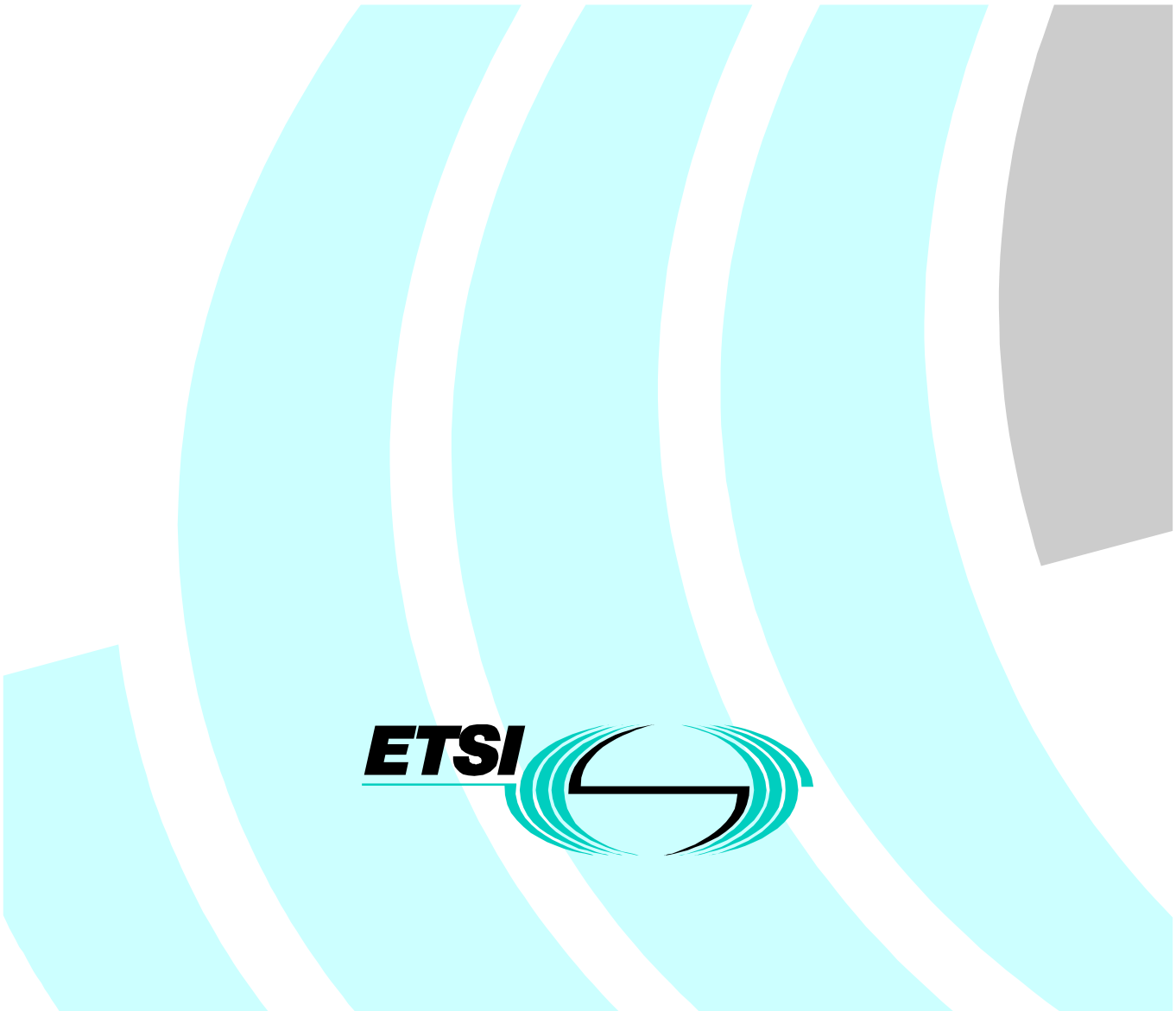
ETSI TS 101 321 V2.1.1 (2000-08)

Technical Specification

Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Open Settlement Protocol (OSP) for Inter-Domain pricing, authorization, and usage exchange



Reference
RTS/TIPHON-03004.2

Keywords
internet, network, interoperability, protocol,
telephony, IP

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

Contents

Intellectual Property Rights 7
Foreword 7
Introduction 7
1 Scope 8
2 References 8
3 Abbreviations 9
4 Open Settlement Protocol Architecture 10
4.1 Communication Protocols 10
4.1.1 Secure Sockets Layer/Transport Layer Security 10
4.1.2 Hypertext Transfer Protocol 10
4.2 Message Format 11
4.2.1 Multipurpose Internet Mail Extensions 11
4.2.2 Extensible Markup Language 11
4.2.3 Secure MIME 12
5 Protocol Profiles 12
5.1 Secure Sockets Layer/Transport Layer Security 12
5.1.1 Protocol Version 12
5.1.2 Client/Server Roles 12
5.1.3 CipherSuites 12
5.2 Hypertext Transfer Protocol 12
5.2.1 Protocol Version 12
5.2.2 Client/Server Roles 12
5.2.3 TCP Port 13
5.2.4 HTTP Methods 13
5.2.5 Uniform Resource Identifier 13
5.2.6 HTTP Headers 13
5.2.7 HTTP Entity Body 13
6 XML Content 13
6.1 Document Structure 13
6.1.1 Multipurpose Internet Mail Extensions Conformance 13
6.1.1.1 Content-Type 13
6.1.1.2 Content-Length 14
6.1.1.3 Transfer Encoding 14
6.1.2 XML Conformance 14
6.1.2.1 XML Version 14
6.1.2.2 Well-Formed Constraint 14
6.1.2.3 Character Encoding 14
6.1.3 XML Framework 15
6.1.3.1 Root Entity 15
6.1.3.2 Random Attribute 15
6.1.3.3 Identifier Attribute 15
6.1.3.4 Critical Attribute 16
6.1.3.5 Extensions 16
6.2 Components 16
6.2.1 PricingIndication 16
6.2.2 PricingConfirmation 17
6.2.3 AuthorizationRequest 17
6.2.4 AuthorizationResponse 18
6.2.5 AuthorizationIndication 18
6.2.6 AuthorizationConfirmation 18
6.2.7 UsageIndication 18

6.2.8 UsageConfirmation 18
6.2.9 ReauthorizationRequest 19
6.2.10 ReauthorizationResponse 19
6.2.11 SubscriberAuthenticationRequest 19
6.2.12 SubscriberAuthenticationResponse 19
6.2.13 CapabilitiesIndication 19
6.2.14 CapabilitiesConfirmation 20
6.3 Elements 20
6.3.1 Amount 20
6.3.2 AuthorityURL 20
6.3.3 CallId 20
6.3.4 Code 21
6.3.5 Currency 21
6.3.6 Description 22
6.3.7 Destination 22
6.3.8 DestinationAlternate 22
6.3.9 DestinationInfo 22
6.3.10 DestinationSignalAddress 23
6.3.11 Increment 23
6.3.12 MaximumDestinations 23
6.3.13 Role 23
6.3.14 Service 24
6.3.15 SourceAlternate 24
6.3.16 SourceInfo 24
6.3.17 SourceSignalAddress 24
6.3.18 Status 25
6.3.19 Timestamp 25
6.3.20 Token 25
6.3.21 TransactionId 25
6.3.22 Unit 26
6.3.23 UsageDetail 26
6.3.24 ValidAfter 26
6.3.25 ValidUntil 26
6.3.26 EndTime 26
6.3.27 StartTime 26
6.3.28 TCCode 27
6.3.29 TerminationCause 28
6.3.30 Certificate 28
6.3.31 CertificateChain 28
6.3.32 OSPCapability 29
6.3.33 OSPService 29
6.3.34 OSPServiceURL 29
6.3.35 OSPSignatureRequired 29
6.3.36 OSPVersion 30
6.3.37 SubscriberAuthenticationInfo 30
6.3.38 DeviceInfo 30
6.3.39 DeviceId 30
6.3.40 Resources 30
6.3.41 DataRate 30
6.3.42 NumberOfChannels 31
6.3.43 Bandwidth 31
6.3.44 AlmostOutOfResources 31

7 Signature Format 31
7.1 Canonical Form 31
7.2 Signature Algorithms 32
7.3 Transfer Encoding 32

8 Protocol Behaviour 32
8.1 Message Sequencing 32
8.2 Exception Handling 33
8.2.1 Transmission Control Protocol 33

8.2.2 Secure Socket Layer/Transport Layer Security 33
8.2.3 Hypertext Transfer Protocol 34
8.2.4 Status Element 34

Annex A (normative): Document Type Definition 35

Annex B (normative): Cryptographic Algorithms 38

B.1 SSL/TLS CipherSuites 38

B.2 S/MIME Signatures 38

B.3 Tokens 38

Annex C (normative): Enhanced Usage Reports 39

C.1 Enhanced Usage Elements 39
C.1.1 Statistics 39
C.1.2 LossSent 39
C.1.3 Packets 39
C.1.4 Fraction 39
C.1.5 LossReceived 39
C.1.6 OneWayDelay 39
C.1.7 Minimum 40
C.1.8 Mean 40
C.1.9 Variance 40
C.1.10 Samples 40
C.1.11 RoundTripDelay 40

Annex D (informative): Token Formats 41

D.1 Cryptographic Encoding 41

D.2 Token Content 42
D.2.1 ASN.1 Format 42
D.2.2 XML Format 43
D.2.3 Binary XML Format 43

D.3 Token Carriage 43

D.4 Sample Token 45

Annex E (informative): Example Messages 46

E.1 Pricing Exchange 46

E.2 Authorization Exchange 48

E.3 Usage Exchange 50

E.4 Subscriber Authentication Exchange 51

E.5 Capabilities Exchange 52

Annex F (informative): Billing Format Conversion 54

Annex G (informative): XML Overview 58

G.1 Document Definition 58

G.2 Element Declaration 58

G.3 Attribute Declaration 59

Annex H (informative): Binary XML Content Format for OSP 60

H.1 Global Extension Tokens 61

H.2 Example Application 63

H.2.1 Standard XML Format (505 bytes) 63
H.2.2 Binary XML Content Format (160 bytes) 63

Annex I (normative): PICS proforma for OSP (TS 101 321) v2.1.1 64

I.1 Guidance for completing the PICS proforma 64
I.1.1 Purposes and structure 64
I.1.2 Abbreviations and conventions 64
I.1.3 Instructions for completing the PICS proforma 66

I.2 Identification of the implementation 66
I.2.1 Date of the statement 66
I.2.2 Implementation Under Test (IUT) identification 67
I.2.3 System Under Test (SUT) identification 67
I.2.4 Product supplier 67
I.2.5 Client (if different from product supplier) 68
I.2.6 PICS contact person 68

I.3 PICS 69
I.3.1 Identification of the protocol 69
I.3.2 Global Statement of Conformance 69
I.3.3 Roles 69
I.3.4 Major capabilities 69
I.3.5 Secure Socket Layer Security Capabilities 70
I.3.6 Hypertext Transfer Protocol Capabilities 70
I.3.7 Multipurpose Internet Mail Extensions Capabilities 72
I.3.8 Extensible Markup Language Capabilities 72
I.3.9 Root Message Capabilities 73
I.3.10 Message support 74
I.3.11 Authorization Token Support 77
I.3.12 XML Extensions 82

Annex J (informative): OSP Applications and Implementations 83

J.1 Call Control Protocols 83
J.1.1 Peer-to-Peer Architecture 83
J.1.1.1 H.323 Gateways 83
J.1.1.1.1 Call Routing and Authorization 84
J.1.1.1.2 Usage Reports 86
J.1.1.2 Session Initiation Protocol Gateways 87
J.1.1.2.1 Call Routing and Authorization 88
J.1.1.2.2 Usage Reports 90
J.1.2 Tightly Controlled Distributed Architecture 91
J.1.2.1 H.323 Gatekeeper Routed Calls 92
J.1.2.1.1 Call Routing and Authorization 92
J.1.2.1.2 Usage Reports 94
J.1.2.2 Session Initiation Protocol Proxy Servers 95
J.1.2.2.1 Call Routing and Authorization 96
J.1.2.2.2 Usage Reports 98
J.1.3 Loosely Controlled Distributed Architecture 99
J.1.3.1 H.323 Direct Routed Calls (with Gatekeepers) 100
J.1.3.1.1 Call Routing and Authorization 100
J.1.3.1.2 Usage Reports 102
J.1.3.2 Session Initiation Protocol Redirect Servers 104
J.1.3.2.1 Call Routing and Authorization 105
J.1.3.2.2 Usage Reports 107

J.2 Prepaid Calling Card and Roaming User Support 110
J.2.1 Call Routing and Authorization 111
J.2.2 Reauthorization 112

Bibliography 116

History 117

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://www.etsi.org/ipr).

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 Project Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON).

Introduction

The contents of the present document are the result of contributions and discussions in Working Group 3.

1 Scope

The present document shall be called the Open Settlement Protocol (OSP). The present document specifies a set of protocols and associated profiles to permit the exchange of inter-domain pricing, authorization, and settlement information between internet telephony operators. The protocols specified fulfil the essential requirements of such services, by providing appropriate functionality between multiple administrative domains in a secure manner. The specification also provides for non-standard extensions that permit co-operating parties to augment or replace the basic functionality.

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.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

[1] American National Standards Institute. Accredited Standards Committee X9 Working Draft: American National Standard X9.42-1993: Public Key Cryptography for the Financial Services Industry: Management of Symmetric Algorithm Keys Using Diffie-Hellman. American Bankers Association, September 21, 1994.

[2] RFC 1945 (1996): Hypertext Transfer Protocol - HTTP/1.0. Berners Lee, T., R. Fielding, and H. Frystyk.

[3] Bray, Tim, Jean Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. World Wide Web Consortium (W3C): 10 February 1998. [http://www.w3.org/TR/REC-xml].

[4] RFC 2068 (1997): Hypertext Transfer Protocol - HTTP/1.1. Fielding R., J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee.

[5] RFC 2045 (1996): Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. Freed, N. and N. Borenstein.

[6] Freier, Alan O., Philip Karlton, and Paul C. Kocher. The SSL Protocol Version 3.0 [http://www.netscape.com/eng/ssl3/ssl-toc.html]. Netscape Communications Corporation: March 1996. As amended by SSL 3.0 Errata of August 26, 1996 [http://www.netscape.com/eng/ssl3/ssl-errata.html].

[7] ISO 4217 (1995): "Codes for the representation of currencies and funds".

[8] ISO 8601 (1988): "Data elements and interchange formats -- Information interchange -- Representation of dates and times".

[9] ITU-T Recommendation H.225.0 (1998): "Call signalling protocols and media stream packetization for packet-based multimedia communication systems".

[10] ITU-T Recommendation H.245 (1998): "Control protocol for multimedia communication".

[11] ITU-T Recommendation X.691 (1995): "Information technology - ASN.1 encoding rules - Specification of Packed Encoding Rules (PER)".

[12] ITU-T Recommendation E.164 (1997): "The international public telecommunication numbering plan".

[13] ITU-T Recommendation H.323 (1998): "Packet-based multimedia communications systems".

[14] ITU-T Recommendation H.235 (1998): "Security and encryption for H-Series (H.323 and other H.245-based) multimedia terminals".

[15] National Institute of Standards and Technology, U.S. Department of Commerce NIST FIPS PUB 46-1 (January 1988): "Data Encryption Standard".

[16] National Institute of Standards and Technology, U.S. Department of Commerce NIST FIPS PUB 186 (18 May 1994): "Digital Signature Standard"

[17] National Institute of Standards and Technology, U.S. Department of Commerce NIST FIPS PUB 180-1 (31 May 1994): "Secure Hash Standard".

[18] RFC 2311 (1998): S/MIME Version 2 Message Specification. S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka.

[19] RFC 2268 (1998): Description of the RC2® Encryption Algorithm. R. Rivest.

[20] RFC 1321 (1992): The MD5 Message-Digest Algorithm. R. Rivest.

[21] RSA Laboratories. PKCS #1: RSA Encryption Standard. Version 1.5, November 1993.

[22] RSA Laboratories. PKCS #7: Cryptographic Message Syntax Standard. Version 1.5, November 1993.

[23] The Unicode Consortium. The Unicode Standard. Version 2.0.

[24] Dierks, Tim and Christopher Allen. The TLS Protocol Version 1.0. Work in progress.

[25] The Open Trading Protocol Consortium. Internet Open Trading Protocol Part 2: Specification. Version 0.9, 12 January 1998.

[26] ISO/IEC 9646-7 (1995): "Information technology -- Open Systems Interconnection -- Conformance testing methodology and framework -- Part 7: Implementation Conformance Statements".

[27] ISO/IEC 10646 (1993): "Information technology -- Universal Multiple-Octet Coded Character Set (UCS)".

[28] RFC 2630 (1999): Cryptographic Message Syntax. Housley, R.

[29] WAP-154, Binary XML Content Format Specification

[30] ISO/IEC 7812-1: "Identification cards -- Identification of issuers -- Part 1: Numbering system".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

CMS	Cryptographic Message Syntax
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
DTD	Document Type Definition
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPSEC	Internet Protocol Security
MD5	Message Digest 5
MIME	Multipurpose Internet Mail Extensions
OSP	Open Settlement Protocol
PIN	Personal Identification Number (e.g. for automated teller machines)
PKCS	Public Key Cryptography Standard

RAS	Registration Admission and Status
RSA	Rivest Shamir Adleman
SHA	Secure Hash Algorithm
S/MIME	Secure Multipurpose Internet Mail Extensions
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URL	Uniform Resource Locator
UTC	Universal Time Co-ordinated
UTF	Universal Text Format
XML	Extensible Markup Language

4 Open Settlement Protocol Architecture

This clause introduces the protocol architecture for the OSP specification. It identifies the major protocols used by communicating parties, and it outlines their relationship to each other. The clause also describes the overall format of messages exchanged by the protocols. The intent of this clause is to outline the framework for the standard's protocols and message formats; later clauses detail specific profiles for these protocols and the specific message content.

4.1 Communication Protocols

As figure 1 shows, systems conforming to the OSP specification use a combination of the Hypertext Transfer Protocol (HTTP), and either the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) to transfer pricing, authorization, and usage information. As the figure indicates, these protocols are layered on top of the Transmission Control Protocol (TCP) for communication across Internet Protocol (IP) networks.

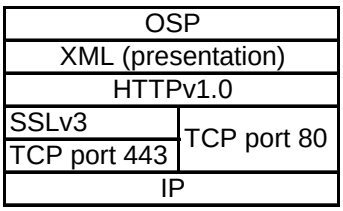


Figure 1: Open Settlement Protocol Architecture for Pricing, Authorization, and Usage Exchange

4.1.1 Secure Sockets Layer/Transport Layer Security

The Secure Sockets Layer and Transport Layer Security protocols add authentication and privacy to TCP connections. SSL is the standard protocol for securing web browsing. As such, it is widely deployed on the Internet and is distinguished by considerable operational experience. SSL also enjoys near universal support from firewalls and proxy servers. TLS is an updated version of SSL currently being developed within the Internet Engineering Task Force (IETF). TLS is heavily based on SSL and, although it is not strictly backwards compatible with SSL, systems supporting both TLS and SSL can automatically recognize either protocol and adapt as required to ensure interoperability.

NOTE: As other industry standard mechanisms for IP-based security (for example, IPSEC) reach maturity, later revisions to the present document may incorporate support for those mechanisms in addition to SSL/TLS. Such revisions to the security mechanisms may also permit the use of an unreliable transport such as UDP.

4.1.2 Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is the standard protocol for web-based communications. HTTP has been adopted for a wide variety of purposes including proxy services, bi-directional content delivery, database access, network management, and metering information. HTTP is by far the most widely used application protocol on the Internet, and is supported by all significant firewalls and proxy servers.

4.2 Message Format

To illustrate the overall format of OSP messages, figure 2 shows an example message. As the figure indicates, the content within the HTTP message is formatted according to the standard for Multipurpose Internet Mail Extensions (MIME). The individual components of the message are a document conforming to the Extensible Markup Language (XML) specification and a Secure MIME (S/MIME) digital signature.

NOTE: The digital signature is optional and, if omitted, the message content consists solely of a XML document.

HTTP Header

```
POST scripts/settlements HTTP/1.0
content-type: multipart/signed;
    protocol="application/pkcs7-signature";
    micalg=sha1;
    boundary=bar
content-length: 844
```

Message Content

```
--bar
Content-Type: text/plain
Content-Length: 524

<?xml version='1.0'?>
<Message messageId="123454321" random="12345678">
    <AuthorizationRequest componentId="9876567890">
        <Timestamp>
            1998-04-24T17:03:00Z
        </Timestamp>
        <CallId>
            1234432198766789
        </CallId>
        <SourceInfo type="e164">
            81458811202
        </SourceInfo>
        <DestinationInfo type="e164">
            4766841360
        </DestinationInfo>
        <Service/>
        <MaximumDestinations>
            5
        </MaximumDestinations>
    </AuthorizationRequest>
</Message>
```

Digital Signature

```
--bar
Content-Type: application/pkcs7-signature
Content-Length: 191

GhyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIG
fHfYT64VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756t
bB9HGTrfvbnjn8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfH
fYT6ghyHhHUujpfyF47GhIGfHfYT64VQbnj756

--bar--
```

Figure 2: Example Message Showing Overall Format

4.2.1 Multipurpose Internet Mail Extensions

All messages exchanged as part of this OSP specification conform to the Multipurpose Internet Mail Extensions (MIME) specification. The MIME specification defines mechanisms to combine individual components of arbitrary format (e.g. text, graphics, audio information, binary data, etc.) into a single message. Originally designed for electronic mail, the MIME specification has been adapted for a variety of communication applications, including web browsing. MIME format is widely supported by existing firewalls and proxy servers.

4.2.2 Extensible Markup Language

The first part of each MIME message is a document conforming to the Extensible Markup Language (XML) standard. As an extension of the widely deployed Hypertext Markup Language (HTML), XML can be readily parsed by firewalls and proxy servers. Unlike HTML, though, XML is readily extensible and can easily support rich, structured data such as pricing and usage information.

4.2.3 Secure MIME

The second part of each MIME message, if present, is a digital signature conforming to the Secure Multipurpose Internet Mail Extensions (S/MIME). S/MIME format includes support for multiple digest and signing algorithms and for variable cryptographic strength (e.g. key lengths). S/MIME format is also self-identifying with respect to these parameters, so that a recipient can derive the necessary information for verifying the signature from the signature data.

NOTE: This does not imply that the recipient is guaranteed to be able to verify the signature, only that the recipient can tell what it needs to perform the verification. (So that, for example, the recipient may identify a signing algorithm that it does not support).

5 Protocol Profiles

This clause specifies the profiles for the protocols required by this OSP specification. It identifies the normative references to those protocols, as well as the specific versions, options, and extensions that the present document requires. The specific protocols described in this clause are the Secure Sockets Layer (SSL) and Transport Layer (TLS) protocols and the Hypertext Transfer Protocol (HTTP). The clause concludes by specifying the overall format of the messages conveyed through these protocols. The following clauses describe the message content in detail.

5.1 Secure Sockets Layer/Transport Layer Security

If secure authentication of the server is desired, or if confidentiality of the information exchanged between client and server is desired, the communication between the devices shall be secured using the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) as described in this clause.

5.1.1 Protocol Version

Conforming systems shall support version 3.0 of the Secure Sockets Layer protocol [6] to secure their communications.

NOTE: As an implementation option, systems may support version 1.0 of the Transport Layer Security protocol [24] or later versions.

5.1.2 Client/Server Roles

When initiating a communication as part of the present document, the initiating system shall act as an SSL/TLS client while the responding system shall act as an SSL/TLS server.

5.1.3 CipherSuites

Annex B documents the cryptographic algorithms required and recommended by the present document, including SSL/TLS ciphersuites.

5.2 Hypertext Transfer Protocol

5.2.1 Protocol Version

Conforming systems shall support version 1.0 of the Hypertext Transfer Protocol [2] as the base transfer protocol for their messages.

NOTE: As an implementation option, systems may support HTTP version 1.1 [4].

5.2.2 Client/Server Roles

When initiating a communication as part of the present document, the initiating system shall act as an HTTP client, while the responding system shall act as an HTTP server.

5.2.3 TCP Port

Clients shall support sending their requests to TCP port 443 if SSL/TLS is being used, and to TCP port 80 otherwise. As an implementation option, communicating parties may agree to communicate via other TCP ports.

5.2.4 HTTP Methods

Requests from clients to a server shall be in the form of HTTP request messages using the POST method. Responses from a server shall consist of valid HTTP response messages.

5.2.5 Uniform Resource Identifier

The uniform resource identifier included in the POST request is not specified in the present document, but rather is subject to prior agreement between the communicating parties.

5.2.6 HTTP Headers

The HTTP header of the POST method shall minimally consist of the request-line. All request-header and general-header fields are optional. If present, they shall conform to the HTTP standard [2]. The status-line for the HTTP responses shall be present in those responses, and it shall conform to the HTTP standard, including status-code and reason-phrase values. All response-header and general-header fields are optional. If present, they shall conform to the HTTP standard.

5.2.7 HTTP Entity Body

Each message (i.e. HTTP entity body) conveyed as part of the present document shall conform to the Multipurpose Internet Mail Extensions standard [5], and shall, if signed, consist of exactly two parts, an Extensible Markup Language document and a Secure Multipurpose Internet Mail Extensions digital signature, as specified in the following two clauses. The highest level structure for each message shall conform to the multipart/signed syntax defined in S/MIME [18]. The message's media type shall be "multipart/signed" with appropriate parameters (e.g. protocol of "application/pkcs7-signature" and micalg of "sha1.") The entity shall indicate the correct content-length value, as defined in the HTTP standard [2].

If not signed, each message shall simply consist of a single, text/plain part.

6 XML Content

This clause specifies the actual message format used by the OSP to exchange pricing, authentication and authorization, and usage information. It outlines the overall XML document structure, lists the individual XML elements, and describes how those elements are combined into exchanges.

6.1 Document Structure

6.1.1 Multipurpose Internet Mail Extensions Conformance

As the first part of a Multipurpose Internet Mail Extensions (MIME) message, each message content shall conform to the MIME standard [5] as indicated below.

6.1.1.1 Content-Type

The message's content-type shall be designated text/plain.

NOTE: It is anticipated that the Internet Engineering Task Force (IETF) will eventually define a MIME content-type for XML documents (e.g. text/xml). When such a definition is available, subsequent revisions of the present document may specify the use of that content-type instead of text/plain.

6.1.1.2 Content-Length

All messages shall indicate the correct content-length value as defined in the MIME standard.

6.1.1.3 Transfer Encoding

As XML documents can be carried within the HTTP protocol in their native format, no transfer encoding (e.g. quoted-printable or base-64) shall be used.

NOTE: Since XML content is carried in its native encoding, that encoding will not conform to the recommendations for the text/plain content-type. In particular, the XML encoding specifies the use of a single line-feed character (#xA) to indicate line ending rather than the return/line-feed pair. Although not the default encoding for text/plain, such an encoding does not violate the MIME standard.

6.1.2 XML Conformance

The actual message content itself shall conform to the XML standard [3]. As part of that conformance, systems shall follow the well-formedness and character encoding requirements as follows.

6.1.2.1 XML Version

Message content shall conform to version 1.0 of the XML standard, and shall indicate that version with the required XML prologue of `<?xml version='1.0'?>`.

6.1.2.2 Well-Formed Constraint

All messages shall be well-formed XML documents, as defined in the [3] standard. Messages may be valid XML documents as well, by referencing the appropriate XML document type definitions (DTDs). Strict validity (as defined by [3]) is not required, however.

NOTE: The terms "well-formed" and "valid" have specific meanings with the XML standard, and are used to indicate specific degrees of conformance to the standard.

6.1.2.3 Character Encoding

Messages may use any character set permitted by the XML standard. As specified in that standard, however, all implementations shall be capable of generating and interpreting UTF-8 and UTF-16 encodings. In the absence of explicit knowledge that the receiving system can support other character encodings, sends shall use UTF-8 or UTF-16 encoding [23].

6.1.3 XML Framework

All messages shall conform to the overall framework illustrated in figure 3. As the figure shows, messages consist of a single root entity, which contains one or more components, each of which consists in turn of one or more elements. These elements may include XML attributes.

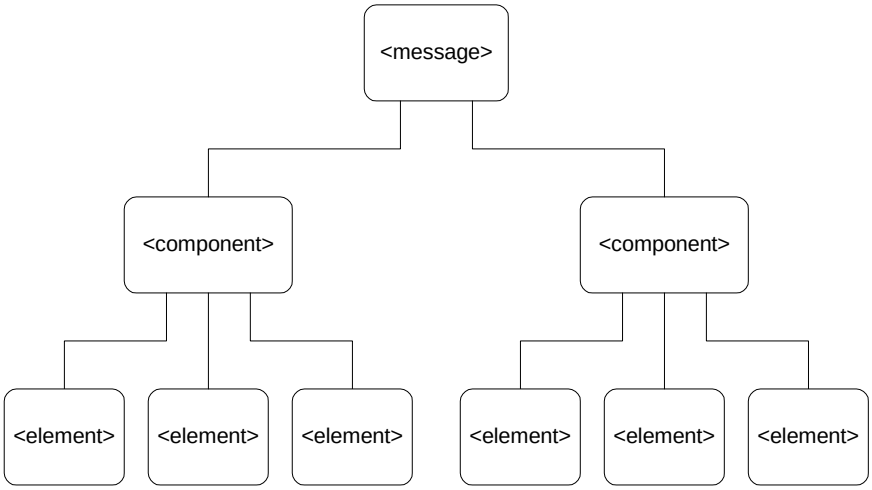


Figure 3: Overall XML Framework

6.1.3.1 Root Entity

```
<!DOCTYPE Message [
<!ELEMENT Message ( ( PricingIndication | PricingConfirmation | AuthorizationRequest |
    AuthorizationResponse | AuthorizationIndication | AuthorizationConfirmation | UsageIndication |
    UsageConfirmation | ReauthorizationRequest | ReauthorizationResponse |
    SubscriberAuthenticationRequest | SubscriberAuthenticationResponse | CapabilitiesIndication |
    CapabilitiesConfirmation )+ )>
<!ATTLIST Message messageId ID    #REQUIRED
                  random    CDATA #REQUIRED>
...
]>
```

The root entity for each message shall be a `Message` element. It shall contain the components documented above, as well as a unique identifier attribute and a random attribute.

6.1.3.2 Random Attribute

Each `Message` element shall include a random attribute. This attribute's value shall be a random number, encoded as a decimal character string. The attribute ensures that each message contains some random content, and it therefore increases the security of the S/MIME digital signature.

