3GPP TSG CN Plenary Meeting #25 **NP-040356**
08-10 September 2004, Palm Springs, CA, USA

Source:	**CN5 (OSA)**
Title:	**7 Rel-6 CR 29.198-xy Add Description of Backwards Compatibility rules & Remove deprecated items**
Agenda item:	**9.7 (OSA Enhancements [OSA3])**
Document for:	**APPROVAL**

Doc-1st-Level	Spec	CR	R	Phase	Subject	Cat	Version-Current	Doc-2nd-Level	Workitem
NP-040356	29.198-01	038	--	Rel-6	Add Description of Backwards Compatibility rules	B	6.1.0	N5-040620	OSA3
NP-040356	29.198-03	125	--	Rel-6	Remove unused Deprecated items	F	6.1.0	N5-040621	OSA3
NP-040356	29.198-04-2	022	--	Rel-6	Remove unused Deprecated items	F	6.1.0	N5-040622	OSA3
NP-040356	29.198-04-3	029	--	Rel-6	Remove unused Deprecated items	F	6.2.0	N5-040623	OSA3
NP-040356	29.198-05	053	--	Rel-6	Remove unused Deprecated items	F	6.1.0	N5-040624	OSA3
NP-040356	29.198-08	035	--	Rel-6	Remove unused Deprecated items	F	6.1.0	N5-040625	OSA3
NP-040356	29.198-11	031	--	Rel-6	Remove unused Deprecated items	F	6.1.0	N5-040626	OSA3

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040620**
Meeting #28, Piscataway, New Jersey, USA, 09-13 August 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-01**	**CR**	**038**	⌘ **rev**	**-**	⌘	Current version:	**6.1.0** ⌘

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

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

Title:	⌘	Add Description of Backwards Compatibility rules		
Source:	⌘	CN5 Ultan Mulligan, ETSI PTCC		
Work item code:	⌘	OSA3	***Date:*** ⌘	27/08/2004
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: ⌘ Currently the development and updating of the OSA specifications is performed according to some backwards compatibilty rules. These rules were originally established in 2002, but were never included in the specifications. Today the knowledge and understanding of these rules is not always shared among those who contribute to the development of the OSA specifications.
Adding rules for backwards compatibility will provide guidelines for those who wish to contribute to the development of the OSA specifications.
Adding rules for the deprecation and removal of 'old' parts of the OSA specifications will provide future visibility for developers who wish to implement the OSA specifications.

Summary of change: ⌘ Add clause 8 which contains guidelines and rules for further development of the OSA specifications, in order to preserve backwards compatibility, and which contains rules for the phased removal of 'old' items from the specifications.

Consequences if not approved: ⌘ There will be no guidance for contributors to the OSA specifications as to what changes will impact backwards compatibility, and what changes will not.
There will be no guidance for developers which indicate for how long any deprecated items will remain in the specifications.

Clauses affected: ⌘ New clause 8

Other specs affected: ⌘

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

Other comments: ⌘

How to create CRs using this form:

New Clause 8

8 Backwards Compatibility Considerations

The backwards compatibility rules described below are intended to enable an older client to continue to interwork with a newer server or gateway.

8.1 Guidelines to enable backwards compatibility in implementations

1) The Gateway should require the usage of Framework versions and service versions. All Applications should use these parameters.

2) The IDL version parameter should not be used when generating the IDL.

3) If there are multiple versions of an SCF they should be all registered with the Framework, and the SCF should create an instance of the version requested by the application when a new service manager is created.

8.2 Rule summary

The following types of changes can be made to these specifications while preserving backwards compatibility, everything beyond these changes is not allowed.

8.2.1 Server side permitted changes:

- Addition of a new interface
- Addition of a new method to an existing or new interface
- Addition and removal of exceptions if the implementation uses the Application version as described above.

8.2.2 Client side permitted changes:

- Addition of a new interface
- Addition of a new method

Note: The version the client requests should be used to indicate which interfaces and methods are supported on the client side.

- Addition and removal of exceptions if the implementation uses the Application version as described above.

8.2.3 Data type permitted changes:

- Elements can be added to “sequence” data types. Care should be taken when adding elements to data types that are sent back to the client: The client may be outdated and thus not be able to interpret the new element. Only information that has not been available before (and therefore is not expected by the client) may be transferred in added elements. Information that has been available before (and therefore possibly expected by the client) may not be modified in any way.
- Elements can be added to “tagged choice of data elements” data types, if they are always sent from client to server (either within a parameter of a server side method, or within the result of a client side method.

Every change beyond the rules listed above is forbidden. In particular, changes like the following should not be done:

- Changing the order of enumerated types
- Changing method signatures

- Removing or renaming methods

8.3 Implementation Guidelines for Server Programmers

- If methods are added at the client side, the server should call them only if it can be sure that the client has implemented them. Basically, this means the server needs to make sure that the client supports the release, where the new methods have been introduced, or a later one.
- Servers could ensure that references to dynamically created objects (service managers or calls) remain valid even after a server upgrade. An alternative method is to be able to make so called graceful close of old versions and running the new version in parallel. The old version will not allow any new requests but will allow existing ones to execute until they are finished.

8.4 Implementation Guidelines for Client Programmers

- The backwards compatibility rules allow for “smooth” upgrades to new Parlay/OSA releases in the Gateway. All existing functionality should still work without any changes in the client. Client programmers need to change code only to enhance it; they should never need to change code just to adopt it to the new release. Care should be taken when supporting features of a new release. The moment a client application use newer release features, it should then support all of the client side features for that newer release, otherwise the sever may invoke newer release methods on the client and the client will not respond.

8.5 Tracking the changes in the specifications

8.5.1 New Tag

If a client side interface is added, or methods are added to an existing interface, the new methods are marked with a UML stereotype “New”.

This tag is merely a hint for the programmer.

8.5.2 Deprecated Tag

If interfaces, methods or service properties are deemed outdated or broken, the items are marked with a UML stereotype “Deprecated”. The tag indicates that they are supported by this Framework or SCF release, but that they will not necessarily be supported in subsequent releases. The respective items may be removed in the specification release.

The tag is a hint for the client programmer that an update to their client applications may be necessary.

8.6 Technology realization rules

8.6.1 Corba IDL Rules

In addition to the rules identified above, in order to ensure backwards compatibility of the IDL code, the following rules shall be followed in updating this specification:

- IDL version numbering should not be used when generating the IDL.

8.6.2 Java rules

In addition to the rules identified above, in order to ensure backwards compatibility of the J2EE and J2SE code, the following rules shall be followed in updating this specification:

- When elements are added to “sequence” data types, the Java constructor for these data types is updated with the new elements when the Java code is re-generated. The old constructor, without the new elements, shall be manually included in the generated Java code and marked as deprecated.

8.7 Rules for removal of deprecated items from the specifications

- At the start of each major 3GPP Release n (during the development phase of a new release, prior to the closure of the release to new functionality), we delete, using change requests, all deprecated methods and other deprecated items, which are identified as deprecated in the most recent version of the Release n-2 edition of specifications.
- When deleting deprecated methods, any unused data types can be deleted using the CR process,.
- At the start of each major release, the <<new>> stereotypes that were present in the specifications prior to this release are deleted. CRs are not required for this.
- Methods or stereotypes are never deleted following closure of a release to new functionality.
- Names of deleted methods are never re-used.
- Exceptionally, we may choose not to delete certain deprecated methods, in the interest of preserving backwards compatibility.

End of NewClause 8

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040621**
Meeting #28, Piscataway, New Jersey, USA, 09-13 August 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-03**	**CR**	**125**	⌘**rev**	**-**	⌘	Current version:	**6.1.0**	⌘

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

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

Title:	⌘	Remove unused Deprecated items		
Source:	⌘	CN5 Ultan Mulligan, ETSI PTCC		
Work item code:	⌘	OSA3	***Date:*** ⌘	27/08/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*
		Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *Rel-4 (Release 4)* *Rel-5 (Release 5)* *Rel-6 (Release 6)*

Reason for change:	⌘	3GPP CN5 uses a procedure of deprecation in its specifications to identify methods and other items which cannot be used or have been replaced. Items are deprecated when they have been found not to work, or when replaced in a controlled, backwards compatible manner. Such deprecated items should be removed one day from the specifications: 3GPP CN5 keeps such deprecated items in its specifications for 2 full releases in order to preserve backwards compatibility. At the delivery of Release 6, it is time to remove those items from the Release 6 specifications which are deprecated in the current Release 4 specifications and the Release 5 specifications. This removal has no impact on the Release 4 or Release 5 specifications - backwards compatibility is preserved for these.
Summary of change:	⌘	Delete the deprecated Service Property Type ADDRESSRANGE_SET
Consequences if not approved:	⌘	Old, unusable items will continue to clutter up the OSA specifications. The non-removal of deprecated items will encourage developers to implement them, when in fact deprecation of an item is intended to do the exact opposite. Furthermore, vendors will continue to be obliged to retain these deprecated items in their implementations, forcing them to continue to include these often broken items in their test plans.

Clauses affected:	⌘	9				
		Y	N			
Other specs affected:	⌘	X		Other core specifications	⌘	29.198-1, -4-2, -4-3, -5, -8 and -11
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

How to create CRs using this form:

Change in Clause 9

9 Service Properties

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.

9.2 Service Property Types

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 (Deprecated)**~~	~~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.
XML_ADDRESS_RANGE_SET	set of values of TpAddressRange. Values of TpAddressRange are described using XML. An XML schema is provided below for this purpose.	{"<AddressRangeSet> <AddressRange> <Plan>P_ADDRESS_PLAN_E164</Plan> <AddrString>123*</AddrString> </AddressRange> <AddressRange> <Plan>P_ADDRESS_PLAN_E164</Plan> <AddrString>234*</AddrString> </AddressRange> </AddressRangeSet>"}	Any addresses starting with 123 or starting with 456 in the E.164 Address Plan

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.

