3GPP TSG CN Plenary Meeting #22 **NP-030543**
10 - 12 December 2003, Maui, Hawaii, USA

Source: **CN5 (OSA)**

Title: **Rel-4/5 CRs 29.198-02 OSA API Part 2: Common data**

Agenda item: **7.10**

Document for: **APPROVAL**

Doc-1st-Level	Spec	CR	R	Ph	Subject	Cat	Version-Current	Doc-2nd-Lev	WI
NP-030543	29.198-02	037	-	Rel-4	Description correction of TpOctetSet	F	4.6.0	N5-030620	OSA1
NP-030543	29.198-02	038	-	Rel-5	Description correction of TpOctetSet	A	5.4.0	N5-030621	OSA1
NP-030543	29.198-02	039	-	Rel-4	Clarify use of base interface Reference within an inheritance relationship	F	4.6.0	N5-030622	OSA1
NP-030543	29.198-02	040	-	Rel-5	Clarify use of base interface Reference within an inheritance relationship	A	5.4.0	N5-030623	OSA1

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-02 CR 037 ⌘rev - ⌘ Current version: 4.6.0 ⌘

*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:	⌘	Description correction of TpOctetSet
Source:	⌘	CN5 (ETSI PTCC)
Work item code:	⌘	OSA1
Date:	⌘	31/10/2003
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)
Rel-6 (Release 6)

Reason for change:	⌘	TpOctetSet is described as a Numbered Set of TpOctet. The definition of a Numbered Set requires that it be an unordered set, with no duplication of data elements. All current usage of TpOctetSet in OSA requires that it be an ordered list, with duplication of data elements permitted. This means that a TpOctetList type would be more appropriate. The specification as it is currently written is seriously misleading. Developers have complained about this particular error. TpOctetList and TpOctetSet resolve to the same base type in IDL, WSDL and Java versions of OSA. So TpOctetSet can be redefined to mean the same a s a TpOctetList without any backwards compatibility issues.
Summary of change:	⌘	Redefine TpOctetSet to be a Numbered List of Data Elements. Introduce a new TpUnorderedOctetSet to be a Numbered Set of Data Elements. Add a description of Numbered List of Data Elements, missing in Release 4.
Consequences if not approved:	⌘	A serious contradiction between the definition of a type and its use will remain in the specification. Questions have been asked by developers about the stability and coherence of the OSA specifications, with such an error in them.

Clauses affected:	⌘	5.1, 5.1.6, 5.2, 5.2.3			
Other specs affected:	⌘	Y: X / N:	Other core specifications	⌘	Rel-5 29.198-02
		Y: / N: X	Test specifications		
		Y: / N: X	O&M Specifications		
Other comments:	⌘	Rel-5 Mirror CR in N5-030621			

How to create CRs using this form:

Change in Clause 5

5.1.5 TpOctet

Defines an 8-bit quantity that is not translated during transmission.

5.1.6 TpOctetSet

Defines a Numbered ~~Set~~ List of Data elements of TpOctet. Note that this is an ordered list.

5.1.7 TpString

Defines a Byte string, comprising length and data. The length shall be at least a 16-bit integer.

5.1.8 TpAssignmentID

Defines an assignment ID with a value that is unique within any instance of an implementation of a given interface, irrespective of the method invoked on that interface instance. This ID may be used for example to identify single or multiple event notifications enabled by a requesting interface implementation, or may be used by a requesting interface implementation to modify or stop functionality associated with a previously supplied assignment ID, e.g event notifications, call load control, abort requests.

The assignment ID is identical to a TpInt32 type.

5.1.9 TpSessionID

Defines a session ID with a value that is at least unique within the context of a specific instance of an SCF. An instance of an SCF is a single service manager instance plus the associated subordinate instances. For example, a single MultiPartyCallControlManager instance plus all associated MultiPartyCall and MultiPartyCallLeg instances. The session ID is used to identify different sessions (e.g. different call or call leg sessions) of an interface capable of handling multiple sessions.

Example 1, myCallObject may implement the IpCall interface. If so, myCallObject may handle multiple call sessions, and each call session will be identified by a call session ID value (e.g. 1, 2, 3) that is unique within the context of the SCF instance.

Example 2, myCallAndCallLegObject may implement the IpCall and IpCallLeg interfaces. If so, myCallAndCallLegObject may handle multiple call sessions and multiple call leg sessions. Each call session will be identified by a call session ID value (e.g. 1, 2, 3) that is unique within the context of the SCF instance. Similarly, each call leg session will be identified by a call leg session ID value (e.g. 1, 2, 3, 4, 5, 6) that is also unique within the context of the SCF instance. Because call session IDs and call leg session IDs are different data types, overlapping values are permitted and their uniqueness still remains.