NOTE: Systems should use a cryptographically strong random number generator as the source for this attribute's value. Standard library random number functions (such as from the ANSI C standard) are generally ***not*** cryptographically strong, and should not be used.

6.1.3.3 Identifier Attribute

Each message and each component within a message shall include a unique identifier for that element. The system that initiates communication (through either a request or an indication) shall ensure that the identifier is unique. The system that replies (through either a response or a confirmation) shall use the identifier value to associate messages and components in its response or confirmation with the corresponding elements in the request or indication. The identifier attribute consists of an arbitrary character string, and is indicated by the attribute names `messageId` or `componentId`, as appropriate.

6.1.3.4 Critical Attribute

All XML elements shall include a special attribute with the name `critical` that takes values of `"true"` or `"false"`. This attribute indicates whether or not processing of the message can safely proceed if the particular element is not supported by the receiver.

If a system receives a component containing a critical element that it does not support, that system shall not accept or process the component.

If the critical attribute is omitted from an element (and its parents), that element shall be treated as if the critical attribute was present with the value `"true"`.

The value of the critical attribute (whether explicit or implied) for an element shall apply to all sub-elements of the element unless a sub-element explicitly indicates otherwise.

6.1.3.5 Extensions

Organizations may define additional elements or components beyond those documented in the present document. To ensure that no two organizations use the same named element, all private extensions shall have, as a prefix to their name, an officially registered Internet domain name belonging to the defining party. That domain name may be separated from the element name by a colon. If, for example, the organization that owns the Internet domain name acme.com wishes to add an element named `PrivateOption`, the tags delineating that element will be `<acme.com:PrivateOption>` and `</acme.com:PrivateOption>`. As noted above, the `critical` attribute may be used to indicate to a recipient whether or not it can safely ignore a private extension that it does not understand or support.

6.2 Components

Components are the main elements within the each message. The `<Message>` element shall contain at least one and may contain more than one component. The components defined in this revision of the standard include pairs to effect pricing exchange (`PricingIndication` and `PricingConfirmation`), obtain authorization (`AuthorizationRequest` and `AuthorizationResponse`), verify authorization (`AuthorizationIndication` and `AuthorizationConfirmation`), refresh authorization (`ReauthorizationRequest` and `ReauthorizationResponse`), report usage (`UsageIndication` and `UsageConfirmation`), authentication subscribers (`SubscriberAuthenticationRequest` and `SubscriberAuthenticationResponse`) and exchange capabilities (`CapabilitiesIndication` and `CapabilitiesConfirmation`).

6.2.1 PricingIndication

```
<!ELEMENT PricingIndication ( Timestamp, SourceInfo, DestinationInfo, Currency, Amount, Increment,
    Unit, Service, ValidAfter, ValidUntil )>
<!ATTLIST PricingIndication componentId ID #REQUIRED>
```

A `PricingIndication` component identifies the price for a particular service. It is composed of several elements, as indicated above.

For example, the following XML content states that the cost of basic telephone calls from the United States (country code 1) to France (country code 33) is US$ 0,50 per minute, effective immediately and indefinitely.

```
<PricingIndication componentId="1234567890">
    <Timestamp>
       1997-05-02T19:03:00Z
    </Timestamp>
   <SourceInfo type="e164prefix">
        1
    </SourceInfo>
   <DestinationInfo type="e164prefix">
        33
    </DestinationInfo>
   <Currency>
        USD
```

```
        </Currency>
      <Amount>
           0.5
        </Amount>
      <Increment>
           60
        </Increment>
      <Unit>
           s
        </Unit>
      <Service/>
        <ValidAfter/>
      <ValidUntil/>
    </PricingIndication>
```

6.2.2 PricingConfirmation

```
<!ELEMENT PricingConfirmation ( Timestamp, Status )>
<!ATTLIST PricingConfirmation componentId ID #REQUIRED>
```

A `PricingConfirmation` component indicates acceptance or rejection of the corresponding `PricingIndication`. The `componentId` attribute associates the confirmation with the indication. The only elements within the `PricingConfirmation` are the `Timestamp` and `Status` elements.

6.2.3 AuthorizationRequest

```
<!ELEMENT AuthorizationRequest ( Timestamp, CallId+, SourceInfo, SourceAlternate*,
    DestinationInfo, DestinationAlternate*, Service, MaximumDestinations, Token*,
    SubscriberAuthenticationInfo* )>
<!ATTLIST AuthorizationRequest componentId ID #REQUIRED>
```

An `AuthorizationRequest` asks for authorization to use resources. In the context of basic Internet telephony service, it asks for authorization to complete a phone call. The call, for example, may be identified by ITU-T Recommendation E.164 [12] numbers in the `SourceInfo` and `DestinationInfo` elements. The requesting system may leave the choice of peer endpoints up to the authorizing server, or it may specify the peer endpoint itself in a `DestinationAlternate` element.

The client shall provide one or more `CallId` elements in the request. A single `CallId` element implies that the client wishes to use that call identifier for all possible destinations returned in the `AuthorizationResponse`. Multiple `CallId` elements imply that the client wishes each potential destination to have its own call identifier. If the client wishes to specify multiple call identifiers, the number of `CallId` elements shall be equal to the value of the `MaximumDestinations` element.

The `AuthorizationRequest` message may request either authorization information, call routing information, or both. When requesting authorization information only, clients shall specify a `MaximumDestinations` value of zero (0). When requesting call routing information only, clients may omit any `SourceAlternate` elements that identify a subscriber, and they may include a `Token` received as the result of a prior authorization. In addition, clients may include previously received `SubscriberAuthenticationInfo` elements.

NOTE 1: The use of the `AuthorizationRequest` message is not intended to be a replacement for, or supersede any ITU-T Recommendation H.323 [13] Registration Admission and Status (RAS) messages. Although the elements of the `AuthorizationRequest` are similar to information elements in RAS messages, `AuthorizationRequest` is intended for use in the bulk transfer of authorization tokens or other scenarios in which RAS signalling would not be appropriate.

NOTE 2: The present document permits either party in an authorization exchange to determine the call routing information (e.g. identifying the peer endpoint for the call). If the client wishes to explicitly specify call routing, it does so by including one or more `DestinationAlternate` elements (e.g. of type `"transport"` containing the IP address and port number of the peer endpoint) in the `AuthorizationRequest`. If the server is performing call routing, it returns the information in the `AuthorizationResponse`. The `DestinationAlternate` elements are advisory during the request. The server should attempt to use one of the endpoints specified, but it is free to substitute a different set of endpoints if it is unable to settle the call using those specified.

If no routing is required, this shall be explicitly indicated by setting `MaximumDestinations` to "0".

6.2.4 AuthorizationResponse

```
<!ELEMENT AuthorizationResponse ( Timestamp, Status, TransactionId, Token*, Destination* )>
<!ATTLIST AuthorizationResponse componentId ID #REQUIRED>
```

An `AuthorizationResponse` component returns authorization information corresponding to an `AuthorizationRequest`. It includes a `Timestamp`, a `Status` element, a `TransactionId`, optional tokens, and zero or more `Destination` elements. Any tokens present are assumed to apply to all destinations. The response includes a `componentId` attribute to associate it with the appropriate `AuthorizationRequest`.

NOTE: If a client has expressed its own call routing preferences in the `AuthorizationResponse` (e.g. with `DestinationAlternate` elements of type `transport`), then the server should make every attempt to honour those preferences by returning appropriate `Destination` elements. The server may also include alternative destinations in its response.

6.2.5 AuthorizationIndication

```
<!ELEMENT AuthorizationIndication ( Timestamp, Role, CallId, SourceInfo, SourceAlternate*,
    DestinationInfo, DestinationAlternate*, Service, Token* )>
<!ATTLIST AuthorizationIndication componentId ID #REQUIRED>
```

An `AuthorizationIndication` asks for verification of previously issued authorization, typically by asking for verification of an authorization token. Because tokens may be opaque to the terminating endpoint, that endpoint may not be able to determine the originator of a particular token. It is therefore acceptable to pass an entire sequence of tokens from a setup message in this component, and it is acceptable to send simultaneous `AuthorizationIndication` messages to multiple servers. The server shall be capable of recognizing authorization tokens in addition to validating them.

NOTE: Even though the present document defines messages for validating authorization tokens, it does not require their use. In particular, some tokens may be constructed so that they can be completely verified in the peer system (e.g. through the use of digital signatures).

6.2.6 AuthorizationConfirmation

```
<!ELEMENT AuthorizationConfirmation ( Timestamp, Status, ValidAfter, ValidUntil )>
<!ATTLIST AuthorizationConfirmation componentId ID #REQUIRED>
```

An `AuthorizationConfirmation` component indicates whether or not an authorization is valid. It includes a `Timestamp`, a `Status` element and validation time limits. The confirmation also includes a `componentId` attribute to associate it with the appropriate `AuthorizationIndication`.

6.2.7 UsageIndication

```
<!ELEMENT UsageIndication ( Timestamp, Role, TransactionId, CallId, SourceInfo, SourceAlternate*,
    DestinationInfo, DestinationAlternate*, UsageDetail*)>
<!ATTLIST UsageIndication componentId ID #REQUIRED>
```

The `UsageIndication` component reports resource usage. In the context of basic Internet telephony service, this is typically call duration.

6.2.8 UsageConfirmation

```
<!ELEMENT UsageConfirmation ( Timestamp, Status )>
<!ATTLIST UsageConfirmation componentId ID #REQUIRED>
```

A `UsageConfirmation` component indicates acceptance or rejection of the corresponding `UsageIndication`. The `componentId` attribute associates the confirmation with the indication. The only elements within the `UsageConfirmation` are the `Timestamp` and `Status` elements.

6.2.9 ReauthorizationRequest

```
<!ELEMENT ReauthorizationRequest ( Timestamp, Role, CallId, SourceInfo?, SourceAlternate*,
    DestinationInfo?, DestinationAlternate*, TransactionId, UsageDetail*, Token* )>
<!ATTLIST ReauthorizationRequest componentId ID #REQUIRED>
```

A `ReauthorizationRequest` component requests a reauthorization of a previously authorized service. A client may use this message, for example, if a previous authorization has expired.

6.2.10 ReauthorizationResponse

```
<!ELEMENT ReauthorizationResponse ( Timestamp, Status, TransactionId, Token*, Destination* )>
<!ATTLIST ReauthorizationResponse componentId ID #REQUIRED>
```

A `ReauthorizationResponse` component indicates acceptance or rejection of the corresponding `ReauthorizationRequest`. Any tokens present are assumed to apply to all destinations. The `componentId` attribute associates the response with the request.

6.2.11 SubscriberAuthenticationRequest

```
<!ELEMENT SubscriberAuthenticationRequest (Timestamp, SourceInfo, SourceAlternate*,
    DestinationInfo?, Service?)>
<!ATTLIST SubscriberAuthenticationRequest componentId ID #REQUIRED>
```

A SubscriberAuthenticationRequest asks for authentication of a subscriber's credentials, typically in the form of a ISO/IEC 7812-1 [30] identification card number and associated personal identification number.

6.2.12 SubscriberAuthenticationResponse

```
<!ELEMENT SubscriberAuthenticationResponse (Timestamp, Status, SubscriberAuthenticationInfo*)>
<!ATTLIST SubscriberAuthenticationResponse componentId ID #REQUIRED>
```

A SubscriberAuthenticationResponse component returns an indication of whether the subscriber's credentials are considered authentic.

6.2.13 CapabilitiesIndication

```
<!ELEMENT CapabilitiesIndication (DeviceInfo*, OSPVersion, OSPCapability*, Resources?)>
<!ATTLIST CapabilitiesIndication componentId ID               #REQUIRED
                                 critical    ( true | false )  #FIXED    "false">
```

To identify the OSP features it should use with a particular server, a client sends that server a `CapabilitiesIndication` message. That message indicates the highest version of OSP that the client is willing to support, as well as the specific capabilities it wishes to use. The server responds with a `CapabilitiesConfirmation` message. This message indicates the version of OSP that the two parties should use for their communication, and it provides the client details about how to use the services it requires.

If the server responds to a `CapabilitiesIndication` message with anything other than a `CapabilitiesConfirmation message` (e.g. an error status), the client should assume that the server is only capable of support for version 1.4.3 of the specification.

6.2.14 CapabilitiesConfirmation

```
<!ELEMENT CapabilitiesConfirmation (Timestamp, Status, OSPVersion, OSPService*, CertificateChain*,
    DeviceId?)>
<!ATTLIST CapabilitiesConfirmation componentId ID                #REQUIRED
                                   critical (true | false)       #FIXED    "false">
```

The `CapabilitiesConfirmation` message allows a server to specify the OSP capabilities that clients should use to communicate with it. It is sent in response to a `CapabilitiesIndication`. The information consists of a version number, an optional list of services supported, and an optional set of certificate chains. The version element specifies the version of OSP to which future communications shall conform. This version number shall be equal to or less than the version number proposed in the client's `CapabilitiesIndication`. The services list provides the client with information it needs to access various OSP services. The `CertificateChain` elements provide certificate chains for public keys that the server will use to sign any authorization tokens it supplies.

6.3 Elements

This subclause defines the individual elements that make up each message component.

6.3.1 Amount

```
<!ELEMENT Amount (#PCDATA)>
<!ATTLIST Amount critical (true | false) "true">
```

The `Amount` element identifies a numeric value and is often associated with the `Increment` and `Unit` elements, as well as the `Currency` element. Amounts are expressed using the period (.) as a decimal separator and with no punctuation as the thousands separator. The following excerpt, for example, expresses a rate of 50 cents (US) per minute.

```
<Currency>
   USD
</Currency>
<Amount>
   0.5
</Amount>
<Increment>
   60
</Increment>
<Unit>
   s
</Unit>
```

6.3.2 AuthorityURL

```
<!ELEMENT AuthorityURL (#PCDATA)>
<!ATTLIST AuthorityURL critical (true | false) "true">
```

The `AuthorityURL` element identifies a uniform resource locator (URL) by which authorization may be verified or refreshed.

6.3.3 CallId

```
<!ELEMENT CallId (#PCDATA)>
<!ATTLIST CallId encoding (cdata | base64) "cdata"
                 critical (true | false)   "true">
```

The `CallId` element contains a call's ITU-T Recommendation H.323 [13] CallId value, and is thus used to uniquely identify individual calls. To convey its binary value, a call identifier may either be encoded using XML CDATA format and appropriate escape sequences, or it may be encoded with base64 encoding as per the [5] standard. An encoding attribute indicates the method selected, and the default value for that attribute is XML CDATA.

6.3.4 Code

```
<!ELEMENT Code (#PCDATA)>
<!ATTLIST Code critical (true | false) "true">
```

The `Code` element contains the numeric value that uniquely and unambiguously indicates the sender's response to a request or indication. It is usually paired with a `Description` element within a `Status` element. The `Code` content consists of three numeric digits in the form `NNN`. The first (most significant) digit indicates the success or failure of the operation, subsequent digits provide greater detail. Values defined by the present document include the following:

NOTE: Subsequent revisions to the present document may add other code values, but will always preserve the meaning of a most significant 2 as success, and any other most significant digit as failure.

`2xx` = operation successful
`200` = success (no other information)
`201` = information created (no previous values)
`210` = updated information accepted (previous values replaced)
`4xx` = client error
`400` = bad request (generic problem interpreting message)
`401` = unauthorized
402 = authentication unsuccessful
403 = call authorization unsuccessful
404 = route authorization unsuccessful
`410` = character encoding not supported
`411` = parsing unsuccessful
`412` = critical element not supported
`420` = generic security problem (no other information available)
`421` = signature invalid
`422` = cryptographic algorithm not supported
`423` = certificate invalid
`424` = certificate revoked
`425` = encryption required
`5xx` = server error
`500` = internal server error
`501` = not implemented
`503` = service not available
`510` = transient problem in server
`520` = long term problem in server
`530` = time problem
`531` = valid time too soon
`532` = time interval too small
`999` = generic failure (no other information available)

6.3.5 Currency

```
<!ELEMENT Currency (#PCDATA)>
<!ATTLIST Currency critical (true | false) "true">
```

The `Currency` element defines the financial currency in use for the parent element. It is represented according to the notation of ISO 4217 [7]. In addition, the following definitions not included in ISO 4217 [7] may be used:

ECU	European Currency Unit
EUR	Euro
SDR	Special Drawing Rights

6.3.6 Description

```
<!ELEMENT Description (#PCDATA)>
<!ATTLIST Description critical (true | false) "false">
```

The `Description` element provides a textual description for a response, and is typically paired with a `Code` element as part of a `Status` parent element. The `Description` element is for informational purposes only, as the `Code` element defines the behaviour. Suggested values for the `Description` element are the descriptions given above for each `Code` value.

6.3.7 Destination

```
<!ELEMENT Destination ( DestinationInfo?, DestinationAlternate*, DestinationSignalAddress, Token*,
    ValidAfter?, ValidUntil?, UsageDetail*, AuthorityURL*, CallId)>
<!ATTLIST Destination critical (true | false) "true">
```

The `Destination` element is the parent element for call routing information, and it is returned by servers in an `AuthorizationResponse` messages. As the above definition shows, as many as nine different types of subelements may comprise a `Destination` element.

6.3.8 DestinationAlternate

```
<!ELEMENT DestinationAlternate (#PCDATA)>
<!ATTLIST DestinationAlternate type     ( e164 | h323 | url | email | transport |
                                          international | national | network |
                                          subscriber | abbreviated | e164prefix )    #REQUIRED
                               critical ( true | false )                               "true"       >
```

The `DestinationAlternate` element contains secondary identification of the destination. This information provides an alternative to the `DestinationInfo` element. `DestinationAlternate` uses the same notation as `DestinationInfo`.

6.3.9 DestinationInfo

```
<!ELEMENT DestinationInfo (#PCDATA)>
<!ATTLIST DestinationInfo type     ( e164 | h323 | url | email | transport |
                                     international | national | network |
                                     subscriber | abbreviated | e164prefix )         #REQUIRED
                          critical ( true | false )                                    "true"       >
```

The `DestinationInfo` element gives the primary identification of the destination, or called party, for a call. The element includes a `type` attribute, and can take one of several forms depending on the value of that attribute.

The following list indicates the contents of the element, given each possible attribute type.

`e164`	full ITU-T Recommendation E.164 [12] telephone number containing numeric digits only (i.e. no punctuation)
`h323`	ITU-T Recommendation H.323 [13] identifier
`url`	Uniform Resource Locator [13]
`email`	electronic mail address [13]
`transport`	transport address is the form of name:nn where name is the domain name (or IP address enclosed in square brackets) and :nn is an (optional) TCP or UDP port number, (e.g. [172.16.1.1]:112)
`international`	international party number [13]
`national`	national party number [13]
`network`	network specific party number [13]
`subscriber`	subscriber party number [13]
`abbreviated`	abbreviated party number [13]
`e164prefix`	initial (most significant) digits of an ITU-T Recommendation E.164 [12] number with no punctuation

6.3.10 DestinationSignalAddress

```
<!ELEMENT DestinationSignalAddress (#PCDATA)>
<!ATTLIST DestinationSignalAddress critical (true | false) "true">
```

The DestinationSignalAddress element identifies the call signalling address for the destination. It is represented as name:nn, where name is a domain name or an IP address enclosed in square brackets. The :nn is optional and indicates a TCP port number. For example, call signalling to device gateway.operator.com at TCP port number 112 is represented as follows:

```
        <DestinationSignalAddress>
           gateway.operator.com:112
   </DestinationSignalAddress>
```

6.3.11 Increment

```
<!ELEMENT Increment (#PCDATA)>
<!ATTLIST Increment critical (true | false) "true">
```

The `Increment` element indicates the number of units being accounted. It is typically used in combination with the `Amount` and `Unit` elements. The following excerpt, for example, expresses a duration of 5½ minutes.

```
    <Amount>
        5.5
    </Amount>
    <Increment>
        60
    </Increment>
    <Unit>
        s
    </Unit>
```

6.3.12 MaximumDestinations

```
<!ELEMENT MaximumDestinations (#PCDATA)>
<!ATTLIST MaximumDestinations critical (true | false) "true">
```

The `MaximumDestinations` element appears in the `AuthorizationRequest` component to indicate the maximum number of potential destinations the client wishes to receive in the response.

6.3.13 Role

```
<!ELEMENT Role (#PCDATA)>
<!ATTLIST Role critical (true | false) "true">
```

The `Role` element indicates the role of the system generating a message. It shall contain one of the following values:

`source` message generated by source;

`destination` message generated by destination;

`other` message generated by systems other than source or destination.

6.3.14 Service

```
<!ELEMENT Service ( Bandwidth? )>
<!ATTLIST Service critical (true | false) "true">
```

The `Service` element indicates a type of service being priced, authorized, or reported. If present, the `Bandwidth` element indicates the bandwidth required for or consumed by the service.

6.3.15 SourceAlternate

```
<!ELEMENT SourceAlternate (#PCDATA)>
<!ATTLIST SourceAlternate type     ( e164 | h323 | url | email | transport |
                                     international | national | network |
                                     subscriber | abbreviated | e164prefix | iso7812 |
                                     pin | epin | deviceId ) #REQUIRED
                          critical ( true | false )                                    "true"     >
```

The `SourceAlternate` element contains secondary identification of the source of a call. It conforms to the same notation as the `DestinationInfo` element.

The additional types specific to source information are `iso7812`, `pin`, and `epin`. They are defined as follows:

`iso7812` ISO/IEC standard 7812-1 [30] identification card number containing numeric digits only (i.e. no punctuation)

`pin` Personal identification number containing numeric digits only (i.e. no punctuation)

`epin` Encrypted personal identification number, base64 encoded. The encryption procedure is as follows:

1. Pad the PIN, represented as ASCII decimal digits, with nulls (binary zeroes) to a multiple of 16 octets.
2. Concatenate a 128-bit random number to the shared secret (shared by the OSP server and client).
3. Calculate a one-way hash of the resulting concatenation using the Message Digest 5 algorithm.
4. Exclusive-OR the result with the padded PIN.
5. Concatenate the resulting value after the original 128-byte random and base64 encode the resulting 256-byte quantity.

`deviceId` server-assigned identifier.

6.3.16 SourceInfo

```
<!ELEMENT SourceInfo (#PCDATA)>
<!ATTLIST SourceInfo type     ( e164 | h323 | url | email | transport |
                                international | national | network | subscriber |
                                abbreviated | e164prefix | iso7812 | pin | epin | deviceId )
                                #REQUIRED
                     critical ( true | false )                                      "true"       >
```

The `SourceInfo` element contains the primary identification of the source of a call. It uses the same notation as the `DestinationInfo` element, along with the additional types defined for `SourceAlternate`.

6.3.17 SourceSignalAddress

```
<!ELEMENT SourceSignalAddress (#PCDATA)>
<!ATTLIST SourceSignalAddress critical (true | false) "true">
```

The `SourceSignalAddress` element identifies the call signalling address of the source of a call. It uses the same notation as the `DestinationSignalAddress` element.

6.3.18 Status

```
<!ELEMENT Status ( Code, Description? )>
<!ATTLIST Status critical (true | false) "true">
```

The `Status` element reports the results of a response or confirmation. It is composed of a `Code` element and an optional `Description` element.

For example, the following excerpt indicates a successful response or confirmation:

```
<Status>
    <Code>
        200
    </Code>
    <Description>
        success (no other information)
    </Description>
</Status>
```

6.3.19 Timestamp

```
<!ELEMENT Timestamp (#PCDATA)>
<!ATTLIST Timestamp critical (true | false) "true">
```

The Timestamp element indicates the time at which the component was generated. It is represented by a restricted form of ISO 8601 [8] format. In particular, time is always represented in co-ordinated universal time (UTC) using the notation `YYYY-MM-DDThh:mm:ssZ` where:

`YYYY` = four-digit year (for example, 1998);

`MM` = two-digit month (01=January, etc.);

`DD` = two-digit day of month (01 through 31);

`T` = indicates division between date and time;

`hh` = two-digit hour (00 through 23);

`mm` = two-digit minute (00 through 59);

`ss` = two-digit second (00 through 59);

`Z` = indicates co-ordinated universal time.

For example, exactly 3:03 P.M. on May 2, 1997, Eastern Daylight Time in the United States, is represented as:

```
<Timestamp>
    1997-05-02T19:03:00Z
</Timestamp>
```

6.3.20 Token

```
<!ELEMENT Token (#PCDATA)>
<!ATTLIST Token encoding (cdata | base64) "cdata"
                critical (true | false)   "true">
```

The `Token` element conveys a security token. To convey its binary value, a token may either be encoded using XML CDATA format and appropriate escape sequences, or it may be encoded with base64 encoding as per the [5] standard. An encoding attribute indicates the method selected, and the default value for that attribute is XML CDATA.

6.3.21 TransactionId

```
<!ELEMENT TransactionId (#PCDATA)>
<!ATTLIST TransactionId critical (true | false) "true">
```

The `TransactionId` element contains an integer, decimal valued identifier assigned to a specific authorized transaction. It is represented without any punctuation (e.g. no thousands separator).

6.3.22 Unit

```
<!ELEMENT Unit (#PCDATA)>
<!ATTLIST Unit critical (true | false) "true">
```

The `Unit` element indicates the units by which pricing is measured or usage recorded. It shall contain one of the following values:

`s` seconds;

`pkt` packets (datagrams);

`byte` bytes.

6.3.23 UsageDetail

```
<!ELEMENT UsageDetail (Service, Amount, Increment, Unit, StartTime?, EndTime?, TerminationCause?,
    Statistics? )>
<!ATTLIST UsageDetail critical (true | false) "true">
```

The `UsageDetail` element collects information describing the usage of a service. Individual transactions may combine multiple `UsageDetail` elements as part of their usage report. This capability supports both parallel services (e.g. audio and video streams during a video conference) and serial services (e.g. low bit-rate codec switching to a higher quality codec as network conditions deteriorate).

The `UsageDetail` element may also be present in either an `AuthorizationResponse` or a `ReauthorizationResponse`. In that case, it indicates a limit to the authorization. For example, an `AuthorizationResponse` that includes a `UsageDetail` of (amount = 3, increment = 60, unit = s) indicates that the authorization is valid for no more than 3 minutes of service.

6.3.24 ValidAfter

```
<!ELEMENT ValidAfter (#PCDATA)>
<!ATTLIST ValidAfter critical (true | false) "true">
```

The `ValidAfter` element identifies the time and date after which the component's information shall be effective or valid. It is encoded using the same notation as the `Timestamp` element. If this element is empty, component information is assumed to be valid as soon as it is received.

6.3.25 ValidUntil

```
<!ELEMENT ValidUntil (#PCDATA)>
<!ATTLIST ValidUntil critical (true | false) "true">
```

The `ValidUntil` element identifies the time and date after which the component's information is no longer effective or valid. It is encoded using the same notation as the `Timestamp` element. If this element is empty, component information is assumed to be effective indefinitely, or until it is explicitly modified with new information.

6.3.26 EndTime

```
<!ELEMENT EndTime (#PCDATA)>
<!ATTLIST EndTime critical (true | false) #FIXED "false">
```

The `EndTime` element indicates the time at which the service ended. It is encoded using the same notation as the `Timestamp` element.

6.3.27 StartTime

```
<!ELEMENT StartTime (#PCDATA)>
<!ATTLIST StartTime critical (true | false) #FIXED "false">
```

The `StartTime` element indicates the time at which the service started. It is encoded using the same notation as the `Timestamp` element.

6.3.28 TCCode

```
<!ELEMENT TCCode (#PCDATA)>
<!ATTLIST TCCode critical (true | false) #FIXED "false">
```

The `TCCode` element contains the numeric value that uniquely and unambiguously indicates the internet telephony transaction's status code. It is usually paired with a `Description` element within a `TerminationCause` element. The `TCCode` content consists of four numeric digits in the form `NNNN`. The first (most significant) digit indicates the success (1) or failure (0) of the operation, subsequent digits provide greater detail. Values defined by the present document include the following:

`0001` = unallocated (unassigned) number;
`0002` = no route to specified transit network;
`0003` = no route to destination;
`0004` = send special information tone;
`0005` = misdialed trunk prefix;
`0006` = channel unacceptable;
`0007` = call awarded and being delivered in an established channel;
`0008` = preemption;
`0009` = preemption – circuit reserved for re-use;
`1016` = normal call clearing;
`0017` = user busy;
`0018` = no user responding;
`0019` = no answer from user (user alerted);
`0020` = subscriber absent;
`0021` = call rejected;
`0022` = number changed;
`0026` = non-selected user clearing;
`0027` = destination out of order;
`0028` = invalid number format (address incomplete);
`0029` = facility rejected;
`0030` = response to STATUS ENQUIRY;
`0031` = normal, unspecified;
`0032` = no circuit/channel unavailable;
`0038` = network out of order;
`0039` = permanent frame mode connection out of service;
`0040` = permanent frame mode connection operational;
`0041` = temporary failure;
`0042` = switching equipment congestion;
`0043` = access information discarded;
`0044` = requested circuit/channel not available;
`0046` = precedence call blocked;
`0047` = resource unavailable, unspecified;
`0049` = quality of service unavailable;
`0050` = requested facility not subscribed;
`0053` = outgoing calls barred within CUG;
`0055` = incoming calls barred with CUG;
`0057` = bearer capability not authorized;
`0058` = bearer capability not presently available;
`0062` = inconsistency in designated outgoing access information and subscriber class;
`0063` = service or option not available, unspecified;
`0065` = bearer capability not implemented;
`0066` = channel type not implemented;
`0069` = requested facility not implemented;

0070 = only restricted digital information bearer capability is available;
0079 = service or option nit implemented, unspecified;
0081 = invalid call reference value;
0082 = identified channel does not exist;
0083 = a suspended call exists, but this call identity does not;
0084 = call identity in use;
0085 = no call suspended;
0086 = call having the requested call identity has been cleared;
0087 = user not member of CUG;
0088 = incompatible destination;
0090 = non-existent CUG;
0091 = invalid transit network selection;
0095 = invalid message, unspecified;
0096 = mandatory information element is missing;
0097 = message type non-existent or not implemented;
0098 = message not compatible with call state or message type non-existent or not implemented;
0099 = information element/parameter non-existent or not implemented;
0100 = invalid information element contents;
0101 = message not compatible with call state;
0102 = recovery on timer expiry;
0103 = parameter non-existent or not implemented, passed on;
0110 = message with unrecognized parameter, discarded;
0111 = protocol error, unspecified;
0127 = interworking, unspecified.

6.3.29 TerminationCause

```
<!ELEMENT TerminationCause (TCCode, Description?)>
<!ATTLIST TerminationCause critical (true | false) #FIXED "false">
```

The `TerminationCause` element reports the results of an internet telephony transaction. It is composed of the `TCCode` element and an optional `Description` element.

