3GPP TSG CN Plenary Meeting #14 **NP-010600**
Kyoto, Japan, 12-14 December 2001

Source: **CN5 (OSA)**

Title: **Rel-4 CRs 29.198-07**

Agenda item: **8.5**

Document for: **Decision**

Doc-1st-Level	Spec	CR	R	Pha	Subject	Cat	Ver Cur	Ver -New	Doc-2nd-Level	Workitem
NP-010600	29.198-07	003		Rel-4	**Replace Out Parameters with Return Types**	F	4.2.0	4.3.0	N5-010567	OSA1

3GPP TSG_CN5 (Open Service Access – OSA) **N5-010567**
Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001

CR-Form-v4

CHANGE REQUEST

⌘ **29.198-07** **CR** **003** ⌘ ev **-** ⌘ Current version: **4.2.0** ⌘

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

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

Title:	⌘	Replacing Out Parameters with Return Types
Source:	⌘	CN5
Work item code:	⌘	OSA1
Date:	⌘	19/07/2001
Category:	⌘	**F**
Release:	⌘	REL-4

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

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
REL-4 (Release 4)
REL-5 (Release 5)

Reason for change:	⌘	At CN5 and CN it was agreed that Out-parameters should be removed from methods as a means of returning information, to be replaced by Return Types, in line with commonly used programming practice
Summary of change:	⌘	For each method, replace the return parameter TpResult with: 'void' if the method has no out-parameter; or the type of the out-parameter if the method has an out-parameter, and delete the out-parameter from the method.
Consequences if not approved:	⌘	If this particular CR is not agreed, TS 29.198-7 is out of sync. with the other parts of TS 29.198. If the related batch of CRs is not agreed, OSA will have a limited acceptance among the application development community, since it will be more difficult to implement. This presents a risk to the return on investment in development of OSA.
Clauses affected:	⌘	7, 8, Annex B
Other specs affected:	⌘	**X** Other core specifications ☐ Test specifications ☐ O&M Specifications
	⌘	All other parts of TS 29.198 Rel-4
Other comments:	⌘	

How to create CRs using this form:

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

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be

downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

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

7 The Service Interface Specifications

7.1 Interface Specification Format

This section defines the interfaces, methods and parameters that form a part of the API specification. The Unified Modelling Language (UML) is used to specify the interface classes. The general format of an interface specification is described below.

7.1.1 Interface Class

This shows a UML interface class description of the methods supported by that interface, and the relevant parameters and types. The Service and Framework interfaces for enterprise-based client applications are denoted by classes with name `Ip<name>`. The callback interfaces to the applications are denoted by classes with name `IpApp<name>`. For the interfaces between a Service and the Framework, the Service interfaces are typically denoted by classes with name IpSvc<name>, while the Framework interfaces are denoted by classes with name IpFw<name>.

7.1.2 Method descriptions

Each method (API method "call") is described. All methods in the API return a value of type `TpResult`, indicating, amongst other things, if the method invocation was sucessfully executed or not.

Both synchronous and asynchronous methods are used in the API. Asynchronous methods are identified by a '`Req`' suffix for a method request, and, if applicable, are served by asynchronous methods identified by either a '`Res`' or '`Err`' suffix for method results and errors, respectively. To handle responses and reports, the application or service developer must implement the relevant `IpApp<name> or IpSvc<name>` interfaces to provide the callback mechanism.

7.1.3 Parameter descriptions

Each method parameter and its possible values are described. Parameters described as "in" represent those that must have a value when the method is called. Those described as "out" are those that contain the return result of the method when the method returns.

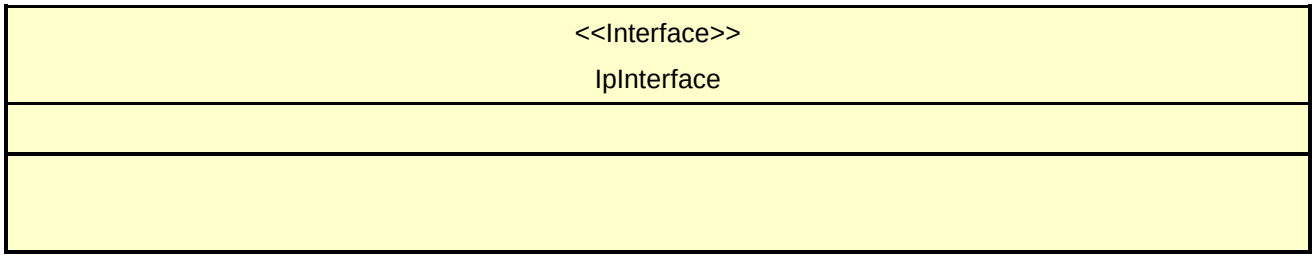
7.1.4 State Model

If relevant, a state model is shown to illustrate the states of the objects that implement the described interface.

7.2 Base Interface

7.2.1 Interface Class IpInterface

All application, framework and service interfaces inherit from the following interface. This API Base Interface does not provide any additional methods.



7.3 Service Interfaces

7.3.1 Overview

The Service Interfaces provide the interfaces into the capabilities of the underlying network - such as call control, user interaction, messaging, mobility and connectivity management.