The session ID is identical to a TpInt32 type.

5.1.10 TpSessionIDSet

Defines a Numbered Set of Data Elements of TpSessionID.

5.1.11 TpAny

Defines a type that can hold any type. This is not restricted to only the primitive types.

5.1.12 TpAttribute

This is a `Sequence of Data Elements` containing the attribute name, type, and value. The attribute Value is interpreted based on the value of the attribute Type.

Sequence Element Name	Sequence Element Type	Notes
`AttributeName`	`TpString`	The name of the attribute.
`AttributeType`	`TpAttributeType`	The type of the attirbute. Valid values for Type must include at least TpString, TpInt32 and TpFloat.
`AttributeValue`	`TpAny`	The values for the attribute. This model allows multi-valued attributes. Cannot be an empty list.

5.1.13 TpAttributeType

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

Character String Value	Description
NULL	An empty (NULL) string indicates no attribute type
`P_STRING`	Attribute type is type `TpString.`
`P_INT32`	Attribute type is type `TpInt32.`
`P_FLOAT`	Attribute type is type `TpFloat.`

5.1.14 TpAttributeList

This is a `Numbered List of Data Elements` of type TpAttribute.

5.1.15 TpAttributeSet

This is a `Numbered Set of Data Elements` of type TpAttribute.

5.1.16 TpUnorderedOctetSet

Defines a Numbered Set of Data Elements of TpOctet. Note that this is an un-ordered set.

Note that this type should not be removed from this specification, even if unused by any part of the OSA specifications. It is included to ensure that TpOctetSet is correctly used as a Numbered List of Data Elements, and not a Numbered Set.

5.2 Other Data sorts

The APIs assumes that the following data syntaxes can be supported:

5.2.1 Sequence of Data Elements

This describes a sequence of data types. This may be defined as a structure (for example, in C++) or simply a sequence of data elements within a structure.

EXAMPLE: The TpAddress data type may be defined in C++ as:

```
typedef struct {
  TpAddressPlan           Plan;
  TpString                AddrString;
  TpString                Name;
  TpAddressPresentation   Presentation;
  TpAddressScreening      Screening;
  TpString                SubAddressString;
```

```
} TpAddress;
```

5.2.2 Tagged Choice of Data Elements

This describes a data type which actually evaluates to one of a choice of a number of data elements. This data element contains two parts: a tag data type (the *tag* part) which is used to identify the chosen data type, and the chosen data type itself (the *union* part). This form of data type is also referred to as a tagged union.

This data type can be implemented (for example, in C++) as a structure containing an integer for the *tag* part, and a union for the *union* part.

This data type is implementation specific. Please refer to the appropriate IDL documents (and the resulting language mappings) to see how this data type is implemented.

EXAMPLE: The `TpCallError` data type may be defined in C++ as:

```
typedef struct {
  TpCallErrorType Tag;
  union {
    TpCallErrorInfoUndefined     Undefined;
    TpCallErrorInfoRoutingAborted  RoutingAborted;
    TpCallErrorInfoCallAbandoned   CallAbandoned;
    TpCallErrorInfoInvalidAddress  InvalidAddress;
    TpCallErrorInfoInvalidState   InvalidState;
    TpCallErrorInfoInvalidCriteria  InvalidCriteria;
  } callErrorInfo;
} TpCallError;
```

5.2.3 Numbered Set of Data Elements

This describes a data type which comprises an integer which indicates the total number of data elements in the set (the *number* part), and an **unordered** set of data elements (the *data* part). *Set* data types do not contain duplicate data elements.

Note that TpOctetSet is a Numbered List of Data Elements, not a Numbered Set.

EXAMPLE: The TpAddressSet data type may be defined in MIDL as:

```
typedef struct TpAddressSet
{
TpInt32 Number; [size_is(Number)] TpAddress Set[];
}
TpAddressSet;
```

5.2.4 Reference

This describes a reference (or pointer) to a data type.

5.2.5 Numbered List of Data Elements

This describes a data type which comprises an integer which indicates the total number of data elements in the set (the *number* part), and an **ordered** set of data elements (the *data* part). *List* data types can contain duplicate data elements.

