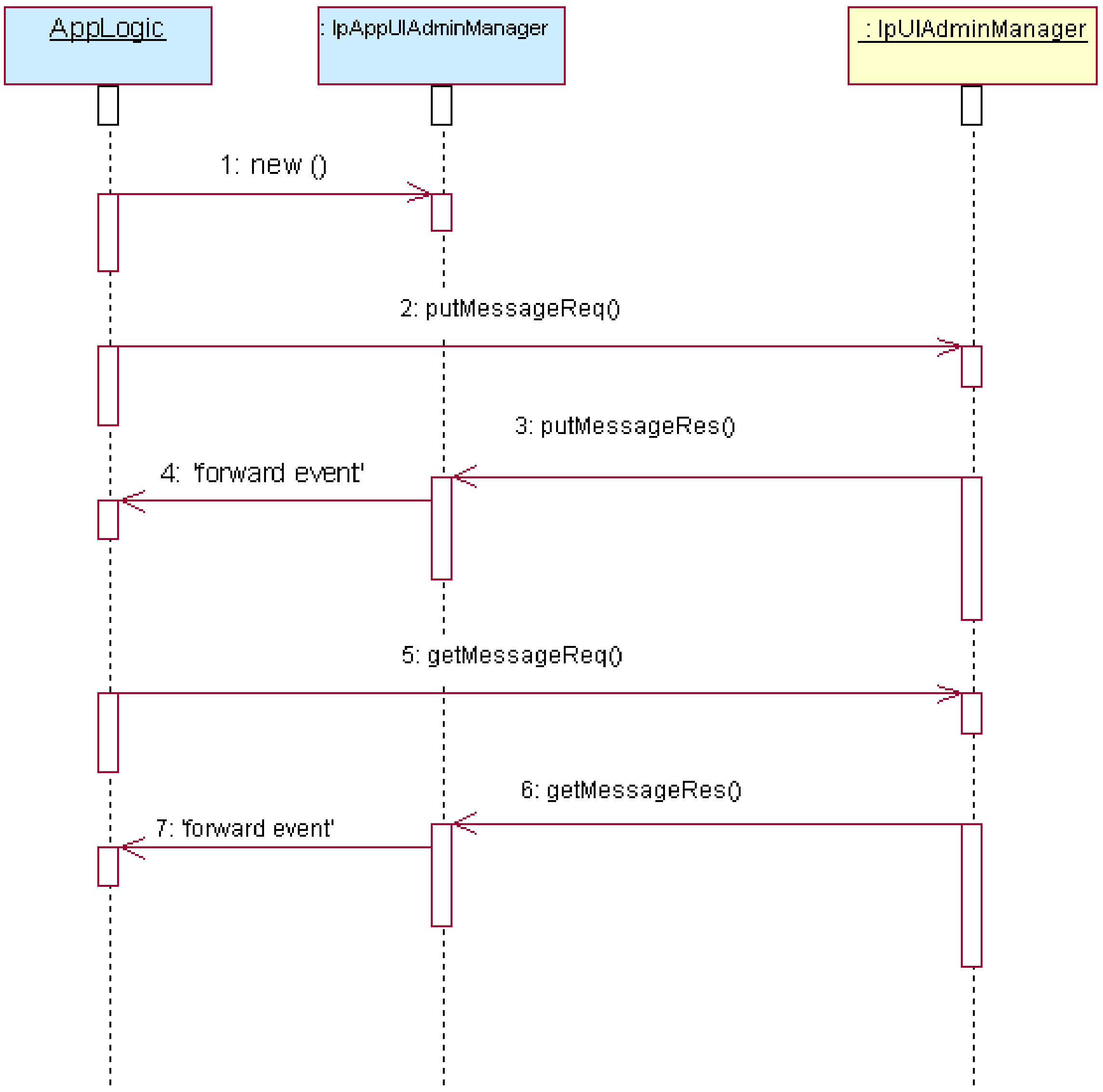
27: With this message the application requests to release the call.

