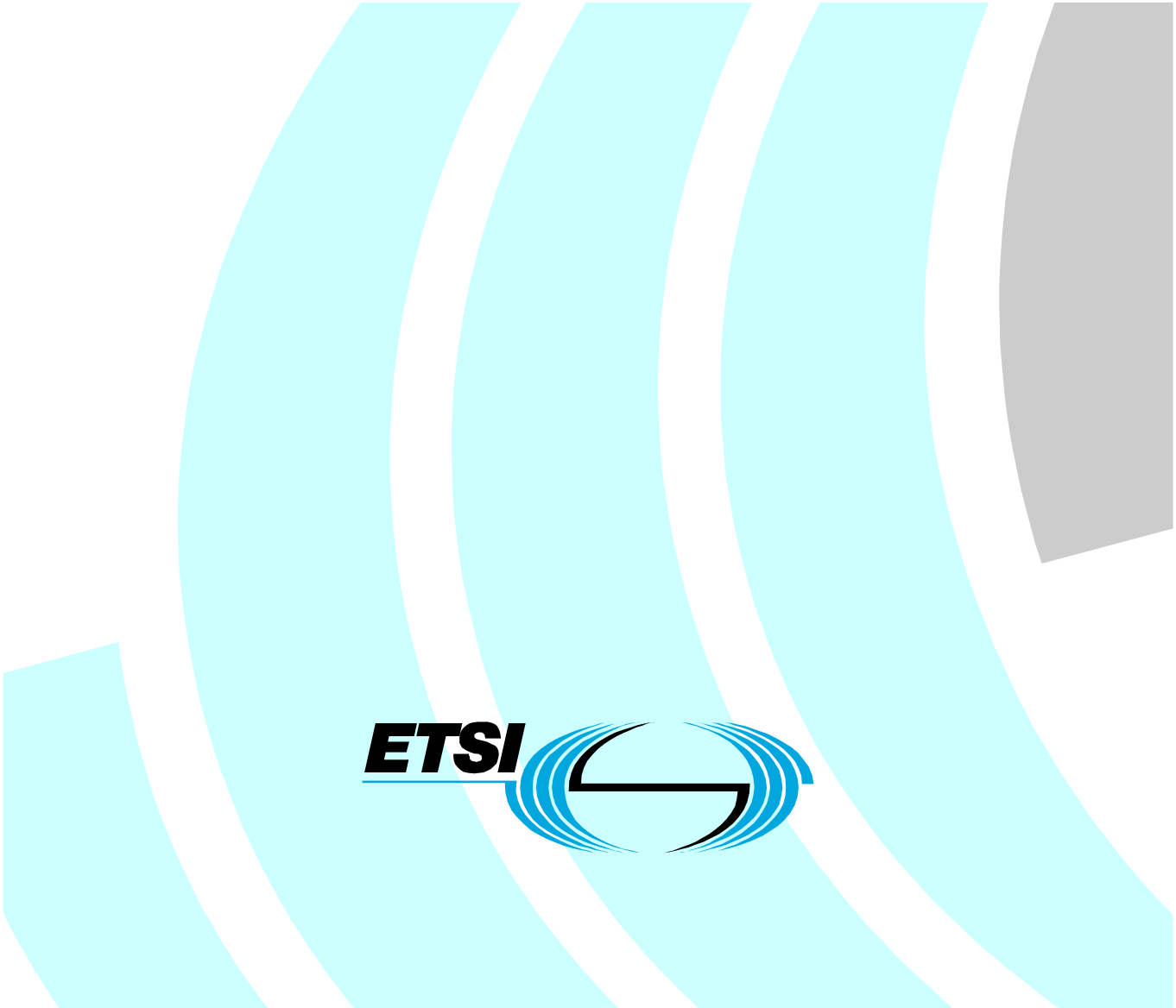
ETSI TS 102 204 V1.1.4 (2003-08)

Technical Specification

Mobile Commerce (M-COMM); Mobile Signature Service; Web Service Interface



Reference
DTS/M-COMM-004

Keywords
commerce, electronic signature, interface,
internet, m-commerce, mobile, service

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

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

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

Foreword

The present document (TS) has been produced by ETSI Project M-Commerce (M-COMM).

Introduction

Citizens around the world are making use increasingly of electronic communications facilities in their daily lives. This often involves interactions between parties who have never previously met - or may never meet - and for whom no pre-established relationship exists. Consequently, communications networks of all kinds are being exploited in new ways to conduct business, to facilitate remote working and to create other "virtual" shared environments.

Consumers, businesses and government departments alike benefit in various ways. For the European Union ("EU"), electronic commerce presents an excellent opportunity to advance its programmes for economic integration. But, such an approach requires an appropriate security mechanism to allow completion of "remote" interactions between parties with confidence. To this end, the European Parliament and Council Directive on Electronic Signatures (1999/93/EC [22]) was published on December-13th, 1999.

The definition of "electronic signature" contained in Article 2 of the Directive facilitated the recognition of data in electronic form in the same manner as a hand-written signature satisfies those requirements for paper-based data. Since electronic signatures can only be as "good" as the technology and processes used to create them, "standardization" activities such as those in Europe by ETSI and CEN within the EESSI framework aim to ensure that a common level of confidence and acceptance can be recognized. The result will be a powerful enabling facility for electronic commerce and, more generally, for completion of transactions of any kind.

In the context of the EU Directive, the present document focuses on electronic signatures created by cryptographic means in a "secure signature creation device". To date (June 2003), security provisions for signature creation and verification systems are such that parties wishing to provide a signature require "special" equipment. Typically, this involves a smartcard and a card reader with sufficient processing power and display capabilities to present full details of the transaction to be "signed". For consumer markets, however, it is doubtful whether individual citizens will want to invest in such equipment, which for the most part may remain connected to (or inserted into) personal computer equipment located in the home.

An alternative approach is to capitalize on the fact that many citizens already possess a device which contains a smartcard and which itself is effectively a personal card reader - their mobile phone. In some European countries, mobile penetration rates are approaching 80 % of the population. As one of the most widely-owned electronic devices, the mobile phone represents the natural choice for implementation of a socially-inclusive, electronic signature solution for the majority of citizens.

Electronic signatures created in this way have become known as "Mobile Signatures" and a number of initiatives are already underway to evaluate the feasibility of such an approach. Only a small number of these have so far been implemented commercially and none have yet been extended to a mass-market scale. Many of those engaged in such activity cite "interoperability" issues as a restraining factor, requiring standardization to avoid market fragmentation.

The concept of a "Mobile Signature" is attractive because it leverages existing commercial models, network infrastructure, mobile device technology (including the SIM-infrastructure) and customer relationships managed by GSM mobile network operators. This offers the prospect that the concept could be adopted by around one billion mobile phone users in 179 countries, world-wide. Extension of the concept to other mobile network technologies is also possible.

Adoption of mobile signature might also assist in the fight against international crimes, such as money "laundering". In this case, the opportunity provided by mobile signature to identify the citizens who are party to a transaction is attractive, subject to provisions concerning Data Protection, Privacy and Legal Interception (as applied to data services).

Acceptance of the concept universally now requires "standardization" of a common service methodology, where signature requests/responses can be issued/received in a "standard" format - irrespective of mobile device characteristics. To this end, the European Commission allocated funds to ETSI to establish a Specialist Task Force (STF-221) to produce a set of deliverables on "Mobile Signature Service".

It is envisaged that mobile signature services will play a pivotal role in reaching an appropriate level of confidence, acceptance and interoperability to support implementation of the European Directive on Electronic Signature - particularly for consumer (mass) markets. This Technical Report focuses on those technologies able to realize a mobile signature the equivalent of an "enhanced electronic signature" as defined by the European Directive.