EXAMPLE: The TpStringList data type may be defined in C++ as:

```
typedef struct {
  TpInt32 Number;
  TpString List[Number];
} TpStringList;
```

End of Change in Clause 5
End of Document

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	--	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	4.0.0
Jun 2001	CN_12	NP-010330	001	--	Corrections to OSA API Rel4 (Exception handling mechanism without ambiguity - Replace TpGeneralException and TpResultInfo with detailed exception classes which can be thrown for each method (N5-010261)	4.0.0	4.1.0
Jun 2001	CN_12	NP-010333	002	--	Introduction of TpOctet (In order to make sure that some data is sent over the “distributed wire” untouched a new data type is needed) (N5-010304)	4.0.0	4.1.0
Sep 2001	CN_13	NP-010465	003	--	Changing references to JAIN	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	004	--	Clarification of common exceptions	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	005	--	Invalid parameter value exception for SLA violation	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	006	--	Storing eventCriteria	4.1.0	4.2.0
Dec 2001	CN_14	NP-010595	007	--	Replace Out Parameters with Return Types	4.2.0	4.3.0
Dec 2001	CN_14	NP-010595	008	--	Correction to Common Data (CD)	4.2.0	4.3.0
Dec 2001	CN_14	NP-010595	009	--	Correction to values of TpAddressPlan	4.2.0	4.3.0
Mar 2002	CN_15	NP-020104	010	--	Ambiguous definition of TpAssignmentID	4.3.0	4.4.0
Mar 2002	CN_15	NP-020104	011	--	Data type alignment in the common data types	4.3.0	4.4.0
Mar 2003	CN_19	NP-030018	023	--	Correction to defintion of sessionID	4.4.0	4.5.0
Mar 2003	CN_19	NP-030018	024	--	Clarification on uniqueness of assignmentID	4.4.0	4.5.0
Mar 2003	CN_19	NP-030018	026	--	Correction to P_INVALID_STATE value	4.4.0	4.5.0
Mar 2003	CN_19	NP-030018	028	--	Addition of Support of National Numbering Plans	4.4.0	4.5.0
Jun 2003	CN_20	NP-030236	033	--	Correction of SIP Address wildcard rules	4.5.0	4.6.0

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-02 CR 038 ⌘rev - ⌘ Current version: 5.4.0 ⌘

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

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

Title:	⌘	Description correction of TpOctetSet
Source:	⌘	CN5 (ETSI PTCC)
Work item code:	⌘	OSA1
Date:	⌘	31/10/2003
Category:	⌘	A
Release:	⌘	*REL-5*

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:	⌘	TpOctetSet is described as a Numbered Set of TpOctet. The definition of a Numbered Set requires that it be an unordered set, with no duplication of data elements. All current usage of TpOctetSet in OSA requires that it be an ordered list, with duplication of data elements permitted. This means that a TpOctetList type would be more appropriate. The specification as it is currently written is seriously misleading. Developers have complained about this particular error. TpOctetList and TpOctetSet resolve to the same base type in IDL, WSDL and Java versions of OSA. So TpOctetSet can be redefined to mean the same a s a TpOctetList without any backwards compatibility issues.
Summary of change:	⌘	Redefine TpOctetSet to be a Numbered List of Data Elements. Introduce a new TpUnorderedOctetSet to be a Numbered Set of Data Elements.
Consequences if not approved:	⌘	A serious contradiction between the definition of a type and its use will remain in the specification. Questions have been asked by developers about the stability and coherence of the OSA specifications, with such an error in them.

Clauses affected:	⌘	5.1, 5.1.6, 5.2.3

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

⌘

Other comments:	⌘	Rel-5 Mirror CR of N5-030620

How to create CRs using this form:

Change in Clause 5

5.1.6 TpOctetSet

Defines a Numbered ~~Set~~ List of Data elements of TpOctet. Note that this is an ordered list.

5.1.7 TpString

Defines a Byte string, comprising length and data. The length shall be at least a 16-bit integer.

5.1.8 TpAssignmentID

Defines an assignment ID with a value that is unique to an instance of an implementation of a given interface (i.e an object), irrespective of the method invoked on it. This ID may be used, for example, to identify single or multiple event notifications enabled by an object; or by a requesting object to modify or stop functionality (e.g event notifications, call load control) associated with a previously supplied assignment ID.

The assignment ID is identical to a TpInt32 type.

5.1.9 TpSessionID

Defines a session ID with a value that is at least unique within the context of a specific instance of an SCF. An instance of an SCF is a single service manager instance plus the associated subordinate instances. For example, a single MultiPartyCallControlManager instance plus all associated MultiPartyCall and MultiPartyCallLeg instances. The session ID is used to identify different sessions (e.g. different call or call leg sessions) of an interface capable of handling multiple sessions.

Example 1, myCallObject may implement the IpCall interface. If so, myCallObject may handle multiple call sessions, and each call session will be identified by a call session ID value (e.g. 1, 2, 3) that is unique within the context of the SCF instance.

