

ETSI TS 129 198-1 V5.0.0 (2002-06)

Technical Specification

**Universal Mobile Telecommunications System (UMTS);
Open Service Access (OSA)
Application Programming Interface (API);
Part 1: Overview
(3GPP TS 29.198-01 version 5.0.0 Release 5)**



Reference

RTS/TSGN-0529198-01v500

Keywords

UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at

<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, send your comment to:

editor@etsi.fr

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2002.
All rights reserved.

DECTTM, **PLUGTESTS**TM and **UMTS**TM are Trade Marks of ETSI registered for the benefit of its Members.
TIPHONTM and the **TIPHON logo** are Trade Marks currently being registered by ETSI for the benefit of its Members.
3GPPTM is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: "*Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards*", which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<http://webapp.etsi.org/IPR/home.asp>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under www.etsi.org/key.

Contents

Intellectual Property Rights	2
Foreword.....	2
Foreword.....	4
Introduction	4
1 Scope	6
2 References	6
3 Definitions and abbreviations.....	7
3.1 Definitions	7
3.2 Abbreviations	8
4 Open Service Access APIs	9
5 Structure of the OSA API (29.198) and Mapping (29.998) documents	10
6 Methodology	11
6.1 Tools and Languages.....	11
6.2 Packaging	11
6.3 Colours	11
6.4 Naming scheme	11
6.5 State Transition Diagram text and text symbols.....	12
6.6 Exception handling and passing results.....	12
6.7 References	12
6.8 Strings and Collections.....	12
6.9 Prefixes.....	13
Annex A (normative): OMG IDL	14
A.1 Tools and Languages.....	14
A.2 Strings and Collections.....	14
A.3 Naming space across CORBA modules	14
Annex B (informative): W3C WSDL.....	15
B.1 Tools and Languages.....	15
B.2 Proposed Namespaces for the OSA WSDL	15
B.3 Object References.....	16
B.4 Mapping UML Data Types to XML Schema	17
B.4.1 Data Types	17
B.4.1.1 <<Constant>>	17
B.4.1.2 <<NameValuePair>>	17
B.4.1.3 <<SequenceOfDataElements>>.....	17
B.4.1.4 <<TypeDef>>	18
B.4.1.5 <<NumberedSetOfDataElements>>	18
B.4.1.6 <<TaggedChoiceOfDataElements>>	19
B.5 Mapping of UML SCF to WSDL.....	19
B.5.1 Mapping of Operations to WSDL <i>message</i> element.....	19
B.5.2 Mapping of Exception to WSDL <i>message</i> element	20
B.5.3 Mapping of CommonExceptions to WSDL <i>message</i> element.....	20
B.5.4 Mapping of Interface Class to WSDL <i>portType</i> and <i>binding</i> elements	20
B.5.5 Mapping of UML SCF to WSDL <i>service</i> element.....	22
Annex C (informative): Java API	23
C.1 Tools and Languages.....	23
C.2 JAIN SPA Overview	23
Annex D (informative): Change history	24
History	25

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
 - 1 presented to TSG for information;
 - 2 presented to TSG for approval;
 - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document is part 1 of a multi-part TS covering the 3rd Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Application Programming Interface (API), as identified below. The **API specification** (3GPP TS 29.198) is structured in the following Parts:

Part 1:	"Overview";	
Part 2:	"Common Data Definitions";	
Part 3:	"Framework";	
Part 4:	"Call Control";	
	Sub-part 1: "Call Control Common Definitions";	(new in 3GPP Release 5)
	Sub-part 2: "Generic Call Control SCF";	(new in 3GPP Release 5)
	Sub-part 3: "Multi-Party Call Control SCF";	(new in 3GPP Release 5)
	Sub-part 4: "Multi-Media Call Control SCF";	(new in 3GPP Release 5)
	Sub-part 5: "Conference Call Control SCF";	(not part of 3GPP Release 5)
Part 5:	"User Interaction SCF";	
Part 6:	"Mobility SCF";	
Part 7:	"Terminal Capabilities SCF";	
Part 8:	"Data Session Control SCF";	
Part 9:	"Generic Messaging SCF";	(not part of 3GPP Release 5)
Part 10:	"Connectivity Manager SCF";	(not part of 3GPP Release 5)
Part 11:	"Account Management SCF";	
Part 12:	"Charging SCF".	
Part 13:	"Policy Management SCF";	(new in 3GPP Release 5)
Part 14:	"Presence and Availability Management SCF";	(new in 3GPP Release 5)

The **Mapping specification of the OSA APIs and network protocols** (3GPP TR 29.998) is also structured as above. A mapping to network protocols is however not applicable for all Parts, but the numbering of Parts is kept. Also in case a Part is not supported in a Release, the numbering of the parts is maintained.