For example, the following excerpt indicates a successful termination cause:

```
<TerminationCause>
    <TCCode>
        1O16
    </TCCode
    <Description>
        normal call clearing
    </Description>
</TerminationCause>
```

6.3.30 Certificate

```
<!ELEMENT Certificate (#PCDATA)>
<!ATTLIST Certificate encoding (cdata | base64) "base64"
       critical (true | false) #FIXED "false">
```

This element contains a public key certificate. To convey its binary value, a certificate may either be encoded using XML CDATA format and appropriate escape sequences, or it may be encoded with base64 encoding as per the [5] standard. An encoding attribute indicates the method selected, and the default value for that attribute is base64.

6.3.31 CertificateChain

```
<!ELEMENT CertificateChain (Certificate*)>
<!ATTLIST CertificateChain critical (true | false) #FIXED "false">
```

This element contains a certificate chain. The first certificate is the subject's certificate. It is followed by any intermediate certificate authorities (in order) and concludes with the certificate for the root authority.

6.3.32 OSPCapability

```
<!ELEMENT OSPCapability (#PCDATA)>
<!ATTLIST OSPCapability critical (true | false) #FIXED "false">
```

This element identifies a particular OSP capability that a server can provide to a client. Capabilities are identified by the name of the component that the client sends to the server to invoke them. A client that wishes to send `AuthorizationRequest` messages to a server, for example, would include the following XML in its `CapabilitiesIndication` message.

<OSPCapability critical="false">
AuthorizationRequest
<OSPCapability>

6.3.33 OSPService

```
<!ELEMENT OSPService (OSPCapability, OSPServiceURL*, OSPSignatureRequired)>
<!ATTLIST OSPService critical (true | false) #FIXED "false">
```

The `OSPService` element provides information a client needs to obtain a particular service from an OSP server. The specific service is identified by the `OSPCapability` element. `OSPServiceURL` elements, if present, indicate the URL(s) the client should use to request the service. They appear in order of decreasing priority (e.g. the primary URL appears first), and, if none are present in the `OSPService` element, then the client may rely on other means to obtain the information. (It may, for example, simply use the same URL it used for the `CapabilitiesIndication`. Message.) The final element, `OSPSignatureRequired,` is a Boolean value that indicates whether or not the server requires that request for the specified service to the identified URLs to be digitally signed. This explicitly allows for a server to indicate one set of URLs for signed messages of a particular type and a different set for unsigned messages of that same type. When such an option is available, the decision as to whether or not to sign messages is an implementation choice for the client. One possible strategy would be to first attempt to use the unsigned option, falling back to the signed option if the server responds with a security error.

6.3.34 OSPServiceURL

```
<!ELEMENT OSPServiceURL (#PCDATA)>
<!ATTLIST OSPServiceURL critical (true | false) #FIXED "false">
```

The `OSPServiceURL` element identifies the uniform resource locator (URL) a client should use for a particular OSP service.

6.3.35 OSPSignatureRequired

```
<!ELEMENT OSPSignatureRequired (#PCDATA)>
<!ATTLIST OSPSignatureRequired critical (true | false) #FIXED "false">
```

The `OSPSignatureRequired` element indicates whether or not a server requires that requests for the indicated service be digitally signed. It shall contain one of the following values:

`true`	signatures are required;
`false`	signatures are not required.

6.3.36 OSPVersion

```
<!ELEMENT OSPVersion (#PCDATA)>
<!ATTLIST OSPVersion critical (true | false) #FIXED "false">
```

The `OSPVersion` element identifies the highest version of TS 101 321 that the sender is willing to support. The sender is assumed to support all publicly released versions of the specification up to and including the indicated value. For example, the following XML fragment indicates that the sender can support OSP versions 1.4.2, 1.4.3, and 2.1.1.

```
<OSPVersion critical = "false">
    2.1.1
</OSPVersion>
```

6.3.37 SubscriberAuthenticationInfo

```
<!ELEMENT SubscriberAuthenticationInfo (#PCDATA)>
<!ATTLIST SubscriberAuthenticationInfo encoding (cdata | base64) "cdata"
                                       critical (true | false)   #FIXED "false">
```

Opaque information that a server creates to confirm the authentication of a subscriber. The server returns this element as part of the `SubscriberAuthenticationResponse` message, and clients should include it in subsequent `AuthorizationRequest` messages for the subscriber.

6.3.38 DeviceInfo

```
<!ELEMENT DeviceInfo (#PCDATA) >
<!ATTLIST DeviceInfo type ( e164 | h323 | url | email |
                            transport | serialnumber | customerId ) #REQUIRED
                     critical    ( true | false )                   #FIXED    "false" >
```

The `DeviceInfo` element allows a device to identify itself to a server when sending a `CapabilitiesIndication` message. In addition to the standard `SourceInfo` types, clients may use the following types to identify themselves:

`serialnumber`	manufacturer's serial number or equivalent for the device
`customerId`	user-assigned value for the device.

6.3.39 DeviceId

```
<!ELEMENT DeviceId (#PCDATA) >
<!ATTLIST DeviceId critical (true | false) "false" >
```

The `DeviceId` element allows a server to provide an identifier to a client in a `CapabilitiesConfirmation` message. The client may then use that identifier in subsequent messages to the server.

6.3.40 Resources

```
<!ELEMENT Resources ( DataRate?, AlmostOutOfResources? ) >
<!ATTLIST Resources critical    ( true | false )  #FIXED "false" >
```

The `Resources` element allows a client to indicate its resources and their current status, as part of a `CapabilitiesIndication` message.

6.3.41 DataRate

```
<!ELEMENT DataRate ( NumberOfChannels?, Bandwidth ) >
<!ATTLIST DataRate critical    ( true | false )  #FIXED "false" >
```

`DataRate` describes the load that the client can handle.

6.3.42 NumberOfChannels

```
<!ELEMENT NumberOfChannels (#PCDATA) >
<!ATTLIST NumberOfChannels critical (true | false) #FIXED "false" >
```

The `NumberOfChannels` element is positive integer, indicating the number of channels available on the client to service calls for a particular protocol.

6.3.43 Bandwidth

```
<!ELEMENT Bandwidth (#PCDATA) >
<!ATTLIST Bandwidth critical (true | false) #FIXED "false" >
```

The `Bandwidth` element is a positive integer. When the `NumberOfChannels` element is present, it indicates the available bandwidth (in bits/sec) of each channel on the client to service calls for a particular protocol. When the `NumberOfChannels` element is not present, `Bandwidth` represents the total bandwidth (in bits/sec).

6.3.44 AlmostOutOfResources

```
<!ELEMENT AlmostOutOfResources (#PCDATA) >
<!ATTLIST AlmostOutOfResources critical (true | false) #FIXED "false" >
```

The `AlmostOutOfResources` element is a flag from the client to the server. It shall contain one of the following values:

`true`	if the client might run out of resources shortly to accept new calls
`false`	if the client has sufficient resources to accept new calls.

7 Signature Format

If present, the digital signature within OSP shall conform to the application/pkcs7-signature format specified in the Secure Multipurpose Internet Mail Extensions (S/MIME) standard [18]. This clause specifies how that signature is created, including the canonicalization procedure, signature algorithm, and transfer encoding.

7.1 Canonical Form

Digital signatures described in the present document are signatures of the entire, transmitted XML document, beginning with (and including) the leftmost "<" of the XML declaration and ending with (and including) the rightmost ">" of the end tag of the root XML entity. Furthermore, conforming implementations should construct their XML documents using the following procedures to arrive at a canonical form. Such a form will enable the reconstruction of an XML document (and the verification of a digital signature) from the abstract information of the document.

NOTE 1: The following procedure borrows heavily from the Internet Open Trading Protocol (IOTP) specification [25].

1) Isolate the element to be signed. For the present document, this consists of the entire XML document, beginning with the leftmost "<" of the XML declaration and ending with (and including) the rightmost ">" of the end tag of the root XML entity;

2) convert all characters in the element to canonical form for the character encoding;

3) apply all external XML entities and all character and entity references in the element so that they are completely resolved;

4) exclude comments and processing instructions;

5) reduce all attributes to their canonical form using the attribute type in the Document Type Definition (DTD). Replace all single and double quotes present in attributes with ' and " respectively so that attributes can be enclosed in double quotes;

6) create attributes, using their default value, which are not present in the original but have default values in the DTD;

7) sort the original and generated attributes in ascending attribute name order according to character encoding of the attribute name;

8) for whitespace inside markup but not inside attribute values, generate it as minimally as possible. Specifically, (1) remove non-essential whitespace, and (2) represent required whitespace by a single space character;

9) generate the content of all start tags using only the element name and the attributes as described above. If the element is an empty element, then generate it using the single empty tag format (i.e. a trailing slash). Generate end tags using only element name with no added whitespace;

10) remove all whitespace in the element content;

11) assemble start tags, end tags, empty tags, CDATA sections, and text sections in the same order as the original document.

NOTE 2: The above procedure results in a canonicalized XML document rather than a canonicalized MIME text part. In particular, the line ending is encoded as a single line-feed character (#xA) rather than the S/MIME convention of a return/line-feed pair.

7.2 Signature Algorithms

Annex B provides a list of cryptographic algorithms required and recommended by the present document, including S/MIME signature algorithms.

7.3 Transfer Encoding

Unlike some electronic mail protocols, HTTP is capable of transferring raw binary data. Consequently, no transfer encoding (e.g. quoted-printable or base-64) shall be used for the signature.

8 Protocol Behaviour

This clause specifies the relationship between the OSP message components. It defines message sequencing, interdependence of messages, and exception handling.

8.1 Message Sequencing

The present document specifies a simple client/server protocol. All protocol exchanges shall be initiated by clients, who send one or more of the eight client components in a single message. The server shall reply with its own single message, which contains one server component for each client component. Figure 4 shows the seven different component pairs.

When multiple components are combined in a single message, each component shall be treated independently of all others in the message. Logically, this treatment shall have the same effect as if each component is carried in its own message.

In addition, each component exchange shall be considered independent of all others; there shall be no dependence between message exchanges. This is true even of Authorization exchanges. Specifically, both AuthorizationIndication/Confirmation and ReauthorizationRequest/Response exchanges may refer to authorization previously obtained through means other than an AuthorizationRequest and AuthorizationResponse.

8.2 Exception Handling

Systems conforming to the present document should use all levels of error handling available in the present document.

8.2.1 Transmission Control Protocol

If TCP indicates that communication cannot be established, or that communication has failed during a transmission, the communicating parties should not assume the delivery of any partial information.

8.2.2 Secure Socket Layer/Transport Layer Security

If SSL or TLS indicates that compatible encryption parameters cannot be established, the communicating parties should not assume the delivery of any partial information. In addition, if SSL/TLS is unable to successfully authenticate the server, the client should not proceed with the transfer of any information.

8.2.3 Hypertext Transfer Protocol

Communicating parties should treat HTTP status codes as defined in RFC 1945 [2]. In particular, unless the status code is in the range 200-299, the client should not assume delivery of any information.

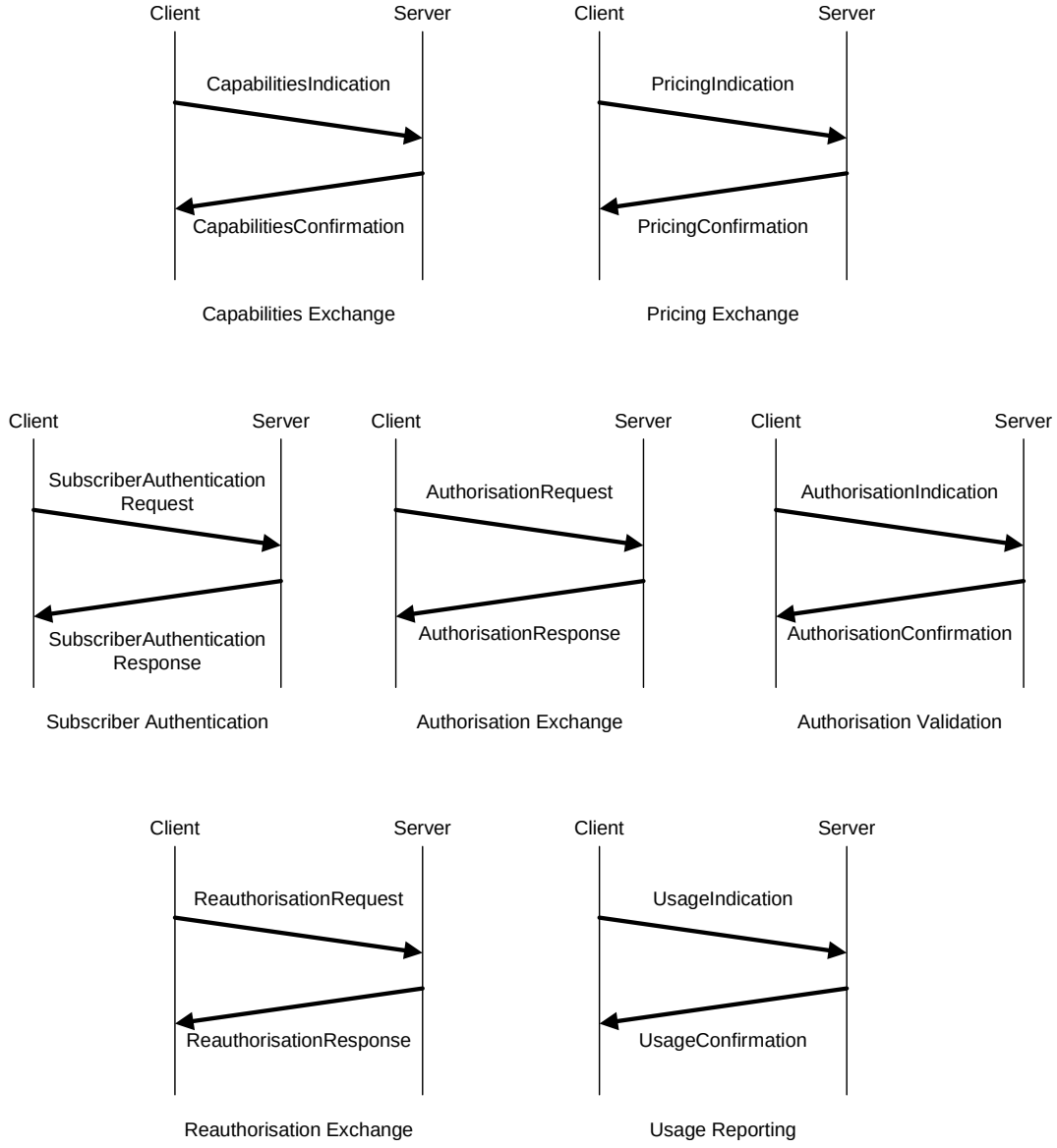


Figure 4: Component Pairs

8.2.4 Status Element

XML `Code` elements within responses and confirmations should be treated appropriately according to the definitions of subclause 6.3.3. The associated `Description` element should be used solely for informational purposes.

Annex A (normative): Document Type Definition

This annex contains the complete XML document type definition for the messages described in the present document. The information is repeated from clause 6 and annex C, but is collected in one place for convenience of reference.

```
<!ELEMENT Message ( ( PricingIndication | PricingConfirmation | AuthorizationRequest |
AuthorizationResponse | AuthorizationIndication | AuthorizationConfirmation | UsageIndication |
UsageConfirmation | ReauthorizationRequest | ReauthorizationResponse |
SubscriberAuthenticationRequest | SubscriberAuthenticationResponse | CapabilitiesIndication |
CapabilitiesConfirmation )+ )>
<!ATTLIST Message messageId ID #REQUIRED
 random CDATA #REQUIRED>
<!ELEMENT PricingIndication ( Timestamp, SourceInfo, DestinationInfo, Currency, Amount, Increment,
Unit, Service, ValidAfter, ValidUntil )>
<!ATTLIST PricingIndication componentId ID #REQUIRED>
<!ELEMENT PricingConfirmation ( Timestamp, Status )>
<!ATTLIST PricingConfirmation componentId ID #REQUIRED>
<!ELEMENT AuthorizationRequest ( Timestamp, CallId+, SourceInfo, SourceAlternate*, DestinationInfo,
DestinationAlternate*, Service, MaximumDestinations, Token*, SubscriberAuthenticationInfo* )>
<!ATTLIST AuthorizationRequest componentId ID #REQUIRED>
<!ELEMENT AuthorizationResponse ( Timestamp, Status, TransactionId, Token*, Destination* )>
<!ATTLIST AuthorizationResponse componentId ID #REQUIRED>
<!ELEMENT AuthorizationIndication ( Timestamp, Role, CallId, SourceInfo, SourceAlternate*,
DestinationInfo, DestinationAlternate*, Service, Token* )>
<!ATTLIST AuthorizationIndication componentId ID #REQUIRED>
<!ELEMENT AuthorizationConfirmation ( Timestamp, Status, ValidAfter, ValidUntil )>
<!ATTLIST AuthorizationConfirmation componentId ID #REQUIRED>
<!ELEMENT UsageIndication ( Timestamp, Role, TransactionId, CallId, SourceInfo, SourceAlternate*,
DestinationInfo, DestinationAlternate*, UsageDetail*)>
<!ATTLIST UsageIndication componentId ID #REQUIRED>
<!ELEMENT UsageConfirmation ( Timestamp, Status )>
<!ATTLIST UsageConfirmation componentId ID #REQUIRED>
<!ELEMENT ReauthorizationRequest ( Timestamp, Role, CallId, SourceInfo?, SourceAlternate*,
DestinationInfo?, DestinationAlternate*, TransactionId, UsageDetail*, Token* )>
<!ATTLIST ReauthorizationRequest componentId ID #REQUIRED>
<!ELEMENT ReauthorizationResponse ( Timestamp, Status, TransactionId, Token*, Destination* )>
<!ATTLIST ReauthorizationResponse componentId ID #REQUIRED>
<!ELEMENT SubscriberAuthenticationRequest (Timestamp, SourceInfo, SourceAlternate*,
DestinationInfo?, Service?)>
<!ATTLIST SubscriberAuthenticationRequest componentId ID #REQUIRED>
<!ELEMENT SubscriberAuthenticationResponse (Timestamp, Status, SubscriberAuthenticationInfo*)>
<!ATTLIST SubscriberAuthenticationResponse componentId ID #REQUIRED>
<!ELEMENT CapabilitiesIndication (DeviceInfo*, OSPVersion, OSPCapability*, Resources*)>
<!ATTLIST CapabilitiesIndication componentId ID #REQUIRED
 critical ( true | false ) #FIXED "false">
<!ELEMENT CapabilitiesConfirmation (Timestamp, Status, OSPVersion, OSPService*, CertificateChain*,
DeviceId?)>
<!ATTLIST CapabilitiesConfirmation componentId ID #REQUIRED
 critical (true | false) #FIXED "false">
<!ELEMENT Amount (#PCDATA)>
<!ATTLIST Amount critical (true | false) "true">
<!ELEMENT AuthorityURL (#PCDATA)>
<!ATTLIST AuthorityURL critical (true | false) "true">
<!ELEMENT CallId (#PCDATA)>
<!ATTLIST CallId encoding (cdata | base64) "cdata"
 critical (true | false) "true">
<!ELEMENT Code (#PCDATA)>
<!ATTLIST Code critical (true | false) "true">
<!ELEMENT Currency (#PCDATA)>
<!ATTLIST Currency critical (true | false) "true">
<!ELEMENT Description (#PCDATA)>
<!ATTLIST Description critical (true | false) "false">
<!ELEMENT Destination ( DestinationInfo?, DestinationAlternate*, DestinationSignalAddress, Token*,
ValidAfter?, ValidUntil?, UsageDetail*, AuthorityURL*, CallId )>
<!ATTLIST Destination critical (true | false) "true">
<!ELEMENT DestinationAlternate (#PCDATA)>
<!ATTLIST DestinationAlternate type ( e164 | h323 | url | email | transport | international |
national | network | subscriber | abbreviated | e164prefix ) #REQUIRED
 critical ( true | false ) "true" >
<!ELEMENT DestinationInfo (#PCDATA)>
<!ATTLIST DestinationInfo type ( e164 | h323 | url | email | transport | international | national |
network | subscriber | abbreviated | e164prefix ) #REQUIRED
```

```
 critical ( true | false ) "true" >
<!ELEMENT DestinationSignalAddress (#PCDATA)>
<!ATTLIST DestinationSignalAddress critical (true | false) "true">
<!ELEMENT Increment (#PCDATA)>
<!ATTLIST Increment critical (true | false) "true">
<!ELEMENT MaximumDestinations (#PCDATA)>
<!ATTLIST MaximumDestinations critical (true | false) "true">
<!ELEMENT Role (#PCDATA)>
<!ATTLIST Role critical (true | false) "true">
<!ELEMENT Service ( Bandwidth? )>
<!ATTLIST Service critical (true | false) "true">
<!ELEMENT SourceAlternate (#PCDATA)>
<!ATTLIST SourceAlternate type ( e164 | h323 | url | email | transport | international | national |
network | subscriber | abbreviated | e164prefix | iso7812 | pin | epin | deviceId ) #REQUIRED
 critical ( true | false ) "true" >
<!ELEMENT SourceInfo (#PCDATA)>
<!ATTLIST SourceInfo type ( e164 | h323 | url | email | transport | international | national |
network | subscriber | abbreviated | e164prefix | iso7812 | pin | epin | deviceId ) #REQUIRED
 critical ( true | false ) "true" >
<!ELEMENT SourceSignalAddress (#PCDATA)>
<!ATTLIST SourceSignalAddress critical (true | false) "true">
<!ELEMENT Status ( Code, Description? )>
<!ATTLIST Status critical (true | false) "true">
<!ELEMENT Timestamp (#PCDATA)>
<!ATTLIST Timestamp critical (true | false) "true">
<!ELEMENT Token (#PCDATA)>
<!ATTLIST Token encoding (cdata | base64) "cdata"
 critical (true | false) "true">
<!ELEMENT TransactionId (#PCDATA)>
<!ATTLIST TransactionId critical (true | false) "true">
<!ELEMENT Unit (#PCDATA)>
<!ATTLIST Unit critical (true | false) "true">
<!ELEMENT UsageDetail (Service, Amount, Increment, Unit, StartTime?, EndTime?, TerminationCause?,
Statistics? )>
<!ATTLIST UsageDetail critical (true | false) "true">
<!ELEMENT ValidAfter (#PCDATA)>
<!ATTLIST ValidAfter critical (true | false) "true">
<!ELEMENT ValidUntil (#PCDATA)>
<!ATTLIST ValidUntil critical (true | false) "true">
<!ELEMENT EndTime (#PCDATA)>
<!ATTLIST EndTime critical (true | false) #FIXED "false">
<!ELEMENT StartTime (#PCDATA)>
<!ATTLIST StartTime critical (true | false) #FIXED "false">
<!ELEMENT TCCode (#PCDATA)>
<!ATTLIST TCCode critical (true | false) #FIXED "false">
<!ELEMENT TerminationCause (TCCode, Description?)>
<!ATTLIST TerminationCause critical (true | false) #FIXED "false">
<!ELEMENT Certificate (#PCDATA)>
<!ATTLIST Certificate encoding (cdata | base64) "base64"
 critical (true | false) #FIXED "false">
<!ELEMENT CertificateChain (Certificate*)>
<!ATTLIST CertificateChain critical (true | false) #FIXED "false">
<!ELEMENT OSPCapability (#PCDATA)>
<!ATTLIST OSPCapability critical (true | false) #FIXED "false">
<!ELEMENT OSPService (OSPCapability, OSPServiceURL*, OSPSignatureRequired)>
<!ATTLIST OSPService critical (true | false) #FIXED "false">
<!ELEMENT OSPServiceURL (#PCDATA)>
<!ATTLIST OSPServiceURL critical (true | false) #FIXED "false">
<!ELEMENT OSPSignatureRequired (#PCDATA)>
<!ATTLIST OSPSignatureRequired critical (true | false) #FIXED "false">
<!ELEMENT OSPVersion (#PCDATA)>
<!ATTLIST OSPVersion critical (true | false) #FIXED "false">
<!ELEMENT SubscriberAuthenticationInfo (#PCDATA)>
<!ATTLIST SubscriberAuthenticationInfo encoding (cdata | base64) "cdata"
 critical (true | false) #FIXED "false">
<!ELEMENT DeviceInfo (#PCDATA) >
<!ATTLIST DeviceInfo type ( e164 | h323 | url | email | transport | serialnumber | customerId )
#REQUIRED
 critical ( true | false ) #FIXED "false" >
<!ELEMENT DeviceId (#PCDATA) >
<!ATTLIST DeviceId critical (true | false) "false" >
<!ELEMENT Resources ( DataRate?, AlmostOutOfResources? ) >
<!ATTLIST Resources critical ( true | false ) #FIXED "false" >
<!ELEMENT DataRate ( NumberOfChannels?, Bandwidth ) >
<!ATTLIST DataRate critical ( true | false ) #FIXED "false" >
<!ELEMENT NumberOfChannels (#PCDATA) >
<!ATTLIST NumberOfChannels critical (true | false) #FIXED "false" >
<!ELEMENT Bandwidth (#PCDATA) >
```

```
<!ATTLIST Bandwidth critical (true | false) #FIXED "false" >
<!ELEMENT AlmostOutOfResources (#PCDATA) >
<!ATTLIST AlmostOutOfResources critical (true | false) #FIXED "false" >
<!ELEMENT Statistics ( LossSent?, LossReceived?, OneWayDelay?, RoundTripDelay? )>
<!ATTLIST Statistics critical (true | false) #FIXED "false">
<!ELEMENT LossSent ( Packets, Fraction )>
<!ATTLIST LossSent critical (true | false) #FIXED "false">
<!ELEMENT Packets (#PCDATA)>
<!ATTLIST Packets critical (true | false) #FIXED "false">
<!ELEMENT Fraction (#PCDATA)>
<!ATTLIST Fraction critical (true | false) #FIXED "false">
<!ELEMENT LossReceived ( Packets, Fraction )>
<!ATTLIST LossReceived critical (true | false) #FIXED "false">
<!ELEMENT OneWayDelay ( Minimum, Mean, Variance, Samples )>
<!ATTLIST OneWayDelay critical (true | false) #FIXED "false">
<!ELEMENT Minimum (#PCDATA)>
<!ATTLIST Minimum critical (true | false) #FIXED "false">
<!ELEMENT Mean (#PCDATA)>
<!ATTLIST Mean critical (true | false) #FIXED "false">
<!ELEMENT Variance (#PCDATA)>
<!ATTLIST Variance critical (true | false) #FIXED "false">
<!ELEMENT Samples (#PCDATA)>
<!ATTLIST Samples critical (true | false) #FIXED "false">
<!ELEMENT RoundTripDelay ( Minimum, Mean, Variance, Samples )>
<!ATTLIST RoundTripDelay critical (true | false) #FIXED "false">
```

Annex B (normative): Cryptographic Algorithms

For convenience and ease of reference, this annex provides the specification of cryptographic algorithms referenced in the standard. Cryptographic algorithms are essential to SSL/TLS communications, S/MIME signatures, and token formats.

B.1 SSL/TLS CipherSuites

Systems conforming to the present document may negotiate any mutually supported SSL/TLS CipherSuite using the standard SSL/TLS handshake mechanisms. To ensure maximum interoperability, all compliant systems should support the SSL_RSA_WITH_3DES_EDE_CBC_SHA CipherSuite. In environments where triple DES data encryption is not desired or is otherwise unavailable, systems should support the SSL_RSA_EXPORT_WITH_DES40_CBC_SHA CipherSuite. If no data encryption is desired or available, systems should use the SSL_RSA _WITH_NULL_SHA CipherSuite.

B.2 S/MIME Signatures

Communicating systems may use any digest and signing algorithms defined by the S/MIME standard [18]. To ensure interoperability, all compliant systems shall support the Secure Hash Algorithm (SHA) [17] for message digests, and the Digital Signature Algorithm (DSA) [16] for signing.

NOTE: To take maximum advantage of deployed cryptographic software, systems should also support the Message Digest 5 (MD5) digest algorithm [20] and the Rivest Shamir Adleman (RSA) signature algorithm [21].

B.3 Tokens

Tokens used for authorization, call progress, and call completion may be signed and encrypted. For message digest, signing, and encryption, implementations may use any algorithm defined by the Public Key Cryptography Standard #7 (PKCS7) [22]. To ensure interoperability, systems should support Secure Hash Algorithm [17], Diffie-Hellman key exchange [1], Digital Signature Algorithm [16], and the Data Encryption Standard [15].

NOTE: To take maximum advantage of deployed cryptographic software, systems should also support the Message Digest 5 [20], Rivest Shamir Aldeman [21], and Rivest Cipher 2 [19] algorithms.

Annex C (normative): Enhanced Usage Reports

The following optional Enhanced Usage elements may be added to the UsageDetail element of UsageIndication messages.

C.1 Enhanced Usage Elements

The definition of Enhanced Usage elements, and the definition of sub-elements, follows:

C.1.1 Statistics

```
<!ELEMENT Statistics ( LossSent?, LossReceived?, OneWayDelay?, RoundTripDelay? )>
<!ATTLIST Statistics critical (true | false) #FIXED "false">
```

The `Statistics` element collects network performance statistics for the call. It may include packet loss statistics (in either direction) and delay statistics (one-way or round trip.) The entire element is non-critical, and may thus be safely ignored by systems that do not support it.

C.1.2 LossSent

```
<!ELEMENT LossSent ( Packets, Fraction )>
<!ATTLIST LossSent critical (true | false) #FIXED "false">
```

The `LossSent` element contains packet loss information for datagrams transmitted by the reporting system that were not received by its peer, as reported in the peer's RTCP sender and receiver reports. It includes the two sub-elements indicated above and described in the following subsections.

C.1.3 Packets

```
<!ELEMENT Packets (#PCDATA)>
<!ATTLIST Packets critical (true | false) #FIXED "false">
```

The `Packets` element contains a count of the total number of packets. The value is formatted as a decimal number without punctuation.

C.1.4 Fraction

```
<!ELEMENT Fraction (#PCDATA)>
<!ATTLIST Fraction critical (true | false) #FIXED "false">
```

The `Fraction` element contains a value for a fraction of packets, expressed as an integer `number` from 0 (no packets) to 255 (all packets), and it is formatted as a decimal number without punctuation.

C.1.5 LossReceived

```
<!ELEMENT LossReceived ( Packets, Fraction )>
<!ATTLIST LossReceived critical (true | false) #FIXED "false">
```

The `LossReceived` element contains packet loss information for datagrams that should `have` been received by the reporting system receive but were not, as reported in the system's RTCP sender and receiver reports. It includes the two sub-elements indicated above and described in the previous subsections.