The mobile signature service is considered suitable for the administration and management of all aspects relating to:

- Advising and guiding citizens about the use of mobile signature
- Acquiring mobile signature capability
- Managing citizen identity (including Data protection and individual privacy)
- Processing of signature requests from application providers (and providing responses)
- Maintaining signature transaction records for the citizen.
- Managing all aspects of signature lifecycle (e.g. validity, expiry)
- Supporting service administration and maintenance activities

The definition of the Mobile Signature Service comprises the following report and specifications:

- TR 102 203 [12]:

  "Mobile Commerce (M-COMM); Mobile Signature; Business & Functional Requirements"

- TS 102 204:

  "Mobile Commerce (M-COMM); Mobile Signature;Web Service Interface".

- TR 102 206 [13]:

  "Mobile Commerce (M-COMM); Mobile Signature Service; Security Framework".

- TS 102 207 [14]:

  "Mobile Commerce (M-COMM); Mobile Signature Service; Specifications for Roaming in Mobile Signature Services".

Together, the Technical Reports (TRs) and the Technical Specifications (TSs) allow the design and implementation of interoperable mobile signature solutions.

1 Scope

The present document specifies the Mobile Signature Service as a Web Service: MOBILE SIGNATURE WEB SERVICE.

From the business and functional requirements of TR 102 203 [12], the present document identifies the methods that must be provided by a Mobile Signature Web Service Provider.

The present document specifies the data structures and messaging models related to these methods thanks to XML Schema and WSDL. Documentations about these technologies can be found in clause 2. The complete MSS XML Schema is provided in Annex A.

A SOAP 1.2 binding is proposed as the mandatory protocol binding. The corresponding WSDL 1.1 description document of such a web service is specified.

In defining the Web service, the present document makes reference to interactions between different parties and to the end user experience of a mobile signature service at the mobile device. This is done to illustrate concepts and facilitate definition of the Web service - only. Readers are referred to other sources of information as indicated in clause 2 regarding definitions and specifications for these topics.

Structure of the present document:

- **Scope:** a description of the goals and objectives of the present document.
- **Document Administration:** an explanation of the structure, definitions, symbols and abbreviations used in the present document.
- **Introduction to mobile signature:** positions the Mobile Signature project and EC funding etc leading to overview of why mobile signature has a way to accelerate deployment of electronic signatures as originally envisaged by the EU Directive.
- **Mobile Signature Service Functions:** this section describes the high-level functionalities provided by a Mobile Signature Service Provider.
- **Mobile Signature Web Service:** the Mobile Signature Service is specified as a Web Service in this section.
- **Message Formats:** the XML messages exchanged between an Application Provider and a Mobile Signature Service Provider are presented.
- **Auxiliary XML Data Types:** the messages presented in the previous chapter are based upon the XML data types specified here.
- **Communication Protocol Binding:** the protocol binding for the Mobile Signature Service is specified as SOAP 1.2 over HTTP.
- **Web Service - Security and Privacy Considerations:** Security and Privacy considerations with respect to the Mobile Signature Service are presented.

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 and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] IETF RFC 2045: "Multipurpose Internet Mail Extensions (MIME); Part One: Format of Internet Message Bodies".

[2] IETF RFC 2046: "Multipurpose Internet Mail Extensions (MIME); Part Two: Media Types".

[3] IETF RFC 2119: "Key words for use in RFCs to Indicate Requirement Levels".

[4] IETF RFC 2246: "The TLS protocol Version 1.0".

[5] IETF RFC 2396: "Uniform Resource Identifier (URI): Generic Syntax".

[6] IETF RFC 2630: "Cryptographic Message Syntax".

[7] IETF RFC 3275: "(Extensible Markup Language) XML-Signature Syntax and Processing".

[8] SOAP 1.2: "SOAP Version 1.2 Part 0: Primer", W3C Recommendation 24 June 2003, http://www.w3.org/TR/soap12-part0/

[9] SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation 24 June 2003, http://www.w3.org/TR/soap12-part1/

[10] "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation 24 June 2003, http://www.w3.org/TR/soap12-part2/

[11] "SOAP Version 1.2 Specification Assertions and Test Collection, W3C Recommendation 24 June 2003, http://www.w3.org/TR/soap12-testcollection/

[12] ETSI TR 102 203: "Mobile Commerce (M-COMM); Mobile Signatures; Business and Functional Requirements".

[13] ETSI TR 102 206: "Mobile Commerce (M-COMM); Mobile Signatures; Mobile Signature Service: Security Framework".

[14] ETSI TS 102 207: "Mobile Commerce (M-COMM); Mobile Signatures; Specifications for Roaming in M-signature Services".

[15] W3C Note 15 March 2001: "Web Services Description Language (WSDL) 1.1", http://www.w3.org/TR/wsdl

[16] W3C Recommendation 10 December 2002: "XML-Encryption Syntax and Processing", http://www.w3.org/TR/xmlenc-core/

[17] W3C Recommendation 2 May 2001: "XML Schema Part 1: Structures", http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[18] W3C Recommendation 2 May 2001: "XML Schema Part 2 : Datatypes", http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[19] W3C Recommendation 12 February 2002: "XML-Signature Core Syntax and Processing", http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/.

[20] ETSI TS 101 903 V1.1.1: "XML Advanced Electronic Signatures (XAdES)".

[21] IETF RFC 3552: "Guidelines for writing RFC Text on Security Considerations", July 2003.

[22] "Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures (Official Journal L 013 , 19/01/2000 pp. 12-20)".

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

[24] RSA PKCS#10 Version 1.7: "Certification Request Syntax Standard", RSA Laboratories, May 2000.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Application Provider (AP): person or organization who develops and/or sells and/or supports a service used by a citizen

asymmetric cryptography: means to encrypt messages in a manner that does not require from the encrypting entity to know the key used to decrypt the cipher-text

NOTE: Asymmetric cryptography also allows to sign messages in a manner that does not require from entity that verifies the signature to know the key used to produce the signature.

buffer over-run: attack consisting in corrupting a program by overflowing its internal variables

NOTE: Can be avoided if the program checks that only data of appropriate length is stored in variables.

Certification Authority (CA): authority that produces signatures on public-keys (certificates)

NOTE: The process of signing one's public-key is called "certification".

dependent application (or service): See definition in TR 102 203 [12], clause 10.3.4.

dispute resolution: process of resolving disputed transactions

electronic signature: data in electronic form which are attached to or logically associated with other electronic data message and which serve as a method of authentication

NOTE: Electronic signatures come are of three sorts: General, Qualified and Enhanced as defined in clause 6.1.

end-user or citizen: person (or device) in possession of (or embedded in) the mobile device (and/or SIM-card) to which a mobile signature is associated

NOTE: End-user and citizen are used interchangeably throughout the present document.