Table: Overview of the OSA APIs & Protocol Mappings 29.198 & 29.998-family

OSA API specifications 29.198-family					OSA API Mapping - 29.998-family	
29.198-01	Overview				29.998-01	Overview
29.198-02	Common Data Definitions				29.998-02	<i>Not Applicable</i>
29.198-03	Framework				29.998-03	<i>Not Applicable</i>
Call Control (CC) SCF	29.198-04-1	29.198-04-2	29.198-04-3	29.198-04-4	29.998-04-1	Generic Call Control – CAP mapping
	Common CC data definitions	Generic CC SCF	Multi-Party CC SCF	Multi-media CC SCF	29.998-04-2	<i>Generic Call Control – INAP mapping</i>
					29.998-04-3	<i>Generic Call Control – Megaco mapping</i>
					29.998-04-4	Multiparty Call Control – SIP mapping
29.198-05	User Interaction SCF				29.998-05-1	User Interaction – CAP mapping
					29.998-05-2	<i>User Interaction – INAP mapping</i>
					29.998-05-3	<i>User Interaction – Megaco mapping</i>
					29.998-05-4	User Interaction – SMS mapping
29.198-06	Mobility SCF				29.998-06	User Status and User Location – MAP mapping
29.198-07	Terminal Capabilities SCF				29.998-07	<i>Not Applicable</i>
29.198-08	Data Session Control SCF				29.998-08	Data Session Control – CAP mapping
29.198-09	<i>Generic Messaging SCF</i>				29.998-09	<i>Not Applicable</i>
29.198-10	<i>Connectivity Manager SCF</i>				29.998-10	<i>Not Applicable</i>
29.198-11	Account Management SCF				29.998-11	<i>Not Applicable</i>
29.198-12	Charging SCF				29.998-12	<i>Not Applicable</i>
29.198-13	Policy Management SCF				29.998-13	<i>Not Applicable</i>
29.198-14	Presence & Availability Management SCF				29.998-14	<i>Not Applicable</i>

1 Scope

The present document is the first part of the 3GPP Specification defining the Application Programming Interface (API) for Open Service Access (OSA), and provides an overview of the content and structure of the various parts of this specification, and of the relation to other standards documents .

The OSA-specifications define an architecture that enables service application developers to make use of network functionality through an open standardised interface, i.e. the OSA APIs. The concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

This specification has been defined jointly between ETSI SPAN12, 3GPP TSG CN WG5 and the Parlay consortium [24], in co-operation with a number of JAIN™ Community [25] member companies. [25].

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

- [1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".
- [2] 3GPP TS 22.127: "Service Requirement for the Open Services Access (OSA)".
- [3] 3GPP TS 23.127: "Virtual Home Environment / Open Service Access (OSA)".
- [4] 3GPP TS 23.078: "Customised Applications for Mobile network Enhanced Logic (CAMEL); Stage 2".
- [5] 3GPP TS 22.101: "Service Aspects; Service Principles".
- [6] World Wide Web Consortium "Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation" (<http://www.w3.org/TR/NOTE-CCPP/>).
- [7] 3GPP TS 29.002: "Mobile Application Part (MAP) specification".
- [8] 3GPP TS 29.078: "Customised Applications for Mobile network Enhanced Logic (CAMEL); CAMEL Application Part (CAP) specification".
- [9] Wireless Application Protocol (WAP), Version 2.0: "User Agent Profiling Specification" (WAP-248) (<http://www.wapforum.org/what/technical.htm>).
- [10] Wireless Application Protocol (WAP), Version 2.0: "WAP Service Indication Specification" (WAP-167) (<http://www.wapforum.org/what/technical.htm>).
- [11] Wireless Application Protocol (WAP), Version 2.0: "Push Architectural Overview" (WAP-250) (<http://www.wapforum.org/what/technical.htm>).
- [12] Wireless Application Protocol (WAP), Version 2.0: "Wireless Application Protocol Architecture Specification" (WAP-210) (<http://www.wapforum.org/what/technical.htm>).
- [13] IDL to Java Compiler (<http://www.javasoft.com/products/jdk/idl/index.html>).
- [14] UML Unified Modelling Language (<http://www.omg.org/uml>).