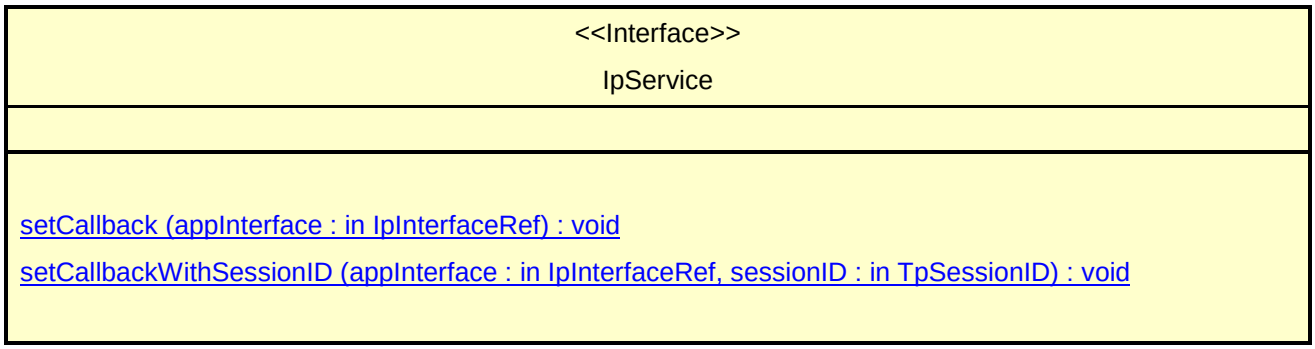
The interfaces that are implemented by the services are denoted as "Service Interface". The corresponding interfaces that must be implemented by the application (e.g. for API callbacks) are denoted as "Application Interface".

7.4 Generic Service Interface

7.4.1 Interface Class IpService

Inherits from: IpInterface

All service interfaces inherit from the following interface.



Method
`setCallback()`

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionID's.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks

Raises

`TpCommonExceptions`

Method
`setCallbackWithSessionID()`

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not uses SessionID's.

Parameters

appInterface : in IpInterfaceRef

Specifies a reference to the application interface, which is used for callbacks

sessionID : in TpSessionID

Specifies the session for which the service can invoke the application's callback interface.

Raises

TpCommonExceptions, P_INVALID_SESSION_ID

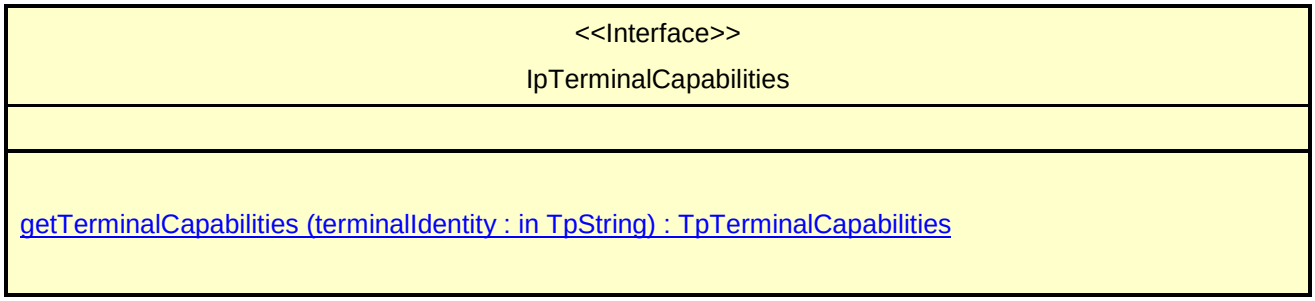
8 Terminal Capabilities Interface Classes

The Terminal Capabilities SCF enables the application to retrieve the terminal capabilities of the specified terminal. The Terminal Capabilities service provides a SCF interface that is called IpTerminalCapabilities. There is no need for an application interface, since IpTerminalCapabilities only contains the synchronous method getTerminalCapabilities.

8.1 Interface Class IpTerminalCapabilities

Inherits from: IpInterface.

The Terminal Capabilities SCF interface IpTerminalCapabilities contains the synchronous method getTerminalCapabilities. The application has to provide the terminaIdentity as input to this method. The result indicates whether or not the terminal capabilities are available in the network and, in case they are, it will return the terminal capabilities (see the data definition of TpTerminalCapabilities for more information).



Method

getTerminalCapabilities()

This method is used by an application to get the capabilities of a user's terminal. Direction: Application to Network.

Returns result : Specifies the latest available capabilities of the user´s terminal.

This information, if available, is returned as CC/PP headers as specified in W3C [1] and adopted in the WAP UAProf specification [2]. It contains URLs; terminal attributes and values, in RDF format; or a combination of both.

Parameters

terminalIdentity : in TpString

Identifies the terminal. It may be a logical address known by the WAP Gateway/PushProxy.

Returns

TpTerminalCapabilities

Raises

TpCommonExceptions, P_INVALID_TERMINAL_ID

==========**Next changed section**==========

Annex B (informative): Differences between this draft and 3GPP TS 29.198 R99

getTerminalCapabilities now throws TpCommonExceptions and individual, identified exceptions

All methods now return void or the former out parameter.