EU Directive: Text of the Directive 1999/93/EC [22] of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

mobile Signature (m-Signature): universal method for using a mobile device to confirm the intention of a citizen to proceed with a transaction

mobile signature process: logical sequence of acquiring and making use of a mobile signature

mobile signature service: facility that coordinates and manages the mobile signature process represents an opportunity for the card-issuer to provide a mobile signature service to citizens and application providers

Mobile Signature Service Provider (MSSP): person or entity that provides a mobile signature service

Mobile Signature Service Provider (Home MSSP): MSSP associated to the mobile network in the citizen's normal country of residence

Mobile Signature Service Provider (Roaming MSSP): intermediary body that may provide interoperability between Home MSSPs

proof of possession: proof that the citizen possesses or owns a given mobile device

Registration Authority (RA): authority in charge of capturing personal attributes from a citizen used to form the security profile

signature gateway: platform operated by the MSSP to enable mobile signature functionality

signing-PIN: numeric code known only to the citizen entered by that citizen on his/her mobile device keypad in order to confirm his/her intention with respect to transaction details displayed on the screen of the citizen's mobile device. OR: A sequence of digits used to verify the identity of the holder of a token. It is a numeric "password".

signature request: message received at the citizen's mobile device from an Application Provider

SIM-Card: smartcard located inside a mobile telephone used to manage the subscriber's access to the mobile telephone network

NOTE: The spare available memory on the SIM-card is often used to provide other services to the subscriber (e.g. telephone address book).

smartcard: card containing a tamper-resistant microprocessor (also called chip-card)

smartcard issuer: entity who manages all aspects relating to the smartcard used to create mobile signatures (e.g. the mobile network operator)

Specialist Task Force (STF): ETSI temporary team of specialist assigned for specific purposes

transaction roaming: transaction where a citizen maybe using a dependent application from an AP aligned with a visited network

NOTE: In this case the visited network needs to communicate with the home network in order to obtain a mobile signature from the citizen.

use case: describes the services that are enabled by mobile signature functionality

value added service: any additional facility offered by an MSSP in addition to the core mobile signature

web service: Internet technology

3.2 Abbreviations

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

AP	Application Provider
CA	Certification Authority
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
EESSI	European Electronic Signature Standardisation Initiative
HTTPS	Hypertext Transfer Protocol Secured
LAN	Local Area Network
m-Signature	Mobile Signature
MNO	Mobile Network Operator
MSISDN	The calling number for a citizen's mobile device
MSS	Mobile Signature Service
MSSP	Mobile Signature Service Provider
PIN	Personal Identification Number
PKI	Public Key Infrastructure
RA	Registration Authority
SigREQ-STD	Signature request received from an AP
SigRESP-STD	Response to a SigREQ-STD
SK	Symmetric Key Infrastructure
SMS	Short Message Service
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URI	Uniform Resource Identifier
WSDL	Web Service Description Language
XAdES	XML Advanced Electronic Signature

XAdES-C	XAdES with complete validation data
XAdES-T	XAdES with Time-Stamp
XML	extensible Markup Language

4 Void

5 Introduction to mobile signature

5.1 Overview

This overview is an extract from the TR 102 203 [12]. The intention is to provide the reader with sufficient information about the concepts introduced in [12].

5.1.1 Mobile Signature

The following working definition is proposed for the concept of mobile signature:

- *"A universal method for using a mobile device to confirm the intention of a citizen to proceed with a transaction."*

In constructing this definition, the following concepts and ideas were considered:

Universal method:

- a consistent end user experience;
- the largest interactive community for endusers and application providers;
- an architecture promoting interoperability and lowest deployment costs;
- an architecture offering the lowest transaction costs.

Mobile device:

- any device using a mobile network as a communications channel;
- mobile telephone, PDAs, Laptop-PCs;
- integral (e.g. MNO SIM card) and external (e.g. Dual slot) smartcards;
- with or without smartcards.

Citizen intention:

- a legitimate transaction instruction;
- citizen's authorization/permission to proceed with a transaction;
- engineered in such a way that the citizen cannot have been confused or misled (cf. what you see is what you sign).
- compliance (or otherwise) with legal effect provisions of EU Directive.

Transaction:

- an interaction requiring the citizen's confirmation in order to proceed, details of which are transmitted to the citizen's mobile device and displayed on the mobile device screen prior to authorization.

5.1.2 Using mobile signature

Mobile signature is a concept that is applicable to all kinds of "applications" and not just those applications which can be accessed through mobile devices. Its use is appropriate for applications requiring a citizen's permission to proceed with completion of a transaction that may be initiated by a voice-call, via interactive voice response systems, via the internet and other electronic communications channels and even face-to-face situations. In this respect, the mobile device may be considered as a "signing-tool" - the electronic equivalent of a pen.

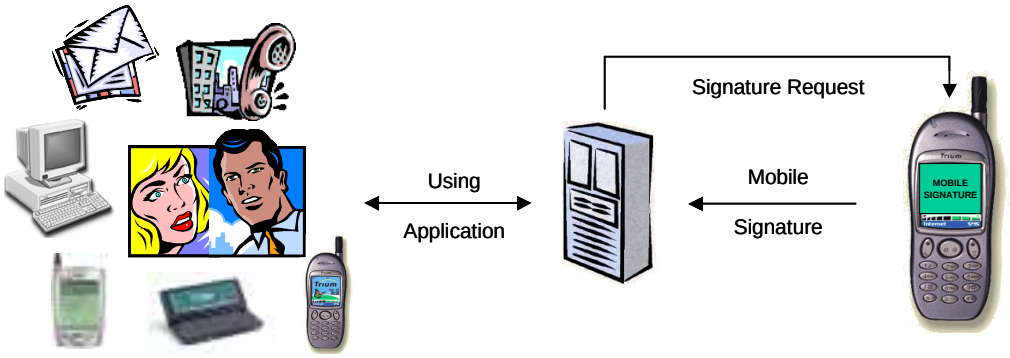


Figure 1: Mobile Device as "Signing Tool" (an electronic pen…)

In considering the use of mobile signature, we consider only the process of forming an electronic signature in relation to a message presented to the citizen. It specifically excludes application level control concerning the signed message. Provision of a mobile signature indicates only that the citizen would like to proceed with a transaction as presented, regardless of whether the citizen is allowed/entitled to do so.

5.1.3 Mobile Signature Service (MSS)

Coordination and management of the mobile signature process represents an opportunity to define a MOBILE SIGNATURE SERVICE for citizens and application providers alike. Such an approach might:

- Accelerate adoption of mobile signature by APs (and consequently adoption by endusers).
- Allow implementation/deployment of a universal API.
- Permit access to an existing base of endusers possessing smartcards and cardreaders.
- Coordinate activation of mobile signature functionality for endusers.
- Coordinate the processing of signature requests for application providers.
- Add value to core mobile signature service (e.g. Timestamp, receipt storage, signature verification, etc.).
- Leverage existing customer support and communication mechanisms.
- Resolve issues faced by "traditional" operators of CA platforms (user registration process, legalities, service level agreement).
- Reduce service deployment costs.
- Minimize duplication.
- Aggregate (i.e. acquire) signature trafic.
- Provide a manageable approach to risk reduction.
- Promote interoperability.

A mobile signature service might be provided under the terms of a commercial agreement between a Mobile Signature Service Provider (MSSP) and those parties who choose to rely on mobile signatures for whatever reason. The features of the MSSP role and his/her responsibilities are considered in clause 13 of TR 102 203 [12].