- [15] Object Management Group (<http://www.omg.org>).
- [16] 3GPP TS 22.002: "Circuit Bearer Services (BS) supported by a Public Land Mobile Network (PLMN)".
- [17] 3GPP TS 22.003: "Circuit Teleservices supported by a Public Land Mobile Network (PLMN)".
- [18] 3GPP TS 24.002: "GSM - UMTS Public Land Mobile Network (PLMN) Access Reference Configuration".
- [19] ITU-T Q.763: "Signalling System No. 7 – ISDN user part formats and codes".
- [20] ITU-T Q.931: "ISDN user-network interface layer 3 specification for basic call control".
- [21] ISO 8601: "Data elements and interchange formats -- Information interchange -- Representation of dates and times".
- [22] ISO 4217: "Codes for the representation of currencies and funds".
- [23] 3GPP TS 22.121: "Service aspects; The Virtual Home Environment; Stage 1".
- [24] "The Parlay Group homepage" (<http://www.parlay.org>)
- [25] "JAIN Community homepage" (<http://www.java.sun.com/products/jain>)
- [26] 3GPP TS 23.057: "Mobile Execution Environment (MExE); Functional Description; Stage 2".
- [27] "JSR Overview" (<http://jcp.org/jsr/overview/index.en.jsp>)
- [28] "Java 2 SDK, Standard Edition" (<http://java.sun.com/j2se/1.4/docs/relnotes/features.html>)
- [29] "Java Community Process" (<http://jcp.org/>)
- [30] "World Wide Web Consortium homepage" (<http://www.w3c.org>)
- [31] 3GPP TS 23.271 "Functional stage 2 description of location services (3GPP TS 23.271)"

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TS 22.101 [5] and the following apply.

Applications: Services, which are designed using Service Capability Features (SCFs).

Gateway: Synonym for Service Capability Server (SCS). From the viewpoint of applications, an SCS can be seen as a gateway to the core network.

HE-VASP: Home Environment Value Added Service Provider. This is a VASP that has an agreement with the Home Environment to provide services.

Home Environment: responsible for overall provision of services to users.

Local Service: A service, which can be exclusively provided in the current serving network by a Value Added Service Provider.

OSA Interface: Standardised Interface used by application to access service capability features.

Personal Service Environment (PSE): contains personalised information defining how subscribed services are provided and presented towards the user. The Personal Service Environment is defined in terms of one or more User Profiles.

Service Capabilities: Bearers defined by parameters, and/or mechanisms needed to realise services. These are within networks and under network control.

Service Capability Feature (SCF): Functionality offered by service capabilities that are accessible via the standardised OSA interface.

Service Capability Server (SCS): Functional Entity providing OSA interfaces towards an application.

Service: term used as an alternative for Service Capability Feature in this specification.

User Interface Profile: Contains information to present the personalised user interface within the capabilities of the terminal and serving network.

User Profile: This is a label identifying a combination of one user interface profile, and one user services profile.

User Services Profile: Contains identification of subscriber services, their status and reference to service preferences.

Value Added Service Provider: provides services other than basic telecommunications service for which additional charges may be incurred.

Virtual Home Environment: A concept for personal service environment portability across network boundaries and between terminals.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply.

API	Application Programming Interface
CAMEL	Customised Application for Mobile network Enhanced Logic
CAP	CAMEL Application Part
CSE	CAMEL Service Environment
FW	Framework
HE	Home Environment
HE-VASP	Home Environment - Value Added Service Provider
HLR	Home Location Register
INAP	Intelligent Networks Application Part
IDL	Interface Description Language
JSR	Java Specification Request
MAP	Mobile Application Part
ME	Mobile Equipment
MExE	Mobile Station (Application) Execution Environment
MS	Mobile Station
MSC	Mobile Switching Centre
OSA	Open Service Access
PLMN	Public Land Mobile Network
PSE	Personal Service Environment
RMI	Java Remote Method Invocation
SAT	SIM Application Tool-Kit
SCF	Service Capability Feature
SCP	Service Control Point
SCS	Service Capability Server
SIM	Subscriber Identity Module
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SPA	Service Provider API
UE	User Equipment
USIM	Universal Subscriber Identity Module
VLR	Visited Location Register
VASP	Value Added Service Provider
VHE	Virtual Home Environment
WAP	Wireless Application Protocol
WGP	Wireless Gateway Proxy

WPP	Wireless Push Proxy
WSDL	Web Services Definition Language
XML	Extensible Markup Language

4 Open Service Access APIs

The OSA-specifications define an architecture that enables service application developers to make use of network functionality through an open standardised interface, i.e. the OSA APIs. The network functionality is describes as Service Capability Features (SCFs) or Services. The OSA Framework is a general component in support of Services (Service Capabilities) and Applications. The concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The OSA API is split into three types of interface classes, Service and Framework (FW).