C.1.6 OneWayDelay

```
<!ELEMENT OneWayDelay ( Minimum, Mean, Variance, Samples )>
<!ATTLIST OneWayDelay critical (true | false) #FIXED "false">
```

The `OneWayDelay` element reports measurements of one way delay ***to*** the reporting system ***from*** its peer, as measured during the communication. It is suggested that the measurement be made by comparing the network time protocol (NTP) timestamp included in `RTCP` messages sent by the peer with the local NTP time. The element consists of the following four sub-elements.

C.1.7 Minimum

```
<!ELEMENT Minimum (#PCDATA)>
<!ATTLIST Minimum critical (true | false) #FIXED "false">
```

The `Minimum` element reports the minimum measured value, expressed in `milliseconds`. It is formatted as an integer decimal number without punctuation.

C.1.8 Mean

```
<!ELEMENT Mean (#PCDATA)>
<!ATTLIST Mean critical (true | false) #FIXED "false">
```

The `Mean` element reports the statistical mean of all measured values. It is expressed in milliseconds, and it is formatted as an integer decimal number without punctuation.

C.1.9 Variance

```
<!ELEMENT Variance (#PCDATA)>
<!ATTLIST Variance critical (true | false) #FIXED "false">
```

The `Variance` element reports the statistical variance of all measured values. It is expressed in squared milliseconds, and it is formatted as an integer decimal number without punctuation.

C.1.10 Samples

```
<!ELEMENT Samples (#PCDATA)>
<!ATTLIST Samples critical (true | false) #FIXED "false">
```

The `Samples` element reports the number of samples measured by the reporting system. It is formatted as a decimal number without punctuation.

C.1.11 RoundTripDelay

```
<!ELEMENT RoundTripDelay ( Minimum, Mean, Variance, Samples )>
<!ATTLIST RoundTripDelay critical (true | false) #FIXED "false">
```

The `RoundTripDelay` element reports measurements of round trip delay between the reporting system and its peer, as measured during the communication. Such measurements may be made, for example, by ITU-T Recommendation H.245 [10] round trip delay exchanges during the call. The element consists of the four sub-elements described above.

Annex D (informative): Token Formats

This annex presents suggested formats for authorization tokens included in the authorization information defined in the present document. Once obtained by an endpoint, such tokens may be passed as part of the call signalling exchange. The annex describes suggested cryptographic encodings as well as suggested contents for these authorization tokens, and it identifies methods for carrying tokens within call signalling protocols. The final section shows a complete, sample token.

D.1 Cryptographic Encoding

Depending on the business roles and network environment, tokens may be digitally signed and/or encrypted. If digitally signed, tokens should conform to Cryptographic Message Syntax (CMS) specification [28] for signed-data content. If encrypted, tokens should conform to the CMS specification for enveloped-data content. If tokens are both signed and encrypted, they should be constructed in two phases. First, the token should be digitally signed, resulting in CMS signed-data content. The resulting signed content should then be encrypted, resulting in CMS enveloped-data content.

If the token syntax and semantics are understood by all relevant parties, then authorization tokens may use the generic id-data OBJECT IDENTIFER (1.2.480.113549.1.7.1) as the data content type. If a token contents are consistent with one of the formats described in this annex, and if the token wishes to explicitly identify that format, it may use one of the following object identifiers as the data content type:

osp-token-contents-asn1 OBJECT IDENTIFER ::= { itu-t(0) identified-organization(4) etsi(0) ts-101-321(1321) token-contents(2) 1 }

osp-token-contents-xml OBJECT IDENTIFER ::= { itu-t(0) identified-organization(4) etsi(0) ts-101-321(1321) token-contents(2) 2 }

osp-token-contents-bxml OBJECT IDENTIFER ::= { itu-t(0) identified-organization(4) etsi(0) ts-101-321(1321) token-contents(2) 3 }

Annex B documents the cryptographic algorithms recommended by the present document, including digest, signing, symmetric encryption, and asymmetric encryption algorithms.

To minimize the size of resulting tokens, appropriate certificates may be conveyed to endpoints as part of the `CapabilitiesConfirmaion` message, and the optional CMS CertificateSet information element may be omitted from individual tokens. In such cases, the most efficient way to identify the signer is to use the SubjectKeyIdentifier, where its contents are a SHA-1 hash of the signer's public key. An example of such a token is:

SignedData ::= SEQUENCE {	
version CMSVersion,	*version 3*
digestAlgorithms DigestAlgorithmIdentifiers,	*e.g. md5 (1.2.840.113594.2.5)*
encapContentInformation EncapsulatedContentInfo,	*e.g. osp-token-contents-bxml*
signerInfos SignerInfos }	*see below*

Where, the SignerInfos consists of the following single element:

SignerInfo ::= SEQUENCE {	
version CMSVersion,	*version 3*
sid SignerIdentifier,	*see below*
digestAlgorithm DigestAlgorithmIdentifier,	*e.g. md5 (1.2.840.113594.2.5)*
signatureAlgorithm SignatureAlgorithmIdentifier,	*e.g. rsa (1.2.840.113594.1.1.1)*
signature SignatureValue }	*128-octet signature*

And the SignerIdentifier element is:

SignerIdentifier ::= CHOICE {	
SubjectKeyIdentifier [0] }	*20-octet SHA-1 hash of signer's public key*

This encoding will result in a token whose size is about 250 octets plus the size of the signed contents.

NOTE: The SubjectKeyIdentifier element may be used even when the signer's public key certificate does not explicitly include that specific extended attribute, as the recipient will be able to compute the appropriate value from the certificate's public key.

D.2 Token Content

This clause suggests two different token formats. The ASN.1 format relies on information elements defined in the ITU-T Recommendation H.323 [13] standards, and is appropriate for native H.323 [13] systems. The XML format is consistent with the body of the present document, and may be used where ASN.1 encoding/decoding is not available or desired. The Binary XML format is a more compact representation of the XML format.

D.2.1 ASN.1 Format

The actual content of the token may conform to the following ASN.1 specifications, as augmented by ASN.1 constructs from the ITU-T Recommendation H.323 [13] standards [13], [9], [14], and [10]. The tokens should be encoded using the Packed Encoding Rules (Aligned) of ITU-T Recommendation X.691 [11].

```
ETSI-TIPHON-TOKENS DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
IMPORTS
        AliasAddress,
        CallIdentifier,
        ReleaseCompleteReason,
        TimeStamp
    FROM H323-MESSAGES;
ServiceDescription :: = CHOICE
{
    basicTelephony      NULL,
    vendorSpecific      NonStandardParameter,
    ...
}
MeasureUnits :: CHOICE
{
    seconds             NULL,
    packets             NULL,
    bytes               NULL,
    vendorSpecific      NonStandardParameter,
    ...
}
UsageData :: = SEQUENCE
{
    amount              INTEGER (0..4294967295)),
    units               MeasureUnits,
    increment           INTEGER (1..4294967295),
    vendorSpecific      NonStandardParameter,
    ...
}
AuthorizationToken :: = SEQUENCE
{
    random              INTEGER (1..4294967295),
    sourceInfo          SEQUENCE OF AliasAddress,
    destinationInfo     SEQUENCE OF AliasAddress,
    callId              CallIdentifier OPTIONAL,
    validAfter          TimeStamp OPTIONAL,
    validUntil          TimeStamp OPTIONAL,
    transactionId       OCTET STRING (SIZE(16)) OPTIONAL,
    serviceAuthorized   SEQUENCE OF SEQUENCE
    {
        service         ServiceDescription,
        limit           UsageData OPTIONAL
    }                   OPTIONAL,
    authorityURL        SEQUENCE OF IA5String (SIZE(1..512)) OPTIONAL,
    vendorSpecific      NonStandardParameter OPTIONAL,
    ...
}
END     -- of ASN.1
```

D.2.2 XML Format

Authorization tokens may also be constructed in XML as a standalone XML document. The root element of such a document shall be `TokenInfo`, and it shall conform to the following construction:

```
<!ELEMENT TokenInfo ( SourceInfo?, SourceAlternate*, DestinationInfo?, DestinationAlternate*,
                      CallId*, ValidAfter?, ValidUntil?, TransactionId?, UsageDetail*,
                      AuthorityURL* )>
<!ATTLIST TokenInfo random #REQUIRED>
```

The individual elements of the document, including any private or future extensions, shall conform to the body of the present document. The following text illustrates an example XML token:

```
    <TokenInfo random="12345678">
        <SourceInfo type="e164">81458811202</SourceInfo>
        <DestinationInfo type="e164">4766841360</DestinationInfo>
        <CallId encoding=base64>Aadf981lasdfA8adooif7c3nnsa89</CallId>
        <ValidAfter>1998-04-24T17:01:01Z</ValidAfter>
        <ValidUntil>1998-04-24T17:11:01Z</ValidUntil>
        <TransactionId>6772374</TransactionId>
        <UsageDetail>
            <Service/>
            <Amount>24</Amount>
            <Increment>3600</Increment>
            <Unit>s</Unit>
        </UsageDetail>
    </TokenInfo>
```

D.2.3 Binary XML Format

Binary XML (see annex H) provides a more compact representation of XML documents, and may be applied to authorization token contents. The contents of the token of D.2.2, when represented as Binary XML according to the rules of annex H, are as follows:

```
02 01 03 00
F6 0B 03 '12345678' 00
    EE 0F 01 03 '81458811202' 00 01
    D6 0F 01 03 '4766841360' 00 01
    CD 08 01 03
        'Aadf981lasdfA8adooif7c3nnsa89' 00 01
    7D 03 '1998-04-24T17:01:01Z' 00 01
    7E 03 '1998-04-24T17:11:01Z' 00 01
    78 03 '6772374' 00 01
    7B
        2C
        46 03 '24' 00 01
        5C 03 '3600' 00 01
        79 03 's' 00 01
    01
01
```

D.3 Token Carriage

When carried as part of a call signalling message of an ASN.1-based protocol, authorization tokens may be identified by one of the following object identifiers. The first value is for tokens with self-identifying or otherwise unspecified formats.

osp-token OBJECT IDENTIFER ::= { itu-t(0) identified-organization(4) etsi(0) ts-101-321(1321) token (1) 0 }

osp-token-asn1-format OBJECT IDENTIFER ::= { itu-t(0) identified-organization(4) etsi(0) ts-101-321(1321) token (1) 1 }

osp-token-xml-format OBJECT IDENTIFER ::= { itu-t(0) identified-organization(4) etsi(0) ts-101-321(1321) token(1) 2 }

osp-token-bxml-format OBJECT IDENTIFER ::= { itu-t(0) identified-organization(4) etsi(0) ts-101-321(1321) token(1) 3 }

When carries as part of a call signalling message of a MIME-based protocol, authorization tokens may be identified by the following MIME type:

```
MIME media type name: x-application
MIME subtype name: x-osp-token
Required parameters: none
Optional parameters:
    x-osp-token-format: a value of "asn.1" indicates the token contents use the ASN.1 format
        defined in annex D, section D.2.1 of the OSP specification; a value of "xml"
        indicates the token contents use the XML format defined in annex D, section D.2.2 of
        the OSP specification, and a value of "bxml" indicates the token contents use the
        Binary XML format defined in annex D, section D.2.3 of the OSP specification. In the
        absence of any value for this parameter, the token contents shall be self identifying
        or otherwise understood by appropriate parties.
    x-osp-token-version: a character string indicating the earliest revision of the OSP
        specification to which the token contents conform. In the absence of any value for
        this parameter, the token contents shall conform to version "2.1.1" of the OSP
        specification.
Encoding considerations: OSP tokens are normally carried as binary data by the call
    signalling protocol. Call signalling protocols which cannot reliably transfer binary data
    may use alternate encodings such as base-64, in which case standard MIME content-encoding
    parameters may indicate the particular encoding.
Security considerations: OSP tokens are intended to provide access control to resources of
    other administrative domains, and, as such, are inherently designed to address security
    concerns. For that reason, OSP tokens are digitally signed and, optionally, encrypted, as
    defined in the OSP specification.
Interoperability considerations: The means and/or algorithms by which a receiving system
    determines whether or not an OSP token is valid are a local matter. However, at a
    minimum, receiving systems should verify the digital signature of the token, and they
    should ensure that any call details included in the token contents (e.g. called number,
    calling number, etc.) are appropriate for the contemplated call.
Published specification: "Telecommunications and Internet Protocol Harmonization Over
    Networks (TIPHON); Open Settlement Protocol (OSP) for Inter-domain pricing,
    authorization, and usage exchange". Technical Specification 101 321. European
    Telecommunications Standards Institute. Version 2.1.1
Applications which use this media type: IP telephony call signalling protocols that use MIME
    types to convey additional information during call setup.
Additional information:
    Magic number(s): none
    File extension(s): none
    Macintosh File Type Code(s): none
Person & email address to contact for further information: Stephen Thomas,
    stephen.thomas@transnexus.com
Intended usage: COMMON
Author/Change controller: European Telecommunications Standards Institute
```

NOTE: A corresponding MIME type of osp-token may be registered with IANA in the future.

D.4 Sample Token

```
30 82 01 A1 SEQUENCE (len = 417)
06 09 ...       OBJECT IDENTIFIER = 1.2.840.113594.1.7.2 (id-signedData)
A0 82 01 92     EXPLICIT TAG [0] (len = 402)
30 82 01 8E         SEQUENCE (len = 398)
02 01 03                INTEGER = 3 (version)
31 0E                   SET (len = 14)
30 0C                       SEQUENCE (len = 12)
06 08 ...                       OBJECT IDENTIFER = 1.2.840.113594.2.5 (md5)
05 00                           NULL
30 82 00 B3             SEQUENCE (len = 179)
06 06 ...                   OBJECT IDENTIFIER = 0.4.0.1321.2.3 (osp-token-contents-bxml)
A0 82 00 AB                 EXPLICIT TAG [0] (len = 171)
04 82 00 A7                     OCTET STRING (len = 167)
02 01 03 00 ...                     token contents
31 82 00 C0             SET (len = 192)
30 82 00 BC                 SEQUENCE (len = 188)
02 01 03                        INTEGER = 3 (version)
A0 16                           EXPLICIT TAG [0]
04 14                               OCTET STRING (len = 20)
...                                     SHA-1 hash of signer's public key information
30 0C                           SEQUENCE (len = 12)
06 08 ...                           OBJECT IDENTIFIER = 1.2.840.113594.2.5 (md5)
05 00                               NULL
30 0D                           SEQUENCE (len = 13)
06 09 ...                           OBJECT IDENTIFIER = 1.2.840.113594.1.1.1 (rsa)
05 00                               NULL
04 82 00 80                     OCTET STRING (len = 128)
...                                 signature
```

Annex E (informative): Example Messages

This annex provides complete example messages for several information exchanges. The messages included are full and complete, but subject to the following modifications for readability:

the content-type field of the HTTP header is shown as several lines. In actual implementations, that field will be constrained to a single line in conformance with HTTP specifications;

extra whitespace is included within the example XML documents. In actual implementations, this whitespace shall be removed before the digital signature is generated (according to the constraints of clause 7), and is not likely to be included in the actual transferred data;

the digital signatures within the example messages do not represent the actual digital signature for the example content. Actual signatures would likely result in unprintable characters.

E.1 Pricing Exchange

The following two messages convey pricing information between two parties. The first message provides three separate price indications. Taken together, the three indications represent prices for basic Internet telephony service (thus the empty `<Service>` element) of ½ DM/minute to Berlin (country code 49, national destination code 30) 1 DM/minute elsewhere in Germany (country code 49), and 2 DM/minute elsewhere in the world. In all three cases no constraints are placed on the calling party (thus the empty `<SourceInfo>` elements).

```
POST scripts/settlements HTTP/1.0
content-type: multipart/signed;
    protocol="application/pkcs7-signature";
    micalg=sha1;
    boundary=bar
content-length: 1968

--bar
Content-Type: text/plain
Content-Length: 1647


<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <PricingIndication componentId="b">
        <Timestamp>
            1998-04-20T19:03:00Z
        </Timestamp>
        <SourceInfo type="e164prefix"/>
        <DestinationInfo type="e164prefix"/>
        <Currency>
            DEM
        </Currency>
        <Amount>
            2
        </Amount>
        <Increment>
            60
        </Increment>
        <Unit>
            s
        </Unit>
        <Service/>
        <ValidAfter/>
        <ValidUntil/>
    </PricingIndication>
    <PricingIndication componentId="c">
        <Timestamp>
            1998-04-20T19:03:02Z
        </Timestamp>
        <SourceInfo type="e164prefix"/>
        <DestinationInfo type="e164prefix">
            49
```

```
        </DestinationInfo>
        <Currency>
            DEM
        </Currency>
        <Amount>
            1
        </Amount>
        <Increment>
            60
        </Increment>
        <Unit>
            s
        </Unit>
        <Service/>
        <ValidAfter/>
        <ValidUntil/>
    </PricingIndication>
    <PricingIndication componentId="d">
        <Timestamp>
            1998-04-20T19:03:05Z
        </Timestamp>
        <SourceInfo type="e164prefix"/>
        <DestinationInfo type="e164prefix">
            4930
        </DestinationInfo>
        <Currency>
            DEM
        </Currency>
        <Amount>
            0.5
        </Amount>
        <Increment>
            60
        </Increment>
        <Unit>
            s
        </Unit>
        <Service/>
        <ValidAfter/>
        <ValidUntil/>
    </PricingIndication>
</Message>

--bar
Content-Type: application/pkcs7-signature
Content-Length: 191

GhyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT64VQpfyF467GhIGfHfYT6jH77n8HH
GghyHhHUujhh756tbB9HGTrfvbnjn8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--bar--
```

The following example shows a confirmation in reply to the above message. In the confirmation, all pricing information is accepted.

```
HTTP/1.0 200 OK
content-type: multipart/signed;
    protocol="application/pkcs7-signature";
    micalg=sha1;
    boundary=bar
content-length: 1401

--bar
Content-Type: text/plain
Content-Length: 1080

<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <PricingConfirmation componentId="b">
        <Timestamp>
            1998-04-20T19:04:00Z
        </Timestamp>
        <Status>
            <Code>
                201
            </Code>
```

```
            <Description>
                new pricing created
            </Description>
        </Status>
    </PricingConfirmation>
    <PricingConfirmation componentId="c">
        <Timestamp>
            1998-04-20T19:04:22Z
        </Timestamp>
        <Status>
            <Code>
                210
            </Code>
            <Description>
                revised pricing accepted
            </Description>
        </Status>
    </PricingConfirmation>
    <PricingConfirmation componentId="d">
        <Timestamp>
            1998-04-20T19:04:45Z
        </Timestamp>
        <Status>
            <Code>
                210
            </Code>
            <Description>
                revised pricing accepted
            </Description>
        </Status>
    </PricingConfirmation>
</Message>

--bar
Content-Type: application/pkcs7-signature
Content-Length: 191

GhyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT64VQpfyF467GhIGfHfYT6j
H77n8HHGghyHhHUujhJh756tbB9HGTrfvbnjn8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT
6ghyHhHUujpfyF47GhIGfHfYT64VQbnj756

--bar—
```

E.2 Authorization Exchange

The authorization exchange shows one party requesting and receiving authorization to access another party's devices. In particular, the requestor wishes to complete a phone call in which the calling party is at +81 45 881 1202 and the called party is at +47 66 84 13 60.

```
POST scripts/settlements HTTP/1.0
content-type: text/plain
Content-Length: 600

<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <AuthorizationRequest componentId="b">
        <Timestamp>
            1998-04-24T17:03:00Z
        </Timestamp>
        <CallId encoding="base64">
                YT64VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756t
        </CallId>
        <SourceInfo type="e164">
            81458811202
        </SourceInfo>
        <DestinationInfo type="e164">
            4766841360
        </DestinationInfo>
        <Service/>
        <MaximumDestinations>
            5
        </MaximumDestinations>
    </AuthorizationRequest>
</Message>
```

The reply to this request returns two separate authorizations, each representing a different peer system that is capable of completing the call. For each destination, the response authorizes up to 24 hours of service for the call.

```
HTTP/1.0 200 OK
content-type: text/plain
Content-Length: 1799

<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <AuthorizationResponse componentId="b">
        <Timestamp>
            1998-04-24T17:03:01Z
        </Timestamp>
        <Status>
            <Code>
                200
            </Code>
            <Description>
                success
            </Description>
        </Status>
        <TransactionId>
            67890987
        </TransactionId>
        <Destination>
            <DestinationSignalAddress>
                [172.16.1.2]:112
            </DestinationSignalAddress>
            <Token encoding="base64">
                YT64VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756t
                HGTrfvbnjn8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfH
                6ghyHhHUujpfyF47GhIGfHfYT64VQbnj
            </Token>
            <ValidAfter>
                1998-04-24T17:01:01Z
            </ValidAfter>
            <ValidUntil>
                1998-04-24T17:11:01Z
            </ValidUntil>
            <UsageDetail>
                <Service/>
                <Amount>
                    24
                </Amount>
                <Increment>
                    3600
                </Increment>
                <Unit>
                    s
                </Unit>
            </UsageDetail>
            <CallId encoding="base64">
                YT64VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756t
            </CallId>
        </Destination>
        <Destination>
            <DestinationSignalAddress>
                [10.0.1.2]:112
            </DestinationSignalAddress>
            <Token encoding="base64">
                F467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tYT64VQpfy
                8HHGTrfvhJhjH776tbB9HG4VQbnj756HGTrfvbnjn7GhIGfH
                ujpfyF47GhIGfHfYT64VQbnj6ghyHhHU
            </Token>
            <ValidAfter>
                1998-04-24T17:01:02Z
            </ValidAfter>
            <ValidUntil>
                1998-04-24T17:11:02Z
            </ValidUntil>
            <UsageDetail>
                <Service/>
                <Amount>
                    24
                </Amount>
                <Increment>
```

```
                    3600
                </Increment>
                <Unit>
                    s
                </Unit>
            </UsageDetail>
            <CallId encoding="base64">
                YT64VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756t
            </CallId>
        </Destination>
    </AuthorizationResponse>
</Message>
```

E.3 Usage Exchange

The final examples demonstrate the exchange of usage information. The first message reports a call duration of 10 minutes.

```
POST scripts/settlements HTTP/1.0
content-type: text/plain
Content-Length: 926

<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <UsageIndication componentId="b">
        <Timestamp>
            1998-04-24T22:03:00Z
        </Timestamp>
        <Role>
            source
        </Role>
        <TransactionId>
            67890987
        </TransactionId>
        <CallId encoding="base64">
                YT64VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756t
        </CallId>
        <SourceInfo type="e164">
            81458811202
        </SourceInfo>
        <SourceAlternate type="subscriber">
            8912342718473772
        </SourceAlternate>
        <DestinationInfo type="e164">
            4766841360
        </DestinationInfo>
        <DestinationAlternate type="transport">
            [10.0.1.2]:112
        </DestinationAlternate>
        <UsageDetail>
            <Service/>
            <Amount>
                10
            </Amount>
            <Increment>
                60
            </Increment>
            <Unit>
                s
            </Unit>
            <StartTime critical="false">
                1999-05-02T19:03:00Z
            </StartTime>
            <EndTime critical="false">
                1999-05-02T19:13:00Z
            </EndTime>
            <TerminationCause critical="false">
                <TCCode>
                    1016
                </TCCode>
                <Description>
                    normal call clearing
                </Description>
            </TerminationCause>
        </UsageDetail>
```

```
    </UsageIndication>
</Message>
```

To complete the exchange, the server responds with a UsageConfirmation.

```
HTTP/1.0 200 OK
content-type: text/plain
Content-Length: 404

<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <UsageConfirmation componentId="b">
        <Timestamp>
            1998-04-24T22:44:00Z
        </Timestamp>
        <Status>
            <Code>
                201
            </Code>
            <Description>
                new usage information created
            </Description>
        </Status>
    </UsageConfirmation>
</Message>
```

E.4 Subscriber Authentication Exchange

The following two messages show the authentication of an individual subscriber. The first message requests authentication for ISO/IEC 7812-1 [30] (calling card or charge card) number 5100123456789012.

```
POST scripts/settlements HTTP/1.0
content-type: text/plain
Content-Length: 336

<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <SubscriberAuthenticationRequest componentId="b">
        <Timestamp>
            1998-04-24T17:03:00Z
        </Timestamp>
        <SourceInfo type="iso7812">
            5100123456789012
        </SourceInfo>
    </SubscriberAuthenticationRequest>
</Message>
```

The following response indicates a successful authentication.

```
HTTP/1.0 200 OK
content-type: text/plain
Content-Length: 427

<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <SubscriberAuthenticationResponse componentId="b">
        <Timestamp>
            1998-04-24T17:03:01Z
        </Timestamp>
        <Status>
            <Code>
                200
            </Code>
            <Description>
                success
            </Description>
        </Status>
    </SubscriberAuthenticationResponse>
</Message>
```

E.5 Capabilities Exchange

This section illustrates the exchange of capabilities between client and server. In the first message, the client identifies its manufacturer's serial number, indicates that it can support OSP version 2.1.1, that it expects to send AuthorizationRequest and UsageIndication messages, and that it has a total of 64 Kbit/s of capacity available.

```
POST scripts/settlements HTTP/1.0
content-type: text/plain
Content-Length: 779

<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <CapabilitiesIndication componentId="b" critical="false">
        <DeviceInfo type="serialnumber" critical="false">
            12345678
        </DeviceInfo>
        <OSPVersion critical="false">
            2.1.1
        </OSPVersion>
        <OSPCapability critical="false">
            AuthorizationRequest
        </OSPCapability>
        <OSPCapability critical="false">
            UsageIndication
        </OSPCapability>
        <Resources critical="false">
            <DataRate critical="false">
                <Bandwidth critical="false">
                    64000
                </Bandwidth>
            </DataRate>
        </Resources>
    </CapabilitiesIndication>
</Message>
```

The server replies with a confirmation that it can support OSP version 2.1.1, and it provides URLs for the client to use for OSP services. Note that the server includes an HTTP "Expires" header, indicating that the client should reconfirm its capabilities prior to 1 December 2000.

```
HTTP/1.0 200 OK
Expires: Thu, 01 Dec 2000 16:00:00 GMT
content-type: text/plain
Content-Length: 1542

<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <CapabilitiesConfirmation componentId="b" critical="false">
        <Timestamp>
            1998-04-24T22:44:00Z
        </Timestamp>
        <Status>
            <Code>
                200
            </Code>
        </Status>
        <OSPVersion critical="false">
            2.1.1
        </OSPVersion>
        <OSPService critical="false">
            <OSPCapability critical="false">
                AuthorizationRequest
            </OSPCapability>
            <OSPServiceURL critical="false">
                http://server1.domain.com/osp/authreq
            </OSPServiceURL>
            <OSPServiceURL critical="false">
                http://server2.domain.com/osp/authreq
            </OSPServiceURL>
            <OSPSignatureRequired critical="false">
                false
            </OSPSignatureRequired>
        </OSPService>
        <OSPService critical="false">
            <OSPCapability critical="false">
```

```
            UsageIndication
        </OSPCapability>
        <OSPServiceURL critical="false">
            http://server1.domain.com/osp/usageind
        </OSPServiceURL>
        <OSPServiceURL critical="false">
            http://server2.domain.com/osp/usageind
        </OSPServiceURL>
        <OSPSignatureRequired critical="false">
            false
        </OSPSignatureRequired>
    </OSPService>
</CapabilitiesConfirmation>
</Message>
```

Annex F (informative): Billing Format Conversion

A wide variety of billing formats are currently in use within telecommunications and data networking systems. One of the advantages of using Extensible Markup Language within the present document is that it allows extremely easy conversion of usage information to and from other formats. As a demonstration of that flexibility, this annex includes example software to covert from <UsageIndication> components to a proprietary call detail record format of one particular, commercially-available system—the VocalTec Internet Telephony Gateway version 3.1. The software is written in Java, and relies on an XML parser available from Microsoft at http://www.microsoft.com/xml. Both the gateway and XML parser are selected solely to demonstrate the ease of the format conversion; this annex does not imply the suitability or fitness of either product for any purpose.

```
import com.ms.xml.parser.*;
import com.ms.xml.om.Document;
import com.ms.xml.om.Element;
import com.ms.xml.util.XMLOutputStream;
import com.ms.xml.util.Name;
import com.ms.xml.om.ElementEnumeration;
import com.ms.xml.om.ElementImpl;

import java.util.Enumeration;
import java.io.*;
import java.io.PrintStream;
import java.net.*;

import java.util.*;

import com.ms.xml.util.Name;
import com.ms.xml.om.ElementFactory;

class CDR
{

    public static void main(String args[])
    {
        // "fix" the version to 2 digits to workaround a bug in the XML parser
        Properties props = new Properties ();
        props = System.getProperties ();
        props.put ("java.version", "1.1");
        System.setProperties (props);

        parseArgs (args);
        if (fileName == null)
            printUsage (System.out);
        else
        {
            URL url = createURL (fileName);

            Document d = null;
            try
            {
                d = new Document ();
                d.setCaseInsensitive (true);
                d.setLoadExternal (true);
                d.load (url);
            }
            catch (ParseException e)
            {
                System.out.println (e);
            }

            if (d != null)
            {
                loadValues (d);

                // type
                cdr = pad ("Voice", 10);

                // date
                cdr += pad (((timestamp == null) ? "  --" :
                    timestamp.substring (5, 10) + "-" + timestamp.substring (2, 4)), 10);
```

```
            // time
            cdr += pad (((timestamp == null) ? "  --" :
                timestamp.substring (11, 19)), 10);

            // duration
            cdr += pad (((unit != null && unit.equals ("s") && amount != null &&
                increment != null) ? "" + Integer.parseInt (amount) *
                Integer.parseInt (increment) : "  --"), 10);

            // username
            cdr += pad ("User Name", 30);

            // dialed number
            cdr += pad (destinationinfo, 30);

            // line
            cdr += pad ("1", 10);

            // remote name
            cdr += pad ("Remote Name", 30);

            // disconnect reason
            cdr += pad ("Disconnect Reason", 40);

            // remote IP
            cdr += pad ("000.000.000.000", 20);

            // remote code
            cdr += pad ("0000", 15);

            // call type
            cdr += pad ((role != null && role.equals ("destination") ? "IPVOD" : "OPVOD"), 15);

            // user ID
            cdr += pad ("000000", 10);

            // fax pages
            cdr += pad ("", 13);

            // remote fax ID
            cdr += pad ("", 25);

            // local fax ref
            cdr += pad ("", 16);

            // remote fax ref
            cdr += pad ("", 16);

            // fax transfer mode
            cdr += pad ("", 18);

            // E.164 number
            cdr += pad (destinationinfo, 24);

            System.out.println (cdr);
        }
    }
    System.exit(0);
}

static void printUsage (PrintStream o)
{
    o.println ("Usage:  java CDR filename");
    o.println ();
}

static void parseArgs (String args[])
{
    if (args.length> 0)
        fileName = args [0];
}

static String pad (String s, int num)
{
    String str;
    int len;
    if (s == null)
    {
```

```
        str = "  --";
        len = 4;
    }
    else
    {
        str = s;
        len = s.length ();
        if (len> num)
        {
            str = "  --";
            len = 4;
        }
    }
    for (int i = len; i <num; i++)
        str += " ";
    return str;
}

static URL createURL (String fileName)
// Microsoft method
{
    URL url = null;
    try
    {
        url = new URL (fileName);
    }
    catch (MalformedURLException ex)
    {
        File f = new File (fileName);
        try
        {
            String path = f.getAbsolutePath();
            String fs = System.getProperty("file.separator");
            if (fs.length() == 1)
            {
                char sep = fs.charAt (0);
                if (sep != '/')
                    path = path.replace (sep, '/');
                    if (path.charAt (0) != '/')
                        path = '/' + path;
            }
            path = "file://" + path;
            url = new URL (path);
        }
        catch (MalformedURLException e)
        {
            System.out.println ("Cannot create url for: " + fileName);
            System.exit(0);
        }
    }
    return url;
}

static void loadValues (Element e)
{
    String attName = "--NULL--";
    String tagName;

    if (e.getTagName () != null)
    {
        attName = e.getText ();
        tagName = e.getTagName ().toString ();

        if (tagName.equals ("TIMESTAMP"))
            timestamp = attName;

        if (tagName.equals ("DESTINATIONINFO"))
            destinationinfo = attName;

        if (tagName.equals ("AMOUNT"))
            amount = attName;

        if (tagName.equals ("INCREMENT"))
            increment = attName;

        if (tagName.equals ("UNIT"))
            unit = attName;
```

```
            if (tagName.equals ("ROLE"))
                role = attName;
        }

        for (ElementEnumeration en = new ElementEnumeration (e); en.hasMoreElements(); )
        {
            Element child = (Element) en.nextElement();
            loadValues (child);
        }
    }

    static String fileName, timestamp, destinationinfo, amount, increment, unit, role, cdr;
}
```

Annex G (informative): XML Overview

As an aid to those readers unfamiliar with Extensible Markup Language, this annex offers a brief overview of the syntax of XML documents. Key components of XML structure are document definitions, element declarations, and attribute declarations.