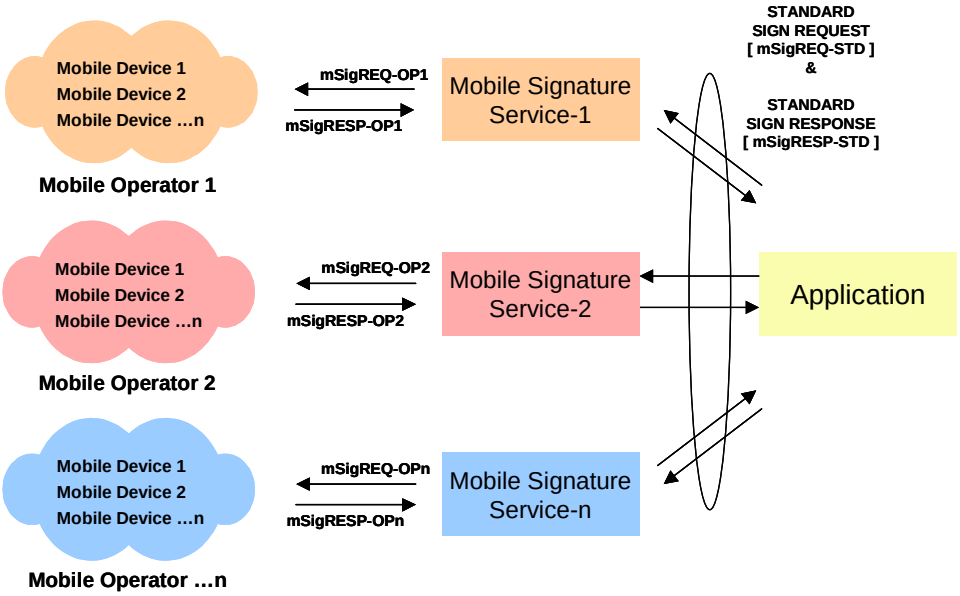


Figure 2: Mobile Signature Service

A Mobile Signature Service has a standardized interface implemented as an Internet Web Service. In this respect, a Mobile Signature Service Provider is an intermediary between endusers and APs that provides and implements a Mobile Signature Web Service.

5.2 Notation

The present document uses schema documents conforming to W3C XML Schema and normative text [17] to describe the syntax and semantics of XML-encoded protocol messages. The Web Service Description Language is used in order to describe the Mobile Signature Web Service.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in the present document are to be interpreted as described in RFC 2119 [3]. When these words are not capitalized, they are meant in their natural-language sense.

5.3 XML schema declaration

The following XML namespace is used for the Mobile Signature Service:

http://uri.etsi.org/TS102204/v1.2.1#

The following namespace declarations apply for the XML schema definitions throughout the present document:

```
<xs:schema targetNamespace="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:env=http://www.w3.org/2003/05/soap-envelope
elementFormDefault="qualified"
>
```

This implies that the prefix "ds" is used throughout the document to denote the namespace of the W3C XML-Signature specification according to [19]. The prefix "xs" denotes the namespace of the XML-Schema specification [XML-Schema], the prefix "xenc" the namespace of the XML Encryption Syntax and Processing recommandation [XML Encryption] and the prefix "env" denotes the namespace of the SOAP specification SOAP 1.2 [8]. The prefix "mss" is used to denote the namespace of the Mobile Signature Service.

The provided XML-Schema in the present document is normative.

6 Mobile Signature Service (MSS) functions

The purpose of this clause is to describe the high-level functionalities provided by the MSSP. These functionalities will help identifying the various methods that the mobile signature web service must implement.

6.1 Mobile Signature

The MSSP MUST support this function that is used for two different purposes:

- Any entity who wants to get a mobile signature of a displayable text from an enduser. It corresponds to the core service provided by the MSSP (see clause 8 of TR 102 203 [12]). The text that is signed is displayed on the enduser's mobile device.
- Any entity who wants to get the mobile signature of a special and non-displayable data from an enduser. For example, in the case where the enduser has an asymmetric key pair, a RA needs to get a PKCS#10 (which is not displayable) with the enduser's private key. A text MUST be displayed to the enduser so that he understands what is happening.

NOTE: Alternatively, the signing key may not be an asymmetric key, so cannot generate for instance a formal PKCS#10. In such circumstances, the RA could just ask the user to sign some friendly text (such as "Please confirm you registration request on <date> <time>") to provide proof of possession of the signing key.

The MSSP may use different technical implementations on the Mobile Network, the MSSP platform and the Mobile Equipment. This leads to the two following points:

- Mobile Signature Profile;
- Mobile Signature Messaging Modes.

6.1.1 Mobile Signature profile

If the endusers have not acquired the same signature capabilities, the Application Provider will not get the same mobile signature quality for a given enduser. The MSSP is able to use the different mobile signature capabilities of the endusers, but he should be able to indicate some kind of a signature quality to the Application Provider. This is what we call the Mobile Signature Profile.

Once the AP gets a Mobile Signature Profile from a MSSP, he must be able to ask for it in the Mobile Signature Request. A Mobile Signature Profile itself is not included in the messages, but a URI pointing on a Mobile Signature Profile is used instead.

TR 102 206 [13] provides a further analysis of what is a Mobile Signature Profile. Basically, a Mobile Signature Profile can be described by any means that is found appropriate by both parties, which can define their own URIs. That is the reason why, to date, TR102 206 [13] only provides with requirements for the content of Mobile Signature Profiles:

- Information about the registration and the certification phase such as:
  - Face to face or distant;
  - Level of credentials;
  - Issuance of Identity;
  - Issuance of keys.
- Information on the Mobile Signature System itself such as:
  - Signature policy;
  - Description of the Mobile Signature technical protection comprising the Mobile Signature Creation Device (e.g. SIM card), the Mobile Signature Creation Application (e.g. Mobile Phone) and the MSSP platform.

6.1.2 Mobile Signature messaging modes

In most of the cases, the MSSP has to push information on the enduser's mobile in order to seek at least the enduser's confirmation. In some cases, we assume that this step may take a while because:

- Network connection is bad where the enduser stands;
- The enduser is not ready to confirm the transaction;
- Etc.

For these reasons, the MSSP may be processing outstanding transactions in an "asynchronous" mode. Then, the relying party is left hanging. It may not be convenient for the relying party to get the mobile signature in one request-response dialogue. The present document proposes that the MSSP MUST be able to use a multi request-response protocol in order to deal with this asynchronous behavior.

The synchronous and asynchronous modes are studied in the present document since all the relying parties do not have the same constraints and needs. Several functional scenarios are possible. However, the MSSP MUST support all the following modes. It is up to the Application Provider to choose the mode used for the transaction processing.

6.1.2.1 Synchronous mode

This mode is based on a one-way messaging system. The relying party is only in position to contact the MSSP. In that case, we consider that the MSSP is a web service providing with a method called **"Mobile Signature Method"**. There is only one connection for the invocation of this method.

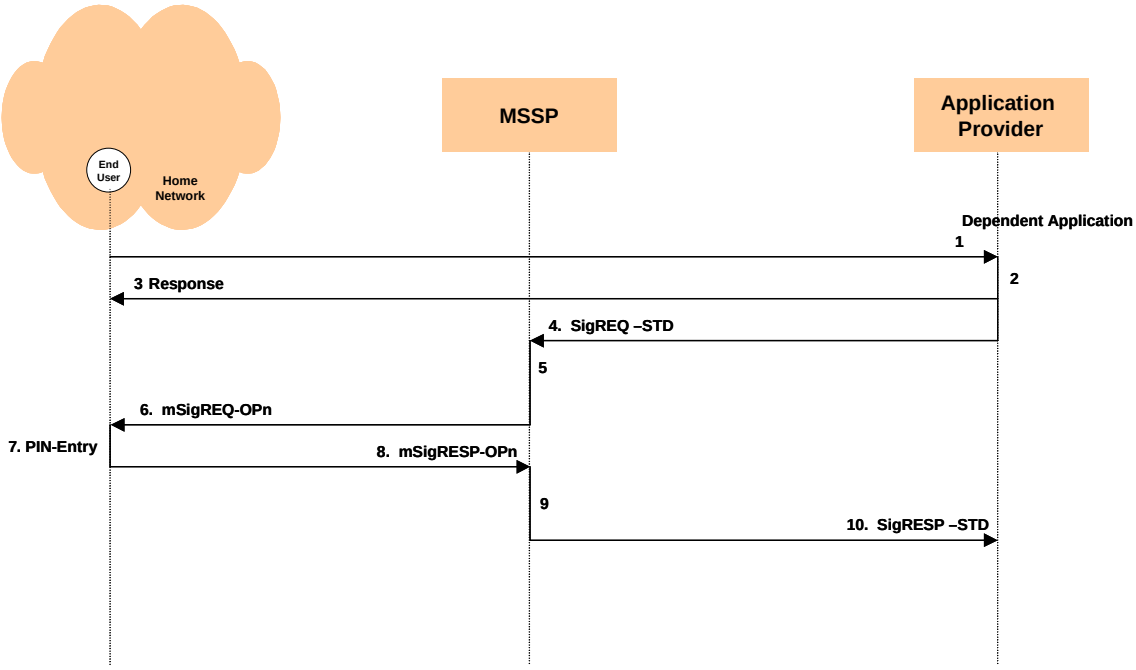


Figure 3: Mobile Signature Method - Synchronous mode

The process steps are described below:

1) Enduser - confirms a transaction or wants to access a service;

2) AP - processes enduser's confirmation; initiates data …;

3) AP - informs the enduser via the applications channel (e.g. Internet) that he is going to invoke his mobile signature application in order to confirm the transaction;

4) AP - Standard signature request [SigREQ-STD] to MSSP's mobile signature method;

5) MSSP - processes SigREQ-STD from AP; evaluates best Sign-acquisition route;

6) MSSP - sends mSig request to enduser (bespoke);

7) Enduser - the mobile handset displays the text to be signed etc. (cf. enduser experience) and computes a digital signature after a confirmation (e.g. PIN-entry, biometric authentication) by the citizen;

8) Mobile device returns mobile signature response to the MSSP;

9) MSSP processes the signature response (including any value added service elements such as secure storage of the signature receipt, signature validation, time stamping, etc.);

10) MSSP transmits the mobile signature method response [SigRESP-STD] back to AP (along with a mobile signature confirmation to the enduser's mobile device - optional) with or without an electronic signature.

After receiving the mobile signature response, the relying party is able to send a receipt thanks to the mobile signature receipt as described in clause 6.5.

6.1.2.2 Asynchronous - ClientServer mode

In this case, the MSSP is still working in a one-way messaging system; the relying party cannot be contacted later by the MSSP. So, MSSP will only acknowledge the call to the **"Mobile Signature Method"** described above and it is up to the relying party to contact MSSP later (and periodically) in order to get a status of the transaction and potentially the signature. This last step is in fact another method that MSSP is providing with, and is called **"Mobile Status Method"**.

There are at least two connections and the relying party initiates both:

- the first one to invoke the mobile signature method; and
- the second one to invoke the mobile signature status method.

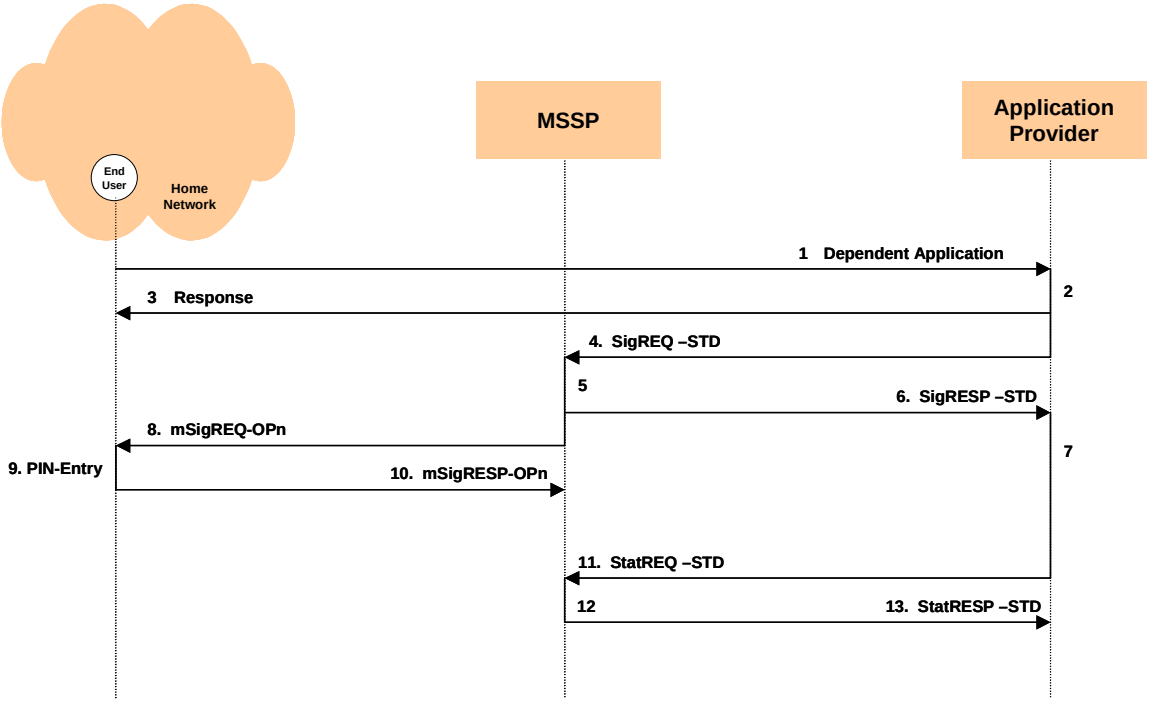


Figure 4: Mobile Signature Method - Asynchronous ClientServer mode

The process steps are described below:

1) Enduser - confirms a transaction or wants to access a service;

2) AP - processes enduser's confirmation; initiates data …;

3) AP - informs the enduser via the applications channel (e.g. Internet) that he is going to invoke his mobile signature application in order to confirm the transaction;

4) AP - Standard signature request [SigREQ-STD] to MSSP's Mobile Signature Method;

5) MSSP - processes [SigREQ-STD] from AP; evaluates best Sign-acquisition route;

6) MSSP transmits the mobile signature method response [SigRESP-STD] back to AP with a status indicating that the transaction is processing;

7) AP keeps track of the transaction;

8) MSSP - sends mSig request to enduser (bespoke);

9) Enduser - the mobile handset displays the text to be signed, etc. (see Annex B of TR 102 203 [12]) and computes a digital signature after confirmation (e.g. PIN-entry, biometric authentication) by the citizen.

10) Mobile device returns mobile signature response to the MSSP who processes the signature response (including any value added service elements such as secure storage of the signature receipt, signature validation, time stamping, etc).

11) AP - Standard status request [StatREQ-STD] to MSSP's mobile status method;

12-13) MSSP transmits the status response [StatRESP-STD] back to AP (along with a mobile signature confirmation to the enduser's mobile device - optional) with current status of the transaction and potentially the mobile signature.

After receiving the mobile signature response, the relying party is able to send a receipt thanks to the mobile signature receipt as described in clause 6.5.

6.1.2.3 Asynchronous - ServerServer mode

In this case, the MSSP will send a mobile signature notification to the relying party when the mobile signature is available or terminated. The relying party needs a "server" capability in order to process this mobile signature notification from the MSSP. In fact, the relying party is providing the MSSP with a **"Mobile Signature Notification Method"** that the MSSP can invoke. Two connections are established independently one from the other: the first one to invoke the Mobile Signature Method (by the AP) and the second one to invoke the Mobile Signature Notification Method (by the MSSP).