28: Terminating the call which has still a UICall object associated will result in a userInteractionFaultDetected. The UICall object is terminated in the gateway and no further communication is possible between the UICall and the application.

5.2 Generic User Interaction Administration Sequence Diagrams

5.2.1 Message Administration

The following sequence diagram shows how an application can manage the user announcement and recorded messages.



1: The application is started. The application creates a new IpAppUIAdminManager to handle callbacks.

2: The putMessageReq method is invoked on the IpUIAdminManager interface to create a new pre-defined message for use by sending to the user.

3: The putMessageRes response notifies the application of the messageID on the callback interface.

 4: The response is forwarded to the application logic.

5: The getMessageReq method is invoked on the IpUIAdminManager interface to retrieve the contents of a user announcement or recorded message.

6: The getMessageRes response notifies the application of the contents of a message.

 7: The event is forwarded to the application.

~~6~~6 Class Diagrams

6.1 Generic and Call User Interaction Class Diagrams

The application generic user interaction service package consists of one IpAppUIManager interface, zero or more IpAppUI interfaces and zero or more IpAppUICall interfaces.

The generic user interaction service package consists of one IpUIManager interface, zero or more IpUI interfaces and zero or more IpUICall interfaces.

The class diagram in the following figure shows the interfaces that make up the application generic user interaction service package and the generic user interaction service package. Communication between these packages is done via the <<uses>> relationships.

The IpUICall implements call related user interaction and it inherits from the non call related IpUI interface. The same holds for the corresponding application interfaces.

