**3GPP TSG CN Plenary Meeting #14** **NP-010594**
**Kyoto, Japan, 12-14 December 2001**

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

**Title:** **Rel-4 CRs 29.198-01**

**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-010594 | 29.198-01 | 003 | | Rel-4 | **Replace Out Parameters with Return Types** | F | 4.2.0 | 4.3.0 | N5-010561 | OSA1 |
| NP-010594 | 29.198-01 | 004 | | Rel-4 | **Remove the perception that the OSA API only uses CORBA for its transport mechanism** | F | 4.2.0 | 4.3.0 | N5-010702 | OSA1 |

CR-Form-v4

# CHANGE REQUEST

⌘ 29.198-01 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:** | ⌘ | General references to number of out parameters, and to method return types, updated to take account of this change. |
| **Consequences if not approved:** | ⌘ | If this particular CR is not agreed, TS 29.198-1 is out of sync. with the other parts of TS 29.198.<br>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:** | ⌘ | 6.6, 6.8 |
| **Other specs affected:** | ⌘ | X Other core specifications<br>☐ Test specifications<br>☐ 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.

## 6.6 Error results

As OMG IDL supports exception handling with high efficiency, OSA methods communicate errors in the form of CORBA exceptions ~~of type TpGeneralException~~ in the IDLs; the CORBA methods themselves ~~always~~ return void or an identified return type.
~~But in the documentation, errors are communicated using a return parameter of type TpGeneralResult.~~

## 6.7 References

In the interface specification whenever parameters are to be passed by reference, the "Ref" suffix is appended to their corresponding data type (e.g. IpAnInterfaceRef anInterface), a reference can also be viewed as a logical indirection. Therefore, structured or primitive data type passed as *out* parameters are references. An interface passed as an *in* parameter is also a reference but an interface passed as an *out* parameter is a double indirection (i.e.: RefRef)

| Original Data type | IN parameter declaration | OUT parameter declaration |
|---|---|---|
| TpPrimitive | parm : IN TpPrimitive | parm : OUT TpPrimitiveRef |
| TpStructured | parm : IN TpStructured | parm : OUT TpStructuredRef |
| IpInterface | parm : IN IpInterfaceRef | parm : OUT IpInterfaceRefRef |

In IDL, however, the following rules apply:

- Interfaces are implicitly passed by reference.
- *out* parameters are also implicitly passed by reference.

This leads to:

- Interface as an *in* parameter: Passed by Reference.
- Structure or primitive type as an *in* parameter: Passed by Value.
- Structure or primitive type as an *out* parameter: Passed by Reference.
- Interface as an *out* parameter: As reference passed by reference.

To simplify the documentation without adding ambiguities, parameters (interfaces, structures and primitive data types) are used as is when specified as *in* or *out* parameters in the IDL. This means that there will be no "Ref" added after the data types of parameters in the IDL.

## 6.8 Number of out parameters

In order to support mapping to as many languages as possible, there are no out ~~is only 1 out~~ parameters allowed per operation. Each operation which returns a value has a defined return type associated with it.

**3GPP TSG_CN5 (Open Service Access – OSA)** **N5-010702**
**Meeting #12, Sophia Antipolis, FRANCE, 16 – 19 July 2001**

*CR-Form-v4*

# CHANGE REQUEST

⌘ **29.198-01** **CR** **004** ⌘ 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:*** | ⌘ | Remove the perception that the OSA API only uses CORBA for its transport mechanism |
| ***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:*** | ⌘ | Remove the perception that the API only uses CORBA for its transport mechanism. This change is in line with the SA1 CR S1-010531. |
| ***Summary of change:*** | ⌘ | Move mention of CORBA from generic parts of the spec to the CORBA parts of the spec |
| ***Consequences if not approved:*** | ⌘ | Perception that the specification only uses CORBA becomes stronger, making it much more difficult to permit other technology mechanisms such as SOAP and Java in the future. Interoperability with Java and SOAP applications could then become more problematic. Also, working relationship with the JAIN Community for the production of common APIs could be jepodised. As these changes have also been accepted by ETSI and Parlay, non-approval would cause mis-alignment |

| | | |
|---|---|---|
| ***Clauses affected:*** | ⌘ | 6, A (new clause) |
| ***Other specs affected:*** | ⌘ | ☐ Other core specifications ⌘<br>☐ Test specifications<br>☐ O&M Specifications |
| ***Other comments:*** | ⌘ | |

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

# 6 Methodology

Following is a description of the methodology used for the establishment of API specification for OSA.