- Interface classes between the Applications and the Framework (FW), that provide applications with basic mechanisms (e.g. Authentication) that enable them to make use of the service capabilities in the network.
- Interface classes between Applications and SCFs, which are individual services that may be required by the client to enable the running of third party applications over the interface e.g. Messaging type service.
- Interface classes between the Framework (FW) and the SCFs, that provide the mechanisms necessary for a multi-vendor environment.

These interfaces represent interfaces 1, 2 and 3 in Figure 1 below. The other interfaces are not yet part of the scope of the work.

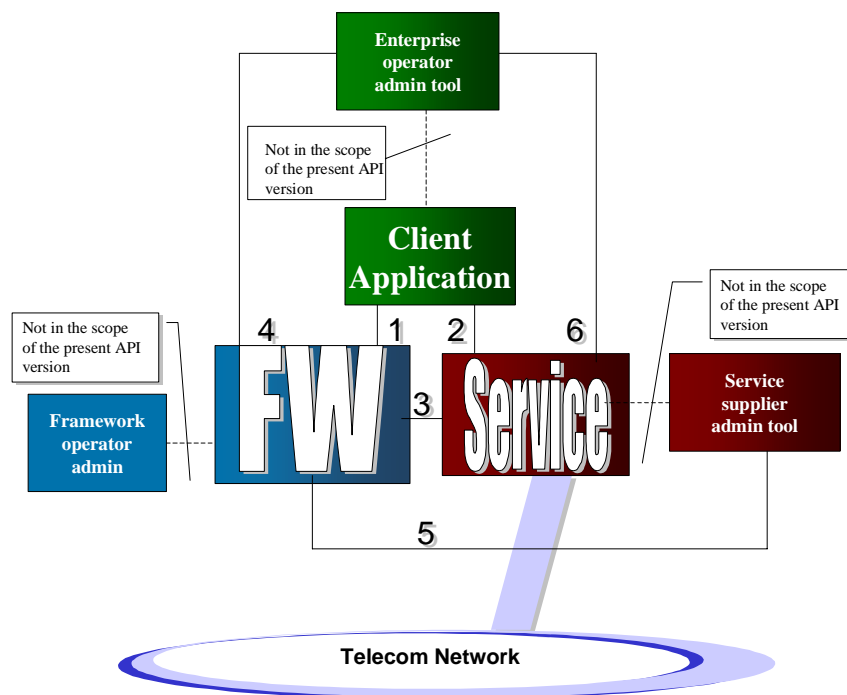


Figure 1:

Within the OSA concept a set of Service Capability Features (SCFs) has been specified. The OSA documentation is structured in parts. The first Part (the present document) contains an overview, the second Part contains common data definitions, the third Part the Framework interfaces and the following Parts contain the description of the SCFs.

NOTE: The terms ‘Service’ and ‘Service Capability Feature’ are used as alternatives for the same concept in the present document. In the OSA API itself the SCFs as identified in the 3GPP requirements and architecture are reflected as ‘service’, in terms like service instance lifecycle manager, service Discovery.

5 Structure of the OSA API (29.198) and Mapping (29.998) documents

The Open Service Access (OSA) Application Programming Interface (API) specifications consist of two sets of documents:

API specification (3GPP TS 29.198)

The Parts of 29.198 - apart from Part 1 (the present document) and Part 2 - define the interfaces, parameters and state models that belong to the API specification. UML (Unified Modelling Language) is used to specify the interface classes.

As such it provides a UML interface class description of the methods (API calls) supported by that interface and the relevant parameters and types. The interfaces are specified both in IDL (Interface Description Language) and in WSDL (Web Services Definition Language). Reference is made to the Java API specification of the interfaces.

Mapping specification of the OSA APIs and network protocols (3GPP TR 29.998)

The Parts of 29.998 contain a possible mapping from the APIs defined in 29.198 to various network protocols (i.e. MAP [7], CAP [8], etc.). It is an informative document, since this mapping is considered as implementation- / vendor-dependent. On the other hand this mapping will provide potential service designers with a better understanding of the relationship of the OSA API interface classes and the behaviour of the network associated to these interface classes.

The purpose of the OSA API is to shield the complexity of the network, its protocols and specific implementation from the applications. This means that applications do not have to be aware of the network nodes, a Service Capability Server interacts with, in order to provide the SCFs to the application. The specific underlying network and its protocols are transparent to the application.