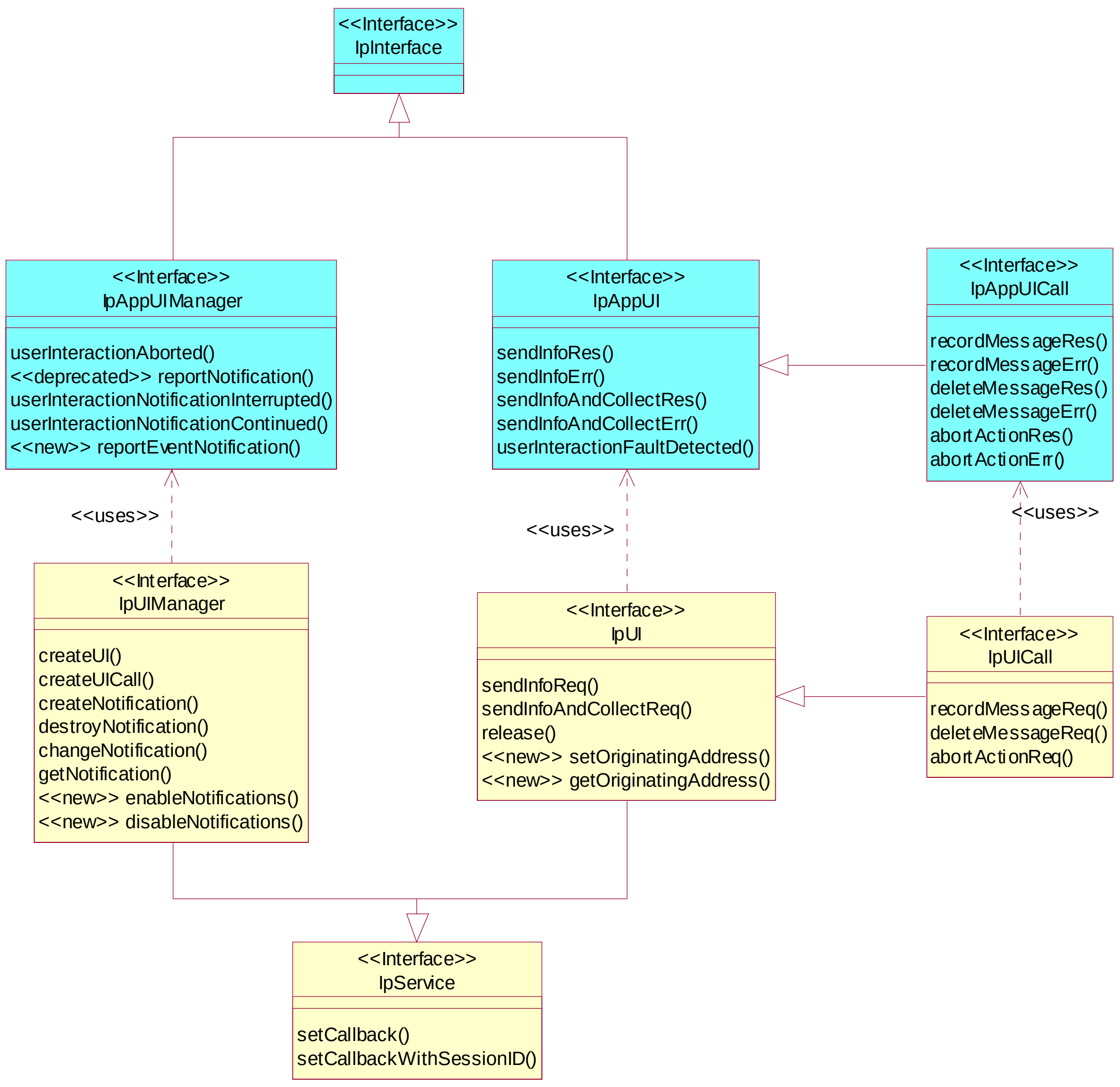


Figure : Generic User Interaction Package Overview

6.2 Generic User Interaction Administration Class Diagrams

The application generic user administration service package consists of one IpAppUIAdminManager interface and one IpUIAdminManager interfaces.

The class diagram in the following figure shows the interfaces that make up the application generic user administration service package. Communication between these packages is done via the <<uses>> relationships.