The value of XML_ADDRESS_RANGE_SET should comply with the following XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="AddressRangeSet">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="AddressRange" maxOccurs="unbounded">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="Plan" type="xs:string" default="P_ADDRESS_PLAN_ANY"/>
							<xs:element name="AddrString" type="xs:string"/>
							<xs:element name="Name" type="xs:string" minOccurs="0"/>
							<xs:element name="SubAddressString" type="xs:string" minOccurs="0"/>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

An example usage could be:

```
{"<?xml version="1.0" encoding="UTF-8"?>
<AddressRangeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="xml_address_range_set.xsd">
	<AddressRange>
		<Plan>P_ADDRESS_PLAN_E164</Plan>
		<AddrString>789*</AddrString>
	</AddressRange>
	<AddressRange>
		<Plan>P_ADDRESS_PLAN_ANY</Plan>
		<AddrString>123*</AddrString>
	</AddressRange>
	<AddressRange>
		<Plan>P_ADDRESS_PLAN_SIP</Plan>
		<AddrString>&lt;sip:*@parlay.org&gt;</AddrString>
		<Name/>
	</AddressRange>
</AddressRangeSet>"}
```

Note that the final address range corresponds to any sip address @parlay.org, i.e. <sip:*@parlay.org>.

End of Change in Clause 9

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040622**
Meeting #28, Piscataway, New Jersey, USA, 09-13 August 2004

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-04-2**	**CR**	**022**	⌘ **rev**	**-**	⌘	Current version:	**6.1.0**	⌘

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

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

Title: ⌘	Remove unused Deprecated items		
Source: ⌘	CN5 Ultan Mulligan, ETSI PTCC		
Work item code: ⌘	OSA3	***Date:*** ⌘	27/08/2004
Category: ⌘	**F**	***Release:*** ⌘	*REL-6*
	Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *Rel-4 (Release 4)* *Rel-5 (Release 5)* *Rel-6 (Release 6)*

Reason for change: ⌘	3GPP CN5 uses a procedure of deprecation in its specifications to identify methods and other items which cannot be used or have been replaced. Items are deprecated when they have been found not to work, or when replaced in a controlled, backwards compatible manner. Such deprecated items should be removed one day from the specifications: 3GPP CN5 keeps such deprecated items in its specifications for 2 full releases in order to preserve backwards compatibility. At the delivery of Release 6, it is time to remove those items from the Release 6 specifications which are deprecated in the current Release 4 specifications and the Release 5 specifications. This removal has no impact on the Release 4 or Release 5 specifications - backwards compatibility is preserved for these.
Summary of change: ⌘	Delete the deprecated Service Property P_TRIGGERING_ADDRESSES
Consequences if not approved: ⌘	Old, unusable items will continue to clutter up the OSA specifications. The non-removal of deprecated items will encourage developers to implement them, when in fact deprecation of an item is intended to do the exact opposite. Furthermore, vendors will continue to be obliged to retain these deprecated items in their implementations, forcing them to continue to include these often broken items in their test plans.

Clauses affected: ⌘			8		
	Y	**N**			
Other specs affected: ⌘	X		Other core specifications	⌘	TS 29.198-1, -3, -4-3, -5, -8 and -11
		X	Test specifications		
		X	O&M Specifications		
Other comments: ⌘					

How to create CRs using this form:

Change in Clause 8

8 Generic Call Control Service Properties

8.1 List of Service Properties

The following table lists properties relevant for the GCC API.

Property	Type	Description / Interpretation
P_TRIGGERING_EVENT_TYPES	INTEGER_SET	Indicates the static event types supported by the SCS. Static events are the events by which applications are initiated.
P_DYNAMIC_EVENT_TYPES	INTEGER_SET	Indicates the dynamic event types supported by the SCS. Dynamic events are the events the application can request for during the context of a call.
P_ADDRESSPLAN	INTEGER_SET	Indicates the supported address plans (defined in TpAddressPlan.) e.g. {P_ADDRESS_PLAN_E164, P_ADDRESS_PLAN_IP}). Note that more than one address plan may be supported.
P_UI_CALL_BASED	BOOLEAN_SET	Value = TRUE : User interaction can be performed on call level and a reference to a Call object can be used in the IpUIManager.createUICall() operation. Value = FALSE: No User interaction on call level is supported.
P_UI_AT_ALL_STAGES	BOOLEAN_SET	Value = TRUE: User Interaction can be performed at any stage during a call . Value = FALSE: User Interaction can be performed in case there is only one party in the call.
P_MEDIA_TYPE	INTEGER_SET	Specifies the media type used by the Service. Values are defined by data-type TpMediaType : P_AUDIO, P_VIDEO, P_DATA

The previous table lists properties related to capabilities of the SCS itself. The following table lists properties that are used in the context of the Service Level Agreement, e.g. to restrict the access of applications to the capabilities of the SCS.

Property	Type	Description
~~P_TRIGGERING_ADDRESSES (Deprecated)~~	~~ADDRESSRANGE_SET~~	~~Indicates for which numbers the notification may be set. For terminating notifications it applies to the terminating number, for originating notifications it applies only to the originating number.~~
P_NOTIFICATION_ADDRESS_RANGES	XML_ADDRESS_RANGE_SET	Indicates for which numbers notifications may be set. More than one range may be present. For terminating notifications they apply to the terminating number, for originating notifications they apply only to the originating number.
P_NOTIFICATION_TYPES	INTEGER_SET	Indicates whether the application is allowed to set originating and/or terminating triggers in the ECN. Set is: P_ORIGINATING P_TERMINATING
P_MONITOR_MODE	INTEGER_SET	Indicates whether the application is allowed to monitor in interrupt and/or notify mode. Set is: P_INTERRUPT P_NOTIFY
P_NUMBERS_TO_BE_CHANGED	INTEGER_SET	Indicates which numbers the application is allowed to change or fill for legs in an incoming call. Allowed value set: {P_ORIGINAL_CALLED_PARTY_NUMBER, P_REDIRECTING_NUMBER, P_TARGET_NUMBER, P_CALLING_PARTY_NUMBER}.
P_CHARGEPLAN_ALLOWED	INTEGER_SET	Indicates which charging is allowed in the setCallChargePlan indicator. Allowed values: {P_TRANSPARANT_CHARGING, P_CHARGE_PLAN}
P_CHARGEPLAN_MAPPING	INTEGER_INTEGER_MAP	Indicates the mapping of chargeplans (we assume they can be indicated with integers) to a logical network chargeplan indicator. When the chargeplan supports indicates P_CHARGE_PLAN then only chargeplans in this mapping are allowed.

8.2 Service Property values for the CAMEL Service Environment.

Implementations of the Generic Call Control API relying on the CSE of CAMEL phase 4 shall have the Service Properties outlined above set to the indicated values :

```
P_OPERATION_SET = {
“IpCallControlManager.createCall”,
“IpCallControlManager.enableCallNotification”,
“IpCallControlManager.disableCallNotification”,
“IpCallControlManager.changeCallNotification”,
“IpCallControlManager.getCriteria”,
“IpCallControlManager.setCallLoadControl”,
“IpCall.routeReq”,
“IpCall.release”,
“IpCall.deassignCall”,
“IpCall.getCallInfoReq”,
“IpCall.setCallChargePlan”,
“IpCall.setAdviceOfCharge”,
“IpCall.superviseCallReq”
}

P_TRIGGERING_EVENT_TYPES = {
P_CALL_REPORT_ALERTING,
P_EVENT_GCCS_ADDRESS_COLLECTED_EVENT,
P_EVENT_GCCS_ADDRESS_ANALYSED_EVENT,
P_EVENT_GCCS_CALLED_PARTY_BUSY,
P_EVENT_GCCS_CALLED_PARTY_UNREACHABLE,
P_EVENT_GCCS_NO_ANSWER_FROM_CALLED_PARTY,
P_EVENT_GCCS_ROUTE_SELECT_FAILURE
}

P_DYNAMIC_EVENT_TYPES = {
P_CALL_REPORT_ANSWER,
P_CALL_REPORT_BUSY,
P_CALL_REPORT_NO_ANSWER,
P_CALL_REPORT_DISCONNECT,
P_CALL_REPORT_SERVICE_CODE,
P_CALL_REPORT_ROUTING_FAILURE,
P_CALL_REPORT_NOT_REACHABLE
}

P_ADDRESS_PLAN = {
P_ADDRESS_PLAN_E164
}

P_UI_CALL_BASED = {
TRUE
}

P_UI_AT_ALL_STAGES = {
FALSE
}

P_MEDIA_TYPE = {
P_AUDIO
}
```

End of Change in Clause 8

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040623**
Meeting #28, Piscataway, New Jersey, USA, 09-13 August 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-04-3**	**CR** **029**	⌘ **rev**	**-**	⌘	Current version: **6.2.0**	⌘

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:	⌘	Remove unused Deprecated items		
Source:	⌘	CN5 Ultan Mulligan, ETSI PTCC		
Work item code:	⌘	OSA3	***Date:*** ⌘	27/08/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	3GPP CN5 uses a procedure of deprecation in its specifications to identify methods and other items which cannot be used or have been replaced. Items are deprecated when they have been found not to work, or when replaced in a controlled, backwards compatible manner. Such deprecated items should be removed one day from the specifications: 3GPP CN5 keeps such deprecated items in its specifications for 2 full releases in order to preserve backwards compatibility. At the delivery of Release 6, it is time to remove those items from the Release 6 specifications which are deprecated in the current Release 4 specifications and the Release 5 specifications. This removal has no impact on the Release 4 or Release 5 specifications - backwards compatibility is preserved for these.
Summary of change:	⌘	Delete the deprecated Service Property P_TRIGGERING_ADDRESSES
Consequences if not approved:	⌘	Old, unusable items will continue to clutter up the OSA specifications. The non-removal of deprecated items will encourage developers to implement them, when in fact deprecation of an item is intended to do the exact opposite. Furthermore, vendors will continue to be obliged to retain these deprecated items in their implementations, forcing them to continue to include these often broken items in their test plans.