Example 2, myCallAndCallLegObject may implement the IpCall and IpCallLeg interfaces. If so, myCallAndCallLegObject may handle multiple call sessions and multiple call leg sessions. Each call session will be identified by a call session ID value (e.g. 1, 2, 3) that is unique within the context of the SCF instance. Similarly, each call leg session will be identified by a call leg session ID value (e.g. 1, 2, 3, 4, 5, 6) that is also unique within the context of the SCF instance. Because call session IDs and call leg session IDs are different data types, overlapping values are permitted and their uniqueness still remains.

The session ID is identical to a TpInt32 type.

5.1.10 TpSessionIDSet

Defines a Numbered Set of Data Elements of TpSessionID.

5.1.11 TpAny

Defines a type that can hold any type. This is not restricted to only the primitive types.

5.1.12 TpAttribute

This is a `Sequence of Data Elements` containing the attribute name, type, and value. The attribute Value is interpreted based on the value of the attribute Type.

Sequence Element Name	Sequence Element Type	Notes
`AttributeName`	`TpString`	The name of the attribute.
`AttributeType`	`TpAttributeType`	The type of the attirbute. Valid values for Type must include at least TpString, TpInt32 and TpFloat.
`AttributeValue`	`TpAny`	The values for the attribute. This model allows multi-valued attributes. Cannot be an empty list.

5.1.13 TpAttributeType

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

Character String Value	Description
NULL	An empty (NULL) string indicates no attribute type
P_STRING	Attribute type is type `TpString`.
P_INT32	Attribute type is type `TpInt32`.
P_FLOAT	Attribute type is type `TpFloat`.

5.1.14 TpAttributeList

This is a `Numbered List of Data Elements` of type TpAttribute.

5.1.15 TpAttributeSet

This is a `Numbered Set of Data Elements` of type TpAttribute.

5.1.16 TpInt64

Defines a signed 64-bit integer.

5.1.17 TpVersion

This data type is identical to `TpString`. It is used to uniquely identify the implemented version of the framework or an SCF. The syntax for this datatype is defined as:

P_<publishing body>_<version number>

Where:

<publishing body> is one of the strings listed in the table below.

Character String Value	Description
PARLAY	Specification released by The Parlay Group.
ETSI	Specification released by ETSI.
3GPP	Specification released by 3GPP.

<version number> consists of numbers separated by underscores (e.g. 3_1). It is recommended that not more than the two most significant numbers (major and minor version) of the version are used.

Examples of version strings are:

Character String Value	Description
P_PARLAY_3_1	Parlay v3.1
P_ETSI_2_0	ETSI v2.0
P_3GPP_4_3	3GPP Release 4.3

Note that different version strings can be aliases of each other all pointing to the same SCF/Framework version.

5.1.18 TpStringSet

Defines a `Numbered Set of Data Elements` of type `TpString`.

5.1.19 TpStringList

Defines a `Numbered List of Data Elements` of type `TpString`.

5.1.16 TpUnorderedOctetSet

Defines a Numbered Set of Data elements of TpOctet. Note that this is an un-ordered set.

Note that this type should not be removed from this specification, even if unused by any part of the OSA specifications. It is included to ensure that TpOctetSet is correctly used as a Numbered List of Data Elements, and not a Numbered Set.

5.2 Other Data sorts

The APIs assumes that the following data syntaxes can be supported:

5.2.1 Sequence of Data Elements

This describes a sequence of data types. This may be defined as a structure (for example, in C++) or simply a sequence of data elements within a structure.

EXAMPLE: The TpAddress data type may be defined in C++ as:

```
typedef struct {
  TpAddressPlan           Plan;
  TpString                AddrString;
  TpString                Name;
  TpAddressPresentation   Presentation;
  TpAddressScreening      Screening;
  TpString                SubAddressString;
} TpAddress;
```

5.2.2 Tagged Choice of Data Elements

This describes a data type which actually evaluates to one of a choice of a number of data elements. This data element contains two parts: a tag data type (the *tag* part) which is used to identify the chosen data type, and the chosen data type itself (the *union* part). This form of data type is also referred to as a tagged union.

This data type can be implemented (for example, in C++) as a structure containing an integer for the *tag* part, and a union for the *union* part.

This data type is implementation specific. Please refer to the appropriate IDL documents (and the resulting language mappings) to see how this data type is implemented.

EXAMPLE: The `TpCallError` data type may be defined in C++ as:

```
typedef struct {
  TpCallErrorType Tag;
  union {
    TpCallErrorInfoUndefined     Undefined;
```

```
    TpCallErrorInfoRoutingAborted  RoutingAborted;
    TpCallErrorInfoCallAbandoned   CallAbandoned;
    TpCallErrorInfoInvalidAddress  InvalidAddress;
    TpCallErrorInfoInvalidState   InvalidState;
    TpCallErrorInfoInvalidCriteria  InvalidCriteria;
  } callErrorInfo;
} TpCallError;
```

5.2.3 Numbered Set of Data Elements

This describes a data type which comprises an integer which indicates the total number of data elements in the set (the *number* part), and an **unordered** set of data elements (the *data* part). *Set* data types do not contain duplicate data elements.