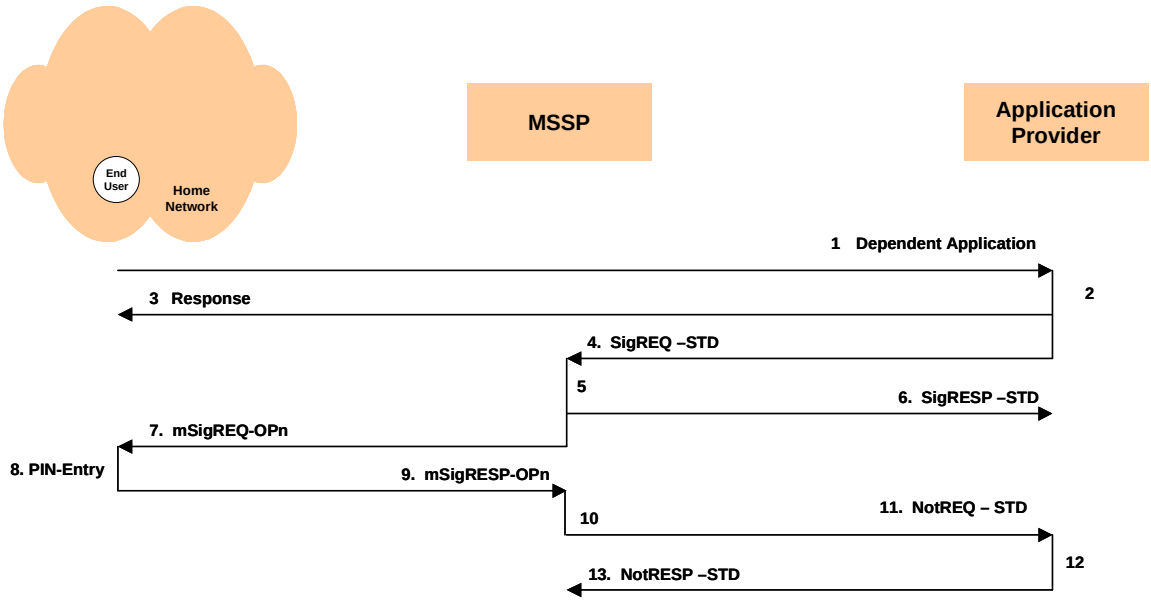


Figure 5: Mobile Signature Method - Asynchronous ServerServer mode

The process steps are described below:

1) Enduser - confirms a transaction or wants to access a service;

2) AP - Processes enduser's confirmation; initiates data …

3) AP - informs the enduser via the applications channel (e.g. Internet) that he is going to invoke his mobile signature application in order to confirm the transaction;

4) AP - Standard signature request [SigREQ-STD] to MSSP's mobile signature method;

5) MSSP - processes [SigREQ-STD] from AP; evaluate best Sign-acquisition route;

6) MSSP transmits the mobile signature method response [SigRESP-STD] back to AP with a status indicating that the transaction is processing;

7) MSSP - sends mSig request to Enduser;

8) Enduser - the mobile handset displays the text to be signed etc. (see Annex B of TR 102 203 [12]), and computes a digital signature after a confirmation (e.g. PIN-entry, biometric authentication) by the citizen;

9) Mobile device returns mobile signature response to the MSSP;

10) MSSP processes the signature response (including any value added service elements such as secure storage of the signature receipt, signature validation, time stamping, etc.);

11) MSSP - sends standard mobile signature notification request [NotREQ-STD] to AP's mobile signature notification method;

12) AP processes the signature notification;

13) AP transmits the mobile signature notification response back to the MSSP with potentially a receipt for the enduser.

6.2 Mobile Signature status query

The MSSP MUST support this functionality, which is used by an Application Provider in order to get information on a mobile signature transaction whether outstanding or not. It corresponds to invoking a **"Mobile Signature Status Query Method"**.

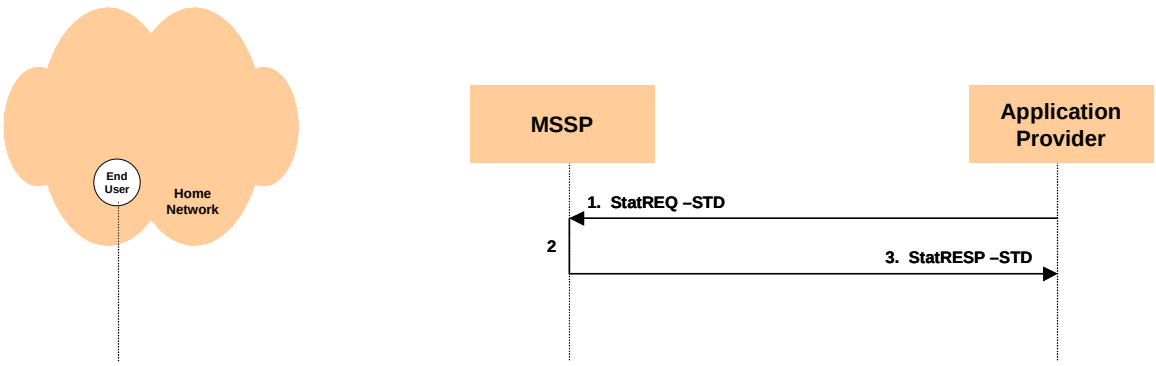


Figure 6: Mobile Signature status query

6.3 Mobile Signature profile query

The MSSP MUST support this function, which is used by an Application Provider in order to get relevant information related to the overall architecture that would be involved in the mobile signature process. Typically, an Application Provider would like to get the mobile signature profile before requesting a signature. The Mobile Signature Profile will be provided as a URI. This functionality corresponds to invoking a **"Mobile Signature Profile Method"**.

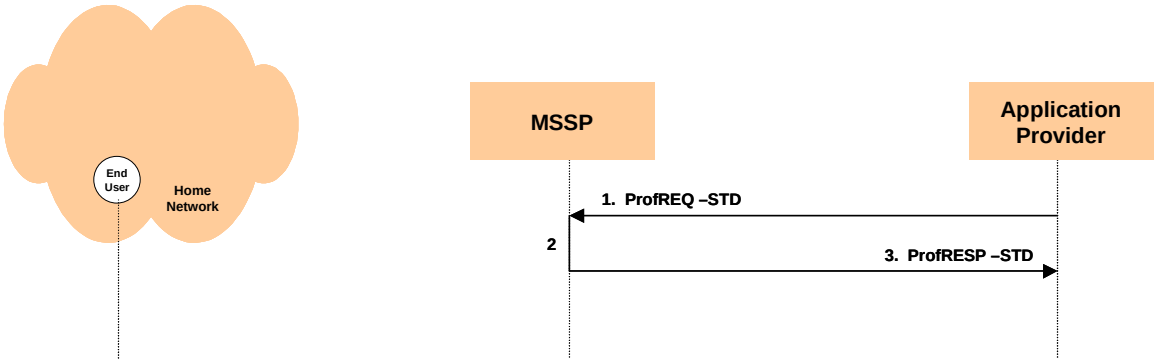


Figure 7: Mobile Signature profile query

6.4 Mobile Signature registration

The MSSP MUST support this function. Any AP (e.g. a smartcard issuer or a Registration Authority) uses this function in order to inform the MSSP about a new enduser willing to register to the Mobile Signature Service. For instance, both Certification Authority and smartcard issuer are in position to provide with relevant information about the security elements of the enduser.

One consequent action of the MSSP would be to download some security elements in the mobile signature application or even to activate the mobile signature application itself. The MSSP can also get information (may be confidential) from the enduser's mobile equipment and send them back to the AP as a response. For example, the RA may ask the MSSP to download a citizen's certificate URI in the SIM Card, activate the signing application, make the enduser initialize the signing PIN, which is then encrypted by the signing application, sent back to the MSSP, and put in the response back to the AP.