Clauses affected:	⌘	8				
		Y	**N**			
Other specs affected:	⌘	**X**		Other core specifications	⌘	TS 29.198-1, -3, -4-2, -5, -8 and -11
			X	Test specifications		
			X	O&M Specifications		
Other comments:	⌘					

How to create CRs using this form:

Change in Clause 8

8 Multi-Party Call Control Service Properties

8.1 List of Service Properties

The following table lists properties relevant for the MPCC API.

Property	Type	Description / Interpretation
P_TRIGGERING_EVENT_TYPES	INTEGER_SET	Indicates the static event types supported by the SCS. Static events are the events by which applications are initiated.
P_DYNAMIC_EVENT_TYPES	INTEGER_SET	Indicates the dynamic event types supported by the SCS. Dynamic events are the events the application can request for during the context of a call.
P_ADDRESSPLAN	INTEGER_SET	Indicates the supported address plans (defined in TpAddressPlan.) e.g. {P_ADDRESS_PLAN_E164, P_ADDRESS_PLAN_IP}). Note that more than one address plan may be supported.
P_UI_CALL_BASED	BOOLEAN_SET	Value = TRUE : User interaction can be performed on call level and a reference to a Call object can be used in the IpUIManager.createUICall() operation. Value = FALSE: No User interaction on call level is supported.
P_UI_AT_ALL_STAGES	BOOLEAN_SET	Value = TRUE: User Interaction can be performed at any stage during a call . Value = FALSE: User Interaction can be performed in case there is only one party in the call.
P_MEDIA_TYPE	INTEGER_SET	Specifies the media type used by the Service. Values are defined by data-type TpMediaType : P_AUDIO, P_VIDEO, P_DATA
P_MAX_CALLLEGS_PER_CALL	INTEGER_SET	Indicates the maximum number of legs in a call for which a connection to a call party exists in the network. The enforcement of this property is done only when a leg is created or routed by the application.
P_UI_CALLLEG_BASED	BOOLEAN_SET	Value = TRUE : User interaction can be performed on leg level and a reference to a CallLeg object can be used in the IpUIManager.createUICall() operation. Value = FALSE : No user interaction on leg level is supported.
P_CALLLEG_PROPERTIES	STRING_SET	Indicates which of the user identity fields are available, valid values are given by TpCallLegPropertiesName.
P_PARALLEL_INITIAL_ROUTING_REQUESTS	BOOLEAN_SET	Indicates whether for application initiated calls it is possible to issue multiple routing request methods in parallel or that the application has to wait for the result of the first request before another one can be invoked. Value = TRUE: Multiple routing requests can be invoked in parallel. Value = FALSE: Result of first request has to be received before another request can be issued.

The previous table lists properties related to capabilities of the SCS itself. The following table lists properties that are used in the context of the Service Level Agreement, e.g. to restrict the access of applications to the capabilities of the SCS.

Property	Type	Description
~~P_TRIGGERING_ADDRESSES (Deprecated)~~	~~ADDRESSRANGE_SET~~	~~Indicates for which numbers the notification may be set. For terminating notifications it applies to the terminating number, for originating notifications it applies only to the originating number. See further explanation on which events are originating and which are terminating, below.~~
P_NOTIFICATION_ADDRESS_RANGES	XML_ADDRESS_RANGE_SET	Indicates for which numbers notifications may be set. More than one range may be present. For terminating notifications they apply to the terminating number, for originating notifications they apply only to the originating number.
P_MONITOR_MODE	INTEGER_SET	Indicates whether the application is allowed to monitor in interrupt and/or notify mode. Set is: P_INTERRUPT P_NOTIFY
P_NUMBERS_TO_BE_CHANGED	INTEGER_SET	Indicates which numbers the application is allowed to change or fill for legs in an incoming call. Allowed value set: {P_ORIGINAL_CALLED_PARTY_NUMBER, P_REDIRECTING_NUMBER, P_TARGET_NUMBER, P_CALLING_PARTY_NUMBER}.
P_CHARGEPLAN_ALLOWED	INTEGER_SET	Indicates which charging is allowed in the setCallChargePlan indicator. Allowed values: {P_TRANSPARANT_CHARGING, P_CHARGE_PLAN}
P_CHARGEPLAN_MAPPING	INTEGER_INTEGER_MAP	Indicates the mapping of chargeplans (we assume they can be indicated with integers) to a logical network chargeplan indicator. When the chargeplan supports indicates P_CHARGE_PLAN then only chargeplans in this mapping are allowed.
P_HIGH_PROBABILITY_OF_COMPLETION	BOOLEAN_SET	Value = TRUE : high probability of call completion field can be set. Value = FALSE : high probability of call completion field can not be set. FALSE is the default value.

The following table explains how the P_TRIGGERING_ADDRESSES property that is inherited via the Generic Call Control properties should be interpreted with respect to which of the notifications apply to originating numbers and which of the notifications apply to terminating numbers.

P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT	Originating
P_CALL_EVENT_ORIGINATING_CALL_ATTEMPT_AUTHORISED	Originating
P_CALL_EVENT_ADDRESS_COLLECTED	Originating
P_CALL_EVENT_ADDRESS_ANALYSED	Originating
P_CALL_EVENT_ORIGINATING_SERVICE_CODE	Originating
P_CALL_EVENT_ORIGINATING_RELEASE	Originating
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT	Terminating
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT_AUTHORISED	Terminating
P_CALL_EVENT_ALERTING	Terminating
P_CALL_EVENT_ANSWER	Terminating
P_CALL_EVENT_TERMINATING_RELEASE	Terminating
P_CALL_EVENT_REDIRECTED	Terminating
P_CALL_EVENT_TERMINATING_SERVICE_CODE	Terminating
P_CALL_EVENT_QUEUED	N/A

8.2 Service Property values for the CAMEL Service Environment.

Implementations of the MultiParty Call Control API relying on the CSE of CAMEL phase 4 shall have the Service Properties outlined above set to the indicated values :

```
P_OPERATION_SET = {
“IpMultiPartyCallControlManager.createCall”,
“IpMultiPartyCallControlManager.createNotification”,
“IpMultiPartyCallControlManager.destroyNotification”,
“IpMultiPartyCallControlManager.changeNotification”,
“IpMultiPartyCallControlManager.getNotification”,
“IpMultiPartyCallControlManager.getNextNotification”,
“IpMultiPartyCallControlManager.enableNotifications”,
“IpMultiPartyCallControlManager.disableNotifications”,
“IpMultiPartyCallControlManager.setCallLoadControl”
“IpMultiPartyCall.getCallLegs”,
“IpMultiPartyCall.createCallLeg”,
“IpMultiPartyCall.createAndRouteCallLegReq”,
“IpMultiPartyCall.release”,
“IpMultiPartyCall.deassignCall”,
“IpMultiPartyCall.getInfoReq”,
“IpMultiPartyCall.setChargePlan”,
“IpMultiPartyCall.setAdviceOfCharge”,
“IpMultiPartyCall.superviseReq”,
“IpCallLeg.routeReq”,
“IpCallLeg.eventReportReq”,
“IpCallLeg.release”,
“IpCallLeg.getInfoReq”,
“IpCallLeg.getCall”,
“IpCallLeg.continueProcessing”
}

P_TRIGGERING_EVENT_TYPES = {
P_CALL_EVENT_ADDRESS_COLLECTED,
P_CALL_EVENT_ADDRESS_ANALYSED,
P_CALL_EVENT_ORIGINATING_RELEASE,
P_CALL_EVENT_TERMINATING_CALL_ATTEMPT_AUTHORISED,
P_CALL_EVENT_TERMINATING_RELEASE
}
```

Note: P_CALL_EVENT_ORIGINATING_RELEASE only for the routing failure case, TpReleaseCause = `P_ROUTING_FAILURE`

```
P_DYNAMIC_EVENT_TYPES = {
P_CALL_EVENT_ALERTING,
P_CALL_EVENT_ANSWER,
P_CALL_EVENT_ORIGINATING_RELEASE,
P_CALL_EVENT_ORIGINATING_SERVICE_CODE,
P_CALL_EVENT_TERMINATING_RELEASE,
P_CALL_EVENT_TERMINATING_SERVICE_CODE
}

P_ADDRESS_PLAN = {
P_ADDRESS_PLAN_E164
}

P_UI_CALL_BASED = {
TRUE
}

P_UI_AT_ALL_STAGES = {
FALSE
}

P_MEDIA_TYPE = {
P_AUDIO
}

P_MAX_CALLLEGS_PER_CALL = {
1,
2,
3,
4,
5,
6
}

P_UI_CALLLEG_BASED = {
TRUE
}

P_MEDIA_ATTACH_EXPLICIT = {
```

```
FALSE
}
```

End of Change in Clause 8

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-05**	**CR**	**053**	⌘ **rev**	**-**	⌘ Current version:	**6.1.0** ⌘

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

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

Title:	⌘	Remove unused Deprecated items		
Source:	⌘	CN5 Ultan Mulligan, ETSI PTCC		
Work item code:	⌘	OSA3	***Date:*** ⌘	27/08/2004
Category:	⌘	**F**	***Release:*** ⌘	*REL-6*

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change: ⌘ 3GPP CN5 uses a procedure of deprecation in its specifications to identify methods and other items which cannot be used or have been replaced. Items are deprecated when they have been found not to work, or when replaced in a controlled, backwards compatible manner.
Such deprecated items should be removed one day from the specifications: 3GPP CN5 keeps such deprecated items in its specifications for 2 full releases in order to preserve backwards compatibility.
At the delivery of Release 6, it is time to remove those items from the Release 6 specifications which are deprecated in the current Release 4 specifications and the Release 5 specifications.
This removal has no impact on the Release 4 or Release 5 specifications - backwards compatibility is preserved for these.