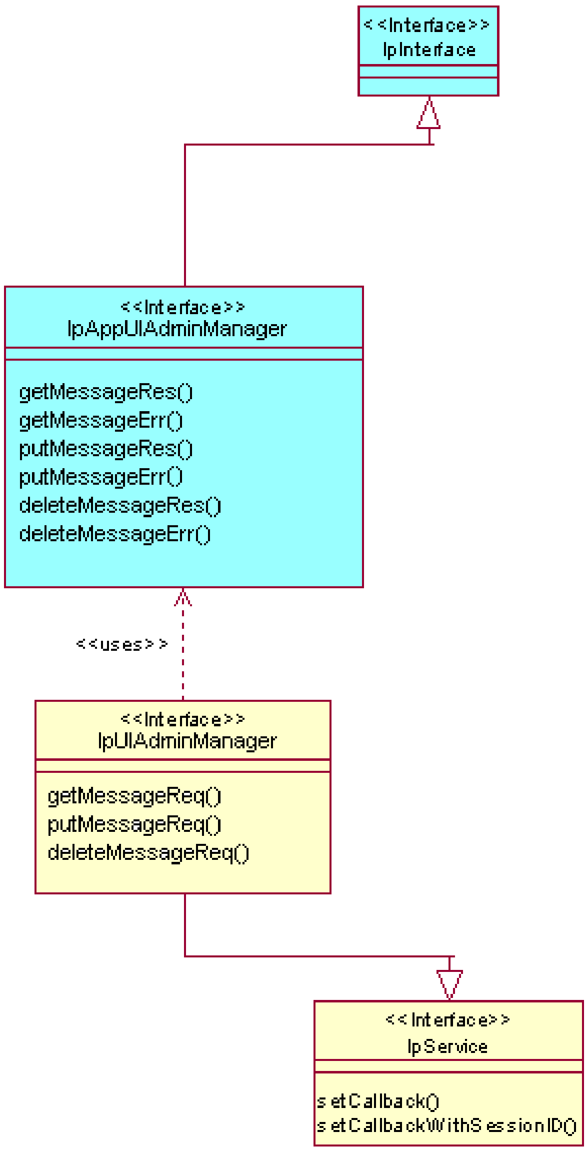


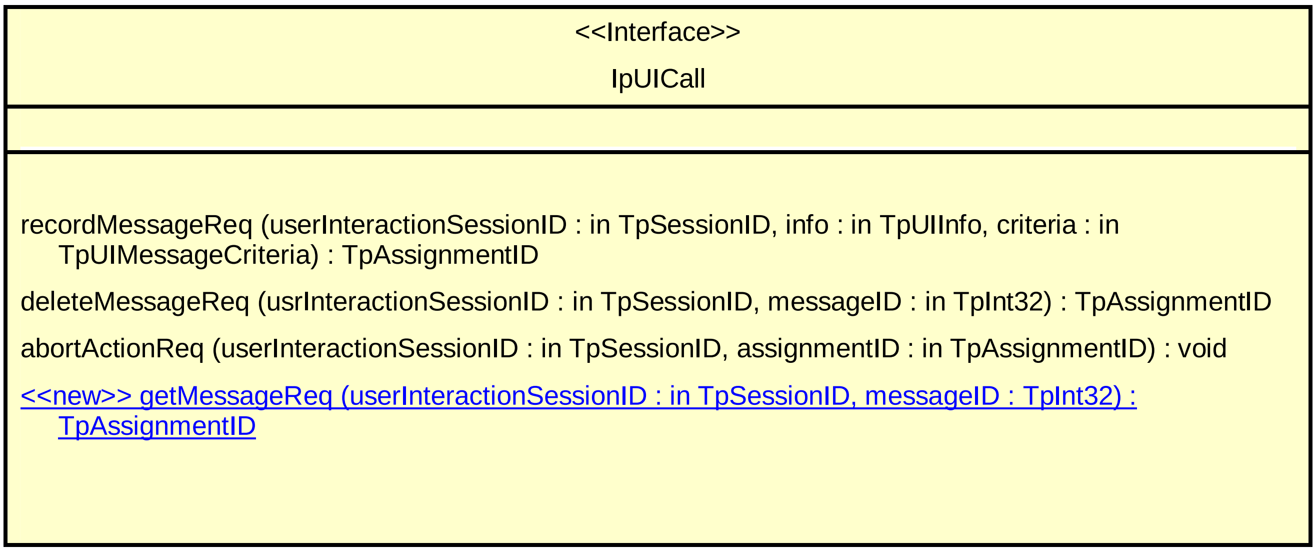
Figure: Generic User Administration Package Overview

8.5 Interface Class IpUICall

Inherits from: IpUI.

The Call User Interaction Service Interface provides functions to send information to, or gather information from the user (or call party) to which a call leg is connected. An application can use the Call User Interaction Service Interface only in conjunction with another service interface, which provides mechanisms to connect a call leg to a user. At present, only the Call Control service supports this capability.

This interface, or the IpUI interface, shall be implemented by a Generic User Interaction SCF as a minimum requirement. The minimum required methods of interface IpUI shall be implemented.



<<Interface>>

IpUICall

recordMessageReq (userInteractionSessionID : in TpSessionID, info : in TpUIInfo, criteria : in TpUIMessageCriteria) : TpAssignmentID

deleteMessageReq (usrInteractionSessionID : in TpSessionID, messageID : in TpInt32) : TpAssignmentID

abortActionReq (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID) : void

<<new>> getMessageReq (userInteractionSessionID : in TpSessionID, messageID : TpInt32) : TpAssignmentID

8.5.4 Method getMessageReq()

This asynchronous method allows retrieving the recorded message content from the gateway. This method is applicable only to recorded messages.

Returns: assignmentID

Specifies the ID assigned by the user interaction interface for a user interaction request.