This functionality corresponds to invoking a **"Mobile Signature Registration Method"**.

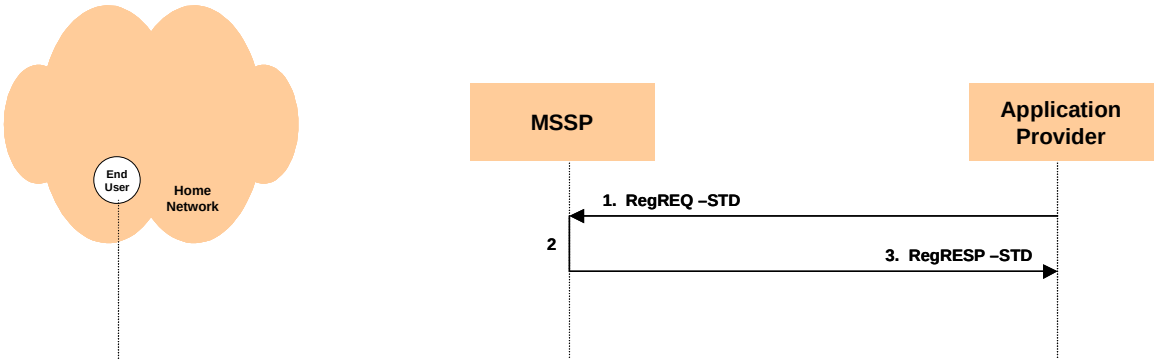


Figure 8: Mobile Signature registration

6.5 Mobile Signature receipt

This method is optional. An Application Provider uses this method at the end of a transaction in order to provide the enduser with some kind of a "receipt" that informs the enduser of the proceeding of the transaction. This corresponds to invoking a **"Mobile Signature Receipt Method"**. The content of a receipt is different according to the AP. Each AP has its own rules for receipt handling. However, this interface gives the possibility to the AP to put text and/or an XML signature (receipt digitally signed by the AP) in the RecReq-STD.

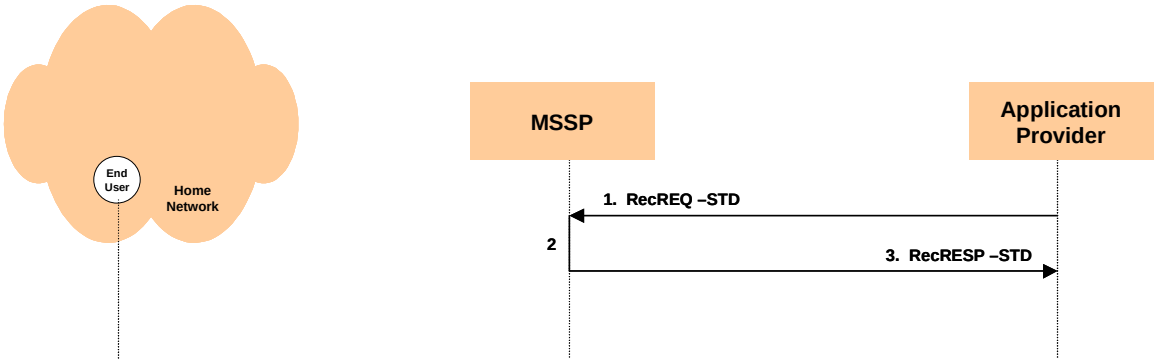


Figure 9: Mobile Signature receipt

6.6 Mobile Signature handshake

The MSSP MUST support this function that is used by both the MSSP and the AP in order to agree on the use of XML signatures in the communication. Clause 11 provides with a description and the rationales for the use of XML signatures in the messages exchanged between the MSSP and the AP. Note that this function corresponds in fact to two use-cases:

- The AP and the MSSP do not know each other, e.g. in the case of an AP communicating to a Home MSSP (see TS 102 207 [14]) through a Mesh. Then, both parties want to discover each other's capabilities.
- The AP and the MSSP have a direct commercial agreement and know exactly each other's capabilities. However, in some circumstances, the AP or the MSSP wants to improve the security involved in the transaction processing by the use of XML signatures, e.g. when there is a financial risk upon the transaction.

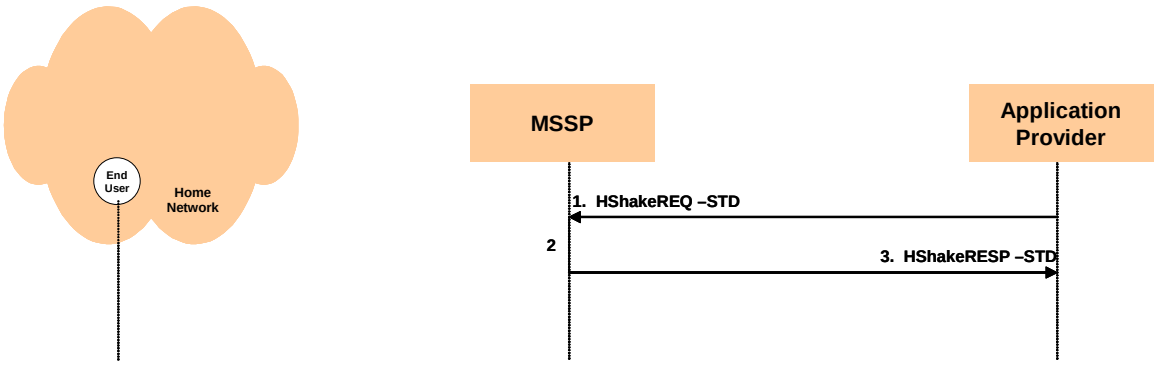


Figure 10: Mobile Signature handshake

7 Mobile Signature web service

The Mobile Signature Service is specified as a web service. All MSSP's functions are implemented over a simple request - response message exchange pattern through methods.

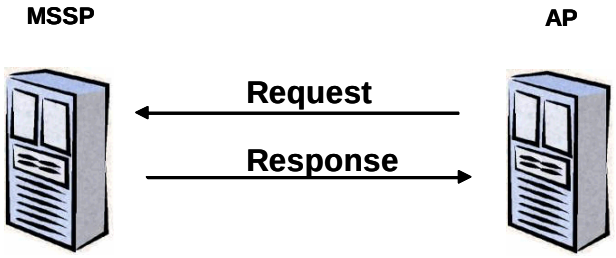


Figure 10a

The methods are described in this clause formally by means of the Web Services Description Language WSDL Version 1.1, see WSDL 1.1 [15], in order to summarize the information presented above. The binding to the network protocol HTTP & SOAP 1.2, see [8], is treated in clause 10.

NOTE: To date (June 2003), WSDL has not been finalized as a standard yet by the W3C, and the most recent version of it is 1.1, submitted a little over a year ago. We are waiting for WSDL Version 1.2 working draft to be approved as the formal W3C recommendation. However, in the meantime, WSDL 1.1 [15] has become commonly accepted as "de facto" standard. That is the reason why we provide the description of the Mobile Signature Web Service in WSDL 1.1 [15]. According to the current work in progress on WSDL 1.2, the following descriptions will not change a lot.