NOTE: This annex is neither authoritative nor complete. A formal definition of XML may be found in [3].

G.1 Document Definition

XML documents are defined using the following format:

```
<!DOCTYPE DocumentName [DocumentStructureDefinition]>
```

`DocumentName` is the name of the XML document, and `DocumentStructureDefnition` is a series of element, attribute, and entity declarations. Those declarations are described below. An example document definition is:

```
<!DOCTYPE Z [
    <!ELEMENT Y (X, W)>
    <!ATTLIST Y ('a' | 'b' | 'c') 'a' #REQUIRED>
    <!ELEMENT X CDATA>
    <!ELEMENT W CDATA>
]>
```

G.2 Element Declaration

XML elements are declared using the following format:

```
<!ELEMENT ElementName (ElementContents)>
```

`ElementName`, as expected, is the name of the element being declared, while `ElementContents` specifies the permissible contents of the element. Elements may include child elements, in which case the `ElementContents` part defines their order and number within the parent element, and element may include character data.

Child elements within the `ElementContents` part of a declaration may be designated as a sequence of child elements or a choice of child elements. The comma (,) separates individual child elements in a sequence, while the vertical bar (|) separates child elements in a choice. For example, `<!ELEMENT Y (X, W)>` indicates that element `Y` consists of a child element `X` followed by a child element `W`. Similarly, `<!ELEMENT Z (X | W)>` indicates that element `Z` consists of either a child element `X` or a child element `W`.

The element declaration indicates the number of child elements permitted by a character following the child element's name. No character indicates that exactly one child element, a question mark (?) indicates zero or one child element, an asterisk (*) indicates zero or more child elements, and a plus (+) indicates one or more child elements.

For example:

```
<!ELEMENT X (A, B?, C*, D+)>
```

defines element `X` as consisting of:

Table G.1

child element	element declaration	meaning
A	*(none)*	exactly one child element
B	?	zero or one child element
C	*	zero or more child elements
D	+	one or more child elements

G.3 Attribute Declaration

XML attributes are associated with elements; they are declared using the format:

```
<!ATTLIST ElementName AttributeName1 DeclaredValue1 DefaultValue1
                      AttributeName2 DeclaredValue2 DefaultValue2
                                          ...
                      AttributeNameN DeclaredValueN DefaultValueN>
```

`ElementName` identifies the XML element with which the attributes may be used. `AttributeName1`, `AttributeName2`, … are the names of the attributes defined for the element. Each attribute has either a list of permissible values or a data type, which the above construction labels `DeclaredValue`. The final part of each attribute's declaration is `DefaultValue`, which indicates the default value for the attribute, as well as constraints on its presence.

Explicit, permissible values for attributes are separated by the vertical bar (|). So that, for example the `Boolean` attribute for element `X` may be declared as:

```
<!ATTLIST X Boolean (true | false)>
```

NOTE 1: No default value is specified in the above declaration. If a default value is specified, then the DefaultValue section may appear as one of:

`#REQUIRED`: some explicit value for the attribute shall be included each time the attribute is used;

`#IMPLIED`: if no explicit value for the attribute is used, then the application may infer a default value;

`'value'` if no explicit value for the attribute is used, then the attribute should be assumed to have the value '`value`';

`#FIXED 'value':` the attribute shall have, and can only have, the value '`value`'.

EXAMPLE: If the `Boolean` attribute should be assumed to be `true` unless otherwise, explicitly indicated, the following declaration would apply.

```
<!ATTLIST X Boolean (true | false) 'true'>
```

NOTE 2: No quotation marks are used in the `DeclaredValue` part, while the `DefaultValue` part does enclose the value in quotations.

Annex H (informative): Binary XML Content Format for OSP

In some implementations, minimizing the size of protocol messages is a critical requirement. The Binary XML Content Format [29], developed by the Wireless Application Protocol (WAP) Forum) provides a standard method for compressing XML content, and it can be directly applied to OSP messages. This annex provides the information necessary to use Binary XML Content Format in an interoperable manner by defining global extension tokens for OSP. It also includes an example message to illustrate the application of the format.

H.1 Global Extension Tokens

The following tables define the global extension tokens for OSP.

Table H.1: Global extension tokens for OSP tags

CODE PAGE	TOKEN	TAG NAME
0	05	AlmostOutOfResources
0	06	Amount
0	07	AuthorityURL
0	08	AuthorizationConfirmation
0	09	AuthorizationIndication
0	0A	AuthorizationRequest
0	0B	AuthorizationResponse
0	0C	Bandwidth
0	0D	CallId
0	0E	Certificate
0	0F	CertificateChain
0	10	Code
0	11	Currency
0	12	DataRate
0	13	Description
0	14	Destination
0	15	DestinationAlternate
0	16	DestinationInfo
0	17	DestinationSignalAddress
0	18	DeviceId
0	19	DeviceInfo
0	1A	EndTime
0	1B	Fraction
0	1C	Increment
0	1D	LossReceived
0	1E	LossSent
0	1F	MaximumDestinations
0	20	Mean
0	21	Message
0	22	Minimum
0	23	NumberOfChannels
0	24	OneWayDelay
0	25	Packets
0	26	ReauthorizationRequest
0	27	ReauthorizationResponse
0	28	Resources
0	29	Role
0	2A	RoundTripDelay
0	2B	Samples
0	2C	Service
0	2D	SourceAlternate
0	2E	SourceInfo
0	2F	SourceSignalAddress
0	30	StartTime
0	31	Statistics
0	32	Status
0	33	TCCode
0	34	TerminationCause
0	35	Timestamp
0	36	Token

CODE PAGE	TOKEN	TAG NAME
0	37	TokenInfo
0	38	TransactionId
0	39	Unit
0	3A	UsageConfirmation
0	3B	UsageDetail
0	3C	UsageIndication
0	3D	ValidAfter
0	3E	ValidUntil
0	3F	Variance
1	05	CapabilitiesConfirmation
1	06	CapabilitiesIndication
1	07	OSPCapability
1	08	OSPService
1	09	OSPServiceURL
1	0A	OSPSignatureRequired
1	0B	OSPVersion
1	0C	PricingConfirmation
1	0D	PricingIndication
1	0E	SubscriberAuthenticationInfo
1	0F	SubscriberAuthenticationRequest
1	10	SubscriberAuthenticationResponse

Table H.2: Global extension tokens for OSP attributes

TOKEN	ATTRIBUTE
05	componentId
06	critical false
07	critical true
08	encoding base64
09	encoding cdata
0A	messageId
0B	random
0C	type abbreviated
0D	type customerId
0E	type deviceId
0F	type e164
10	type e164prefix
11	type email
12	type epin
13	type h323
14	type international
15	type iso7812
16	type national
17	type network
18	type pin
19	type serialnumber
1A	type subscriber
1B	type transport
1C	type url

H.2 Example Application

This section illustrates the process of converting an OSP message to Binary XML Content Format. It uses the example AuthorizationRequest from annex E.

H.2.1 Standard XML Format (505 bytes)

```
<?xml version='1.0'?>
<Message messageId="a" random="1234">
    <AuthorizationRequest componentId="b">
        <Timestamp>
            1998-04-24T17:03:00Z
        </Timestamp>
        <CallId encoding="base64">
                YT64VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756t
        </CallId>
        <SourceInfo type="e164">
            81458811202
        </SourceInfo>
        <DestinationInfo type="e164">
            4766841360
        </DestinationInfo>
        <Service/>
        <MaximumDestinations>
            5
        </MaximumDestinations>
    </AuthorizationRequest>
</Message>
```

H.2.2 Binary XML Content Format (160 bytes)

```
02 01 03 00
E1 0A 03 'a'  00 0B 03 '1234' 00
    CA 05 03 'b' 00
        75 03 '1998-04-24T17:03:00Z' 00 01
        CD 08 01 03
          'YT64VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756t'00 01
        EE 0F 01 03 '81458811202'00 01
        D6 0F 01 03 '4766841360'00 01
        2C
        5F 03 '5'00 01
    01
01
```

Annex I (normative): PICS proforma for OSP (TS 101 321) v2.1.1

I.1 Guidance for completing the PICS proforma

I.1.1 Purposes and structure

The purpose of this PICS proforma is to provide a mechanism whereby a supplier of an implementation of the requirements defined in the present document may provide information about the implementation in a standardized manner.

The PICS proforma is subdivided into subclauses for the following categories of information:

guidance for completing the PICS proforma;

identification of the implementation;

identification of the protocol;

global statement of conformance;

roles;

major capabilities;

subsidiary capabilities;

operations;

arguments, results and errors;

timers.

I.1.2 Abbreviations and conventions

The PICS proforma contained in this annex is comprised of information in tabular form in accordance with the guidelines presented in ISO/IEC 9646-7 [26].

Item column

The item column contains a number which identifies the item in the table.

Item description column

The item description column describes in free text each respective item (e.g. parameters, timers, etc.). It implicitly means "is <item description> supported by the implementation".

Status column

The following notations, defined in ISO/IEC 9646-7 [26], are used for the status column:

m mandatory - the capability is required to be supported;

o optional - the capability may be supported or not;

n/a	not applicable - in the given context, it is impossible to use the capability;
x	prohibited (excluded) - there is a requirement not to use this capability in the given context;
o.i	qualified optional - for mutually exclusive or selectable options from a set. "i" is an integer which identifies a unique group of related optional items and the logic of their selection which is defined immediately following the table;
ci	conditional - the requirement on the capability ("m", "o", "x" or "n/a") depends on the support of other optional or conditional items. "i" is an integer identifying a unique conditional status expression which is defined immediately following the table;
i	irrelevant (out-of-scope) - capability outside the scope of the reference specification. No answer is requested from the supplier.

Reference column

The reference column makes reference to ETSI TS 101 321, except where explicitly stated otherwise.

Support column

The support column shall be filled in by the supplier of the implementation. The following common notations, defined in ISO/IEC 9646-7 [26], are used for the support column:

Y or y	supported by the implementation;
N or n	not supported by the implementation;
N/A, n/a or -	no answer required (allowed only if the status is n/a, directly or after evaluation of a conditional status).

If this PICS proforma is completed in order to describe a multiple-profile support in a system, it is necessary to be able to answer that a capability is supported for one profile and not supported for another. In that case, the supplier shall enter the unique reference to a conditional expression, preceded by "?" (e.g. ?3). This expression shall be given in the space for comments provided at the bottom of the table. It uses predicates defined in the SCS, each of which refers to a single profile and which takes the value TRUE if and only if that profile is to be used.

EXAMPLE 1: ?3: IF prof1 THEN Y ELSE N.

It is also possible to provide a comment to an answer in the space provided at the bottom of the table.

NOTE: As stated in ISO/IEC 9646-7 [26], support for a received PDU requires the ability to parse all valid parameters of that PDU. Supporting a PDU while having no ability to parse a valid parameter is non-conformant. Support for a parameter on a PDU means that the semantics of that parameter are supported.

Values allowed column

The values allowed column contains the type, the list, the range, or the length of values allowed. The following notations are used:

range of values: <min value> .. <max value>:

EXAMPLE 2: 5 .. 20;

list of values: <value1>, <value2>, ..., <valueN>:

EXAMPLE 3: 2 ,4 ,6 ,8, 9;

EXAMPLE 4: '1101'B, '1011'B, '1111'B;

EXAMPLE 5: '0A'H, '34'H, '2F'H;

list of named values: <name1>(<val1>), <name2>(<val2>), ..., <nameN>(<valN>:

EXAMPLE 6: reject(1), accept(2);

length: size (<min size> .. <max size>):

EXAMPLE 7: size (1 .. 8).

Values supported column

The values supported column shall be filled in by the supplier of the implementation. In this column, the values or the ranges of values supported by the implementation shall be indicated.

References to items

For each possible item answer (answer in the support column) within the PICS proforma a unique reference exists, used, for example, in the conditional expressions. It is defined as the table identifier, followed by a solidus character "/", followed by the item number in the table. If there is more than one support column in a table, the columns are discriminated by letters (a, b, etc.), respectively.

EXAMPLE 8: A.5/4 is the reference to the answer of item 4 in table 5 of annex A.

EXAMPLE 9: A.6/3b is the reference to the second answer (i.e. in the second support column) of item 3 in table 6 of annex A.

Prerequisite line

A prerequisite line takes the form: Prerequisite: <predicate>.

A prerequisite line after a clause or table title indicates that the whole clause or the whole table is not required to be completed if the predicate is FALSE.

I.1.3 Instructions for completing the PICS proforma

The supplier of the implementation shall complete the PICS proforma in each of the spaces provided. In particular, an explicit answer shall be entered, in each of the support or supported column boxes provided, using the notation described in subclause I.1.2.

If necessary, the supplier may provide additional comments in space at the bottom of the tables, or separately on sheets of paper.

More detailed instructions are given at the beginning of the different subclauses of the PICS proforma.

I.2 Identification of the implementation

Identification of the Implementation Under Test (IUT) and the system in which it resides (the System Under Test (SUT)) should be filled in so as to provide as much detail as possible regarding version numbers and configuration options.

The product supplier information and client information should both be filled in if they are different.

A person who can answer queries regarding information supplied in the PICS should be named as the contact person.

I.2.1 Date of the statement

.....

I.2.2 Implementation Under Test (IUT) identification

IUT name:

.....

.....

IUT version:

.....

I.2.3 System Under Test (SUT) identification

SUT name:

.....

.....

Hardware configuration:

.....

.....

.....

Operating system:

.....

I.2.4 Product supplier

Name:

.....

Address:

.....

.....

.....

Telephone number:

.....

Facsimile number:

.....

E-mail address:

.....

Additional information:

.....

.....

.....

I.2.5 Client (if different from product supplier)

Name:

.....

Address:

.....

.....

.....

Telephone number:

.....

Facsimile number:

.....

E-mail address:

.....

Additional information:

.....

I.2.6 PICS contact person

(A person to contact if there are any queries concerning the content of the PICS)

Name:

.....

Telephone number:

.....

Facsimile number:

.....

E-mail address:

.....

Additional information:

.....

.....

.....

I.3 PICS

I.3.1 Identification of the protocol

The PICS proforma applies to the following standard:

TS 101 321: "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Open Settlement Protocol (OSP) for Inter-Domain pricing, authorization, and usage exchange" (the present document).

I.3.2 Global Statement of Conformance

Are all mandatory capabilities implemented? (Yes/No).

NOTE: Answering "No" to this question indicates non-conformance to the protocol specification. Non-supported mandatory capabilities are to be identified in the PICS, with an explanation of why the implementation is non-conforming, on pages attached to the PICS proforma.

I.3.3 Roles

Table I.1: Roles

Item	Role	Reference	Status	Support Y | N | n/a
1	OSP Client		o1	
2	OSP Server		o1	

o1: Support of at least one of these options is required.

I.3.4 Major capabilities

Table I.2: Major capabilities

Item	Capability	Reference	Status	Support Y | N | n/a
1	SSL v3		o	
2	HTTP v1.0		m	
3	MIME RFC 2045		m	
4	XML v1.0		m	
5	HTTP v1.1		o	

I.3.5 Secure Socket Layer Security Capabilities

Table I.3: SSL Capabilities

Prerequisite: Table I.2/1 – If SSLv3 is supported then this table shall be completed				
Item	**Capability**	**Reference**	**Status**	**Support Y | N | n/a**
1	SSL version 3	5.1.1	m	
2	TLS version 1	5.1.1	o	
3	SSL/TLS client authentication	5.13	o	
4	SSL ciphering	5.14	C301	
5	SSL Session Re-use		C301	

C302: IF I.3/1
THEN o
ELSE n/a

Table I.4: SSL Cipher Suite support Capabilities

Item	Capability	Reference	Status	Support Y | N | n/a
1	SSL_RSA_WITH_NULL_SHA	annex B	m	
2	SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	annex B	o	
3	SSL_RSA_WITH_3DES_EDE_CBC_SHA	annex B	o	

I.3.6 Hypertext Transfer Protocol Capabilities

Table I.5: TCP Port capabilities

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	TCP port	5.2.3	C701		443	
2	TCP port	5.2.3	C702		80	

C701: IF I.2/1
THEN m
ELSE n/a

C702: IF NOT I.2/1
THEN m
ELSE n/a

Table I.6: HTTP method capability

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Client messages as POST	5.2.4	m			
2	Server messages as response	5.2.4	m			

Table I.7: HTTP URI Support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Uniform Resource Identifier	5.2.5	m			

Table I.8: HTTPv1.0 POST Header Capabilities

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	request-line	5.2.6	m			
2	authorization	HTTP1.0 10.2	o			
3	from	HTTP1.0 10.8	o			
4	if-modified-since	HTTP1.0 10.9	o			
5	referer	HTTP1.0 10.13	o			
6	user-agent	HTTP1.0 10.15	o			
7	allow	HTTP1.0 10.1	o			
8	content-encoding	HTTP1.0 10.3	o			
9	content-length	HTTP1.0 10.4	o			
10	content-type	HTTP1.0 10.5	o			
11	expires	HTTP1.0 10.7	o			
12	last-modified	HTTP1.0 10.10	o			

Table I.9: HTTPv1.0 Response Header Capabilities

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	status-line	5.2.6	m			
2	location	HTTP1.0 10.11	o			
3	server	HTTP1.0 10.14	o			
4	www-authenticate	HTTP1.0 10.16	o			
5	allow	HTTP1.0 10.1	o			
6	content-encoding	HTTP1.0 10.3	o			
7	content-length	HTTP1.0 10.4	o			
8	content-type	HTTP1.0 10.5	o			
9	expires	HTTP1.0 10.7	o			
10	last-modified	HTTP1.0 10.10	o			

Table I.10: HTTPv1.0 Status code support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	200 OK	HTTP 1.0 9.2	m			
2	201 Created	HTTP 1.0 9.2	o			
3	202 Accepted	HTTP 1.0 9.2	o			
4	204 No Content	HTTP 1.0 9.2	o			
5	300 Multiple Choices	HTTP 1.0 9.3	o			
6	301 Moved Permanently	HTTP 1.0 9.3	o			
7	302 Moved Temporarily	HTTP 1.0 9.3	o			
8	304 Not Modified	HTTP 1.0 9.3	o			
9	400 Bad Request	HTTP 1.0 9.4	o			
10	401 Unauthorized	HTTP 1.0 9.4	o			
11	403 Forbidden	HTTP 1.0 9.4	o			
12	404 Not Found	HTTP 1.0 9.4	o			
13	500 Internal Server Error	HTTP 1.0 9.5	o			
14	501 Not Implemented	HTTP 1.0 9.5	o			
15	502 Bad Gateway	HTTP 1.0 9.5	o			
16	503 Service Unavailable	HTTP 1.0 9.5	o			

I.3.7 Multipurpose Internet Mail Extensions Capabilities

Table I.11: MIME primary message content capabilities

Prerequisite: Table I.2/3 – MIME is supported				
Item	**Capability**	**Reference**	**Status**	**Support Y | N | n/a**
1	Multipart/signed	5.2.7	o	
2	Text/plain	6.1.1.1	o	

Table I.12: S/MIME signature capabilities

Prerequisite: Table I.2/3 – MIME is supported				
Item	**Capability**	**Reference**	**Status**	**Support Y | N | n/a**
1	S/MIME version 2	5.2.7	C1201	
2	Canonical Form	7.1	C1201	
3	Binary transfer encoding	7.3	C1201	

C1201: IF I.11/1 -- If multipart/signed primary message content of MIME is supported
THEN m -- then mandatory, else not applicable
ELSE n/a

Table I.13: S/MIME Signature algorithm capabilities

Prerequisite: Table I.2/3 – MIME is supported				
Item	**Capability**	**Reference**	**Status**	**Support Y | N | n/a**
1	Md2WithRSAEncryption	S/MIME A.4	C1301	
2	Md5WithRSAEncryption	S/MIME A.4	C1301	
3	Sha-1WithRSAEncryption	S/MIME A.4	C1301	

C1301: IF I.11/1 -- If multipart/signed primary message content of MIME is supported
THEN o -- then optional, else not applicable
ELSE n/a

Table I.14: MIME XML Transfer Capabilities

Prerequisite: Table I.2/3 – MIME is supported				
Item	**Capability**	**Reference**	**Status**	**Support Y | N | n/a**
1	Binary transfer encoding	6.1.1.3	m	

I.3.8 Extensible Markup Language Capabilities

Table I.15: XML Character Coding Support

Prerequisite: Table I.2/4 – XML is supported						
Item	**Capability**	**Reference**	**Status**	**Support Y | N | n/a**	**Values allowed**	**Values supported**
1	UTF-8	6.1.2.3	m			
2	UTF-16	6.1.2.3	m			
3	other(s)	6.1.2.3	o		note 1	
4	well-formed	6.1.2.2	m			
NOTE 1: Permissible character encodings defined by ISO/IEC 10646 [27]						

I.3.9 Root Message Capabilities

Table I.16: Root Message Capabilities

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Message as root entity	6.1.3.1	m			
2	Maximum components sent per client message	6.1.3	C1601		1 ..	
3	Maximum components recognized per client message	6.1.3	C1602		1 ..	
4	Equal components in server messages as received in client messages	6.1.3	C1602			
5	Random attribute	6.1.3.2	m			
6	Identifier attribute	6.1.3.3	m			

C1601: IF I.1/1 --Client role
THEN m
ELSE n/a

C1802: IF I.1/2 --Server role
THEN m
ELSE n/a

Table I.17: Random attribute Capabilities

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Random attribute present in messages	6.1.3.2	m			
2	Random value statistically valid	6.1.3.2	m			

Table I.18: Identifier attribute Capabilities

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Identifier present in client messages	6.13.3	C1801			
2	Identifier unique	6.1.3.3	C1801		1 ..	
3	Identifier returned in server messages	6.1.3.3	C1802		1 ..	
4	Identifier associates server messages with client request/indication	6.1.4.4	C1802			

C1801: IF I.1/1 --Client role
THEN m
ELSE n/a

C1602: IF I.1/2 --Server role
THEN m
ELSE n/a

I.3.10 Message support

It is mandatory to support the encoding and decoding of messages used by OSP.

Table I.19: PDU en/decoding

Item	Description	Reference	Status	Support
1	XML message encoding	6.2	m	
2	XML message decoding	6.2	m	

Table I.20: Message support

Item	Description	Reference	Status	Support
1	PricingIndication	6.2.1	o	
2	PricingConfirmation	6.2.2	o	
3	AuthorizationRequest	6.2.3	o	
4	AuthorizationResponse	6.2.4	o	
5	AuthorizationIndication	6.2.5	o	
6	AuthorizationConfirmation	6.2.6	o	
7	UsageIndication	6.2.7	o	
8	UsageConfirmation	6.2.8	o	
9	ReauthorizationRequest	6.2.9	o	
10	ReauthorizationResponse	6.2.10	o	
11	SubscriberAuthenticationRequest	6.2.11	o	
12	SubscriberAuthenticationResponse	6.2.12	o	
13	CapabilitiesIndication	6.2.13	o	
14	CapabilitiesConfirmation	6.2.14	o	

Table I.21: `PricingIndication` Support

Item	Information element	Reference	Status	Support Y | N | n/a
1	Timestamp	6.3.19	m	
2	SourceInfo	6.3.16	m	
3	DestinationInfo	6.3.9	m	
4	Currency	6.3.5	m	
5	Amount	6.3.1	m	
6	Increment	6.3.11	m	
7	Unit	6.3.22	m	
8	Service	6.3.14	m	
9	ValidAfter	6.3.24	m	
10	ValidUntil	6.3.25	m	

Table I.22: `PricingConfirmation` Support

Item	Information element	Reference	Status	Support Y | N | n/a
1	Timestamp	6.3.19	m	
2	Status	6.3.18	m	

Table I.23: `AuthorizationRequest` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Timestamp	6.3.19	m			
2	CallId	6.3.3	m			
3	SourceInfo	6.3.16	m			
4	SourceAlternate	6.3.15	o			
5	DestinationInfo	6.3.9	m			
6	DestinationAlternate	6.3.8	o			
7	Service	6.3.14	m			
8	MaximumDestinations	6.3.12	m			
9	Token	6.3.20	o			
10	SubscriberAuthenticationInfo	6.3.37	o			

Table I.24: `AuthorizationResponse` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Timestamp	6.3.19	m			
2	Status	6.3.18	m			
3	TransactionId	6.3.21	m			
4	Destination	6.3.7	o			
5	Token	6.3.20	o			

Table I.25: `AuthorizationIndication` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Timestamp	6.3.19	m			
2	Role	6.3.13	m			
3	CallId	6.3.3	m			
4	SourceInfo	6.3.16	m			
5	SourceAlternate	6.3.15	o			
6	DestinationInfo	6.3.9	m			
7	DestinationAlternate	6.3.8	o			
8	Service	6.3.14	m			
9	Token	6.3.20	o			

Table I.26: `AuthorizationConfirmation` Support

Item	Information element	Reference	Status	Support Y | N | n/a
1	Timestamp	6.3.19	m	
2	Status	6.3.18	m	
3	ValidAfter	6.3.24	m	
4	ValidUntil	6.3.25	m	

Table I.27: `UsageIndication` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Timestamp	6.3.19	m			
2	Role	6.3.13	m			
3	TransactionId	6.3.21	m			
4	CallId	6.3.3	m			
5	SourceInfo	6.3.16	m			
6	SourceAlternate	6.3.15	o			
7	DestinationInfo	6.3.9	m			
8	DestinationAlternate	6.3.8	o			
9	UsageDetail	6.3.23	o			

Table I.28: `UsageConfirmation` Support

Item	Information element	Reference	Status	Support Y | N | n/a
1	Timestamp	6.3.19	m	
2	Status	6.3.18	m	

Table I.29: `ReauthorizationRequest` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Timestamp	6.3.19	m			
2	Role	6.3.13	m			
3	CallId	6.3.3	m			
4	SourceInfo	6.3.16	o			
5	SourceAlternate	6.3.15	o			
6	DestinationInfo	6.3.9	o			
7	DestinationAlternate	6.3.8	o			
8	TransactionId	6.3.21	m			
9	UsageDetail	6.3.23	o			
10	Token	6.3.20	o			

Table I.30: `ReauthorizationResponse` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Timestamp	6.3.19	m			
2	Status	6.3.18	m			
3	TransactionId	6.3.21	m			
4	Destination	6.3.7	o			
5	Token	6.3.20	o			

Table I.31: `SubscriberAuthenticationRequest` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Timestamp	6.319	m			
2	SourceInfo	6.3.16	m			
3	SourceAlternate	6.3.15	m			
4	DestinationInfo	6.3.9	o			
5	Service	6.3.14	o			

Table I.32: `SubscriberAuthenticationResponse` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Timestamp	6.3.19	m			
2	Status	6.3.18	m			
3	SubscriberAuthenticationInfo	6.3.37	o			

Table I.33: `CapabilitiesIndication` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	DeviceInfo	6.3.38	o			
2	OSPVersion	6.3.36	m			
3	OSPCapability	6.3.32	o			
4	Resources	6.3.40	o			
5	Data Rate	6.3.41	o			
6	NumberOfChannels	6.3.42	o			
7	Bandwidth	6.3.43	o			
8	AlmostOutOfResources	6.3.44	o			

Table I.34: `CapabilitiesConfirmation` Support

Item	Information element	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Timestamp	6.3.19	m			
2	Status	6.3.18	m			
3	OSPVersion	6.3.36	m			
4	OSPService	6.3.33	o			
5	OSPServiceURL	6.3.34	o			
6	OSPSignatureRequired	6.3.35	o			
7	CertificateChain	6.3.31	o			
8	Certificate	6.3.30	o			
9	DeviceId	6.3.39	o			

I.3.11 Authorization Token Support

Table I.35: Authorization Token Support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Cryptographic Encoding	D.1	o			
2	ASN.1 format	D.2.1	o.1			
3	XML format	D.2.2	o.1			
4	Binary XML format	D.2.3	o.1			

o.1: It is essential to support at least one of these formats.

Table I.36: Cryptographic encoding type support

Prerequisite: I.29/1						
Item	**Capability**	**Reference**	**Status**	**Support Y | N | n/a**	**Values allowed**	**Values supported**
1	signed-data	PKCS7	o			
2	encrypted-data	PKCS7	o			

Table I.37: Authorization Token Support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	DigestAlgorithm	PKCS7	o			
2	DigestEncryptionAlgorithm	PKCS7	o			

Table I.38: Authorization Token Support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	md2	X.509	o.1			
2	md4	X.509	o.1			
3	md5	X.509	o.1			
4	sha-1	ISO	o.1			

o.1: It is essential to support at least one of these formats.