Note that TpOctetSet is a Numbered List of Data Elements, not a Numbered Set.

EXAMPLE: The TpAddressSet data type may be defined in C++ as:

```
typedef struct {
  TpInt32 Number;
  TpAddress Set[Number];
} TpAddressSet;
```

5.2.4 Reference

This describes a reference (or pointer) to a data type.

5.2.5 Numbered List of Data Elements

This describes a data type which comprises an integer which indicates the total number of data elements in the set (the *number* part), and an **ordered** set of data elements (the *data* part). *List* data types can contain duplicate data elements.

EXAMPLE: The TpStringList data type may be defined in C++ as:

```
typedef struct {
  TpInt32 Number;
  TpString List[Number];
} TpStringList;
```

End of Change in Clause 5
End of Document

Annex E (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	--	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	4.0.0
Jun 2001	CN_12	NP-010330	001	--	Corrections to OSA API Rel4 (Exception handling mechanism without ambiguity - Replace TpGeneralException and TpResultInfo with detailed exception classes which can be thrown for each method (N5-010261)	4.0.0	4.1.0
Jun 2001	CN_12	NP-010333	002	--	Introduction of TpOctet (In order to make sure that some data is sent over the "distributed wire" untouched a new data type is needed) (N5-010304)	4.0.0	4.1.0
Sep 2001	CN_13	NP-010465	003	--	Changing references to JAIN	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	004	--	Clarification of common exceptions	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	005	--	Invalid parameter value exception for SLA violation	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	006	--	Storing eventCriteria	4.1.0	4.2.0
Dec 2001	CN_14	NP-010595	007	--	Replace Out Parameters with Return Types	4.2.0	4.3.0
Dec 2001	CN_14	NP-010595	008	--	Correction to Common Data (CD)	4.2.0	4.3.0
Dec 2001	CN_14	NP-010595	009	--	Correction to values of TpAddressPlan	4.2.0	4.3.0
Mar 2002	CN_15	NP-020104	010	--	Ambiguous definition of TpAssignmentID	4.3.0	4.4.0
Mar 2002	CN_15	NP-020104	011	--	Data type alignment in the common data types	4.3.0	4.4.0
Jun 2002	CN_16	NP-020185	011	--	Allowing the use of tel URL in TpAddressPlan	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	012	--	Adding TpInt64 in order to aling with the new Rel-5 TS 29.198-14	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	013	--	Addition of undefined Data types: TpStringList and TpStringSet	4.4.0	5.0.0
Jun 2002	CN_16	NP-020181	014	--	Addition of support for Java API technology realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020182	015	--	Addition of support for WSDL realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	016	--	Deletion of P_SET_LENGTH_EXCEEDED	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	017	--	Removal of MIDL	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	018	--	Revise the scope of TpSessionID and TpAssignmentID	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	019	--	Deprecate P_ADDRESS_PLAN_MSMAIL	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	020	--	Addition of support for an Exception Hierarchy	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	021	--	Addition of type TpVersion in common data	4.4.0	5.0.0
Sep 2002	CN_17	NP-020395	022	--	Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications	5.0.0	5.1.0
Oct 2002	--	--	--	--	Added the two missing attachments (osa.idl contained in archive 2919802IDL.ZIP) (osa.wsdl contained in archive 2919802WSDL.ZIP)	5.1.0	5.1.1
Mar 2003	CN_19	NP-030018	025	--	Clarification on uniqueness of assignmentID	5.1.1	5.2.0
Mar 2003	CN_19	NP-030018	027	--	Correction to P_INVALID_STATE value	5.1.1	5.2.0
Mar 2003	CN_19	NP-030018	029	--	Addition of Support of National Numbering Plans	5.1.1	5.2.0
Mar 2003	CN_19	NP-030027	030	--	Addition of Numbered List of Data Elements definition	5.1.1	5.2.0
Mar 2003	CN_19	NP-030027	031	--	Correction of Exception Hierarchy to align with Java Realisation	5.1.1	5.2.0
Mar 2003	CN_19	NP-030027	032	--	Promotion of TpDataSessionQosClass dat type definition to the Common Data Types	5.1.1	5.2.0
Jun 2003	CN_20	NP-030236	034	--	Correction of SIP Address wildcard rules	5.2.0	5.3.0
Jun 2003	CN_20	NP-030240	035	--	Add the type TpURN to Common Data Types	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	036	--	Correction to Java Realisation Annex	5.3.0	5.4.0