The **API specification** (3GPP TS 29.198) is structured in the following Parts:

29.198-1	Part 1:	Overview
29.198-2	Part 2:	Common Data Definitions
29.198-3	Part 3:	Framework
29.198-4	Part 4:	Call Control SCF
29.198-5	Part 5:	User Interaction SCF
29.198-6	Part 6:	Mobility SCF
29.198-7	Part 7:	Terminal Capabilities SCF
29.198-8	Part 8:	Data Session Control SCF
29.198-9	Part 9:	Generic Messaging SCF
29.198-10	Part 10:	Connectivity Manager SCF
29.198-11	Part 11:	Account Management SCF
29.198-12	Part 12:	Charging SCF
29.198-13	Part 13:	Policy Management SCF
29.198-14	Part 14:	Presence & Availability Management SCF

The **Mapping specification of the OSA APIs and network protocols** (3GPP TR 29.998) is also structured as above. A mapping to network protocols is however not applicable for all Parts, but the numbering of Parts is kept. Also in case a Part is not supported in a Release, the numbering of the parts is maintained.

Structure of the Parts of 29.198

The Parts with API specification themselves are structured 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 progression of internal processes either in the application, or Gateway.
- The Data definitions clauses show a detailed expansion of each of the data types associated with the methods within the classes. It is to be noted that some data types are used in other methods and classes and are therefore defined within the Common Data types part of this specification.

The OSA API is defined using UML and as such is technology independent. OSA can be realised in a number of ways and in addition to the UML defined OSA API, the OSA specification includes:

- A normative annex with the OSA API in IDL that specifies the CORBA distribution technology realisation
- An informative annex with the OSA API in WSDL that specifies the SOAP/HTTP distribution technology realisation
- An informative annex that references the OSA API in Java (known as JAIN™ Service Provider API) that specifies the Java local API technology realisation

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.

6.2 Packaging

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

org.csapi

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 documentation.

packages

lowercase.

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

classes, structures and types. Start with T

TpCapitalizedWithInternalWordsAlsoCapitalized

Exception class:

TpClassNameEndsWithException and P_UPPER_CASE_WITH_UNDERSCORES_AND_START_WITH_P

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 Exception handling and passing results

OSA methods communicate errors in the form of exceptions. 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.

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 type (e.g. IpAnInterfaceRef anInterface), a reference can also be viewed as a logical indirection.

Table:

Original type	IN parameter declaration
IpInterface	parm : IN IpInterfaceRef

6.8 Strings and Collections

For character strings, the *String* data type is used without regard to the maximum length of the string.

For homogeneous collections of instances of a particular data type the following naming scheme is used: <datatype>Set

6.9 Prefixes

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

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= P_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 B (informative): W3C WSDL

B.1 Tools and Languages

The W3C [30] WSDL (Web Services Definition Language) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. WSDL files are generated from the UML model using scripts. The generated WSDL files are verified using WSDL compilers.. The WSDL is based on W3C WSDL 1.1

B.2 Proposed Namespaces for the OSA WSDL

Namespaces are an important part of an XML Schema. They are used to qualify the source of a particular XML element.

There are several XML/SOAP/WSDL related Namespaces which are used within each of the WSDL documents. The Namespace Prefix and the Namespace are noted below.

```
xmlns:wSDL = 'http://schemas.xmlsoap.org/wSDL/'
xmlns:soap='http://schemas.xmlsoap.org/wSDL/soap/'
xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
xmlns:xsd='http://www.w3c.org/2001/XMLSchema'
```

There are also OSA specific namespaces which are used within the OSA WSDL documents. The OSA related namespaces present within each WSDL document depends on the WSDL document and which WSDL documents it imports. The guidelines used to derive these namespaces are:

- The root namespace for the OSA WSDL and XML schemas is `http://www.csapi.org/`
- There is one document generated for each component (Module) within the Analysis UML model. The document will have the name of the UML component with the extension `.wSDL` For each wSDL document generated the following additional namespaces will be included:
 - `xmlns:<component name>='http://www.csapi.org/<component name>/wSDL'`
 - `xmlns:<component name>xsd='http://www.csapi.org/<component name>/schema'`
 - For each OSA wSDL document which is referenced by an import statement within the current wSDL document then the following additional namespaces will be included.
 - `xmlns:<imported component name>='http://www.csapi.org/<imported component name>/wSDL'`
 - `xmlns:<imported component name>xsd='http://www.csapi.org/<imported component name>/schema'`
- Attributes which require a QName value shall use the appropriate Namespace Prefix (as defined in the definitions element of the wSDL file) to qualify the element being referenced.