Summary of change: ⌘ Delete the deprecated Service Property P_TRIGGERING_ADDRESSES, the deprecated method reportNotification() and the type TpUIEventInfo

Consequences if not approved: ⌘ Old, unusable items will continue to clutter up the OSA specifications. The non-removal of deprecated items will encourage developers to implement them, when in fact deprecation of an item is intended to do the exact opposite.
Furthermore, vendors will continue to be obliged to retain these deprecated items in their implementations, forcing them to continue to include these often broken items in their test plans.

Clauses affected: ⌘ 4, 5, 8, 9, 10, 11

		Y	N			
Other specs affected:	⌘	X		Other core specifications	⌘	29.198-1, -3, -4-2, -4-3, -8 and -11
			X	Test specifications		
			X	O&M Specifications		

Other comments: ⌘

How to create CRs using this form:

Start of change in Clause 4

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
reportEventNotification	sendInfoAndCollectRes
userInteractionAborted	sendInfoAndCollectErr
userInteractionNotificationInterrupted	release
userInteractionNotificationContinued	userInteractionFaultDetected
changeNotification	
getNotification	
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~~ Generic 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 and Generic User Interaction Administration Service Capability Features (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 of this specification.

4.4~~1~~ General requirements on support of methods

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

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

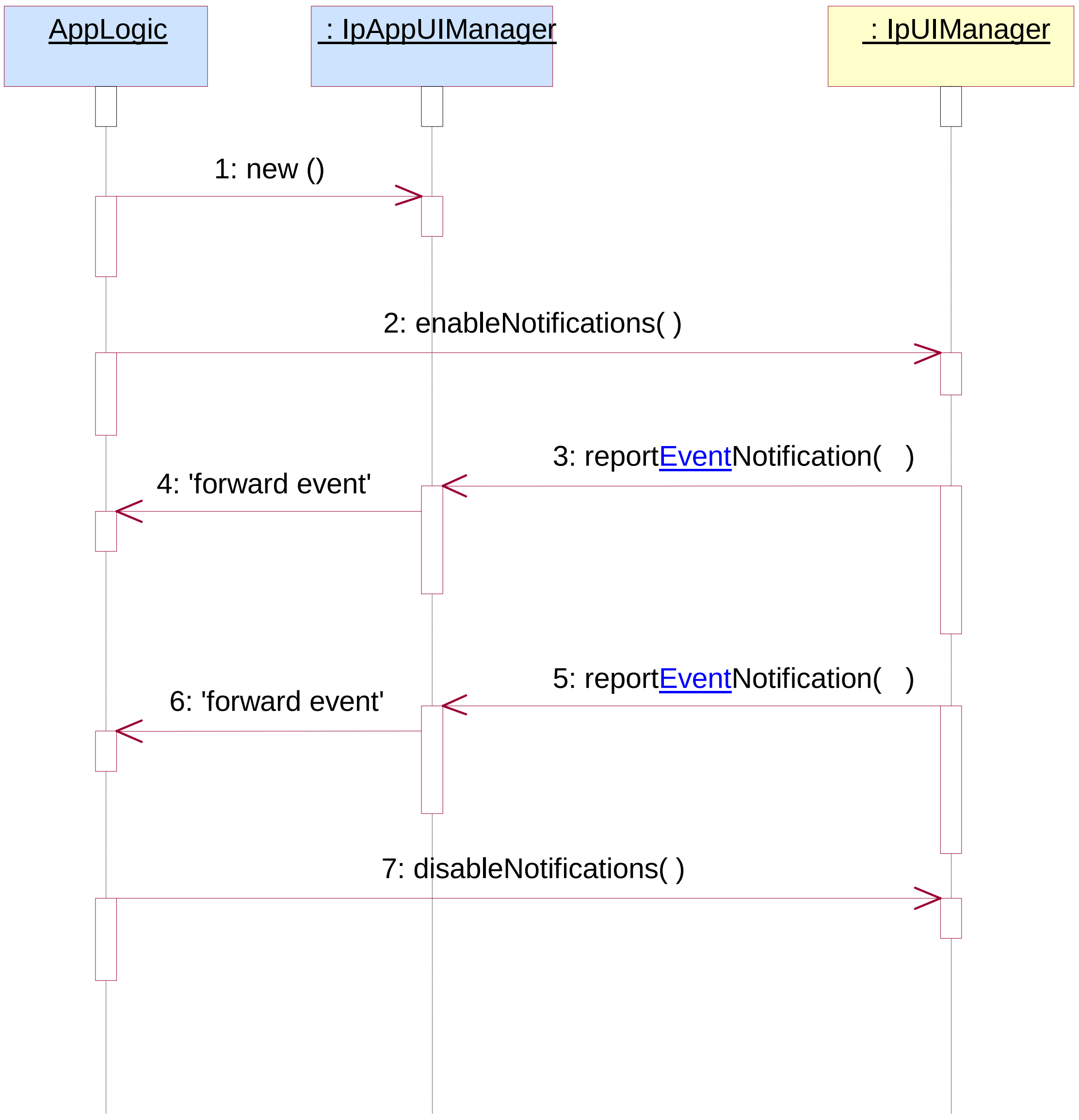
Where a method is not supported by an implementation of an Application interface, a call to that method shall be possible, and no exception shall be returned.

End of change in Clause 4

Start of change in Clause 5

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.


AppLogic
: IpAppUIManager
: IpUIManager
1: new ()
2: enableNotifications( )
3: reportEventNotification( )
4: 'forward event'
5: reportEventNotification( )
6: 'forward event'
7: disableNotifications( )

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.

End of change in Clause 5

Start of change in Clause 8

8.1.1.7 Method enableNotifications()

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

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

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

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

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

Parameters

`appUIManager : in IpAppUIManagerRef`

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

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions`

8.1.1.8 Method disableNotifications()

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

Parameters

No Parameters were identified for this method

Raises

TpCommonExceptions

8.1.4 Interface Class IpAppUIManager

Inherits from: IpInterface.

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

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

8.1.4.1 Method userInteractionAborted()

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

Parameters

userInteraction : in TpUIIdentifier

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

~~8.1.4.2Method <<deprecated>> reportNotification()~~

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

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

~~Returns: appUI~~

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

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

~~*Parameters*~~

~~**`userInteraction : in TpUIIdentifier`**~~

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

~~**`eventInfo : in TpUIEventInfo`**~~

~~Specifies data associated with this event.~~

~~**`assignmentID : in TpAssignmentID`**~~

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

~~*Returns*~~

~~**`IpAppUIRef`**~~

~~8.1.4.3~~8.1.4.2 Method userInteractionNotificationInterrupted()

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

Parameters

No Parameters were identified for this method

~~8.1.4.4~~8.1.4.3 Method userInteractionNotificationContinued()

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

Parameters

No Parameters were identified for this method

~~8.1.4.5~~8.1.4.4 Method reportEventNotification()

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

Returns: appUI

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

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

Parameters

`userInteraction : in TpUIIdentifier`

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

`eventNotificationInfo : in TpUIEventNotificationInfo`

Specifies data associated with this event.

`assignmentID : in TpAssignmentID`

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

Returns

`IpAppUIRef`

End of change in Clause 8

Start of change in Clause 9

9 State Transition Diagrams

9.1 Generic and Call User Interaction State Transition Diagrams

9.1.3 State Transition Diagrams for IpUIManager

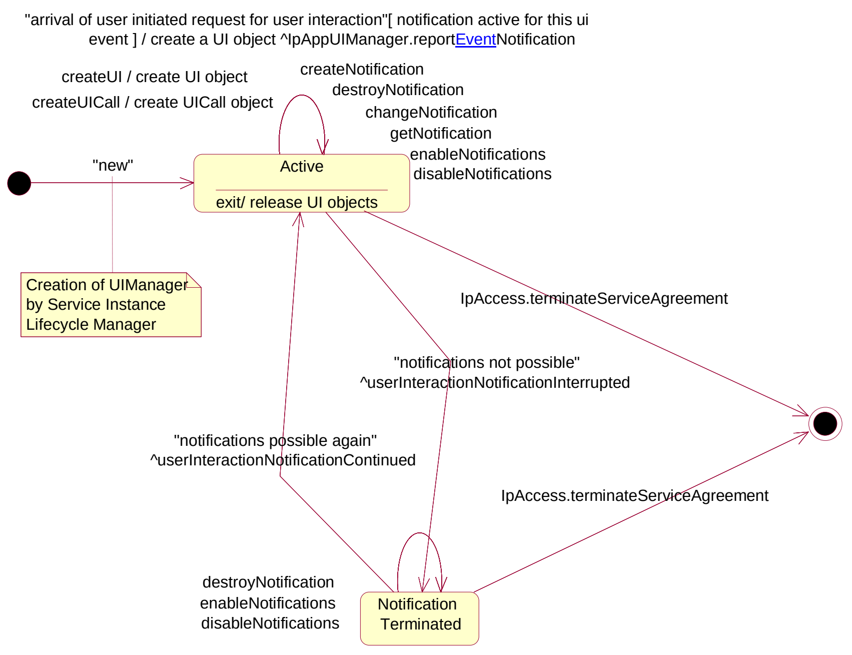

"arrival of user initiated request for user interaction"[notification active for this ui event] / create a UI object ^IpAppUIManager.reportEventNotification
createUI / create UI object
createUICall / create UICall object
createNotification
destroyNotification
changeNotification
getNotification
enableNotifications
disableNotifications
"new"
Active
exit/ release UI objects
Creation of UIManager by Service Instance Lifecycle Manager
IpAccess.terminateServiceAgreement
"notifications not possible" ^userInteractionNotificationInterrupted
"notifications possible again" ^userInteractionNotificationContinued
IpAccess.terminateServiceAgreement
destroyNotification
enableNotifications
disableNotifications
Notification Terminated

Figure : Application view on the UI Manager

9.1.3.1 Active State

In this state a relation between the Application and a User Interaction Service Capability Feature (Generic User Interaction or Call User Interaction) has been established. The application is now able to request creation of UI and/or UICall objects.