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-02 CR 039 ⌘rev - ⌘ Current version: 4.6.0 ⌘

*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:	⌘	Clarify use of base interface Reference within an inheritance relationship
Source:	⌘	CN5 (Marconi)
Work item code:	⌘	OSA1
Date:	⌘	31/10/2003
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)*
Rel-6	*(Release 6)*

Reason for change:	⌘	Within an inheritance relationship, a reference to the inheriting interface may be used in place of the base interface Reference. The current text in the Common Data definitions (Part 2) makes no reference to this behaviour
Summary of change:	⌘	Text is added to section 5.2.4 which clarifies that within an inheritance relationship, a reference to the inheriting interface may be used in place of the base interface Reference.
Consequences if not approved:	⌘	The current text is insufficient in describing how the Reference parameter can be used. Implementations could develop other workarounds to achieve the same result as re-use (e.g. define a new proprietary parameter) however the use of such workarounds would result in interworking difficulties between implementations thus mis-operation will occur.

Clauses affected:	⌘	5.2.4			
Other specs affected:	⌘	**Y** / **N**	Other core specifications	⌘	Rel-5 29.198-02
		X /	Other core specifications		
		/ **X**	Test specifications		
		/ **X**	O&M Specifications		
Other comments:	⌘	Mirror Rel-5 CR 29.198-02 in N5-030623			

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

Change in Clause 5.2

5.2 Other Data sorts

The APIs assumes that the following data syntaxes can be supported:

5.2.1 Sequence of Data Elements

This describes a sequence of data types. This may be defined as a structure (for example, in C++) or simply a sequence of data elements within a structure.

EXAMPLE: The TpAddress data type may be defined in C++ as:

```
typedef struct {
  TpAddressPlan           Plan;
  TpString                AddrString;
  TpString                Name;
  TpAddressPresentation   Presentation;
  TpAddressScreening      Screening;
  TpString                SubAddressString;
} TpAddress;
```

5.2.2 Tagged Choice of Data Elements

This describes a data type which actually evaluates to one of a choice of a number of data elements. This data element contains two parts: a tag data type (the *tag* part) which is used to identify the chosen data type, and the chosen data type itself (the *union* part). This form of data type is also referred to as a tagged union.

This data type can be implemented (for example, in C++) as a structure containing an integer for the *tag* part, and a union for the *union* part.

This data type is implementation specific. Please refer to the appropriate IDL documents (and the resulting language mappings) to see how this data type is implemented.

EXAMPLE: The `TpCallError` data type may be defined in C++ as:

```
typedef struct {
  TpCallErrorType Tag;
  union {
    TpCallErrorInfoUndefined     Undefined;
    TpCallErrorInfoRoutingAborted  RoutingAborted;
    TpCallErrorInfoCallAbandoned   CallAbandoned;
    TpCallErrorInfoInvalidAddress  InvalidAddress;
    TpCallErrorInfoInvalidState   InvalidState;
    TpCallErrorInfoInvalidCriteria  InvalidCriteria;
  } callErrorInfo;
} TpCallError;
```

5.2.3 Numbered Set of Data Elements

This describes a data type which comprises an integer which indicates the total number of data elements in the set (the *number* part), and an **unordered** set of data elements (the *data* part). *Set* data types do not contain duplicate data elements.

EXAMPLE: The TpAddressSet data type may be defined in MIDL as:

```
typedef struct TpAddressSet
{
TpInt32 Number; [size_is(Number)] TpAddress Set[];
}
TpAddressSet;
```

5.2.4 Reference

This describes a reference (or pointer) to a data type. When one interface inherits directly from another interface, a reference to the inheriting interface can be used when the base interface reference is required. For example, the

references to Multi-Media Call Control interfaces can be used in Multi-Party Call Control methods that have been inherited by Multi-Media Call Control from Multi-Party Call Control interfaces.