## 6.1 Tools and Languages

The Unified Modelling Language (UML) [14] is used as the means to specify class and state transition diagrams. ~~Additionally, Object Management Group's (OMG) [15] Interface Definition Language (IDL) is used as the means to programmatically define the interfaces. IDL files are either generated manually from class diagrams or by using a UML tool. In the case IDLs are manually written and/or being corrected manually, correctness has been verified using a CORBA2 (orbos/97-02-25) compliant IDL compiler, e.g. [13].~~

## 6.2 Packaging

A hierarchical packaging scheme is used to avoid polluting the global name space. The root is defined as:

org.csapi

~~NOTE: the CORBA module hierarchy defined in the IDLs does not necessarily parallels the logical UML package hierarchy.~~

## 6.3 Colours

For clarity, class diagrams follow a certain colour scheme. Blue for application interface packages and yellow for all the others.

## 6.4 Naming scheme

The following naming scheme is used for ~~both~~ documentation ~~and IDLs~~.

**packages**

lowercase.

Using the domain-based naming (For example, org.csapi)

**classes, structures and types. Start with T**

TpCapitalizedWithInternalWordsAlsoCapitalized

**Exception class:**

TpClassNameEndsWithException

**Interface. Start with Ip:**

IpThisIsAnInterface

**constants:**

P_UPPER_CASE_WITH_UNDERSCORES_AND_START_WITH_P

**methods:**

firstWordLowerCaseButInternalWordsCapitalized()

**method's parameters**

firstWordLowerCaseButInternalWordsCapitalized

**collections (set, array or list types)**

TpCollectionEndsWithSet

**class/structure members**

FirstWordAndInternalWordsCapitalized

Spaces in-between words are not allowed.

## 6.5 State Transition Diagram text and text symbols

The descriptions of the State Transitions in the State Transition Diagrams follow the convention:

when_this_event_is_received [guard condition is true] /do_this_action ^send_this_message

Furthermore, text underneath a line through the middle of a State indicates an exit or entry event (normally specified which one).

## 6.6 ~~Error~~ Exception handling and passing results

~~As OMG IDL supports exception handling with high efficiency,~~ OSA methods communicate errors in the form of ~~CORBA~~ exceptions ~~of type TpGeneralException in the IDLs; the CORBA~~ OSA methods themselves always use the return parameter to pass results. If no results are to be returned a ~~-~~ void is used instead of the return parameter. In order to support mapping to as many languages as possible, no method *out* parameters are allowed.
~~But in the documentation, errors are communicated using a return parameter of type TpGeneralResult.~~

## 6.7 References

In the interface specification whenever Interface parameters are to be passed as an *in* parameter they are done so by reference, and the "Ref" suffix is appended to their corresponding ~~data~~ type (e.g. IpAnInterfaceRef anInterface), a reference can also be viewed as a logical indirection. ~~Therefore, structured or primitive data type passed as *out* parameters are references. An interface passed as an *in* parameter is also a reference but an interface passed as an *out* parameter is a double indirection (i.e.: RefRef)~~

| Original ~~Data~~ type | IN parameter declaration | ~~OUT parameter declaration~~ |
|---|---|---|
| ~~TpPrimitive~~ | ~~parm : IN TpPrimitive~~ | ~~parm : OUT TpPrimitiveRef~~ |
| ~~TpStructured~~ | ~~parm : IN TpStructured~~ | ~~parm : OUT TpStructuredRef~~ |
| IpInterface | parm : IN IpInterfaceRef | ~~parm : OUT IpInterfaceRefRef~~ |

~~In IDL, however, the following rules apply:~~

- ~~Interfaces are implicitly passed by reference.~~
- ~~*out* parameters are also implicitly passed by reference.~~

~~This leads to:~~

- ~~Interface as an *in* parameter: Passed by Reference.~~
- ~~Structure or primitive type as an *in* parameter: Passed by Value.~~
- ~~Structure or primitive type as an *out* parameter: Passed by Reference.~~
- ~~Interface as an *out* parameter: As reference passed by reference.~~

~~To simplify the documentation without adding ambiguities, parameters (interfaces, structures and primitive data types) are used as is when specified as *in* or *out* parameters in the IDL. This means that there will be no "Ref" added after the data types of parameters in the IDL.~~

## 6.8 Number of out parameters

In order to support mapping to as many languages as possible, there is only 1 out parameter allowed per operation.