Parameters

`userInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`messageID : in TpInt32`

Specifies the message ID.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_NETWORK_STATE,P_ILLEGAL_ID,P_ID_NOT_FOUND`

8.6 Interface Class IpAppUICall

Inherits from: IpAppUI.

The Call User Interaction Application Interface is implemented by the client application developer and is used to handle call user interaction request responses and reports.

<<Interface>>

IpAppUICall

recordMessageRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, response : in TpUIReport, messageID : in TpInt32) : void

recordMessageErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

deleteMessageRes (usrInteractionSessionID : in TpSessionID, response : in TpUIReport, assignmentID : in TpAssignmentID) : void

deleteMessageErr (usrInteractionSessionID : in TpSessionID, error : in TpUIError, assignmentID : in TpAssignmentID) : void

abortActionRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID) : void

abortActionErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

<<new>> getMessageRes (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, message : in TpUIInfo) : void

<<new>> getMessageErr (userInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

8.6.7 Method getMessageRes()

This method returns the message content if the message was retrieved successfully.

Parameters

`userInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the user interaction interface for a user interaction request.

`message : in TpUIInfo`

Specifies the UI Information containing the message content information.

8.6.8 Method getMessageErr()

This method indicates that the request to retrieve a message was not successful.

Parameters

`userInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the user interaction interface for a user interaction request.

error : in TpUIError

Specifies the error which led to the original request failing.

8.7 Interface Class IpUIAdminManager

The Generic User Interaction Administration Manager Service interface is used by applications to manage user announcement and recorded messages on the gateway. This Service is represented by the IpUIAdminManager interface that interfaces to the service provided by the network. To handle responses and reports, the developer must implement IpAppUIAdminManager interface to provide the callback mechanism.

The application context will ensure that one application doesn't interfere with the messages of another application.

The User Interaction Administration Manager Service Interface provides functions to manage the messages.

<<Interface>> IpUIAdminManager
getMessageReq (usrInteractionSessionID : in TpSessionID, messageID : in TpInt32) : TpAssignmentID putMessageReq (usrInteractionSessionID : in TpSessionID, info : in TpUIInfo) : TpAssignmentID deleteMessageReq (usrInteractionSessionID : in TpSessionID, messageID : in TpInt32) : TpAssignmentID

8.7.1 Method getMessageReq()

This asynchronous method allows retrieving the user announcement or recorded message content from the gateway.

Returns: assignmentID

Specifies the ID assigned by the user interaction interface for a user interaction request.

Parameters

usrInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

messageID : in TpInt32

Specifies the message ID.

Returns

TpAssignmentID

Raises

TpCommonExceptions,P_INVALID_SESSION_ID,P_INVALID_NETWORK_STATE,P_ILLEGAL_ID,P_ID_NOT_FOUND

8.7.2 Method putMessageReq()

This asynchronous method allows putting a user announcement message content onto the gateway. The gateway will allocate the messageID and return it to the application on the putMessageRes() confirmation.

Returns: assignmentID

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`info : in TpUIInfo`

Specifies the information to send to the user. This information can be either an ID (for pre-defined announcement or text), a text string, or an URL (indicating the information to be sent, e.g. an audio stream).

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_ILLEGAL_ID,P_ID_NOT_FOUND`

8.7.3 Method deleteMessageReq()

This asynchronous method allows deleting a user announcement or recorded message.

Returns: assignmentID

Specifies the ID assigned by the generic user interaction interface for a user interaction request.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`messageID : in TpInt32`

Specifies the message ID.

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions,P_INVALID_SESSION_ID,P_ILLEGAL_ID,P_ID_NOT_FOUND`

8.8 Interface Class IpAppUIAdminManager

The User Interaction Administration Manager Application Interface is implemented by the client application and is used to handle administration user interaction request responses and reports.