9.1.3.2 Notification Terminated State

When the UI manager is in the Notification terminated state, events requested with createNotification()/enableNotifications() will not be forwarded to the application. There can be multiple reasons for this: for instance it might be that the application receives more notifications than defined in the Service Level Agreement. Another example is that the SCS has detected it receives no notifications from the network due to e.g. a link failure. In this state no requests for new notifications will be accepted.

End of change in Clause 9

Start of changes in Clauses 10 & 11

10 Service Properties

10.1 User Interaction Service Properties

The following table lists properties relevant for the User Interaction API.

Property	Type	Description
P_INFO_TYPE	INTEGER_SET	Specifies whether the UI SCS supports text or URLs etc. Allowed values are defined by TpUIInfoType.
P_SPEECH_RECOGNITION_SUPPORTED	BOOLEAN	Value: TRUE when the speech recognition features are supported

The previous table lists properties related to capabilities of the SCS itself. The following table lists properties that are used in the context of the Service Level Agreement, e.g. to restrict the access of applications to the capabilities of the SCS.

Property	Type	Description
~~P_TRIGGERING_ADDRESSES (Deprecated)~~	~~ADDRESSRANGE_SET~~	10 ~~Specifies which numbers the notification may be set~~
P_SERVICE_CODE	INTEGER_SET	Specifies the service codes that may be used for notification requests.
P_NOTIFICATION_ADDRESS_RANGES	XML_ADDRESS_RANGE_SET	Indicates for which numbers notifications may be set. More than one range may be present.

11 Data Definitions

The following data types referenced in this clause are defined in 3GPP TS 29.198-4:

```
TpCallIdentifier
TpMultiPartyCallIdentifier
TpCallLegIdentifier
```

All other data types referenced but not defined in this clause are common data definitions which may be found in 3GPP TS 29.198-2.

11.1 TpUIFault

Defines the cause of the UI fault detected.

Name	Value	Description
P_UI_FAULT_UNDEFINED	0	Undefined
P_UI_CALL_ENDED	1	The related Call object has been terminated. Therefore, the UICall object is also terminated. No further interaction is possible with this object.

11.2 IpUI

Defines the address of an `IpUI` Interface.

11.3 IpUIRef

Defines a `Reference` to type `IpUI`.

11.4 IpAppUI

Defines the address of an `IpAppUI` Interface.

11.5 IpAppUIRef

Defines a `Reference` to type `IpAppUI`.

11.6 IpAppUIManager

Defines the address of an `IpAppUIManager` Interface.

11.7 IpAppUIManagerRef

Defines a `Reference` to type `IpAppUIManager.`

11.8 TpUICallIdentifier

Defines the Sequence of Data Elements that unambiguously specify the UICall object.

Structure Element Name	Structure Element Type	Structure Element Description
UICallRef	IpUICallRef	This element specifies the interface reference for the UICall object.
UserInteractionSessionID	TpSessionID	This element specifies the User Interaction session ID.

11.9 TpUICollectCriteria

Defines the `Sequence of Data Elements` that specify the additional properties for the collection of information, such as the end character, first character timeout, inter-character timeout, and maximum interaction time. The CollectMode element defines the type of data that is to be collected. DTMF and Voice Recognition can be used separately or in combination. The P_SPEECH_RECOGNITION_SUPPORTED property defines whether the voice recognition features are supported.

Structure Element Name	Structure Element Type
MinLength	TpInt32
MaxLength	TpInt32
EndSequence	TpString
StartTimeout	TpDuration
InterCharTimeout	TpDuration
CollectMode	TpUICollectMode
RecognitionCriteria	TpUIRecognitionCriteria

The structure elements specify the following criteria:

`MinLength:`	Defines the minimum number of characters (e.g. digits) to collect. Applies to DTMF collection and voice recognition.
`MaxLength:`	Defines the maximum number of characters (e.g. digits) to collect. Applies to DTMF collection and voice recognition.
`EndSequence:`	Defines the character or characters which terminate an input of variable length, e.g. phone numbers. Applies to DTMF collection only.
`StartTimeout:`	specifies the value for the first character time-out timer. The timer is started when the announcement has been completed or has been interrupted. The user should enter the start of the response (e.g. first digit) before the timer expires. If the start of the response is not entered before the timer expires, the input is regarded to be erroneous. After receipt of the start of the response, which may be valid or invalid, the timer is stopped. Applies to DTMF collection and voice recognition.
`InterCharTimeOut:`	specifies the value for the inter-character time-out timer. The timer is started when a response (e.g. digit) is received, and is reset and restarted when a subsequent response is received. The responses may be valid or invalid. the announcement has been completed or has been interrupted. Applies to DTMF collection only.
`CollectMode:`	Defines the type of collection to do. Applies to DTMF collection and voice recognition. The default is DTMF collection only.
`RecognitionCriteria:`	Defines the criteria for voice recognition.

Input is considered successful if the following applies:

If the `EndSequence` is not present (i.e. an empty string):

- when the `InterCharTimeOut` timer expires; or

- when the number of valid digits received equals the `MaxLength`.

If the `EndSequence` is present:

- when the `InterCharTimeOut` timer expires; or

- when the `EndSequence` is received; or

- when the number of valid digits received equals the `MaxLength`.

In the case the number of valid characters received is less than the `MinLength` when the `InterCharTimeOut` timer expires or when the `EndSequence` is received, the input is considered erroneous.

The collected characters (including the `EndSequence`) are sent to the client application when input has been successful.

11.10 TpUIError

Defines the UI error codes.

Name	Value	Description
`P_UI_ERROR_UNDEFINED`	0	Undefined error
`P_UI_ERROR_ILLEGAL_INFO`	1	The specified information (InfoId, InfoData, or InfoAddress) is invalid
`P_UI_ERROR_ID_NOT_FOUND`	2	A legal InfoId is not known to the User Interaction service
`P_UI_ERROR_RESOURCE_UNAVAILABLE`	3	The information resources used by the User Interaction service are unavailable, e.g. due to an overload situation.
`P_UI_ERROR_ILLEGAL_RANGE`	4	The values for minimum and maximum collection length are out of range
`P_UI_ERROR_IMPROPER_USER_RESPONSE`	5	Improper user response
`P_UI_ERROR_ABANDON`	6	The specified leg is disconnected before the send information completed
`P_UI_ERROR_NO_OPERATION_ACTIVE`	7	There is no active User Interaction for the specified leg. Either the application did not start any User Interaction or the User Interaction was already finished when the `abortActionReq()` was called.
`P_UI_ERROR_NO_SPACE_AVAILABLE`	8	There is no more storage capacity to record the message when the `recordMessageReq()` operation was called
`P_UI_ERROR_RESOURCE_TIMEOUT`	9	The request has been accepted by the resource but it did not report a result.

The call User Interaction object will be automatically de-assigned if the error P_UI_ERROR_ABANDON is reported, as a corresponding call or call leg object no longer exists.

11.11 TpUIEventCriteria

Defines the `Sequence of Data Elements` that specify the additional criteria for receiving a UI notification

Structure Element Name	Structure Element Type	Description
`OriginatingAddress`	`TpAddressRange`	Defines the originating address for which the notification is requested.
`DestinationAddress`	`TpAddressRange`	Defines the destination address or address range for which the notification is requested.
`ServiceCode`	`TpString`	Defines a 2-digit code indicating the UI to be triggered. The value is operator specific.

11.12 TpUIEventCriteriaResultSet

Defines a set of TpUIEventCriteriaResult.

11.13 TpUIEventCriteriaResult

Defines a sequence of data elements that specify a requested event notification criteria with the associated assignmentID.

Structure Element Name	Structure Element Type	Structure Element Description
`EventCriteria`	`TpUIEventCriteria`	The event criteria that were specified by the application.
`AssignmentID`	`TpInt32`	The associated assignmentID. This can be used to disable the notification.

~~11.14TpUIEventInfo~~

~~Defines the Sequence of Data Elements that specify a UI notification~~

~~Structure Element Name~~	~~Structure Element Type~~	~~Structure Element Description~~
~~OriginatingAddress~~	~~TpAddress~~	~~Defines the originating address.~~
~~DestinationAddress~~	~~TpAddress~~	~~Defines the destination address.~~
~~ServiceCode~~	~~TpString~~	~~Defines a 2-digit code indicating the UI to be triggered. The value is operator specific.~~
~~DataTypeIndication~~	~~TpUIEventInfoDataType~~	~~Identifies the type of contents in DataString.~~
~~DataString~~	~~TpString~~	~~Freely defined data string with a limited length e.g. 160 bytes according to the network policy.~~

~~11.15~~11.14 TpUIEventInfoDataType

Defines the type of the dataString parameter in the method userInteractionEventNotify.