End of Change in Clause 5.2
End of Document

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	--	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	4.0.0
Jun 2001	CN_12	NP-010330	001	--	Corrections to OSA API Rel4 (Exception handling mechanism without ambiguity - Replace TpGeneralException and TpResultInfo with detailed exception classes which can be thrown for each method (N5-010261)	4.0.0	4.1.0
Jun 2001	CN_12	NP-010333	002	--	Introduction of TpOctet (In order to make sure that some data is sent over the “distributed wire” untouched a new data type is needed) (N5-010304)	4.0.0	4.1.0
Sep 2001	CN_13	NP-010465	003	--	Changing references to JAIN	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	004	--	Clarification of common exceptions	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	005	--	Invalid parameter value exception for SLA violation	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	006	--	Storing eventCriteria	4.1.0	4.2.0
Dec 2001	CN_14	NP-010595	007	--	Replace Out Parameters with Return Types	4.2.0	4.3.0
Dec 2001	CN_14	NP-010595	008	--	Correction to Common Data (CD)	4.2.0	4.3.0
Dec 2001	CN_14	NP-010595	009	--	Correction to values of TpAddressPlan	4.2.0	4.3.0
Mar 2002	CN_15	NP-020104	010	--	Ambiguous definition of TpAssignmentID	4.3.0	4.4.0
Mar 2002	CN_15	NP-020104	011	--	Data type alignment in the common data types	4.3.0	4.4.0
Mar 2003	CN_19	NP-030018	023	--	Correction to defintion of sessionID	4.4.0	4.5.0
Mar 2003	CN_19	NP-030018	024	--	Clarification on uniqueness of assignmentID	4.4.0	4.5.0
Mar 2003	CN_19	NP-030018	026	--	Correction to P_INVALID_STATE value	4.4.0	4.5.0
Mar 2003	CN_19	NP-030018	028	--	Addition of Support of National Numbering Plans	4.4.0	4.5.0
Jun 2003	CN_20	NP-030236	033	--	Correction of SIP Address wildcard rules	4.5.0	4.6.0

CR-Form-v7

CHANGE REQUEST

⌘ 29.198-02 CR 040 ⌘rev - ⌘ Current version: 5.4.0 ⌘

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

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

Title:	⌘	Clarify use of base interface Reference within an inheritance relationship
Source:	⌘	CN5 (Marconi)
Work item code:	⌘	OSA1
Date:	⌘	31/10/2003
Category:	⌘	**A**
Release:	⌘	*REL-5*

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:	⌘	Within an inheritance relationship, a reference to the inheriting interface may be used in place of the base interface Reference. The current text in the Common Data definitions (Part 2) makes no reference to thisbehaviour.
Summary of change:	⌘	Text is added to section 5.2.4 which clarifies that within an inheritance relationship, a reference to the inheriting interface may be used in place of the base interface Reference.
Consequences if not approved:	⌘	The current text is insufficient in describing how the Reference parameter can be used. Implementations could develop other workarounds to achieve the same result as re-use (e.g. define a new proprietary parameter) however the use of such workarounds would result in interworking difficulties between implementations thus mis-operation will occur.

Clauses affected:	⌘	5.2.4

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

⌘

Other comments:	⌘	Rel-5 Mirror CR 29.198-02 of N5-030622

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

Change in Clause 5.2

5.2 Other Data sorts

The APIs assumes that the following data syntaxes can be supported:

5.2.1 Sequence of Data Elements

This describes a sequence of data types. This may be defined as a structure (for example, in C++) or simply a sequence of data elements within a structure.

EXAMPLE: The TpAddress data type may be defined in C++ as:

```
typedef struct {
  TpAddressPlan       Plan;
  TpString         AddrString;
  TpString         Name;
  TpAddressPresentation.....Presentation;
....TpAddressScreening........Screening;
....TpString..................SubAddressString;
} TpAddress;
```

5.2.2 Tagged Choice of Data Elements

This describes a data type which actually evaluates to one of a choice of a number of data elements. This data element contains two parts: a tag data type (the *tag* part) which is used to identify the chosen data type, and the chosen data type itself (the *union* part). This form of data type is also referred to as a tagged union.

This data type can be implemented (for example, in C++) as a structure containing an integer for the *tag* part, and a union for the *union* part.

This data type is implementation specific. Please refer to the appropriate IDL documents (and the resulting language mappings) to see how this data type is implemented.

EXAMPLE: The `TpCallError` data type may be defined in C++ as:

```
typedef struct {
  TpCallErrorType Tag;
  union {
    TpCallErrorInfoUndefined     Undefined;
    TpCallErrorInfoRoutingAborted  RoutingAborted;
    TpCallErrorInfoCallAbandoned   CallAbandoned;
    TpCallErrorInfoInvalidAddress  InvalidAddress;
    TpCallErrorInfoInvalidState   InvalidState;
    TpCallErrorInfoInvalidCriteria  InvalidCriteria;
  } callErrorInfo;
} TpCallError;
```

5.2.3 Numbered Set of Data Elements

This describes a data type which comprises an integer which indicates the total number of data elements in the set (the *number* part), and an **unordered** set of data elements (the *data* part). *Set* data types do not contain duplicate data elements.