The namespaces are defined within the 'definitions' element of a wSDL document. For example, the definitions element of the `am.wSDL` document would look like:

```
<definitions
  name='am'
  targetNamespace='http://www.csapi.org/am/wSDL'
  xmlns='http://schemas.xmlsoap.org/wSDL/'
  xmlns:wSDL='http://schemas.xmlsoap.org/wSDL/'
  xmlns:soap='http://schemas.xmlsoap.org/wSDL/soap/'
```



```

xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:am='http://www.csapi.org/am/wsdl'
xmlns:amxsd='http://www.csapi.org/am/schema'
xmlns:osa='http://www.csapi.org/osa/wsdl'
xmlns:osaxsd='http://www.csapi.org/osa/schema'>

```

```

<import namespace='http://www.csapi.org/osa/wsdl'
      location='osa.wsdl' />

```

B.3 Object References

Object references are used to identify an particular remote object instance. Object references are used in two ways:

1. Passed as a parameter within a method to a remote object.
2. Included within a message to identify the object for which the message is intended.

Within the context of SOAP, an object reference can be represented as a URL appended with a String. The String suffix identifies the particular remote object instance in the context of the URL.

An object reference will be represented by the new type ObjectRef. The ObjectRef type is defined within osa.wsdl as:

```

<xsd:simpleType name='Objectref'>
  <xsd:restriction base='xsd:anyURI' />
</xsd:simpleType>

```

When an object reference is passed as a parameter, the parameter type is defined as a reference to an interface. Each interface will have a corresponding reference type associated with it. The interface reference will be defined as:

```

<xsd:simpleType name='InterfaceNameRef'>
  <xsd:restriction base='osaxsd:ObjectRef' />
</xsd:simpleType>

```

where *InterfaceName* is the name of the particular interface.

When an object reference is used to identify the intended recipient of a message, then the object reference is included in the SOAP Header element as an ObjectRefHeader. The ObjectRefHeader is defined in the osa.wsdl document as follows:

```

<message name='ObjectRefHeader'>
  <part name='header' element='osaxsd:ObjectRef' />
</message>

```

Within each method, the ObjectRefHeader is bound to the message within the wsdl soap:header element of the input message of the binding element. For example:

```

<binding name='IpAccountManagerBinding' type='am:IpAccountManager'>
  <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http' />
  <operation name='createNotification'>
    <soap:operation
      soapAction='http://www.csapi.org/am/IpAccountManager#createNotification' />
    <input>
      <soap:body
        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'
        namespace='http://www.csapi.org/am/wsdl'
        use='encoded' />
      <soap:header
        message='osaxsd:ObjectRefHeader' part='header' />
    </input>
  </operation>
</binding>

```

B.4 Mapping UML Data Types to XML Schema

B.4.1 Data Types

B.4.1.1 <<Constant>>

The UML Constant data type contains the following attributes:

- Name
- Constant Value

This type would then map to the following XML Schema construct:

This mapping assumes that all constants are of type TpInt32

```
<xsd:simpletype name="Name">
  <xsd:restriction base="osaxsd:TpInt32">
    <xsd:minInclusive value="Constant Value" />
    <xsd:maxInclusive value="Constant Value" />
  </xsd:restriction>
</xs:simpleType>
```

B.4.1.2 <<NameValuePair>>

The UML NameValuePair data type contains the following attributes:

- Name
- Attributes
 - Name

This type would then map to the following XML Schema construct:

```
<xsd:simpleType base="xsd:string" name="Name">
  <xsd:restriction base="xsd:String">
    <xsd:enumeration value="Attribute-Name" />
    <xsd:enumeration value="Attribute-Name" />
    ...
    <xsd:enumeration value="Attribute-Name" />
  </xsd:restriction>
</xsd:simpleType>
```

B.4.1.3 <<SequenceOfDataElements>>

The UML SequenceOfDataElements data type contains the following attributes:

- Name
- Roles
 - Name
 - Type

This type would then map to the following XML Schema construct:

```
<xsd:complexType name="Name"
  <xsd:sequence>
    <xsd:element
      Name="Role-Name"
      type="Role-Type" />
    <xsd:element
      Name="Role-Name"
      type="Role-Type" />
    ...
    <xsd:element
      Name="Role-Name"
      type="Role-Type" />
  </xsd:sequence>
</xsd:complexType>
```