Name	Value	Description
P_UI_EVENT_DATA_TYPE_UNDEFINED	0	Undefined (e.g. binary data)
P_UI_EVENT_DATA_TYPE_UNSPECIFIED	1	Unspecified data
P_UI_EVENT_DATA_TYPE_TEXT	2	Text
P_UI_EVENT_DATA_TYPE_USSD_DATA	3	USSD data starting with coding scheme

~~11.16~~11.15 TpUIIdentifier

Defines the Sequence of Data Elements that unambiguously specify the UI object

Structure Element Name	Structure Element Type	Structure Element Description
UIRef	IpUIRef	This element specifies the interface reference for the UI object.
UserInteractionSessionID	TpSessionID	This element specifies the User Interaction session ID.

~~11.17~~11.16 TpUIInfo

Defines the Tagged Choice of Data Elements that specify the information to send to the user.

	Tag Element Type	
	TpUIInfoType	

Tag Element Value	Choice Element Type	Choice Element Name
P_UI_INFO_ID	TpInt32	InfoId
P_UI_INFO_DATA	TpString	InfoData
P_UI_INFO_ADDRESS	TpURL	InfoAddress
P_UI_INFO_BIN_DATA	TpOctetSet	InfoBinData
P_UI_INFO_UUENCODED	TpString	InfoUUEncData
P_UI_INFO_MIME	TpOctetSet	InfoMimeData
P_UI_INFO_WAVE	TpOctetSet	InfoWaveData
P_UI_INFO_AU	TpOctetSet	InfoAuData
P_UI_INFO_VXML	TpString	InfoVXMLData
P_UI_INFO_SYNTHESIS	TpUISynthesisInfoData	InfoSynthData

The choice elements represent the following:

`InfoID:`	defines the ID of the user information script or stream to send to an end-user. The values of this data type are operator specific.
`InfoData:`	defines the data to be sent to an end-user's terminal. The data is free-format and the encoding is depending on the resources being used..
`InfoAddress:`	defines the URL of the text or stream to be sent to an end-user's terminal.
`InfoBinData:`	defines the binary data to be sent to an end-user's terminal. The data is a free-format, 8-bit quantity that is guaranteed not to undergo any conversion when transmitted.
`InfoUUEncData:`	defines the UUEncoded data to be sent to an end-user's terminal.
`InfoMimeData:`	defines the MIME data to be sent to an end-user's terminal.
`InfoWaveData:`	defines the WAVE data to be sent to an end-user's terminal.
`InfoAuData:`	defines the AU data to be sent to an end-user's terminal.
`InfoVXMLData:`	defines the TpString that describes the VXML (Voice XML) page that is sent to the server for execution and interaction with the end-user. See http://www.w3.org/TR/2000/NOTE-voicexml-20000505/ for more information.
`InfoSynthData:`	defines the TpUISynthesisInfoData that describes the content and how the speech synthesis will be done.

InfoSynthData allows the application to utilize the fundamental speech synthesis capabilities of the server without dependency VXML, while InfoVXMLData allows the application to send a complex VXML program (including call control, flow control, dynamic content, menuing, etc) to the server for execution with little change to the OSA application itself.

~~11.18~~11.17 TpUIInfoType

Defines the type of the information to be sent to the user.

Name	Value	Description
P_UI_INFO_ID	0	The information to be send to an end-user consists of an ID
P_UI_INFO_DATA	1	The information to be send to an end-user consists of a data string
P_UI_INFO_ADDRESS	2	The information to be send to an end-user consists of a URL.
P_UI_INFO_BIN_DATA	3	The information to be sent to an end-user consists of a 8 bit binary data set
P_UI_INFO_UUENCODED	4	The information to be sent to an end-user consists of UUEncoded data
P_UI_INFO_MIME	5	The information to be sent to the end-user consists of MIME encoded data
P_UI_INFO_WAVE	6	The information to be sent to the end-user is .wav waveform data
P_UI_INFO_AU	7	The information to be sent to the end-user is .au audio data
P_UI_INFO_VXML	8	The information to be sent to the end-user is controlled by this VXML
P_UI_INFO_SYNTHESIS	9	The information to be sent to an end-user is synthesized from text.

~~11.19~~11.18 TpUIMessageCriteria

Defines the `Sequence of Data Elements` that specify the additional properties for the recording of a message.

Structure Element Name	Structure Element Type
EndSequence	TpString
MaxMessageTime	TpDuration
MaxMessageSize	TpInt32

The structure elements specify the following criteria:

`EndSequence:`	Defines the character or characters which terminate an input of variable length, e.g. phone numbers.
`MaxMessageTime:`	specifies the maximum duration in seconds of the message that is to be recorded.
`MaxMessageSize:`	If this parameter is non-zero, it specifies the maximum size in bytes of the message that is to be recorded.

~~11.20~~11.19 TpUIReport

Defines the UI reports if a response was requested.

Name	Value	Description
P_UI_REPORT_UNDEFINED	0	Undefined report
P_UI_REPORT_INFO_SENT	1	Confirmation that the information has been sent
P_UI_REPORT_INFO_COLLECTED	2	Information collected., meeting the specified criteria.
P_UI_REPORT_NO_INPUT	3	No information collected. The user immediately entered the delimiter character. No valid information has been returned
P_UI_REPORT_TIMEOUT	4	No information collected. The user did not input any response before the input timeout expired
P_UI_REPORT_MESSAGE_STORED	5	A message has been stored successfully
P_UI_REPORT_MESSAGE_NOT_STORED	6	The message has not been stored successfully
P_UI_REPORT_MESSAGE_DELETED	7	A message has been deleted successfully
P_UI_REPORT_MESSAGE_NOT_DELETED	8	A message has not been deleted successfully

~~11.21~~11.20 TpUIResponseRequest

Defines the situations for which a response is expected following the User Interaction.

Name	Value	Description
P_UI_RESPONSE_REQUIRED	1	The User Interaction Call shall send a response when the request has completed.
P_UI_LAST_ANNOUNCEMENT_IN_A_ROW	2	This is the final announcement within a sequence. It might, however, be that additional announcements will be requested at a later moment. The User Interaction Call service may release any used resources in the network. The UI object will not be released.
P_UI_FINAL_REQUEST	4	This is the final request. The UI object will be released after the information has been presented to the user.

This parameter represents a so-called bitmask, i.e. the values can be added to derived the final meaning.

~~11.22~~11.21 TpUITargetObjectType

Defines the type of object where User Interaction should be performed upon.

Name	Value	Description
P_UI_TARGET_OBJECT_CALL	0	User-interaction will be performed on a complete Call.
P_UI_TARGET_OBJECT_MULTI_PARTY_CALL	1	User-interaction will be performed on a complete Multi-party Call.
P_UI_TARGET_OBJECT_CALL_LEG	2	User-interaction will be performed on a single Call Leg. The media of this call leg should be detached at the moment any user interaction is done.

~~11.23~~11.22 TpUITargetObject

Defines the `Tagged Choice of Data Elements` that specify the object to perform User Interaction on.

	Tag Element Type	
	TpUITargetObjectType	

Tag Element Value	Choice Element Type	Choice Element Name
P_UI_TARGET_OBJECT_CALL	TpCallIdentifier	Call
P_UI_TARGET_OBJECT_MULTI_PARTY_CALL	TpMultiPartyCallIdentifier	MultiPartyCall
P_UI_TARGET_OBJECT_CALL_LEG	TpCallLegIdentifier	CallLeg

~~11.24~~11.23 TpUIVariableInfo

Defines the `Tagged Choice of Data Elements` that specify the variable parts in the information to send to the user.

	Tag Element Type	
	TpUIVariablePartType	

Tag Element Value	Choice Element Type	Choice Element Name
P_UI_VARIABLE_PART_INT	TpInt32	VariablePartInteger
P_UI_VARIABLE_PART_ADDRESS	TpString	VariablePartAddress
P_UI_VARIABLE_PART_TIME	TpTime	VariablePartTime
P_UI_VARIABLE_PART_DATE	TpDate	VariablePartDate
P_UI_VARIABLE_PART_PRICE	TpPrice	VariablePartPrice

~~11.25~~11.24 TpUIVariableInfoSet

Defines a `Numbered Set of Data Elements` of TpUIVariableInfo.

~~11.26~~11.25 TpUIVariablePartType

Defines the type of the variable parts in the information to send to the user.

Name	Value	Description
P_UI_VARIABLE_PART_INT	0	Variable part is of type integer
P_UI_VARIABLE_PART_ADDRESS	1	Variable part is of type address
P_UI_VARIABLE_PART_TIME	2	Variable part is of type time
P_UI_VARIABLE_PART_DATE	3	Variable part is of type date
P_UI_VARIABLE_PART_PRICE	4	Variable part is of type price

~~11.27~~11.26 TpUIEventNotificationInfo

Defines the `Sequence of Data Elements` that specify a UI event notification

Structure Element Name	Structure Element Type	Structure Element Description
`OriginatingAddress`	`TpAddress`	Defines the originating address.
`DestinationAddress`	`TpAddress`	Defines the destination address.
`ServiceCode`	`TpString`	Defines a 2-digit code indicating the UI to be triggered. The value is operator specific.
`DataTypeIndication`	`TpUIEventInfoDataType`	Identifies the type of contents in UIEventData
`UIEventData`	`TpOctetSet`	Freely defined data according to the network policy. e.g 7 bit USSD encoded

~~11.28~~11.27 TpUISynthesisInfoData

Defines the `Sequence of Data Elements` that specify the information to use in generating the desired effects on the generated voice. The speech synthesis parameters or processing tags will be interpreted as hints and may be ignored by the speech synthesis engine. Note that the language is specified on the sendInfoReq() and sendInfoAndCollectReq() method calls.