EXAMPLE: The TpAddressSet data type may be defined in C++ as:

```
typedef struct {
  TpInt32 Number;
  TpAddress Set[Number];
} TpAddressSet;
```

5.2.4 Reference

This describes a reference (or pointer) to a data type. When one interface inherits directly from another interface, a reference to the inheriting interface can be used when the base interface reference is required. For example, the

references to Multi-Media Call Control interfaces can be used in Multi-Party Call Control methods that have been inherited by Multi-Media Call Control from Multi-Party Call Control interfaces.

5.2.5 Numbered List of Data Elements

This describes a data type which comprises an integer which indicates the total number of data elements in the set (the *number* part), and an **ordered** set of data elements (the *data* part). *List* data types can contain duplicate data elements.

EXAMPLE: The TpStringList data type may be defined in C++ as:

```
typedef struct {
  TpInt32 Number;
  TpString List[Number];
} TpStringList;
```

End of Change in Clause 5.2
End of Document

Annex E (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Mar 2001	CN_11	NP-010134	047	--	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	4.0.0
Jun 2001	CN_12	NP-010330	001	--	Corrections to OSA API Rel4 (Exception handling mechanism without ambiguity - Replace TpGeneralException and TpResultInfo with detailed exception classes which can be thrown for each method (N5-010261)	4.0.0	4.1.0
Jun 2001	CN_12	NP-010333	002	--	Introduction of TpOctet (In order to make sure that some data is sent over the "distributed wire" untouched a new data type is needed) (N5-010304)	4.0.0	4.1.0
Sep 2001	CN_13	NP-010465	003	--	Changing references to JAIN	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	004	--	Clarification of common exceptions	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	005	--	Invalid parameter value exception for SLA violation	4.1.0	4.2.0
Sep 2001	CN_13	NP-010465	006	--	Storing eventCriteria	4.1.0	4.2.0
Dec 2001	CN_14	NP-010595	007	--	Replace Out Parameters with Return Types	4.2.0	4.3.0
Dec 2001	CN_14	NP-010595	008	--	Correction to Common Data (CD)	4.2.0	4.3.0
Dec 2001	CN_14	NP-010595	009	--	Correction to values of TpAddressPlan	4.2.0	4.3.0

Mar 2002	CN_15	NP-020104	010	--	Ambiguous definition of TpAssignmentID	4.3.0	4.4.0
Mar 2002	CN_15	NP-020104	011	--	Data type alignment in the common data types	4.3.0	4.4.0
Jun 2002	CN_16	NP-020185	011	--	Allowing the use of tel URL in TpAddressPlan	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	012	--	Adding TpInt64 in order to aling with the new Rel-5 TS 29.198-14	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	013	--	Addition of undefined Data types: TpStringList and TpStringSet	4.4.0	5.0.0
Jun 2002	CN_16	NP-020181	014	--	Addition of support for Java API technology realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020182	015	--	Addition of support for WSDL realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	016	--	Deletion of P_SET_LENGTH_EXCEEDED	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	017	--	Removal of MIDL	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	018	--	Revise the scope of TpSessionID and TpAssignmentID	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	019	--	Deprecate P_ADDRESS_PLAN_MSMAIL	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	020	--	Addition of support for an Exception Hierarchy	4.4.0	5.0.0
Jun 2002	CN_16	NP-020185	021	--	Addition of type TpVersion in common data	4.4.0	5.0.0
Sep 2002	CN_17	NP-020395	022	--	Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications	5.0.0	5.1.0
Oct 2002	--	--	--	--	Added the two missing attachments (osa.idl contained in archive 2919802IDL.ZIP) (osa.wsdl contained in archive 2919802WSDL.ZIP)	5.1.0	5.1.1
Mar 2003	CN_19	NP-030018	025	--	Clarification on uniqueness of assignmentID	5.1.1	5.2.0
Mar 2003	CN_19	NP-030018	027	--	Correction to P_INVALID_STATE value	5.1.1	5.2.0
Mar 2003	CN_19	NP-030018	029	--	Addition of Support of National Numbering Plans	5.1.1	5.2.0
Mar 2003	CN_19	NP-030027	030	--	Addition of Numbered List of Data Elements definition	5.1.1	5.2.0
Mar 2003	CN_19	NP-030027	031	--	Correction of Exception Hierarchy to align with Java Realisation	5.1.1	5.2.0
Mar 2003	CN_19	NP-030027	032	--	Promotion of TpDataSessionQosClass dat type definition to the Common Data Types	5.1.1	5.2.0
Jun 2003	CN_20	NP-030236	034	--	Correction of SIP Address wildcard rules	5.2.0	5.3.0
Jun 2003	CN_20	NP-030240	035	--	Add the type TpURN to Common Data Types	5.2.0	5.3.0
Sep 2003	CN_21	NP-030352	036	--	Correction to Java Realisation Annex	5.3.0	5.4.0