Table I.39: Authorization Token Support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	RsaEncryption	PKCS1	o.1			
2	id-dsa-with-sha1	ISO	o.1			

o.1: It is essential to support at least one of these formats.

Table I.40: Authorization Token Support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	KeyEncryptionAlgorithm	PKCS7	o			
2	ContentEncryptionAlgorithm	PKCS7	o			

Table I.41: Authorization Token Support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	rsaEncryption	PKCS1	o			

Table I.42: Authorization Token Support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	des-ede3-cbc	ISO	o.1			
2	rc2-cbc	ISO	o.1			

o.1: It is essential to support at least one of these formats.

Table I.43: ASN.1 format token support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Random	D.2.1	m			
2	SourceInfo	D.2.1	m			
3	DestinationInfo	D.2.1	m			
4	CallId	D.2.1	o			
5	ValidAfter	D.2.1	o			
6	ValidUntil	D.2.1	o			
7	TransactionId	D.2.1	o			
8	ServiceAuthorized	D.2.1	o			
9	AuthorityURL	D.2.1	o			

Table I.44: ASN.1 format source info and destination info coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Maximum instances	D.2.1	m		0..	
2	e164	H.225.0 [9]	o.1			
3	h323-ID	H.225.0 [9]	o.1			
4	url-ID	H.225.0 [9]	o.1			
5	transport-ID	H.225.0 [9]	o.1			
6	email-ID	H.225.0 [9]	o.1			
7	PartyNumber	H.225.0 [9]	o.1			

o.1: It is essential to support at least one of these formats.

Table I.45: ASN.1 format transport-id coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	IpAddress	H.225.0 [9]	o.1			
2	IpSourceRoute	H.225.0 [9]	o.1			
3	Ipxaddress	H 225.0 [9]	o.1			
4	ip6Address	H.225.0 [9]	o.1			
5	NetBios	H.225.0 [9]	o.1			
6	Nsap	H.225.0 [9]	o.1			
7	NonStandardAddress	H.225.0 [9]	o.1			

o.1: It is essential to support at least one of these formats.

Table I.46: ASN.1 format ServiceAuthorized coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Maximum instances	D.2.1	m		0 ..	
2	Service	D.2.1	m			
3	Limit	D.2.1	o			

Table I.47: ASN.1 format service coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	BasicTelephony	D.2.1	o			
2	VendorSpecific	D.2.1	o			

Table I.48: ASN.1 format limit coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Amount	D.2.1	m			
2	Units	D.2.1	m			
3	Increment	D.2.1	m			
4	VendorSpecific	D.2.1	o			

Table I.49: ASN.1 format unit coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Seconds	D.2.1	o.1			
2	Packets	D.2.1	o.1			
3	Bytes	D.2.1	o.1			
4	VendorSpecific	D.2.1	o.1			

o.1: It is essential to support at least one of these formats.

Table I.50: XML format token support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	SourceInfo	6.3.16	o			
2	SourceAlternate	6.3.15	o			
3	DestinationInfo	6.3.9	o			
4	DestinationAlternate	6.3.8	o			
5	CallId	6.3.3	o			
6	ValidAfter	6.3.24	o			
7	ValidUntil	6.3.25	o			
8	TransactionId	6.3.21	o			
9	UsageDetail	6.3.23	o			
10	AuthorityURL	6.3.2	o			
11	SourceSignalAddress	6.3.17	o			
12	DestinationSignalAddress	6.3.10	o			

Table I.51: XML format source or destination token coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	e164	6.3.9	o.1			
2	h323	6.3.9	o.1			
3	url	6.3.9	o.1			
4	Email	6.3.9	o.1			
5	Transport	6.3.9	o.1			
6	International	6.3.9	o.1			
7	National	6.3.9	o.1			
8	Network	6.3.9	o.1			
9	Subscriber	6.3.9	o.1			
10	Abbreviated	6.3.9	o.1			
11	e164prefix	6.3.9	o.1			
12	iso7812	6.3.15	o.1			
13	pin	6.3.15	o.1			
14	epin	6.3.15	o.1			
15	deviceID	6.3.15	o.1			

o.1: It is essential to support at least one of these formats.

Table I.52: XML source alternate or destination alternate coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Maximum instances	D.2.2	m			
2	e164	6.3.9	o.1			
3	h323	6.3.9	o.1			
4	url	6.3.9	o.1			
5	Email	6.3.9	o.1			
6	Transport	6.3.9	o.1			
7	International	6.3.9	o.1			
8	National	6.3.9	o.1			
9	Network	6.3.9	o.1			
10	Subscriber	6.3.9	o.1			
11	Abbreviated	6.3.9	o.1			
12	e164prefix	6.3.9	o.1			
13	iso7812	6.3.15	o.1			
14	pin	6.3.15	o.1			
15	epin	6.3.15	o.1			
16	deviceID	6.3.15	o.1			

o.1: It is essential to support at least one of these formats.

Table I.53: XML call id coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Maximum instances	D.2.2	m		0 ..	
2	CDATA encoding	6.3.3	o.1			
3	base64 encoding	6.3.3	o.1			

o.1: It is essential to support at least one of these formats.

Table I.54: XML UsageDetail coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	Maximum instances	D.2.2	m		0 ..	
2	Service	6.3.14	m			
3	Amount	6.3.1	m			
4	Increment	6.3.11	m			
5	Unit	6.3.22	m			
6	StartTime	6.3.27	o			
7	EndTime	6.3.26	o			
8	TerminationCause	6.3.29	o			
9	TCCode	6.3.28	o			
10	Description	6.3.6	o			
11	Statistics	C.1.1	o			

Table I.55: XML units coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	s	6.3.22	o.1			
2	pkt	6.3.22	o.1			
3	byte	6.3.22	o.1			

o.1: It is essential to support at least one of these formats.

Table I.56: XML Statistics coding support

Item	Capability	Reference	Status	Support Y | N | n/a	Values allowed	Values supported
1	LossSent	C.1.2	o			
2	LossReceived	C.1.5	o			
3	Packets	C.1.3	o			
4	Fraction	C.1.4	o			
5	OneWayDelay	C.1.6	o			
6	Minimum	C.1.7	o			
7	Mean	C.1.8	o			
8	Variance	C.1.9	o			
9	Samples	C.1.10	o			
10	RoundTripDelay	C.1.11	o			
11	Minimum	C.1.7	o			
12	Mean	C.1.8	o			
13	Variance	C.1.9	o			
14	Samples	C.1.10	o			

I.3.12 XML Extensions

List XML extensions generated by implementation (optional):

.....

.....

.....

.....

.....

.....

List XML extensions recognized by implementation (optional):

.....

.....

.....

.....

.....

.....

Annex J (informative): OSP Applications and Implementations

This annex describes the application of ETSI TS 101 321, the Open Settlement Protocol, in various network implementations.

J.1 Call Control Protocols

The Open Settlement Protocol (OSP) is not limited to any particular call control protocol. Rather, it is neutral, and is designed for deployment with devices conforming to H.323 [13] and Session Initiation Protocol (SIP), as well as various proprietary signalling protocols.

NOTE: Some well-known IP telephony protocols, including the Simple Gateway Control Protocol (SGCP), IP Device Control (IPDC), Media Gateway Control Protocol (MGCP), and MEGACO/H.248, do not integrate directly with OSP. Instead, systems relying on those protocols integrate with OSP at the level of protocols between gateway controllers (e.g. call agents). Today, those protocols are primarily based on H.323 [13] or SIP.

This section illustrates how OSP may be used with various call signalling protocols. Rather than considering each protocol separately, however, it considers three different architectures—peer-to-peer, distributed tightly-coupled, and distributed loosely-coupled—that can be applied to various signalling protocols. The H.323 [13] and SIP protocols are used as examples to describe the interaction with OSP in detail. The principals described in these sections can also be used for other protocols.

J.1.1 Peer-to-Peer Architecture

In a peer-to-peer architecture, gateways contact each other directly, without any control or co-ordination from other devices. This architecture corresponds to simple H.323 [13] deployments without gatekeepers as well as basic SIP-based services. In such an environment, the endpoints of a call implement the open settlement protocol to find and authorize each other, and to directly report usage information to a settlement provider. The following subsections illustrate the use of OSP and in both H.323 [13] and SIP architectures.

J.1.1.1 H.323 Gateways

When operating with H.323 [13] gateways, OSP provides services at both the start and end of each call. During initial call setup, gateways can use OSP to obtain call routing information and authorization tokens. Once the call has ended, gateways use OSP to report usage details.

J.1.1.1.1 Call Routing and Authorization

The following figure shows how the open settlement protocol may be used by H.323 [13] gateways to find and authorize each other.

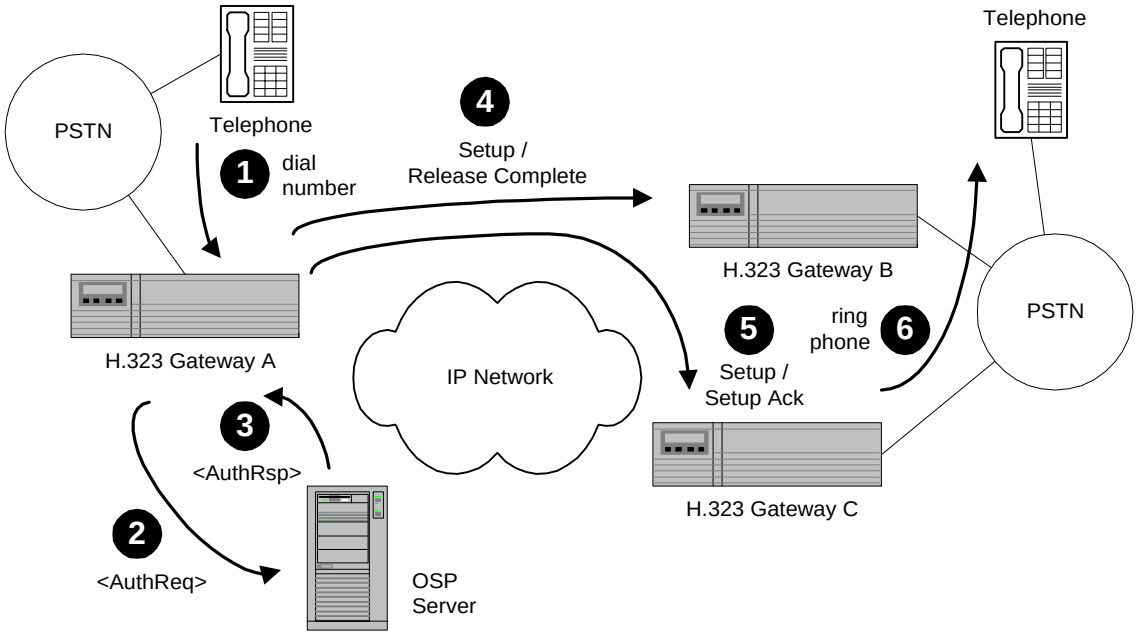


Figure J.1

The figure highlights the following interactions between the devices.

Calling party indicates the desired, destination phone number to Gateway A (by, for example, responding to a DTMF-based IVR, sending an ISDN Q.931 Setup, or transmitting an SS7 Initial Address Message).

Gateway A sends an OSP `<AuthorizationRequest>` message to the OSP Server. The significant elements within the `<AuthorizationRequest>` include:

`<Timestamp>`	Time of request
`<CallId>`	H.323 [13] Call Identifier to be used for the call
`<SourceInfo type="e164">`	Calling party's E.164 [12] number if available; otherwise a local E.164 [12] number controlled by Gateway A, e.g. `14048724887`; this number shall be passed to the destination gateway(s) in Setup messages
`<SourceAlternate type="transport">`	DNS name or IP address of Gateway A, for example `gatewayA.carrier.com`
`<DestinationInfo type="e164">`	Called party's E.164 [12] number, e.g. `33492944299`
`<Service/>`	Empty (for basic service)
`<MaximumDestinations>`	The maximum number of destinations, including alternatives, Gateway A will consider

OSP Server replies with an `<AuthorizationResponse>` message. The message indicates two candidate destinations: Gateway B and Gateway C, in that order. In particular, the `<AuthorizationResponse>` contains the following elements:

`<Timestamp>`	time of response
`<Status>`	result of response, e.g. `<Code>200</Code>`
`<TransactionId>`	transaction identifier assigned by settlement provider
`<Destination>`	first destination gateway to try for call

<Destination SignalAddress type="transport">	DNS name or IP address of Gateway B, for example gatewayB.itsp.fr
<Token>	authorization token to be passed to Gateway B
<ValidAfter>	time after which token for Gateway B is valid
<ValidUntil>	time until which token for Gateway B is valid
<UsageDetail>	how much service is authorized with Gateway B
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	H.323 [13] Call Identifier to be used for the call to Gateway B
<Destination>	second destination gateway to try for call
<Destination SignalAddress type="transport">	DNS name or IP address of Gateway C, for example gatewayC.isp.fr
<Token>	authorization token to be passed to Gateway C
<ValidAfter>	time after which token for Gateway C is valid
<ValidUntil>	time until which token for Gateway C is valid
<UsageDetail>	how much service is authorized with Gateway C
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	H.323 [13] Call Identifier to be used for the call to Gateway C

Gateway A sends an H.225.0 [9] Setup message to Gateway B; however, the Setup is refused with a Release Complete.

NOTE: The H.323 [13] Call Identifier in the Setup message has the same value as in the original <AuthorizationRequest>. The Setup shall also include the authorization token(s) provided in the OSP <AuthorizationResponse>. For maximum interoperability, any tokens should use the following object identifier, constructed according to ETSI guidelines.

itu-t(0), identified-organization(4), etsi(0), ts-101-321(1321), token(1), xml-format(2)

Gateway A then sends a second H.225.0 [9] Setup message, this time to Gateway C, and this time the Setup is accepted with a Setup Acknowledge. This Setup message also uses the H.323 [13] Call Identifier from the <AuthorizationRequest>, and the Setup message shall also include the authorization token(s) provided in the OSP <AuthorizationResponse>. For maximum interoperability, any tokens should use the following object identifier, constructed according to ETSI guidelines.

itu-t(0), identified-organization(4), etsi(0), ts-101-321(1321), token(1), xml-format(2)

Gateway C accepts the Setup and completes the call to the called party; the phone conversation can now take place.

J.1.1.1.2 Usage Reports

Once the call has ended, both gateways report usage details to an OSP server. As the following figure indicates, those reports are conveyed in OSP `<UsageIndication>` messages.

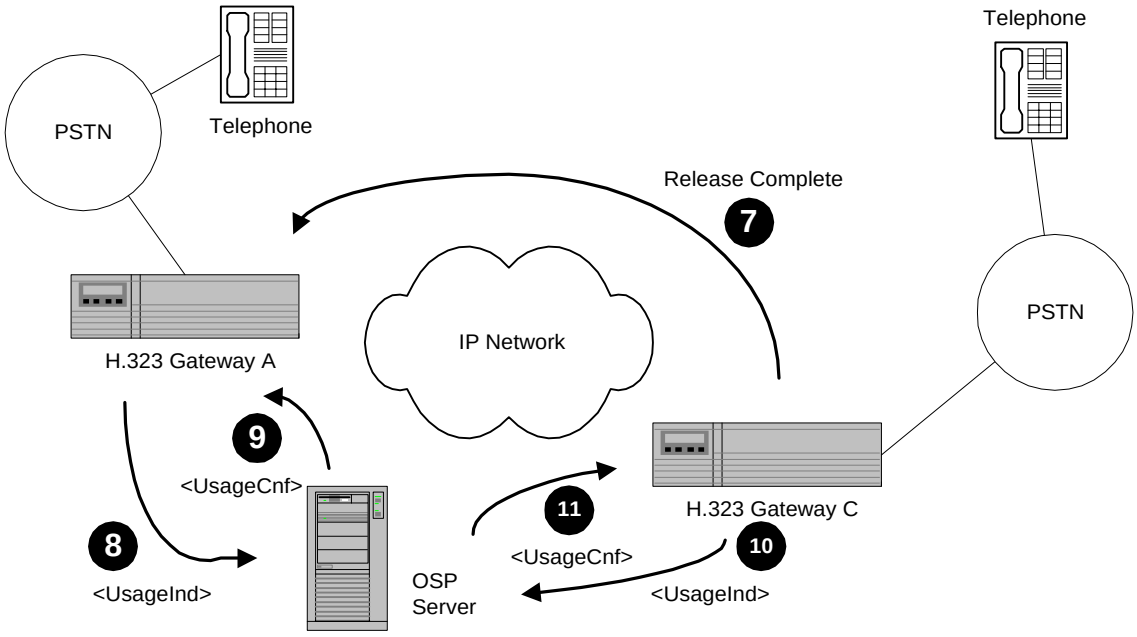


Figure J.2

The steps shown in the figure are straightforward.

Gateways A and C clear the call by exchanging an H.225.0 [9] Release Complete message.

Gateway A sends a `<UsageIndication>` message to the OSP server. If Gateway A is not using any protocol extensions, the message will contain the following elements:

`<Timestamp>`	time of request
`<Role>`	for Gateway A, `source`
`<TransactionId>`	transaction ID assigned by OSP server in authorization response
`<CallId>`	H.323 [13] Call Identifier used for the call
`<SourceInfo type="e164">`	calling party's E.164 [12] number as returned in the authorization response, e.g. `14048724887`
`<SourceAlternate type="transport">`	DNS name or IP address of Gateway A, for example `gatewayA.carrier.com`
`<DestinationInfo type="e164">`	called party's E.164 [12] number, e.g. `33492944299`
`<DestinationAlternate type="transport">`	DNS name or IP address of Gateway C, for example, `gatewayC.isp.fr`
`<UsageDetail>`	usage information for the call
`<Service/>`	empty (for basic service)
`<Amount>`	amount of service used, e.g. `300`
`<Increment>`	increment of service measurement, e.g. `1`
`<Unit>`	unit of service measurement, e.g. `s` for seconds

The OSP server responds with a `<UsageConfirmation>` message. If it has accepted the usage report, that message will contain a successful `<Status>` element (e.g. `<Code>200</Code>`).

Gateway C also sends a <UsageIndication> to the OSP server. That message would include the following elements:

<Timestamp>	time of request
<Role>	for Gateway C, destination
<TransactionId>	transaction ID assigned by OSP server and passed to Gateway C in authorization token
<CallId>	H.323 [13] Call Identifier used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number as presented in the Setup message, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of Gateway A, for example [172.16.1.1]
<DestinationInfo type="e164">	called party's E.164 [12] number, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of Gateway C, for example, gatewayC.isp.fr
<UsageDetail>	usage information for the call
<Service/>	empty (for basic service)
<Amount>	amount of service used, e.g. 300
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<Statistics>	statistical information for call
<LossSent>	loss information for packets sent by Gateway C
<Packets>	number of packets lost from Gateway C to Gateway A
<Fraction>	fraction (from 0 to 255) of packets lost from C to A
<LossReceived>	loss information for packets sent by Gateway A
<Packets>	number of packets lost from Gateway A to Gateway C
<Fraction>	fraction (from 0 to 255) of packets lost from A to C
<OneWayDelay>	one way delay measured from Gateway A to C
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements
<RoundTripDelay>	round trip delay between Gateway A and C measured during call
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements

The OSP server responds with a <UsageConfirmation> message. If it has accepted the usage report, that message will contain a successful <Status> element (e.g. <Code>200</Code>).

J.1.1.2 Session Initiation Protocol Gateways

In a simple peer-to-peer environment, Session Initiation Protocol (SIP) gateways operate in a manner nearly identical to H.323 [13] gateways. As before, it is convenient to consider interaction using OSP before and after the multimedia call.

J.1.1.2.1 Call Routing and Authorization

The following figure shows how the open settlement protocol may be used by SIP gateways to find and authorize each other.

NOTE 1: This function allows a SIP gateway to find a peer gateway for a particular, PSTN-terminated, telephone user. As such, it is not the same as the standard SIP user location function.

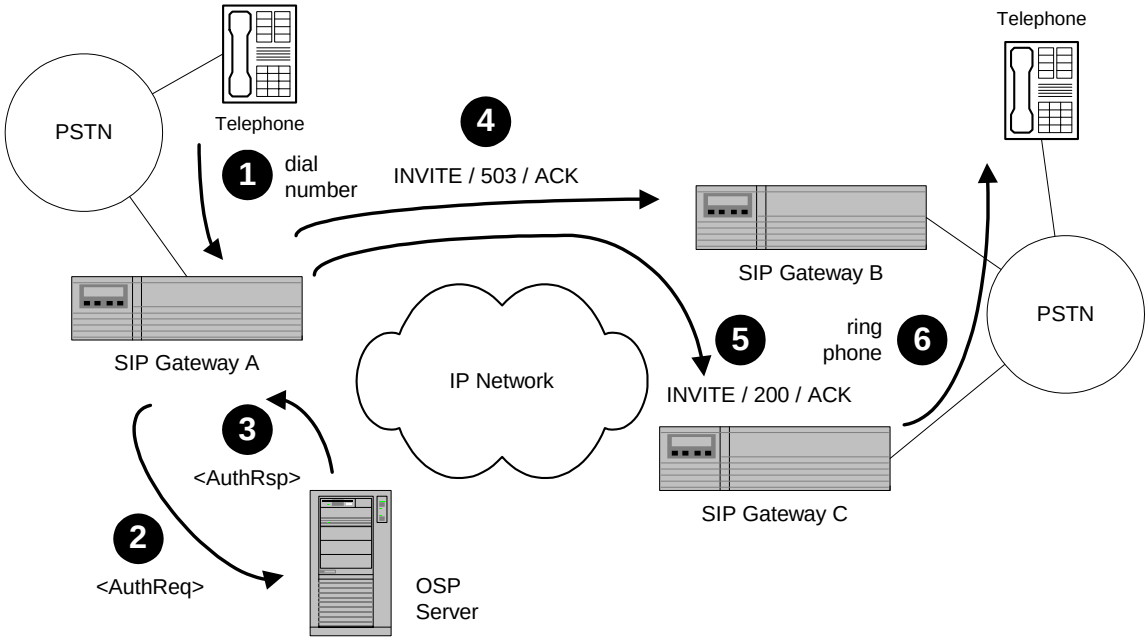


Figure J.3

The figure highlights the following interactions between the devices.

Calling party indicates the desired, destination phone number to Gateway A (by, for example, responding to a DTMF-based IVR, sending an ISDN Q.931 Setup, or transmitting an SS7 Initial Address Message).

Gateway A sends an OSP `<AuthorizationRequest>` message to the OSP Server. The significant elements within the `<AuthorizationRequest>` include:

`<Timestamp>`	time of request
`<CallId>`	SIP Call-ID to be used for the call
`<SourceInfo type="e164">`	calling party's E.164 [12] number if available; otherwise a local E.164 [12] number controlled by Gateway A, e.g. `14048724887`; this number shall be passed to the destination gateway(s) in the subsequent INVITE method
`<SourceAlternate type="transport">`	DNS name or IP address of Gateway A, for example `gatewayA.carrier.com`
`<DestinationInfo type="e164">`	called party's E.164 [12] number, e.g. `33492944299`
`<Service/>`	empty (for basic service)
`<MaximumDestinations>`	the maximum number of destinations, including alternatives, Gateway A will consider

OSP Server replies with an <AuthorizationResponse> message. The message indicates two candidate destinations: Gateway B and Gateway C, in that order. In particular, the <AuthorizationResponse> contains the following elements:

<Timestamp>	time of response
<Status>	result of response, e.g. <Code>200</Code>
<TransactionId>	transaction identifier assigned by settlement provider
<Destination>	first destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of Gateway B, for example gatewayB.itsp.fr
<Token>	authorization token to be passed to Gateway B
<ValidAfter>	time after which token for Gateway B is valid
<ValidUntil>	time until which token for Gateway B is valid
<UsageDetail>	how much service is authorized with Gateway B
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	SIP Call-ID to be used for the call to Gateway B
<Destination>	second destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of Gateway C, for example gatewayC.isp.fr
<Token>	authorization token to be passed to Gateway C
<ValidAfter>	time after which token for Gateway C is valid
<ValidUntil>	time until which token for Gateway C is valid
<UsageDetail>	how much service is authorized with Gateway C
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	SIP Call-ID to be used for the call to Gateway C

Gateway A sends an INVITE message to Gateway B; however, Gateway B cannot accept the request. It responds with a "503 Service Unavailable," which Gateway A acknowledges with an ACK.

NOTE 2: The SIP Call-ID in the INVITE message has the same value as in the original <AuthorizationRequest>. The INVITE shall also include authorization token(s) provided in the OSP <AuthorizationResponse>. Each token should be conveyed in the SIP message body using the application/osp-token MIME type, as initially defined in draft-thomas-mime-osp-token-00.txt.

Gateway A then sends a second INVITE message, this time to Gateway C, and this time the INVITE is accepted with "200 Success," to which Gateway A replies with an ACK. This INVITE message also uses the SIP Call-ID from the <AuthorizationRequest>, and the INVITE shall also include the authorization token(s) provided in the OSP <AuthorizationResponse>. Each token should be conveyed in the SIP message body using the application/osp-token MIME type, as initially defined in draft-thomas-mime-osp-token-00.txt.

Gateway C accepts the INVITE and completes the call to the called party; the phone conversation can now take place.

J.1.1.2.2 Usage Reports

Once the call has ended, both gateways report usage details to an OSP server. As the following figure indicates, those reports are conveyed in OSP `<UsageIndication>` messages.

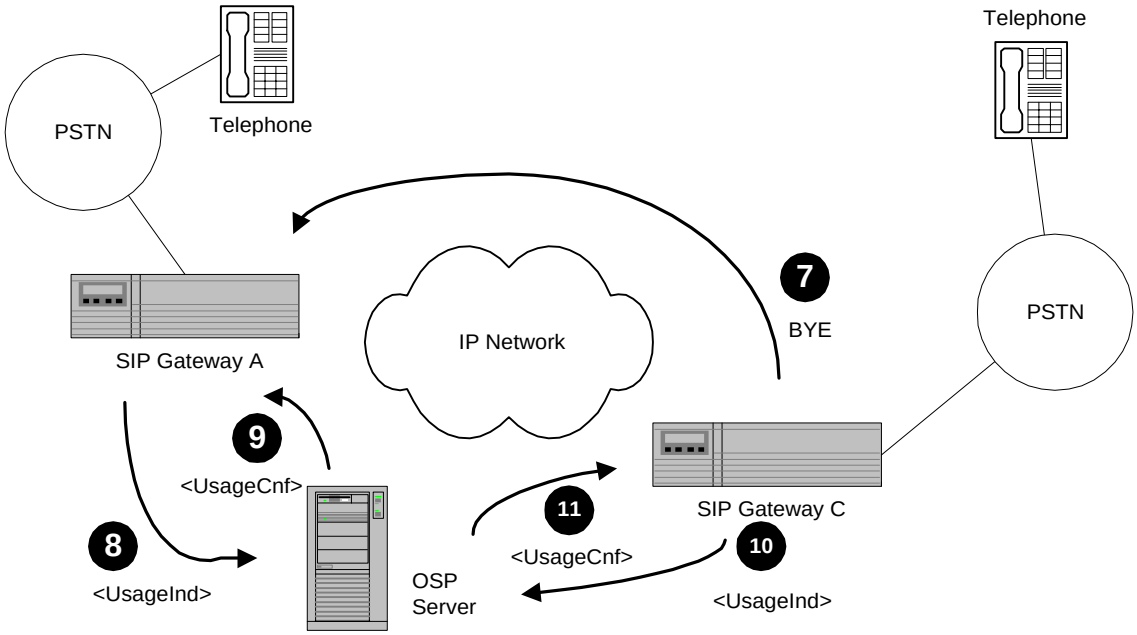


Figure J.4

The steps shown in the figure are straightforward.

Gateways A and C clear the call with a SIP BYE message.

Gateway A sends a `<UsageIndication>` message to the OSP server. If Gateway A is not using any protocol extensions, the message will contain the following elements:

`<Timestamp>`	time of request
`<Role>`	for Gateway A, `source`
`<TransactionId>`	transaction ID assigned by OSP server in authorization response
`<CallId>`	SIP Call-ID used for the call
`<SourceInfo type="e164">`	calling party's E.164 [12] number as returned in the authorization response, e.g. `14048724887`
`<SourceAlternate type="transport">`	DNS name or IP address of Gateway A, for example `gatewayA.carrier.com`
`<DestinationInfo type="e164">`	called party's E.164 [12] number, e.g. `33492944299`
`<DestinationAlternate type="transport">`	DNS name or IP address of Gateway C, for example, `gatewayC.isp.fr`
`<UsageDetail>`	usage information for the call
`<Service/>`	empty (for basic service)
`<Amount>`	amount of service used, e.g. `300`
`<Increment>`	increment of service measurement, e.g. `1`
`<Unit>`	unit of service measurement, e.g. `s` for seconds

The OSP server responds with a <UsageConfirmation> message. If it has accepted the usage report, that message will contain a successful <Status> element (e.g. <Code>200</Code>).

Gateway C also sends a <UsageIndication> to the OSP server.

<Timestamp>	time of request
<Role>	for Gateway C, destination
<TransactionId>	transaction ID assigned by OSP server and passed to Gateway C in authorization token
<CallId>	SIP Call-ID used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number as presented in the INVITE message, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of Gateway A, for example [172.16.1.1]
<DestinationInfo type="e164">	called party's E.164 [12] number, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of Gateway C, for example, gatewayC.isp.fr
<UsageDetail>	usage information for the call
<Service/>	empty (for basic service)
<Amount>	amount of service used, e.g. 300
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<Statistics>	statistical information for call
<LossSent>	loss information for packets sent by Gateway C
<Packets>	number of packets lost from Gateway C to Gateway A
<Fraction>	fraction (from 0 to 255) of packets lost from C to A
<LossReceived>	loss information for packets sent by Gateway A
<Packets>	number of packets lost from Gateway A to Gateway C
<Fraction>	fraction (from 0 to 255) of packets lost from A to C
<OneWayDelay>	one way delay measured from Gateway A to C
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements
<RoundTripDelay>	round trip delay between Gateway A and C measured during call
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements

The OSP server responds with a <UsageConfirmation> message. If it has accepted the usage report, that message will contain a successful <Status> element (e.g. <Code>200</Code>).

J.1.2 Tightly Controlled Distributed Architecture

The Open Settlement Protocol supports operation in a tightly controlled, distributed architecture. That architecture includes some H.323 [13]-based deployments with active gatekeepers and SIP proxy servers (but not SIP redirect servers). The following subsections illustrate the use of OSP in those environments.