## 6.9 Strings and Collections

For character strings, the *String* data type is used without regard to the maximum length of the string. In IDL, the data type *String* is typedefed (see Note below) from the CORBA primitive *string*. This CORBA primitive is made up of a length and a variable array of byte.

NOTE: A *typedef* is a type definition declaration in IDL.

For homogeneous collections of instances of a particular data type the following naming scheme is used: <datatype>Set. In OMG IDL, this maps to a sequence of the data type. A CORBA sequence is implicitly made of a length and a variable array of elements of the same type.

**Example 1:** typedef sequence<TpSessionID> TpSessionIDSet;

Collection types can be implemented (for example, in C++) as a structure containing an integer for the *number* part, and an array for the *data* part.

**Example 2:** The TpAddressSet data type may be defined in C++ as:

```
typedef struct {
    short       number;
    TpAddress   address [];
} TpAddressSet;
```

The array "address" is allocated dynamically with the exact number of required TpAddress elements based on "number".

## 6.10 Prefixes

OSA constants and data types are defined in the global name space: *org.csapi*.

## 6.11 Naming space across CORBA modules

The following shows the naming space used in this specification.

```
module org {
   module csapi {
    /* The fully qualified name of the following constant is
    org::csapi::P_THIS_IS_AN_OSA_GLOBAL_CONST */
    const long P_THIS_IS_AN_OSA_GLOBAL_CONST= 1999;
    // Add other OSA global constants and types here
    module fw {
    /* no scoping required to access  P_THIS_IS_AN_OSA_GLOBAL_CONST */
      const long P_FW_CONST= THIS_IS_AN_OSA_GLOBAL_CONST;
    };
    module mm {
    // scoping required to access P_FW_CONST
      const long P_M_CONST= fw::P_FW_CONST;
    };
   };
};
```

# Annex A (normative): OMG IDL

## A.1 Tools and Languages

The Object Management Group's (OMG) [15] Interface Definition Language (IDL) is used as a means to programmatically define the interfaces. IDL files are either generated manually from class diagrams or by using a UML tool. In the case IDLs are manually written and/or being corrected manually, correctness has been verified using a CORBA2 (orbos/97-02-25) compliant IDL compiler, e.g. [13].

## A.2 Strings and Collections

In IDL, the data type *String* is typedefed (see Note below) from the CORBA primitive *string*. This CORBA primitive is made up of a length and a variable array of byte.

NOTE: A *typedef* is a type definition declaration in IDL.

In OMG IDL, this maps to a sequence of the data type. A CORBA sequence is implicitly made of a length and a variable array of elements of the same type.

**Example 1:** typedef sequence<TpSessionID> TpSessionIDSet;

Collection types can be implemented (for example, in C++) as a structure containing an integer for the *number* part, and an array for the *data* part.

**Example 2:** The TpAddressSet data type may be defined in C++ as:

```
typedef struct {
    short       number;
    TpAddress   address [];
} TpAddressSet;
```

The array "address" is allocated dynamically with the exact number of required TpAddress elements based on "number".

## A.3 Naming space across CORBA modules

The following shows the naming space used in this specification.

```
module org {
   module csapi {
    /* The fully qualified name of the following constant is
    org::csapi::P_THIS_IS_AN_OSA_GLOBAL_CONST */
    const long P_THIS_IS_AN_OSA_GLOBAL_CONST= 1999;
    // Add other OSA global constants and types here
    module fw {
    /* no scoping required to access  P_THIS_IS_AN_OSA_GLOBAL_CONST */
      const long P_FW_CONST= THIS_IS_AN_OSA_GLOBAL_CONST;
    };
    module mm {
    // scoping required to access P_FW_CONST
      const long P_M_CONST= fw::P_FW_CONST;
    };
   };
};
```