B.4.1.4 <<TypeDef>>

The UML TypeDef data type contains the following attributes:

- Name
- ImplementationType

If the Implementation type is a technology specific type, then the following mappings have been made:

TpBoolean – xsd:boolean

TpInt32 – xsd:float

TpFloat – xsd:float

TpLongString – xsd:string

TpString – xsd:string

TpOctet – xsd:hexBinary

This type would then map to the following XML Schema construct:

```
<complexType name="Name" base="ImplementationType" />
```

B.4.1.5 <<NumberedSetOfDataElements>>

The UML NumberedSetOfDataElements data type for sequences types contains the following attributes:

- Name
- ImplementationType

This type would then map to the following XML Schema construct:

```
<xsd:complexType name="Name">
  <xsd:sequence>
    <xsd:element
      name="Name"
      type="ImplementationType"
      minOccurs="0"
      maxOccurs="unbounded" />
  </xsd:sequence>
```

```
</xsd:complexType>
```

B.4.1.6 <<TaggedChoiceOfDataElements>>

The UML TaggedChoiceOfDataElements data type contains the following attributes:

- Name
- SwitchType
- Roles
 - Name
 - Type

This type would then map to the following XML Schema construct:

```
<xsd:complexType name="Name">
  <xsd:element name="SwitchName" type="SwitchType" />
  <xsd:choice>
    <xsd:element name="Role-Name" type="Role-Type" />
    <xsd:element name="Role-Name" type="Role-Type" />
    ...
    <xsd:element name="Role-Name" type="Role-Type" />
  </xsd:choice>
</complexType>
```

B.5 Mapping of UML SCF to WSDL

B.5.1 Mapping of Operations to WSDL *message* element

A UML Operation contains the following attributes:

- Name
- Module Name
- Return Type
- Parameter
 - Name
 - Type

This type would then map to the following XML Schema construct:

```
<message name="Name">
  <part
    name="Parameter-Name"
    type="Parameter-Type" />
  ...
  <part
    name="Parameter-Name"
    type="Parameter-Type" />
</message>

<message name="NameResponse">
```

```
<part name="return" type="ReturnType" />
</message>
```

Note: If the `ReturnType` is void, then no 'type' attribute would be included in the Response message.

B.5.2 Mapping of Exception to WSDL *message* element

A UML Exception has the following attributes:

- Name

All exceptions (except for `CommonException`), contain a parameter called `ExtraInformation` which is of type `TpString`.

This type would then map to the following XML Schema Construct:

```
<message name="Name" >
  <part
    name="ExtraInformation"
    type="osaxsd:TpString" />
</message>
```

B.5.3 Mapping of `CommonExceptions` to WSDL *message* element

The UML `CommonExceptions` type has the following attributes:

- Name ("`CommonExceptions`")

The UML `CommonException` contains two parameters; `ExceptionType` which is of type `osaxsd:TpInt32` and `ExtraInformation` which is of type `osaxsd:TpString`.

This type would then map to the following XML Schema Construct:

```
<message name="CommonExceptions" >
  <part
    name="ExceptionType"
    type="osaxsd:TpInt32" />
  <part
    name="ExtraInformation"
    type="osaxsd:TpString" />
</message>
```

B.5.4 Mapping of Interface Class to WSDL *portType* and *binding* elements

A UML Interface Class contains the following attributes:

- Name
- Associated module (i.e. component)
- Operations
 - Name
 - Parameters
 - Name
 - Exceptions

- Name

This type would then map to the following WSDL portType element:

```
<portType name="Name">
  <operation
    name="Operation-Name"
    <input message="Operation-Name"/>
    <output message="Operation-NameResponse"/>
    <fault message="Operation-Exception- Name" />
    ...
    <fault message="Operation-Exception-Name" />
  </operation>
  ...
  <operation
    name="Operation-Name"
    <input message="Operation-Name"/>
    <output message="Operation-NameResponse"/>
    <fault name="Operation-Exception-Name" message="Operation-Exception-Name"
  />
  ...
  <fault message="Operation-Exception-Name" />
</operation>
</portType>
```

This type would also then map into the following WSDL binding element:

```
<binding
  name="Interface-NameBinding"
  type="Interface-Name">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

  <operation name="Operation-Name">
    <soap:operation soapAction="http://www.csapi.org/am/Name#Operation-Name"/>
    <input>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://www.csapi.org/Module-Name.wsdl"
        use="encoded"/>
      <soap:header message="osaxsd:ObjRefHeader" part="header" />
    </input>
    <output>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace=" http://www.csapi.org/Module-Name.wsdl "
        use="encoded"/>
    </output>
    <fault>
      <soap:fault name="Operation-Exception-Name"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://www.csapi.org/Module-Name.wsdl"
        use="encoded"/>
    </fault>

    ... additional fault elements

  </operation>

  ... additional operation elements
```

```
</binding>
```