<<Interface>> IpAppUIAdminManager
getMessageRes (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, message : in TpUIInfo) : void getMessageErr (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void deleteMessageRes (usrInteractionSessionID : in TpSessionID, response : in TpUIReport, assignmentID : in TpAssignmentID) : void deleteMessageErr (usrInteractionSessionID : in TpSessionID, error : in TpUIError, assignmentID : in TpAssignmentID) : void putMessageRes (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, messageID : in TpInt32) : void putMessageErr (usrInteractionSessionID : in TpSessionID, assignmentID : in TpAssignmentID, error : in TpUIError) : void

8.8.1 Method getMessageRes()

This method returns the message content if the message was retrieved successfully.

Parameters

usrInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the user interaction interface for a user interaction request.

message : in TpUIInfo

Specifies the UI Information containing the message content information.

8.8.2 Method getMessageErr()

This method indicates that the request to retrieve a message was not successful.

Parameters

usrInteractionSessionID : in TpSessionID

Specifies the user interaction session ID of the user interaction.

assignmentID : in TpAssignmentID

Specifies the ID assigned by the user interaction interface for a user interaction request.

error : in TpUIError

Specifies the error which led to the original request failing.

8.8.3 Method deleteMessageRes()

This method indicates that the request to delete a message was successful.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`response : in TpUIReport`

Specifies the type of response received from the device where the message was stored.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the user interaction interface for a user interaction request.

8.8.4 Method deleteMessageErr()

This method indicates that the request to delete a message was not successful.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`error : in TpUIError`

Specifies the error which led to the original request failing.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the user interaction interface for a user interaction request.

8.8.5 Method putMessageRes()

This asynchronous method confirms that the request to put the message content was successful.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the user interaction interface for a user interaction request.

`messageID : in TpInt32`

Specifies the message ID that was allocated by the gateway.

8.8.6 Method putMessageErr()

This asynchronous method indicates that the request to put the message content resulted in an error.

Parameters

`usrInteractionSessionID : in TpSessionID`

Specifies the user interaction session ID of the user interaction.

`assignmentID : in TpAssignmentID`

Specifies the ID assigned by the user interaction interface for a user interaction request.

`error : in TpUIError`

Specifies the error which led to the original request failing.

9.4 State Transition Diagrams for IpUIManagerAdmin

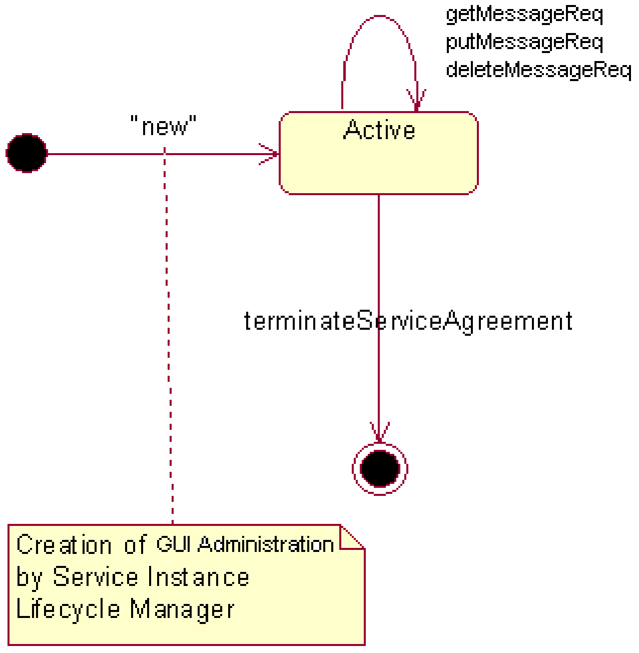

getMessageReq
putMessageReq
deleteMessageReq
"new"
Active
terminateServiceAgreement
Creation of GUI Administration by Service Instance Lifecycle Manager

Figure : State Transition Diagram for User Interaction Administration

9.4.1 Active State

In this state, a relation between the Application and the Generic User Interaction Administration Service Capability Feature has been established. It allows the application to make specific requests of the service.