The TextData field may contain the following tags to affect the processing of the text. The <break> tag specifies a timing pause. The <emp> tag specifies an emphasis on a word or phrase.

```
<break size="milliseconds">      // specifies a timing pause in the
                                 // middle of the text in milliseconds
<emp>A word</emp>               // apply emphasis to a word or words
```

Structure Element Name	Structure Element Type	Structure Element Description
`SpeakerGender`	`TpUISynthesisGender`	Defines the gender of the speaker.
`SpeakerAge`	`TpUISynthesisAge`	Defines the age of the speaker.
`SpeakerRate`	`TpUISynthesisRate`	Defines the rate of the speaker.
`SpeakerRange`	`TpUISynthesisRange`	Defines the range of the speaker.
`TextData`	`TpString`	Defines the text to synthesize into speech.
`WordOverrideSet`	`TpUIWordOverrideSet`	Defines the pronunciation overrides.

~~11.29~~11.28 TpUISynthesisGender

Defines the UI reports if a response was requested.

Name	Value	Description
`P_UI_GENDER_MALE`	`0`	Male
`P_UI_GENDER_FEMALE`	`1`	Female

~~11.30~~11.29 TpUISynthesisAge

Defines the UI reports if a response was requested.

Name	Value	Description
P_UI_AGE_CHILD	0	Child voice
P_UI_AGE_YOUNG_ADULT	1	Young adults voice
P_UI_AGE_ADULT	2	Adult voice
P_UI_AGE_OLDER_ADULT	3	Older adult voice

~~11.31~~11.30 TpUISynthesisRate

Defines the rate of the speech.

Name	Value	Description
P_UI_RATE_SLOW	0	Slow speech rate
P_UI_RATE_AVERAGE	1	Average speech rate
P_UI_RATE_FAST	2	Fast speech rate

~~11.32~~11.31 TpUISynthesisRange

Defines the range or liveliness of the speech.

Name	Value	Description
P_UI_RANGE_CALMER	0	Very Calm or monotone speech
P_UI_RANGE_CALM	1	Moderately calm speech
P_UI_RANGE_AVERAGE	2	Average speech
P_UI_RANGE_EXCITED	3	Moderately excited or lively speech
P_UI_RANGE_MORE_EXCITED	4	Excited or lively speech

~~11.33~~11.32 TpUIWordOverrideSet

Defines a `Numbered Set of Data Elements` of TpUIWordOverride.

~~11.34~~11.33 TpUIWordOverride

Defines the `Sequence of Data Elements` that specify the information to use in overriding the default pronunciation of a word.

Structure Element Name	Structure Element Type	Structure Element Description
Spelling	TpString	Defines the spelling of the word override.
PronounceType	TpUIPronounceType	Defines the type of pronunciation syntax.
PronounceAs	TpString	Defines how the spelling field should be pronounced.

~~11.35~~11.34 TpUIPronounceType

Defines the pronunciation type.

The International Phonetic Alphabet (IPA) representation can be used to specify pronunciations. For more information see:

http://www2.arts.gla.ac.uk/IPA/ipachart.html

http://charts.unicode.org/Unicode.charts/normal/U0250.html

Also, simple sound-alike replacements can be used, such as “I triple E” for IEEE.

Name	Value	Description
P_UI_PRONOUNCE_IPA	0	The IPA pronunciation type
P_UI_PRONOUNCE_SOUNDSLIKE	1	The simple sounds like replacement type

~~11.36~~11.35 TpUICollectMode

Defines the type of collection.

Name	Value	Description
P_UI_COLLECT_MODE_DTMF	0	Collect DTMF digits only
P_UI_COLLECT_MODE_VOICE	1	Collect Voice recognized data
P_UI_COLLECT_MODE_DTMFANDVOICE	2	Collect both DTMF digits and voice recognized data

~~11.37~~11.36 TpUIRecognitionCriteria

Defines the Sequence of Data Elements that specify the additional properties for the collection of information in the form of voice recognition according to the specified grammar.

Structure Element Name	Structure Element Type
SpeakerID	TpUIRecognitionSpeakerID
Properties	TpUIRecognitionPropertySet
Grammar	TpUIRecognitionGrammar

The structure elements specify the following criteria:

SpeakerID: Defines the user identifier associating a user with a speech profile known to the recognition engine, which provides a hint for better quality.

Properties: Defines the properties set for additional information to the speech recognition engine.

Grammar: Defines the syntax of the language to be recognized.

~~11.38~~11.37 TpUIRecognitionSpeakerID

Defines a user identifier string that identifies the speaker and is a hint to whose voice is to be recognized.

~~11.39~~11.38 TpUIRecognitionPropertySet

Defines a Numbered Set of Data Elements of TpUIRecognitionProperty.

~~11.40~~11.39 TpUIRecognitionProperty

Defines the Sequence of Data Elements that specify the additional properties for the recognition engine. The TpUIRecognitionProperty is a hint to the recognition engine on how it should interpret the input.

Structure Element Name	Structure Element Type
PropertyName	TpString
PropertyValue	TpString

The structure elements specify the following criteria:

PropertyName:	Defines the name of the property.
PropertyValue:	Defines the value of the property.

The defined properties are:

P_RECOGNITION_PROPERTY_CONFIDENCE_LEVEL - The speech recognition confidence level, a float value in the range of 0.0 to 1.0. Results are rejected when the recognitions engine's confidence in its interpretation is below this threshold. A value of 0.0 means minimum confidence is needed for a recognition, and a value of 1.0 requires maximum confidence. The default value is 0.5.

P_RECOGNITION_PROPERTY_SENSITIVITY - Set the sensitivity level. A value of 1.0 means that it is highly sensitive to quiet input. A value of 0.0 means it is least sensitive to noise. The default value is 0.5.

P_RECOGNITION_PROPERTY_SPEEDVSACCURACY - A hint specifying the desired balance between speed vs. accuracy. A value of 0.0 means fastest recognition. A value of 1.0 means best accuracy. The default is value 0.5.

P_RECOGNITION_PROPERTY_COMPLETE_TIMEOUT - The speech timeout value to use when an active grammar is matched.

~~11.41~~11.40 TpUIRecognitionGrammar

Defines a string that consists of an inline grammar that specifies the syntax of the speech to be recognized.   The format of this string is a subset of the Voice XML 1.0 grammar element tag. The <grammar> element tag specifies the allowable input that the voice recognition will accept. The Voice XML grammar specifies the set of utterances that a user may speak to perform an action and specifies the corresponding string value for the result.

The following table describes the features that provide a language for describing context-free grammars.

Feature	Purpose
word or words	(terminals, tokens) need not be quoted
[x]	optional x
(...)	Grouping
x {value text}	arbitrary value text may be associated with x
x*	0 or more occurrences of x
x+	1 or more occurrences of x
x y z ...	a sequence of x then y then z then ...
x | y | z | ...	a set of alternatives of x or y or z or ...
<rule>	rule names (non-terminals) are enclosed in <>
<rule> = x;	a private rule definition
public <rule> = x;	a public rule definition

The format of the grammar tag is:

```
<grammar type="application/x-jsgf"> grammar content </grammar>
```

or

```
<grammar type="application/x-jsgf" src="url" />
```

The grammar defines a possible set of utterances. The text of the utterance itself is used as the value, if the value text is not explicitly specified with {value}.

This form is particularly convenient for expressing simple lists of alternative ways of saying the same thing, for example:

```
<grammar type="application/x-jsgf">
     [please] help [me] [please] | [please] I (need | want) help [please]
</grammar>


<grammar type="application/x-jsgf">
    hamburger | burger {hamburger} | (chicken [sandwich]) {chicken}
</grammar>
```

In the first example, any of the ways of saying "help" result in a valid response. In the second example, the user may say "hamburger" or "burger" and the response will be given the value "hamburger", or the user may say "chicken" or "chicken sandwich" and the result will be given the value "chicken".

If the grammar can not be matched, then a sendInfoAndCollectErr will result, with an P_IMPROPER_USER_RESPONSE.

For a better description of Voice XML version 1.0 and the Java™ Speech Grammar Format, see:

http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/index.html

End of changes in Clauses 10&11

CR-Form-v7

CHANGE REQUEST

⌘	**29.198-08**	**CR**	**035**	⌘ **rev**	**-**	⌘	Current version:	**6.1.0** ⌘

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

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

Title: ⌘	Remove unused Deprecated items		
Source: ⌘	CN5 Ultan Mulligan, ETSI PTCC		
Work item code: ⌘	OSA3	***Date:*** ⌘	27/08/2004
Category: ⌘	**F**	***Release:*** ⌘	*REL-6*
	Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *Rel-4 (Release 4)* *Rel-5 (Release 5)* *Rel-6 (Release 6)*

Reason for change: ⌘	3GPP CN5 uses a procedure of deprecation in its specifications to identify methods and other items which cannot be used or have been replaced. Items are deprecated when they have been found not to work, or when replaced in a controlled, backwards compatible manner. Such deprecated items should be removed one day from the specifications: 3GPP CN5 keeps such deprecated items in its specifications for 2 full releases in order to preserve backwards compatibility. At the delivery of Release 6, it is time to remove those items from the Release 6 specifications which are deprecated in the current Release 4 specifications and the Release 5 specifications. This removal has no impact on the Release 4 or Release 5 specifications - backwards compatibility is preserved for these.
Summary of change: ⌘	Delete the deprecated Service Property P_TRIGGERING_ADDRESSES and the deprecated method getNotification()
Consequences if not approved: ⌘	Old, unusable items will continue to clutter up the OSA specifications. The non-removal of deprecated items will encourage developers to implement them, when in fact deprecation of an item is intended to do the exact opposite. Furthermore, vendors will continue to be obliged to retain these deprecated items in their implementations, forcing them to continue to include these often broken items in their test plans.