B.5.5 Mapping of UML SCF to WSDL *service* element

A UML Module contains the following attributes:

- Name
- Interfaces
 - Name

This type would then map to the following WSDL service element:

```
<service name="Name">  
  
  <port binding="Interface-NameBinding" name="Interface-Name">  
    <soap:address location="http://{Service Address}" />  
  </port>  
  
  ... additional port elements  
</service>  
  
</definitions>
```

Annex C (informative): Java API

C.1 Tools and Languages

The Java language is used as a means to programmatically define the interfaces. Java files are either generated manually from class diagrams or by using a UML tool and editing scripts. Either way, the Java files are generated by the JAIN Community [25] in accordance with the Parlay UML to Java API Rulebook [24], which define a set of rules that are used to rapidly generate the Java APIs from the OSA/Parlay UML.

The generated Java files are verified using Java compilers such as javac [28]. The Java API specifications are designed to be compatible with the Java 2 SDK, Standard Edition, version 1.4.0 [28] or later. The Java API Realizations of the OSA/Parlay APIs are known as the JAIN Service Provider APIs (JAIN SPA).

C.2 JAIN SPA Overview

JAIN SPA is a local Java API realization of the OSA/Parlay specifications. The benefits of providing a local API (in addition to a distribution or remote API, such as the OSA/Parlay OMG-IDL or the OSA/Parlay W3C WSDL) is that the API is tailored to a particular programming language (in this case it's Java), which is distribution mechanism independent, meaning that, providing the necessary adapters are put in place, Java applications can be written to this local API that use any form of technology (e.g. CORBA, SOAP, RMI) for the purpose of distributing this API. With remote APIs, although the programmer may be free to write in multiple programming languages, he needs knowledge of, and is committed to, the particular distribution mechanism (e.g. CORBA, SOAP, RMI).

As the OSA/Parlay UML assumes a remote API, many optimizations have been made to the specifications, which, although acceptable to a "specialist" programmer taking distribution into account, would appear alien to the large community of "regular" Java programmers. As such, the JAIN SPA specifications are tailored to the Java language by following Java language naming conventions, design patterns and object oriented practices for a local Java API, while reusing as much Java codebase as possible. JAIN Service Provider APIs are developed by the JAIN Community [25] under the Java Community Process (JCP) [29]. Within the JCP, each JAIN Service Provider API is developed by submitting a Java Specification Request (JSR) [27]. Each JAIN Service Provider API is assigned a JSR number, and an associated webpage, that can be used to identify it.

Each JSR webpage contains a table identifying the relationships between the different versions of the Parlay, ETSI/OSA, 3GPP/OSA and JAIN SPA specifications. In addition, each JAIN SPA specification version indicates to which Parlay, ETSI/OSA and 3GPP/OSA specification versions it corresponds to.

Annex D (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 (Correction to IDL namespace to align with that of ETSI and Parlay equivalent APIs: Change org.open_service_access root namespace to org.csapi) (N5-010267)	4.0.0	4.1.0
Sep 2001	CN_13	NP-010464	002	--	Changing references to JAIN	4.1.0	4.2.0
Dec 2001	CN_14	NP-010594	003	--	Replace Out Parameters with Return Types	4.2.0	4.3.0
Dec 2001	CN_14	NP-010594	004	--	Remove the perception that the OSA API only uses CORBA for its transport mechanism	4.2.0	4.3.0
Mar 2002	--	--	--	--	Editorial update (no CR) following Hong Kong CN5#16	4.3.0	4.3.1
Jun 2002	CN_16	NP-020181	005	--	Addition of support for Java API technology realisation	4.3.1	5.0.0
Jun 2002	CN_16	NP-020182	006	--	Addition of support for WSDL realisation	4.3.1	5.0.0
Jun 2002	CN_16	NP-020184	007	--	Adding the full naming convention for exceptions	4.3.1	5.0.0
Jun 2002	CN_16	NP-020184	008	--	Correction of References in OSA specifications	4.3.1	5.0.0
Jun 2002	CN_16	NP-020184	009	--	Addition of text describing the technology realisations of the Parlay/OSA specification	4.3.1	5.0.0

History

Document history		
V5.0.0	June 2002	Publication