J.1.2.1 H.323 Gatekeeper Routed Calls

In H.323 [13] deployments with gatekeepers, the gatekeeper may play a very active roll in call signalling. In such an environment, gatekeepers not only assume responsibility for call routing and authorization on behalf of their endpoints. They also act as the signalling endpoint for calls into their zone. As the following figure shows, gatekeepers may support multimedia terminals as well as gateways. As that figure and the following figure also highlight, only gatekeepers are required to support the Open Settlement Protocol in this environment.

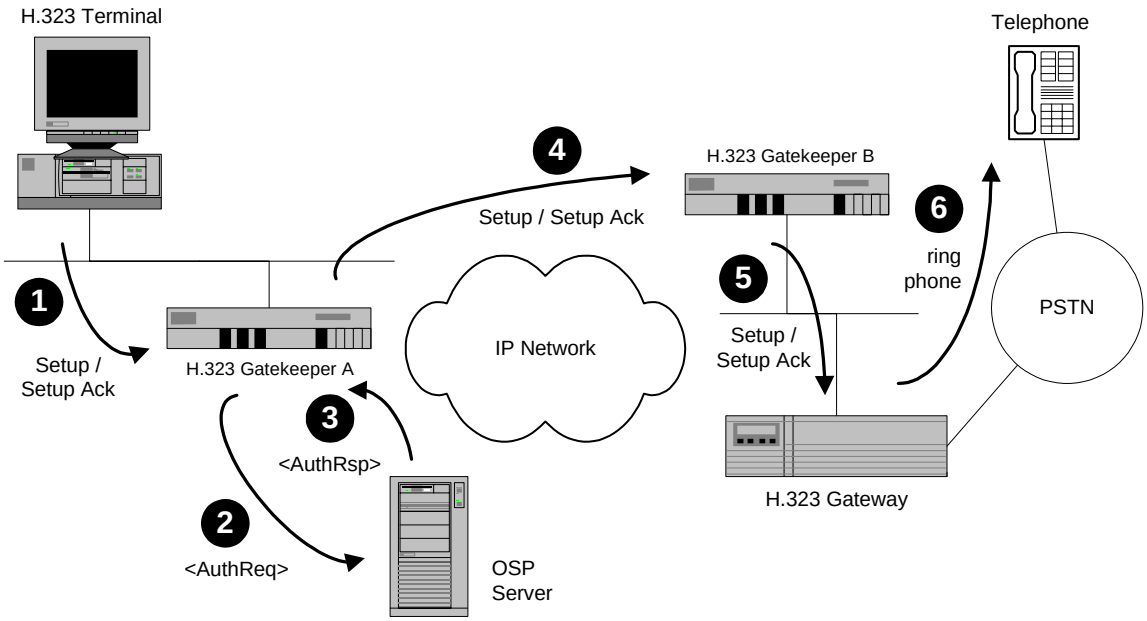


Figure J.5

NOTE: All destination gateways (or endpoints) controlled by the gatekeepers shall be equivalent, as far as the OSP server is concerned. Indeed, the OSP server is not even aware of the existence of any H.323 [13] terminals or gateways. Such an architecture, therefore, places implicit restrictions on the services offered by the operators. An operator, for example, will be unable to price termination services differently for different gateways in the same zone.

J.1.2.1.1 Call Routing and Authorization

H.323 [13] Terminal begins a call by sending an H.225.0 [9] Setup message its gatekeeper, Gatekeeper A.

NOTE: This scenario assumes the use of pre-granted admission requests.

The Setup indicates that the called party is identified by an E.164 [12] phone number such as +33 4 92 94 42 99.

Gatekeeper A sends an OSP `<AuthorizationRequest>` message to the OSP Server. The significant elements within the `<AuthorizationRequest>` include:

`<Timestamp>`	time of request
`<CallId>`	H.323 [13] Call Identifier to be used for the call
`<SourceInfo type="e164">`	a representation of the H.323 [13] Terminal using an E.164 [12] number; in the absence of other information, this number may be derived using the IP address to E.164 [12] number mapping of ETSI TIPHON; this number shall be passed to the destination gateway(s) in Setup messages
`<SourceAlternate type="transport">`	DNS name or IP address of Gatekeeper A, for example `gatekeeperA.carrier.com`
`<DestinationInfo type="e164">`	called party's E.164 [12] number, e.g. `33492944299`

type="e164">	
<Service/>	empty (for basic service)
<MaximumDestinations>	the maximum number of destinations, including alternatives, Gatekeeper A will consider

OSP Server replies with an <AuthorizationResponse> message. The message identifies Gatekeeper B. In particular, the <AuthorizationResponse> contains the following elements:

<Timestamp>	time of response
<Status>	result of response, e.g. <Code>200</Code>
<TransactionId>	transaction identifier assigned by settlement provider
<Destination>	first destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of Gatekeeper B, for example gatekeeperB.itsp.fr
<Token>	authorization token to be passed to Gatekeeper B
<ValidAfter>	time after which token for Gatekeeper B is valid
<ValidUntil>	time until which token for Gatekeeper B is valid
<UsageDetail>	how much service is authorized with Gatekeeper B
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	H.323 [13] Call Identifier to be used for the call to Gatekeeper B

Gatekeeper A extends the call with its own Setup message to Gatekeeper B.

Gatekeeper B accepts the Setup and extends the call to the destination gateway with another Setup/Setup Acknowledge exchange.

The Gateway adds the final leg of the call by dialling the called party; the phone conversation can now take place.

J.1.2.1.2 Usage Reports

At the conclusion of the phone call, both gatekeepers shall report usage information to the OSP server. The following figure identifies the principle steps of a typical call completion.

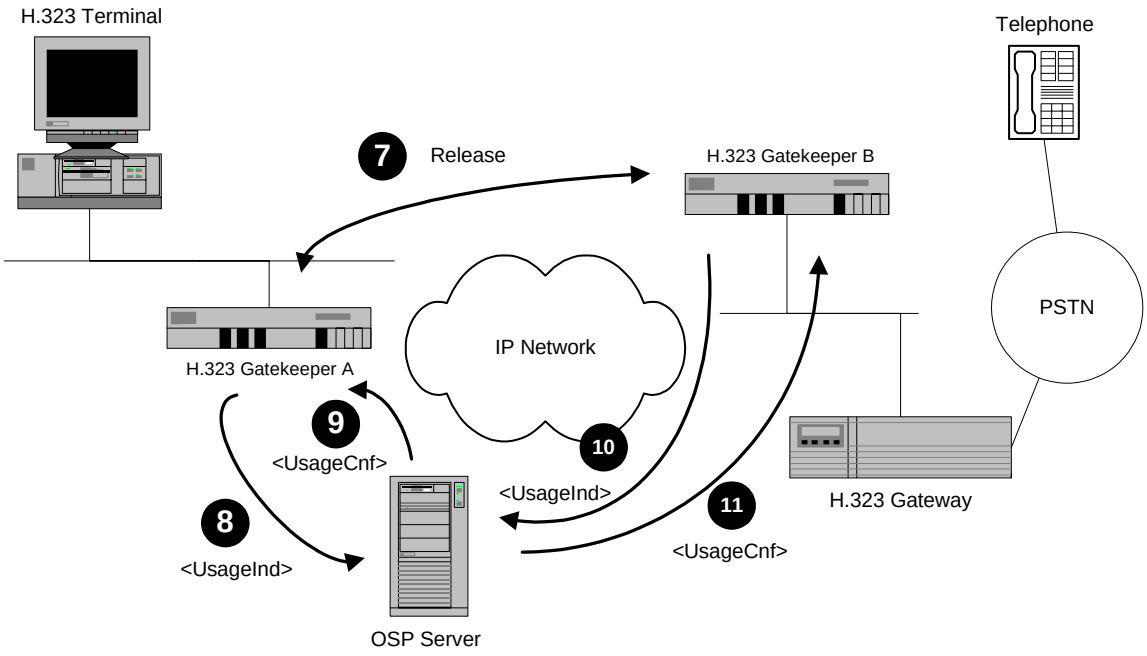


Figure J.6

The steps shown in the figure are straightforward.

The gatekeepers release the call with an exchange of H.225.0 [9] Release and Release Complete messages.

Gatekeeper A then sends a `<UsageIndication>` message to the OSP server. In this example Gatekeeper A may not be able to support protocol extensions such as statistics. Its message, therefore contains just the following elements:

`<Timestamp>`	time of request
`<Role>`	for Gatekeeper A, `source`
`<TransactionId>`	transaction ID assigned by OSP server in authorization response
`<CallId>`	H.323 [13] Call Identifier used for the call
`<SourceInfo type="e164">`	calling party's E.164 [12] number as returned in the authorization response, e.g. `14048724887`
`<SourceAlternate type="transport">`	DNS name or IP address of Gatekeeper A, for example `gatekeeperA.carrier.com`
`<DestinationInfo type="e164">`	called party's E.164 [12] number, e.g. `33492944299`
`<DestinationAlternate type="transport">`	DNS name or IP address of Gatekeeper B, for example, `gatekeeperB.itsp.fr`
`<UsageDetail>`	usage information for the call
`<Service/>`	empty (for basic service)
`<Amount>`	amount of service used, e.g. `300`
`<Increment>`	increment of service measurement, e.g. `1`
`<Unit>`	unit of service measurement, e.g. `s` for seconds

The OSP server responds with a `<UsageConfirmation>` message. If it has accepted the usage report, that message will contain a successful `<Status>` element (e.g. `<Code>200</Code>`).

Gatekeeper B also sends a <UsageIndication> to the OSP server. That message would include the following elements:

<Timestamp>	time of request
<Role>	for Gatekeeper B, destination
<TransactionId>	transaction ID assigned by OSP server and passed to Gatekeeper B in authorization token
<CallId>	H.323 [13] Call Identifier used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number as presented in the Setup message, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of H.323 [13] Terminal, for example [172.16.100.1]
<DestinationInfo type="e164">	called party's E.164 [12] number, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of Gatekeeper B, for example, gatekeeperB.itsp.fr
<UsageDetail>	usage information for the call
<Service/>	empty (for basic service)
<Amount>	amount of service used, e.g. 300
<Increment>	Increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<Statistics>	statistical information for call
<LossSent>	loss information for packets sent by Gatekeeper B
<Packets>	number of packets lost from Gatekeeper B to H.323 [13] Terminal
<Fraction>	fraction (from 0 to 255) of packets lost from B to H.323 [13] Terminal
<LossReceived>	loss information for packets sent by H.323 [13] Terminal
<Packets>	number of packets lost from H.323 [13] Terminal to Gatekeeper B
<Fraction>	fraction (from 0 to 255) of packets lost from Terminal to B
<OneWayDelay>	one way delay measured from Terminal to B
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements
<RoundTripDelay>	round trip delay between Terminal and B measured during call
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements

The OSP server responds with a <UsageConfirmation> message. If it has accepted the usage report, that message will contain a successful <Status> element (e.g. <Code>200</Code>).

J.1.2.2 Session Initiation Protocol Proxy Servers

Session Initiation Protocol (SIP) proxy servers may function in a manner similar to H.323 [13] gatekeepers. In such an environment, the proxy servers may support the Open Settlement Protocol, while the systems on whose behalf they act need not.

NOTE: All destination gateways (or endpoints) served by the SIP proxies shall be equivalent, as far as the OSP server is concerned. Indeed, the OSP server is not even aware of the existence of any endpoints or gateways. Such an architecture, therefore, places implicit restrictions on the services offered by the operators. An operator, for example, will be unable to price termination services differently for different gateways served by the same proxy.

J.1.2.2.1 Call Routing and Authorization

The following figure shows how OSP plays a role in environments that use SIP Proxy Servers. In the figure, The SIP Proxy Server, acting on behalf of the SIP client, completes a call to a SIP Gateway, and ultimately to the PSTN.

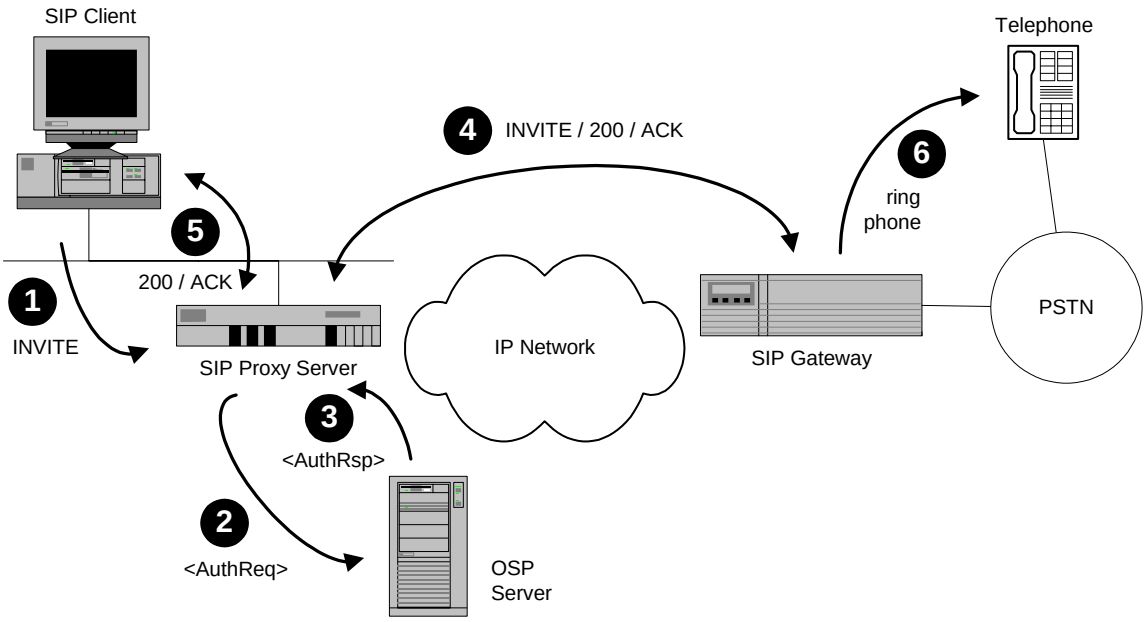


Figure J.7

The SIP Client begins a call by sending an INVITE request to its proxy server. The INVITE indicates that the called party is identified by an E.164 [12] phone number such as +33 4 92 94 42 99.

The SIP Proxy Server sends an OSP `<AuthorizationRequest>` message to the OSP Server. The significant elements within the `<AuthorizationRequest>` include the following:

`<Timestamp>`	time of request
`<CallId>`	SIP Call-ID to be used for the call
`<SourceInfo type="e164">`	a representation of the SIP client using an E.164 [12] number; in the absence of other information, this number may be derived using the IP address to E.164 [12] number mapping of ETSI TIPHON.
`<SourceAlternate type="transport">`	DNS name or IP address of the Proxy Server, for example `proxy.carrier.com`
`<DestinationInfo type="e164">`	called party's E.164 [12] number, e.g. `33492944299`
`<Service/>`	empty (for basic service)
`<MaximumDestinations>`	the maximum number of destinations, including alternatives, the Proxy Server will consider

OSP Server replies with an <AuthorizationResponse> message. The message identifies the SIP gateway. In particular, the <AuthorizationResponse> contains the following elements:

<Timestamp>	time of response
<Status>	result of response, e.g. <Code>200</Code>
<TransactionId>	transaction identifier assigned by settlement provider
<Destination>	first destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of destination gateway, e.g. gateway.itsp.fr
<Token>	authorization token to be passed to Gateway
<ValidAfter>	time after which token for Gateway is valid
<ValidUntil>	time until which token for Gateway is valid
<UsageDetail>	how much service is authorized with Gateway
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	SIP Call-ID to be used for the call to the Gateway

The Proxy Server, acting on behalf of its client, sends an SIP INVITE to the Gateway. In this example, the Gateway accepts the connection with an 200, and the proxy responds with an ACK. This INVITE message also uses the SIP Call-ID from the <AuthorizationRequest>, and the INVITE shall also include the authorization token(s) provided in the OSP <AuthorizationResponse>. Each token should be conveyed in the SIP message body using the application/osp-token MIME type, as initially defined in draft-thomas-mime-osp-token-00.txt.

The Proxy Server, on establishing the connection with the Gateway, completes the connection with its client as well with a SIP 200/ACK exchange.

The Gateway, meanwhile, completes the call to the destination phone number.

J.1.2.2.2 Usage Reports

At the conclusion of the phone call, both the Proxy Server and Gateway report usage information to the OSP server. The following figure identifies the principle steps of a typical call completion.

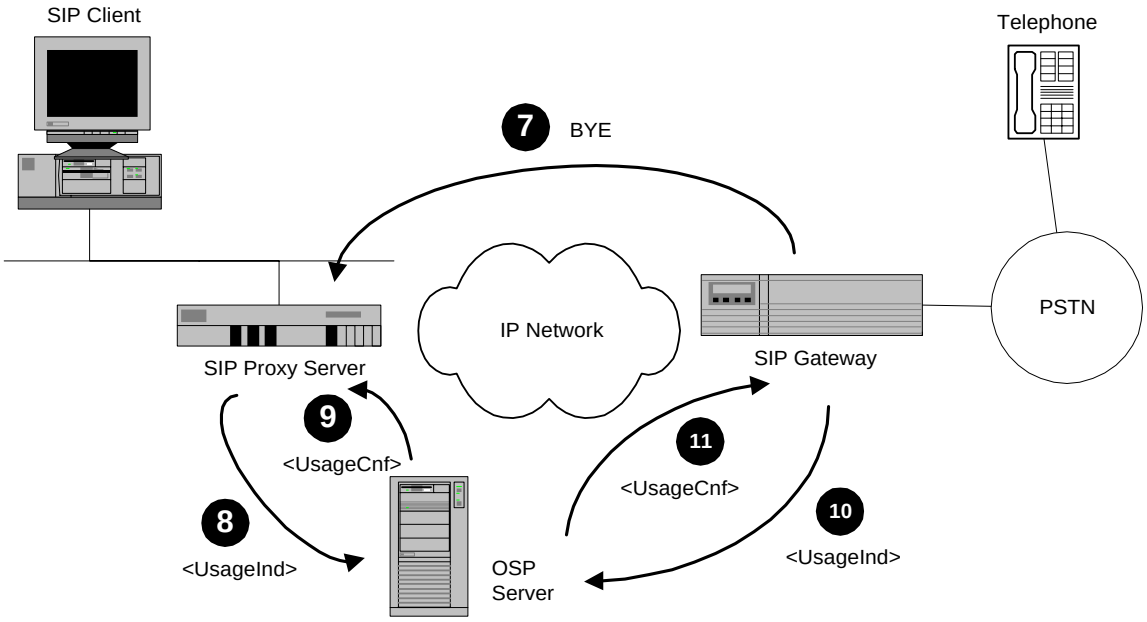


Figure J.8

The steps shown in the figure are straightforward.

The Proxy Server and Gateway release the call by transferring a SIP BYE message.

The Proxy Server then sends a `<UsageIndication>` message to the OSP server. In this example, the Proxy Server does not support new protocol extensions such as statistics. Its message, therefore contains just the following elements:

`<Timestamp>`	time of request
`<Role>`	for Proxy Server, `source`
`<TransactionId>`	transaction ID assigned by OSP server in authorization response
`<CallId>`	SIP Call-ID used for the call
`<SourceInfo type="e164">`	calling party's E.164 [12] number as returned in the authorization response, e.g. `14048724887`
`<SourceAlternate type="transport">`	DNS name or IP address of the Proxy Server, for example `proxy.carrier.com`
`<DestinationInfo type="e164">`	called party's E.164 [12] number, e.g. `33492944299`
`<DestinationAlternate type="transport">`	DNS name or IP address of the Gateway, for example, `gateway.itsp.fr`
`<UsageDetail>`	usage information for the call
`<Service/>`	empty (for basic service)
`<Amount>`	amount of service used, e.g. `300`
`<Increment>`	increment of service measurement, e.g. `1`
`<Unit>`	unit of service measurement, e.g. `s` for seconds

The OSP server responds with a `<UsageConfirmation>` message. If it has accepted the usage report, that message will contain a successful `<Status>` element (e.g. `<Code>200</Code>`).

The Gateway also sends a <UsageIndication> to the OSP server. That message would include the following elements:

<Timestamp>	time of request
<Role>	for the Gateway, destination
<TransactionId>	transaction ID assigned by OSP server and passed to the gateway in authorization token
<CallId>	SIP Call-ID used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number as presented in the INVITE message, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of the Proxy Server, for example [172.16.100.1]
<DestinationInfo type="e164">	called party's E.164 [12] number, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of the Gateway, for example, gateway.itsp.fr
<UsageDetail>	usage information for the call
<Service/>	empty (for basic service)
<Amount>	amount of service used, e.g. 300
<Increment>	Increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<Statistics>	statistical information for call
<LossSent>	loss information for packets sent by the Gateway
<Packets>	number of packets lost from the Gateway to the Proxy Server
<Fraction>	fraction (from 0 to 255) of packets lost from the Gateway to the Proxy Server
<LossReceived>	loss information for packets sent by the Proxy Server
<Packets>	number of packets lost from Proxy Server to the Gateway
<Fraction>	fraction (from 0 to 255) of packets lost from Proxy Server to Gateway
<OneWayDelay>	one way delay measured from Proxy Server to Gateway
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements
<RoundTripDelay>	round trip delay between Proxy Server and Gateway measured during call
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements

The OSP server responds with a <UsageConfirmation> message. If it has accepted the usage report, that message will contain a successful <Status> element (e.g. <Code>200</Code>).

J.1.3 Loosely Controlled Distributed Architecture

The Open Settlement Protocol can also be used in environments based on loosely-coupled, distributed architectures. One such environment is an H.323 [13]-based architecture with direct call signalling. Another example is the use Session Initiation Protocol (SIP) redirect servers. This subsection describes the use of OSP in each of these architectures.

J.1.3.1 H.323 Direct Routed Calls (with Gatekeepers)

When H.323 [13] gateways and gatekeepers both implement the Open Settlement Protocol, it is possible to use OSP in an architecture that relies on H.323 [13] Direct Call Signalling (as opposed to Gatekeeper Call Signalling). Such an approach is a hybrid of the peer-to-peer gateway architecture and the tightly controlled gatekeeper model discussed in the previous two subsections. In this approach, the gatekeepers acts as agents for call routing and authorization, but the gateways themselves are responsible for establishing and disconnecting calls directly with each other. In addition to gateways, H.323 [13] proxy devices may also be used to support this model on behalf of H.323 [13] terminals.

J.1.3.1.1 Call Routing and Authorization

The following figure shows a sample routing and authorization scenario for this model.

NOTE 1: This figure shows a finer level of detail than previous examples in order to clarify several subtle points.

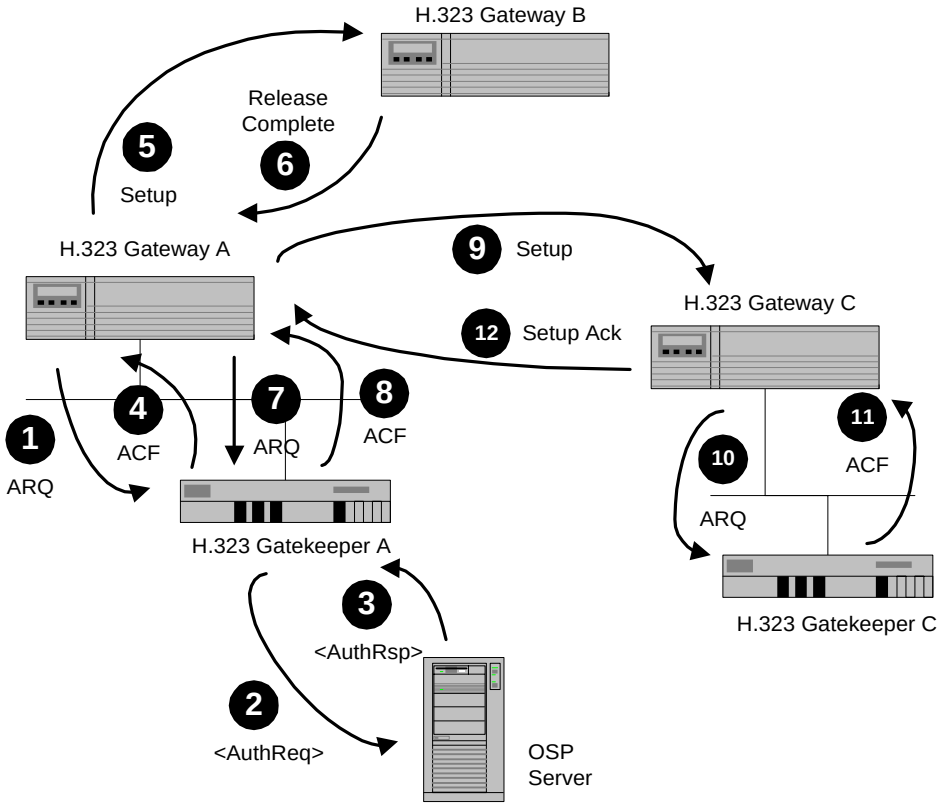


Figure J.9

Gateway A begins a call by sending an H.225.0 [9] Admission Request (ARQ) to its gatekeeper, Gatekeeper A. The ARQ indicates that the called party is identified by an E.164 [12] phone number such as +33 4 92 94 42 99.

Gatekeeper A sends an OSP `<AuthorizationRequest>` message to the OSP Server. The significant elements within the `<AuthorizationRequest>` include:

`<Timestamp>`	time of request
`<CallId>`	H.323 [13] Call Identifier to be used for the call
`<SourceInfo type="e164">`	the called party's E.164 [12] number, or, if that is not available, a local E.164 [12] number belonging to Gateway A; this number shall be passed to the destination gateway(s) in Setup messages
`<SourceAlternate type="transpo`	DNS name or IP address of Gatekeeper A, for example `gatekeeperA.carrier.com`

rt">	
<SourceAlternate type="h323">	H.323 [13] identifier of Gateway A, e.g. 12345678
<DestinationInfo type="e164">	called party's E.164 [12] number, e.g. 33492944299
<Service/>	empty (for basic service)
<MaximumDestinations>	the maximum number of destinations, including alternatives, Gatekeeper A will consider

OSP Server replies with an <AuthorizationResponse> message. The message identifies Gateway B and Gateway C as candidate destinations. In particular, the <AuthorizationResponse> contains the following elements:

<Timestamp>	time of response
<Status>	result of response, e.g. <Code>200</Code>
<TransactionId>	transaction identifier assigned by settlement provider
<Destination>	first destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of Gateway B, for example gatewayB.itsp.fr
<Token>	authorization token to be passed to Gateway B
<ValidAfter>	time after which token for Gateway B is valid
<ValidUntil>	time until which token for Gateway B is valid
<UsageDetail>	how much service is authorized with Gateway B
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	H.323 [13] Call Identifier to be used for the call to Gateway B
<Destination>	second destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of destination Gateway C, e.g. gatewayC.isp.fr
<Token>	authorization token to be passed to Gateway C
<ValidAfter>	time after which token for Gateway C is valid
<ValidUntil>	time until which token for Gateway C is valid
<UsageDetail>	how much service is authorized with Gateway C
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	H.323 [13] Call Identifier to be used for the call to Gateway C

Gatekeeper A sends an H.225.0 [9] Admission Confirm (ACF) message to its gateway. That ACF identifies the destination as Gateway B and it shall include the authorization token for gateway B.

Gateway A sends an H.225.0 [9] Setup message to Gateway B. This Setup message uses the H.323 [13] Call Identifier from the <AuthorizationRequest>, and the message shall include the authorization token(s) provided in the OSP <AuthorizationResponse>.

Gateway B refuses the setup attempt with a Release Complete message, perhaps, for example, because no outbound PSTN ports are available.

Gateway A sends another Admission Request (ARQ) message to its gatekeeper to request an alternate destination for the phone call.

NOTE 2: This ARQ shall use the same H.323 [13] Call Identifier as the original ARQ. For clarity, the example omits other details of the exchange between Gateway A and Gatekeeper A, such as, for example, a Disengage Request (DRQ) and Disengage Confirm (DCF).

Gatekeeper A replies with an Admission Confirm (ACF) message that identifies Gateway C as a potential destination.

NOTE 3: The Gatekeeper need not query the OSP server to get that information. Rather, the information is available in the original `<AuthorizationResponse>` noted in step 3.

Gateway A tries a second setup attempt, this time by sending a Setup message to Gateway C.

Gateway C receives the Setup message and asks its gatekeeper for permission to accept the call. It does so by sending an Admission Request (ARQ) to Gatekeeper C.

NOTE 4: This message shall include the authorization token(s) received in the Setup.

Gatekeeper C authorizes the call and returns an Admission Confirm (ACF) message to Gateway C.

Gateway C receives the ACF and accepts the call by returning a Setup Acknowledge message to Gateway A.

J.1.3.1.2 Usage Reports

At the conclusion of the phone call, both gateways shall report usage information to the OSP server. The following figure identifies the principle steps of a typical call completion.

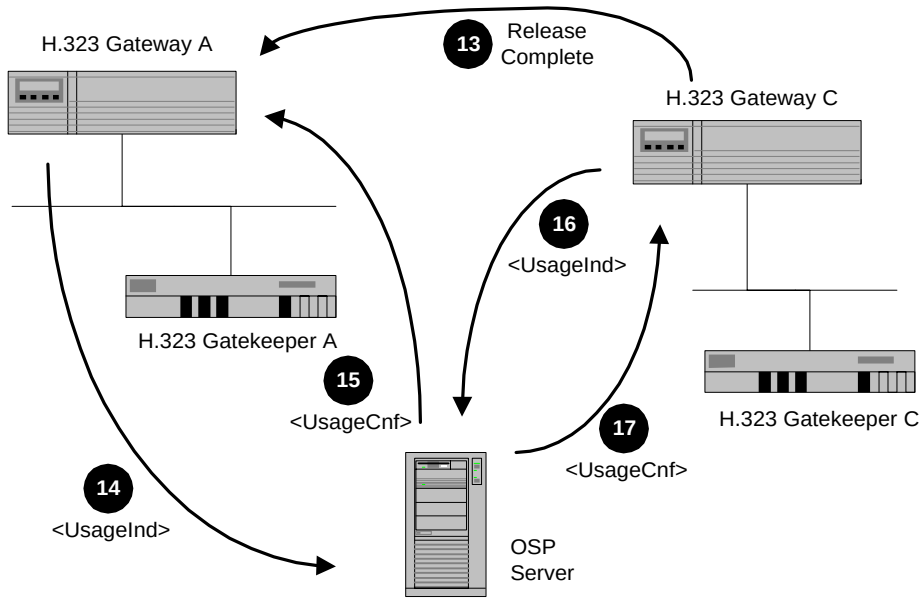


Figure J.10

The steps shown in the figure are straightforward.