Clauses affected: ⌘	4, 8, 10				
	Y	**N**			
Other specs affected: ⌘	**X**		Other core specifications	⌘	TS 29.198-1, -3, -4-2, -4-3, -5 and -11
		X	Test specifications		
		X	O&M Specifications		
Other comments: ⌘					

How to create CRs using this form:

Start of change in Clause 4

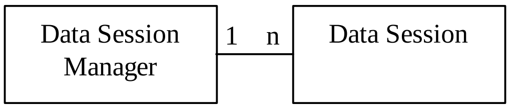
4 Data Session Control SCF

The Data Session Control network SCF consists of two interfaces:

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

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

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



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

Figure 1: Data Session control interfaces usage relationship

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

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

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

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

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

The order is as follows:

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

4.1 General requirements on support of methods

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

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

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

End of change in Clause 4

Start of change in Clause 8

8.4 Interface Class IpDataSessionControlManager

Inherits from: IpService.

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

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

8.4.1 Method <<deprecated>> createNotification()

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

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

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

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

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

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

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

Parameters

`appDataSessionControlManager : in IpAppDataSessionControlManagerRef`

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

`eventCriteria : in TpDataSessionEventCriteria`

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

Returns

`TpAssignmentID`

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE`

8.4.2 Method destroyNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

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

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID`

8.4.3 Method changeNotification()

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

Parameters

`assignmentID : in TpAssignmentID`

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

`eventCriteria : in TpDataSessionEventCriteria`

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

Raises

`TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_ASSIGNMENT_ID, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE`

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

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

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

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

~~*Parameters*~~

~~No Parameters were identified for this method~~

~~Returns~~

~~TpDataSessionEventCriteria~~

~~Raises~~

~~TpCommonExceptions, P_INVALID_NETWORK_STATE~~

~~8.4.5~~8.4.4 Method enableNotifications()

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

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

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

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

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

Parameters

appDataSessionControlManager : in IpAppDataSessionControlManagerRef

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

Returns

TpAssignmentID

Raises

TpCommonExceptions

~~8.4.6~~8.4.5 Method disableNotifications()

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

Parameters

No Parameters were identified for this method

Raises

TpCommonExceptions

~~8.4.7~~8.4.6 Method getNotifications()

~~This method replaces getNotification().~~

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

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

Parameters

No Parameters were identified for this method

Returns

TpDataSessionEventCriteriaResultSet

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE

~~8.4.8~~8.4.7 Method createNotifications()

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

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

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

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

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

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

Parameters

appDataSessionControlManager : in IpAppDataSessionControlManagerRef

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

eventCriteria : in TpDataSessionEventCriteria

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

Returns

TpAssignmentID

Raises

TpCommonExceptions, P_INVALID_NETWORK_STATE, P_INVALID_CRITERIA, P_INVALID_EVENT_TYPE, P_INVALID_INTERFACE_TYPE

End of change in Clause 8

Start of change in Clause 10

10 Data Session Control Service Properties

The following table lists properties relevant for the Data Session Control API.

Property	Type	Description/Interpretation
P_TRIGGERING_EVENT_TYPES	INTEGER_SET	Indicates the static event types supported by the SCS. Static events are the events by which applications are initiated.
P_DYNAMIC_EVENT_TYPES	INTEGER_SET	Indicates the dynamic event types supported by the SCS. Dynamic events are the events the application can request for during the context of a call.
P_ADDRESSPLAN	INTEGER_SET	Indicates the supported address plans (defined in TpAddressPlan.) E.g. P_ADDRESS_PLAN_IP. Note that more than one address plan may be supported.

The previous table lists properties related to the capabilities of the SCS itself. The following table lists properties that are used in the context of the Service Level Agreement, e.g. to restrict the access of applications to the capabilities of the SCS.

Property	Type	Description/Interpretation
~~P_TRIGGERING_ADDRESSES (Deprecated)~~	~~ADDRESSRANGE_SET~~	~~Indicates for which numbers the notification may be set. For terminating notifications it applies to the terminating number, for originating notifications it applies only to the originating number.~~
P_NOTIFICATION_ADDRESS_RANGES	XML_ADDRESS_RANGE_SET	Indicates for which numbers notifications may be set. More than one range may be present. For terminating notifications they apply to the terminating number, for originating notifications they apply only to the originating number.
P_MONITOR_MODE	INTEGER_SET	Indicates whether the application is allowed to monitor in interrupt and/or notify mode. Set is: P_INTERRUPT P_NOTIFY
P_NUMBERS_TO_BE_CHANGED	INTEGER_SET	Indicates which numbers the application is allowed to change or fill for legs in an incoming call. Allowed value set: {P_TARGET_NUMBER}.
P_CHARGEPLAN_ALLOWED	INTEGER_SET	Indicates which charging is allowed in the setDataSessionChargePlan indicator. Allowed values: {P_CHARGE_PER_VOLUME, P_TRANSPARANT_CHARGING, P_CHARGE_PLAN}
P_CHARGEPLAN_MAPPING	INTEGER_INTEGER_MAP	Indicates the mapping of charge plans (we assume they can be indicated with integers) to a logical network charge plan indicator. When the P_CHARGEPLAN_ALLOWED property indicates P_CHARGE_PLAN, then only charge plans in this mapping are allowed.
P_CURRENCY_ALLOWED	STRING_SET	Indicates the currencies that are allowed to be set for the charge plan in the setDataSessionChargePlan. The valid values for the string set are according to ISO-4217:1995. E.g. {“EUR”, “NLG”}.

joint-API-group (Parlay, ETSI Project OSA, 3GPP TSG_CN WG5) **N5-040626**
Meeting #28, Piscataway, New Jersey, USA, 09-13 August 2004

CR-Form-v7

CHANGE REQUEST

⌘ **29.198-11**	**CR**	**031**	⌘ **rev**	**-**	⌘ Current version:	**6.1.0** ⌘

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

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

Title: ⌘	Remove unused Deprecated items		
Source: ⌘	CN5 Ultan Mulligan, ETSI PTCC		
Work item code: ⌘	OSA3	***Date:*** ⌘	27/08/2004
Category: ⌘	**F**	***Release:*** ⌘	*REL-6*
	Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *Rel-4 (Release 4)* *Rel-5 (Release 5)* *Rel-6 (Release 6)*

Reason for change: ⌘	3GPP CN5 uses a procedure of deprecation in its specifications to identify methods and other items which cannot be used or have been replaced. Items are deprecated when they have been found not to work, or when replaced in a controlled, backwards compatible manner. Such deprecated items should be removed one day from the specifications: 3GPP CN5 keeps such deprecated items in its specifications for 2 full releases in order to preserve backwards compatibility. At the delivery of Release 6, it is time to remove those items from the Release 6 specifications which are deprecated in the current Release 4 specifications and the Release 5 specifications. This removal has no impact on the Release 4 or Release 5 specifications - backwards compatibility is preserved for these.
Summary of change: ⌘	Delete the deprecated Service Property P_TRIGGERING_ADDRESSES
Consequences if not approved: ⌘	Old, unusable items will continue to clutter up the OSA specifications. The non-removal of deprecated items will encourage developers to implement them, when in fact deprecation of an item is intended to do the exact opposite. Furthermore, vendors will continue to be obliged to retain these deprecated items in their implementations, forcing them to continue to include these often broken items in their test plans.

Clauses affected: ⌘	10				
	Y	**N**			
Other specs affected: ⌘	X		Other core specifications	⌘	29.198-1, -3, -4-2, -4-3, -5 and -8
		X	Test specifications		
		X	O&M Specifications		
Other comments: ⌘					

How to create CRs using this form:

Start of change in Clause 10

10 Account Management Service Properties

The following table lists properties relevant for the Account Management API.

Property	Type	Description/Interpretation
P_ EVENT_TYPES	INTEGER_SET	Indicates the event types supported by the SCS. Static events are the events by which applications are initiated.
P_ADDRESSPLAN	INTEGER_SET	Indicates the supported address plans (defined in TpAddressPlan.) E.g. {P_ADDRESS_PLAN_E164, P_ADDRESS_PLAN_IP}). Note that more than one address plan may be supported.

The previous table lists properties related to the capabilities of the SCS itself. The following table lists properties that are used in the context of the Service Level Agreement, e.g. to restrict the access of applications to the capabilities of the SCS.

Property	Type	Description/Interpretation
~~P_TRIGGERING_ADDRESSES (Deprecated)~~	~~ADDRESSRANGE_SET~~	~~Indicates for which numbers the notification may be set. For terminating notifications it applies to the terminating number, for originating notifications it applies only to the originating number.~~
P_NOTIFICATION_ADDRESS_RANGES	XML_ADDRESS_RANGE_SET	Indicates for which numbers notifications may be set. More than one range may be present. For terminating notifications they apply to the terminating number, for originating notifications they apply only to the originating number.
P_CURRENCY_ALLOWED	STRING_SET	Indicates the currencies that can be returned in the queryBalanceRes. The valid values for the string set are according to ISO-4217:1995. E.g. {“EUR”, “NLG”}.
P_HISTORY_ALLOWED	STRING_SET	Indicates the length of the transaction history interval that is allowed to be retrieved by the application. The valid values for the string are according to TpDateAndTime. The string-set will be of format {"lower_start_time", "upper_stop_time"}, e.g. {"1998-12-04 10:30", "1999-12-04 10:30"}
P_MAX_ADDRESSES_PER_QUERY	INTEGER_SET	Indicates the maximum number of addresses which can be included in a queryBalanceReq.