The gateways close the connection by exchanging H.323 [13] Release and Release Complete messages.

Gateway A then sends a <UsageIndication> message to the OSP server. In this example Gateway A's message will include two complete <UsageIndication> components, one for the failed attempt and one for the successful call. The sub-elements for each will include the following:

<UsageIndication>	usage information for the failed setup attempt
<Timestamp>	time of request
<Role>	for Gateway A, source
<TransactionId>	transaction ID assigned by OSP server in authorization response
<CallId>	H.323 [13] Call Identifier used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of Gateway A, for example gatewayA.carrier.com
<DestinationInfo type="e164">	called party's E.164 [12] number as returned in the authorization response, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of Gateway B, for example, gatewayB.itsp.fr
<FailureReason>	reason for failure of attempted setup, e.g. 422
<UsageIndication>	usage information for the successful setup attempt
<Timestamp>	time of request
<Role>	for Gateway A, source
<TransactionId>	transaction ID assigned by OSP server in authorization response
<CallId>	H.323 [13] Call Identifier used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of Gateway A, for example gatekeeperA.carrier.com
<DestinationInfo type="e164">	called party's E.164 [12] number as returned in the authorization response, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of Gateway C, for example, gatewayC.isp.fr
<UsageDetail>	usage information for the call
<Service/>	empty (for basic service)
<Amount>	amount of service used, e.g. 300
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds

The OSP server responds with a <UsageConfirmation> message. If it has accepted the usage report, that message will contain a successful <Status> element (e.g. <Code>200</Code>).

Gateway C also sends a <UsageIndication> to the OSP server. That message would include the following elements:

<Timestamp>	time of request
<Role>	for Gateway C, destination
<TransactionId>	Transaction ID assigned by OSP server and passed to Gateway C in authorization token
<CallId>	H.323 [13] Call Identifier used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number as presented in the Setup message, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of Gateway A, for example [172.16.100.1]
<DestinationInfo type="e164">	called party's E.164 [12] number, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of Gateway C, for example, gatewayC.itsp.fr
<UsageDetail>	usage information for the call
<Service/>	empty (for basic service)
<Amount>	amount of service used, e.g. 300
<Increment>	Increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<Statistics>	statistical information for call
<LossSent>	loss information for packets sent by Gateway C
<Packets>	number of packets lost from Gateway C to Gateway A
<Fraction>	fraction (from 0 to 255) of packets lost from C to A
<LossReceived>	loss information for packets sent by Gateway A
<Packets>	number of packets lost from Gateway A to Gateway C
<Fraction>	fraction (from 0 to 255) of packets lost from A to C
<OneWayDelay>	one way delay measured from A to C
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements
<RoundTripDelay>	round trip delay between A and C measured during call
<Minimum>	minimum measured value for delay, in seconds
<Mean>	sample mean of delay measurements, in seconds
<Variance>	sample variance of delay measurements, in squared seconds
<Samples>	number of sample measurements

The OSP server responds with a <UsageConfirmation> message. If it has accepted the usage report, that message will contain a successful <Status> element (e.g. <Code>200</Code>).

J.1.3.2 Session Initiation Protocol Redirect Servers

Session Initiation Protocol (SIP) redirect servers represent another example of the loosely coupled distributed architecture, particularly on the source side of a call. In this environment, the redirect server provides the call routing and authorization on behalf of the systems it serves, but those systems themselves report usage information.

J.1.3.2.1 Call Routing and Authorization

The following figure shows a sample routing and authorization scenario for SIP redirect servers.

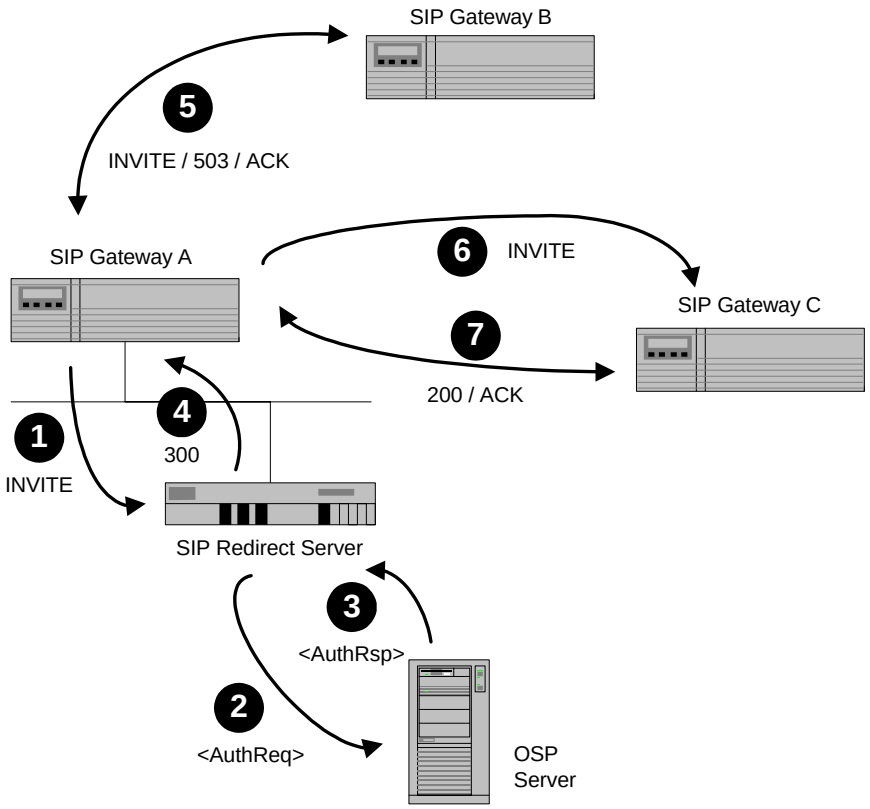


Figure J.11

Gateway A begins a call by sending a SIP INVITE method to its redirect server. The INVITE indicates that the called party is identified by an E.164 [12] phone number such as +33 4 92 94 42 99.

The Redirect Server sends an OSP `<AuthorizationRequest>` message to the OSP Server. The significant elements within the `<AuthorizationRequest>` include:

`<Timestamp>`	time of request
`<CallId>`	SIP Call-ID to be used for the call
`<SourceInfo type="e164">`	calling party's E.164 [12] number if available; otherwise a local E.164 [12] number controlled by Gateway A (e.g. `14048724887`); this number shall be passed to the destination gateway(s) in future INVITE messages
`<SourceAlternate type="transport">`	DNS name or IP address of Redirect Server, for example `redirect.carrier.com`
`<SourceAlternate type="h323">`	DNS name or IP address of Gateway A, for example `gatewayA.carrier.com`; the type `"h323"`, in this case, implies a device-specific ID rather than a particular protocol
`<DestinationInfo type="e164">`	called party's E.164 [12] number, e.g. `33492944299`
`<Service/>`	empty (for basic service)
`<MaximumDestinations>`	the maximum number of destinations, including alternatives, Redirect Server will consider

OSP Server replies with an <AuthorizationResponse> message. The message identifies Gateway B and Gateway C as candidate destinations. In particular, the <AuthorizationResponse> contains the following elements:

<Timestamp>	time of response
<Status>	result of response, e.g. <Code>200</Code>
<TransactionId>	transaction identifier assigned by settlement provider
<Destination>	first destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of Gateway B, for example gatewayB.itsp.fr
<Token>	authorization token to be passed to Gateway B
<ValidAfter>	time after which token for Gateway B is valid
<ValidUntil>	time until which token for Gateway B is valid
<UsageDetail>	how much service is authorized with Gateway B
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	SIP Call-ID to be used for the call to Gateway B
<Destination>	second destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of Gateway C, for example gatewayC.isp.fr
<Token>	authorization token to be passed to Gateway C
<ValidAfter>	time after which token for Gateway C is valid
<ValidUntil>	time until which token for Gateway C is valid
<UsageDetail>	how much service is authorized with Gateway C
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	SIP Call-ID to be used for the call to Gateway C

The Redirect Server returns a "300 Multiple Choices" redirection status code to Gateway A. This response relays the data from the <AuthorizationResponse> to the gateway, explicitly identifying Gateways B and C as candidate destinations. This response shall also include the authorization tokens returned by the OSP Server as application/osp-token MIME types in the response body.

Gateway A sends an INVITE message to Gateway B. This INVITE message uses the SIP Call-ID from the <AuthorizationRequest>, and the message shall include all authorization tokens associated with that destination in the server's OSP <AuthorizationResponse>.

Gateway B refuses the setup attempt with a 503 message, perhaps, for example, because no outbound PSTN ports are available.

Gateway A tries a second setup attempt, this time by sending an INVITE message to Gateway C.

Gateway C receives the INVITE message and accepts the call by responding with a 200 message, to which Gateway A replies with an ACK.

J.1.3.2.2 Usage Reports

Once the call has ended, both gateways report usage details to an OSP server. As the following figure indicates, those reports are conveyed in OSP `<UsageIndication>` messages.

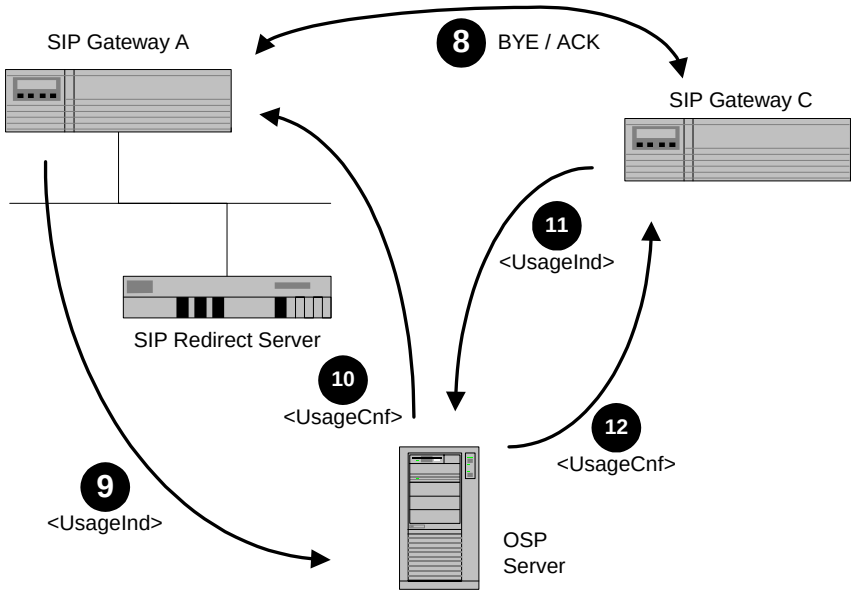


Figure J.12

The steps shown in the figure are straightforward.

Gateways A and C clear the call with a SIP BYE message.

Gateway A sends a <UsageIndication> message to the OSP server. In this example Gateway A's message will include two complete <UsageIndication> components, one for the failed attempt and one for the successful call. The sub-elements for each will include the following:

<UsageIndication>	usage information for the failed setup attempt
<Timestamp>	time of request
<Role>	for Gateway A, source
<TransactionId>	transaction ID assigned by OSP server in authorization response
<CallId>	SIP Call-ID used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of Gateway A, for example gatewayA.carrier.com
<DestinationInfo type="e164">	called party's E.164 [12] number as returned in the authorization response, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of Gateway B, for example, gatewayB.itsp.fr
<FailureReason>	reason for failure of attempted setup, e.g. 422
<UsageIndication>	usage information for the successful setup attempt
<Timestamp>	time of request
<Role>	for Gateway A, source
<TransactionId>	transaction ID assigned by OSP server in authorization response
<CallId>	H.323 [13] Call Identifier used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of Gateway A, for example gatewayA.carrier.com
<DestinationInfo type="e164">	called party's E.164 [12] number as returned in the authorization response, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of Gateway C, for example, gatewayC.isp.fr
<UsageDetail>	usage information for the call
<Service/>	empty (for basic service)
<Amount>	amount of service used, e.g. 300
<Increment>	increment of service measurement, e.g. 1
<Unit>	Unit of service measurement, e.g. s for seconds

The OSP server responds with a <UsageConfirmation> message. If it has accepted the usage report, that message will contain a successful <Status> element (e.g. <Code>200</Code>).

Gateway C also sends a <UsageIndication> to the OSP server. That message would include the following elements:

<Timestamp>	time of request
<Role>	For Gateway C, destination
<TransactionId>	transaction ID assigned by OSP server and passed to Gateway C in authorization token
<CallId>	SIP Call-ID used for the call
<SourceInfo type="e164">	calling party's E.164 [12] number as presented in the INVITE message, e.g. 14048724887
<SourceAlternate type="transport">	DNS name or IP address of Gateway A, for example [172.16.1.1]
<DestinationInfo type="e164">	called party's E.164 [12] number, e.g. 33492944299

`<DestinationAlternate type="transport">`	DNS name or IP address of Gateway C, for example, `gatewayC.isp.fr`
`<UsageDetail>`	usage information for the call
`<Service/>`	empty (for basic service)
`<Amount>`	amount of service used, e.g. `300`
`<Increment>`	increment of service measurement, e.g. `1`
`<Unit>`	unit of service measurement, e.g. `s` for seconds
`<Statistics>`	statistical information for call
`<LossSent>`	loss information for packets sent by Gateway C
`<Packets>`	number of packets lost from Gateway C to Gateway A
`<Fraction>`	fraction (from 0 to 255) of packets lost from C to A
`<LossReceived>`	loss information for packets sent by Gateway A
`<Packets>`	number of packets lost from Gateway A to Gateway C
`<Fraction>`	fraction (from 0 to 255) of packets lost from A to C
`<OneWayDelay>`	one way delay measured from Gateway A to C
`<Minimum>`	minimum measured value for delay, in seconds
`<Mean>`	sample mean of delay measurements, in seconds
`<Variance>`	sample variance of delay measurements, in squared seconds
`<Samples>`	number of sample measurements
`<RoundTripDelay>`	round trip delay between Gateway A and C measured during call
`<Minimum>`	minimum measured value for delay, in seconds
`<Mean>`	sample mean of delay measurements, in seconds
`<Variance>`	sample variance of delay measurements, in squared seconds
`<Samples>`	number of sample measurements

The OSP server responds with a `<UsageConfirmation>` message. If it has accepted the usage report, that message will contain a successful `<Status>` element (e.g. `<Code>200</Code>`).

J.2 Prepaid Calling Card and Roaming User Support

As the following figure shows, the most general environment requires support from four different domains: the source and destination domains of the IP telephony gateways, the settlement service provider, and the end user billing domain. User billing may distinct from the source domain in the case, for example, or a roaming user.

The figure also shows the general operational procedure for authorization divided into seven discrete steps.

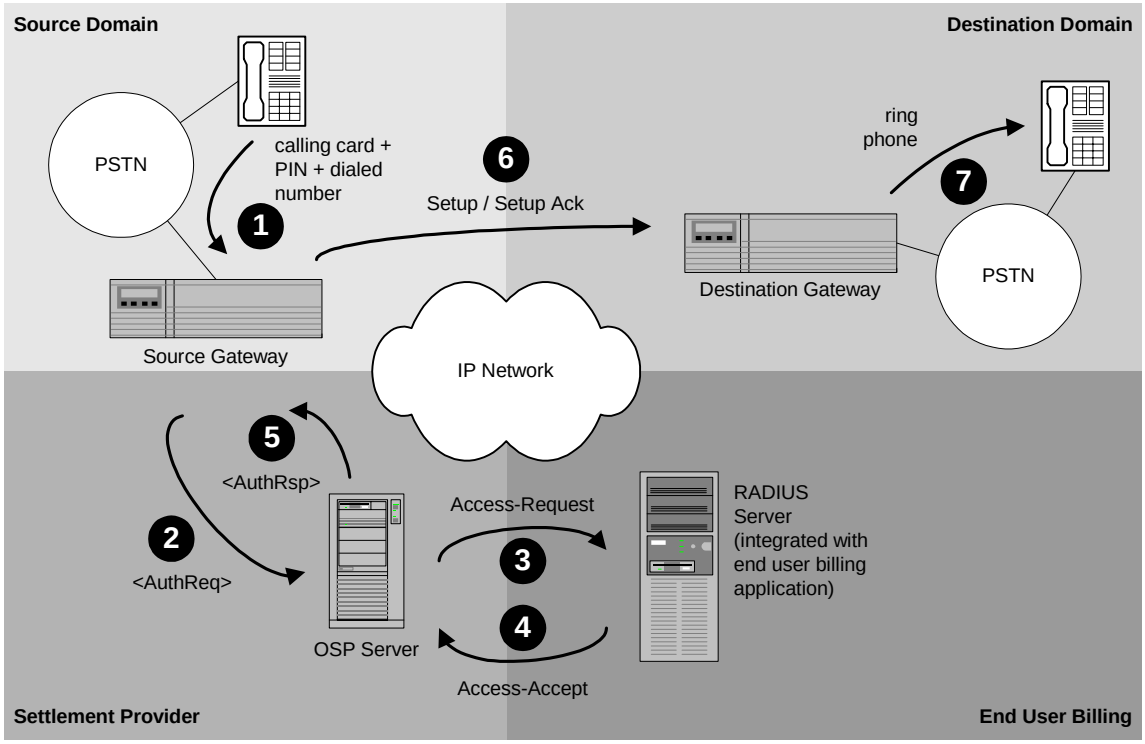


Figure J.13

The user accesses a gateway in the source domain. The gateway, perhaps using an IVR application, collects the user's calling card number and personal identification number, in addition to the called number.

The source gateway forwards this information to the settlement provider in an OSP `<AuthorizationRequest>`.

The settlement provider, in addition to authenticating the source gateway, also authenticates the end user. That authentication procedure is outside the scope of OSP, but may, as the example shows, rely on another standard protocol such as RADIUS (RFC 2138).

The end user billing application authenticates and authorizes the user.

The settlement provider returns an `<AuthorizationResponse>` to the source gateway indicating acceptance of the end user and providing authorization tokens for the destination gateway.

The call proceeds normally with a Setup message from the source to the destination gateway.

The destination gateway completes the call to the called party.

These steps represent the beginning of a calling card transaction. Additional phases, including refreshing the user's authorization and reporting usage information, can be found in the following section on implementation details.

J.2.1 Call Routing and Authorization

The OSP <AuthorizationRequest> message shown in step 2 contains the following significant elements:

<Timestamp>	Time of request
<CallId>	H.323 [13] Call Identifier to be used for the call
<SourceInfo type="e164">	Calling party's E.164 [12] number if available; otherwise a local E.164 [12] number controlled by the source gateway, e.g. 14048724887; this number shall be passed to the destination gateway(s) in Setup messages
<SourceAlternate type="transport">	DNS name or IP address of source gateway, for example sourcegateway.carrier.com
<SourceAlternate type="subscriber">	The user's calling card and PIN; following conventions established by the Voice over IP Forum, these should be combined into a single character string, with the two components separated by the pound sign (#). For example, the calling card number 12345678, combined with the PIN 4444, should be represented as "12345678#4444".
<DestinationInfo type="e164">	Called party's E.164 [12] number, e.g. 33492944299
<Service/>	Empty (for basic service)
<MaximumDestinations>	The maximum number of destinations, including alternatives, the source gateway will consider

The OSP server replies with an <AuthorizationResponse> message as the figure indicates in step 5. The message indicates candidate destination gateways, in order of priority. In this example (which only shows a single destination gateway, the OSP <AuthorizationResponse> contains the following elements:

<Timestamp>	time of response
<Status>	result of response, e.g. <Code>200</Code>
<TransactionId>	transaction identifier assigned by settlement provider
<Destination>	first destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of destination gateway, for example destgateway.itsp.fr
<Token>	authorization token to be passed to destination gateway
<ValidAfter>	time after which token for destination gateway is valid
<ValidUntil>	time until which token for destination gateway is valid
<UsageDetail>	how much service is authorized with destination gateway
<Service/>	empty (for basic service)
<Amount>	amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<CallId>	H.323 [13] Call Identifier to be used for the call to destination gateway

J.2.2 Reauthorization

In many calling card services, it may be necessary to refresh the authorization of a call that is already in progress. For example, some debit card applications will only authorize a limited amount of service at any given time; this can minimize the risk of fraudulent, simultaneous use of the debit card by multiple users. In such scenarios, and when the users wish to continue their conversation past the limited amount of time initially authorized, it will be necessary for the supporting devices to request additional authorization for the call. The following figure shows the message flow for this process. The six steps in the figure begin when the originating gateway recognizes that the currently authorized service limit is approaching.

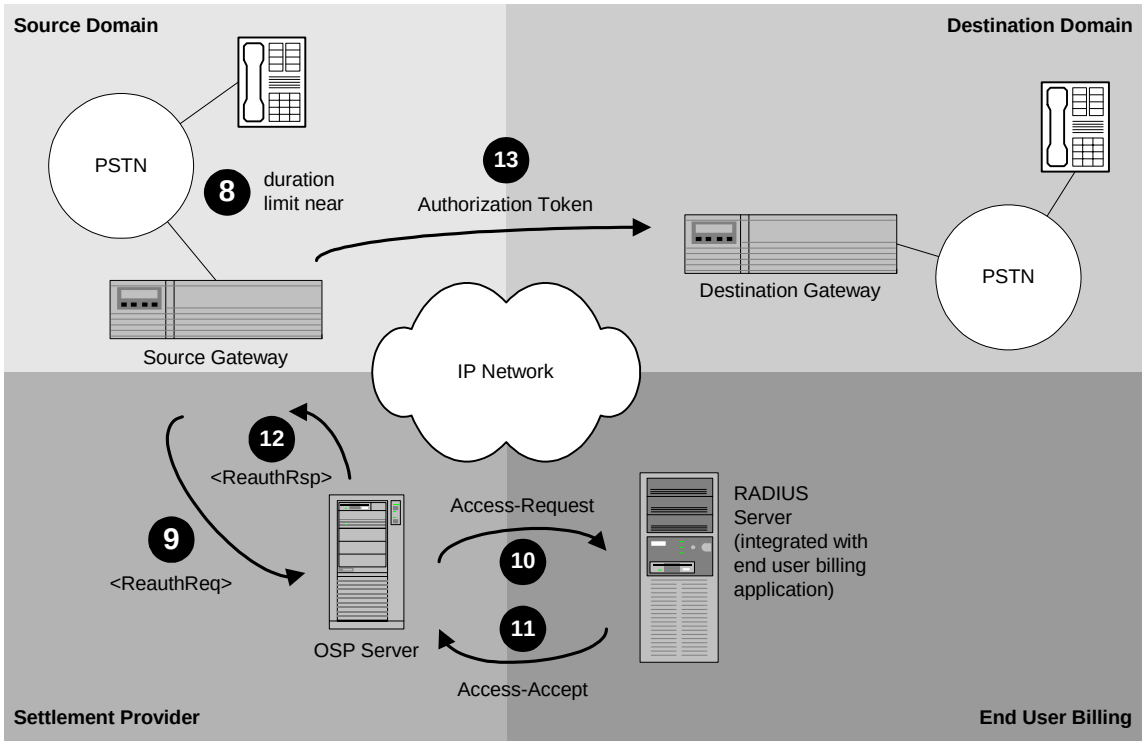


Figure J.14

Source gateway recognizes that the duration currently authorized for the call is nearing its limit.

Source gateway sends an OSP `<ReauthorizationRequest>` to the OSP server.

The OSP server uses some other means (such as RADIUS, in the example) to authorize additional service for the user.

The OSP server confirms that additional service is authorized.

The OSP server returns a `<ReauthorizationResponse>` to the source gateway, granting the additional authorized service. The response tells the source gateway the new authorization limits explicitly, and it includes an updated authorization token.

The source gateway passes the new authorization token to the destination gateway. The exact method of transfer depends on the gateway implementations and the particular call signalling protocol; as an example, the source gateway may include the token in an H.323 [13] Facility message.

The OSP <ReauthorizationRequest> message shown in step 9 contains the following significant elements:

<Timestamp>	Time of request
<Role>	for the source gateway, source
<CallId>	H.323 [13] Call Identifier used for the call
<SourceInfo type="e164">	Calling party's E.164 [12] number if available; otherwise a local E.164 [12] number controlled by the source gateway, e.g. 14048724887; this number shall be passed to the destination gateway(s) in Setup messages
<SourceAlternate type="transport">	DNS name or IP address of source gateway, for example sourcegateway.carrier.com
<SourceAlternate type="subscriber">	The user's calling card and PIN; following conventions established by the Voice over IP Forum, these should be combined into a single character string, with the two components separated by the pound sign (#). For example, the calling card number 12345678, combined with the PIN 4444, should be represented as "12345678#4444".
<DestinationInfo type="e164">	Called party's E.164 [12] number, e.g. 33492944299
<DestinationAlternate type="transport">	DNS name or IP address of destination gateway, for example, destgateway.itsp.fr
<TransactionId>	transaction identifier assigned by settlement provider
<UsageDetail>	usage information for the call so far
<Service/>	empty (for basic service)
<Amount>	amount of service used so far, e.g. 300
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds
<Token>	authorization token to be passed to destination gateway

The OSP server returns that information within an <ReauthorizationResponse> message, as the figure indicates in step 12. The message refreshes the authorization information for the call. In this example, the OSP <AuthorizationResponse> contains the following elements:

<Timestamp>	time of response
<Status>	result of response, e.g. <Code>200</Code>
<TransactionId>	transaction identifier assigned by settlement provider
<Destination>	destination gateway to try for call
<DestinationSignalAddress type="transport">	DNS name or IP address of destination gateway, for example destgateway.itsp.fr
<Token>	updated authorization token to be passed to destination gateway
<ValidAfter>	time after which token for destination gateway is valid
<ValidUntil>	time until which token for destination gateway is valid
<UsageDetail>	how much (cumulative) service is authorized with destination gateway
<Service/>	empty (for basic service)
<Amount>	(cumulative) amount of authorized service, e.g. 3600
<Increment>	increment of service measurement, e.g. 1
<Unit>	unit of service measurement, e.g. s for seconds

Once the call has ended, both gateways report usage details to an OSP server. As the following figure indicates, those reports are conveyed in OSP `<UsageIndication>` messages.

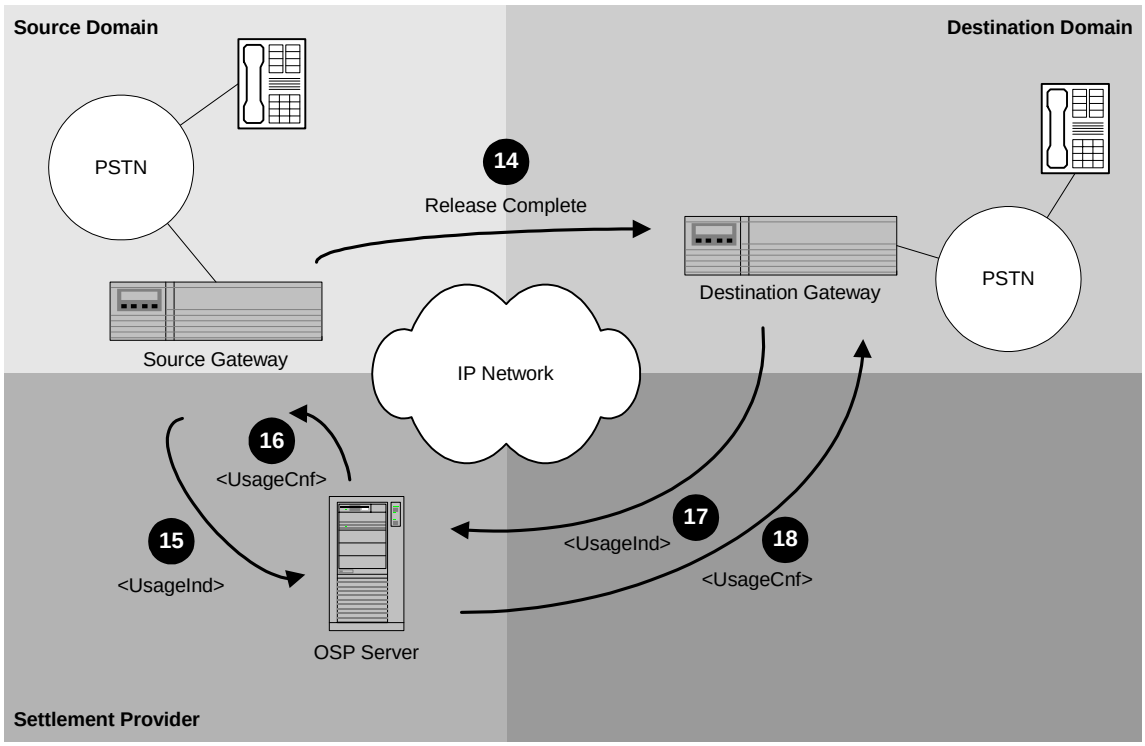


Figure J.15

The steps shown in the figure are straightforward.

The gateways clear the call by exchanging an H.225.0 [9] Release Complete message.

The source gateway sends a `<UsageIndication>` message to the OSP server, reporting its usage details for the call.

The OSP server acknowledges receipt with a `<UsageConfirmation>` message.

The destination gateway also reports its usage details with a `<UsageIndication>` message.

The server acknowledges this message as well with a `<UsageConfirmation>`.

The `<UsageIndication>` messages from both gateways will be substantially the same. As an example, here are the significant fields of the source gateway's message:

`<Timestamp>`	time of request
`<Role>`	for source gateway, `source`
`<TransactionId>`	transaction ID assigned by OSP server in authorization response
`<CallId>`	H.323 [13] Call Identifier used for the call
`<SourceInfo type="e164">`	calling party's E.164 [12] number as returned in the authorization response, e.g. `14048724887`
`<SourceAlternate type="transport">`	DNS name or IP address of source gateway, for example `sourcegateway.carrier.com`
`<DestinationInfo type="e164">`	called party's E.164 [12] number, e.g. `33492944299`
`<DestinationAlternate type="transport">`	DNS name or IP address of destination gateway, for example, `destgateway.isp.fr`
`<UsageDetail>`	usage information for the call

`<Service/>`	empty (for basic service)
`<Amount>`	amount of service used, e.g. `600`
`<Increment>`	increment of service measurement, e.g. `1`
`<Unit>`	unit of service measurement, e.g. `S` for seconds

The OSP server responds with a `<UsageConfirmation>` message. If it has accepted the usage report, that message will contain a successful `<Status>` element (e.g. `<Code>200</Code>`).

Bibliography

The following material, though not specifically referenced in the body of the present document (or not publicly available), gives supporting information.

- RFC 2138 (1997): Remote Authentication Dial In User Service (RADIUS). C. Rigney, A. Rubens, W. Simpson, S. Willens.

History

Document history		
V1.4.2	December 1998	Publication
V2.1.1	August 2000	Publication