The following definitions apply for the WSDL schema definitions throughout this clause:

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:mss="http://uri.etsi.org/TS102204/v1.2.1#"
xmlns:tns="http://new.MSS_webservice.namespace"
targetNamespace="http://new.MSS_webservice.namespace"
>
```

Using these definitions the prefix "mss" is still used in this chapter to denote the namespace of the Mobile Signature Service XML-Schema defined in the present document while the prefix "tns" denotes the actual target namespace specified in the definitions and is currently just an example for the namespace of a Mobile Signature Web Service.

In the following the terminology of WSDL used in this chapter is outlined. Only the request-response mechanism used in the Mobile Signature Service is considered.

WSDL refers to this request-response mechanism as an operation. Using the terminology of WSDL this operation is made up of an input message and an output message. In the Mobile Signature Service every WSDL message consists exatly of one WSDL part. These parts are of the types presented in clause 8 (please note that the term message is used in the referenced chapter in a different way than in the WSDL terminology) and defined in the Mobile Signature Service XML-Schema. Not all operations of the Mobile Signature Service do not have a return value.

7.1 Mobile Signature method

The WSDL operation named MSS_Signature is made up of a WSDL input message and an output message. This operation has no return value. The WSDL input message consists simply of an element of the type mss:MSS_SignatureReqType while the WSDL output message consists of an element of the type mss:MSS_SignatureRespType.

```
<message name="MSS_SignatureInput">
    <part name="MSS_SignatureReq" type="mss:MSS_SignatureReqType"/>
</message>
<message name="MSS_SignatureOutput">
    <part name="MSS_SignatureResp" type="mss:MSS_SignatureRespType"/>
</message>

<portType name="MSS_SignaturePortType">
    <operation name="MSS_Signature">
        <input message="tns:MSS_SignatureInput"/>
        <output message="tns:MSS_SignatureOutput"/>
    </operation>
</portType>
```

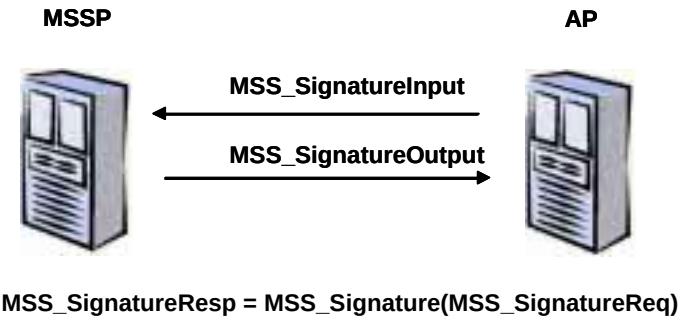


Figure 11: Mobile Signature Method - RPC Call

In the scope of the SOAP binding presented in clause 10 this means that the MSS_SignatureReq message is the input parameter of a remote procedure call and the MSS_SignatureResp message is the corresponding output parameter.

7.2 Mobile Signature status query method

A MSS_Status operation can be described exactly in the same way as the MSS_Signature operation using as part of the input message an element of the type mss:MSS_StatusReqType and as part of the output message an element of the type mss: MSS_StatusRespType.

```
<message name="MSS_StatusQueryInput">
    <part name="MSS_StatusReq" type="mss:MSS_StatusReqType"/>
</message>
<message name="MSS_ StatusQueryOutput ">
    <part name="MSS_StatusResp" type="MSS_StatusRespType"/>
</message>

<portType name="MSS_StatusQueryType">
    <operation name="MSS_StatusQuery">
        <input message="tns:MSS_StatusQueryInput"/>
        <output message="tns:MSS_StatusQueryOutput"/>
    </operation>
</portType>
```

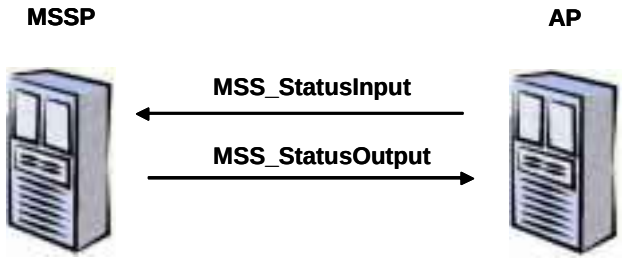


Figure 12: Mobile Signature Status Query - RPC Call

7.3 Mobile Signature Receipt Method

A MSS_Receipt operation can be described exactly in the same way as the MSS_Signature operation using as part of the input message an element of the type mss: MSS_ReceiptReqType and as part of the output message an element of the type mss: MSS_ReceiptRespType.

```
<message name="MSS_ ReceiptInput">
    <part name="MSS_ReceiptReq" type="MSS_ReceiptReqType"/>
</message>
<message name="MSS_ ReceiptOutput ">
    <part name="MSS_ReceiptResp" type="MSS_ReceiptRespType"/>
</message>

<portType name="MSS_ReceiptType">
    <operation name="MSS_Receipt">
        <input message="tns:MSS_ReceiptInput"/>
        <output message="tns:MSS_ReceiptOutput"/>
    </operation>
</portType>
```

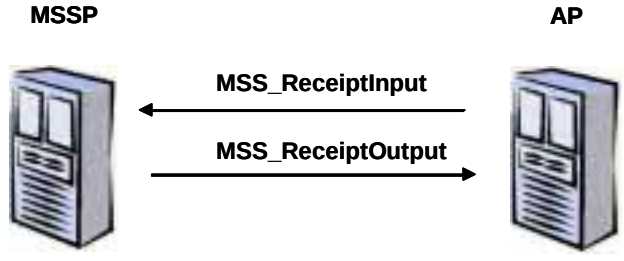


Figure 13: Mobile Signature Receipt - RPC Call

7.4 Mobile Signature Registration Method

A MSS_Registration operation can be described exactly in the same way as the MSS_Signature operation using as part of the input message an element of the type mss: MSS_RegistrationReqType and as part of the output message an element of the type mss: MSS_RegistrationRespType.

```
<message name="MSS_RegistrationInput">
    <part name="MSS_RegistrationReq" type="mss:MSS_RegistrationReqType"/>
</message>
<message name="MSS_RegistrationOutput">
    <part name="MSS_RegistrationResp" type="MSS_RegistrationRespType"/>
</message>

<portType name="MSS_RegistrationType">
    <operation name="MSS_Registration">
        <input message="tns:MSS_RegistrationInput"/>
        <output message="tns:MSS_RegistrationOutput"/>
    </operation>
</portType>
```

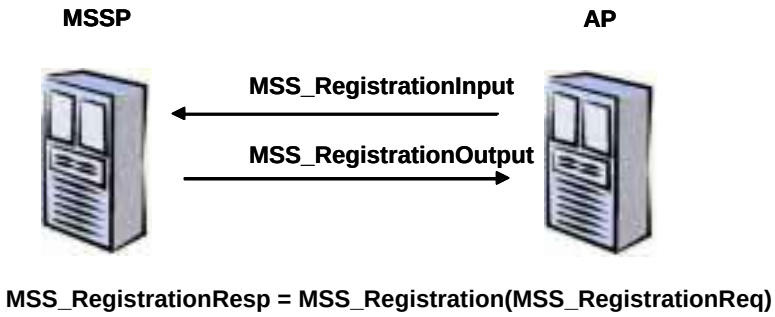


Figure 14: Mobile Signature registration - RPC Call

7.5 Mobile Signature Profile Query Method

A MSS_Profile operation can be described exactly in the same way as the MSS_Signature operation using as part of the input message an element of the type mss: MSS_ProfileReqType and as part of the output message an element of the type mss: MSS_ProfileRespType.

```
<message name="MSS_ProfileQueryInput">
    <part name="MSS_ProfileReq" type="mss:MSS_ProfileReqType"/>
</message>
<message name="MSS_ ProfileQueryOutput ">
    <part name="MSS_ProfileResp" type="MSS_ProfileRespType"/>
</message>

<portType name="MSS_ProfileQueryType">
    <operation name="MSS_ProfileQuery">
        <input message="tns:MSS_ProfileQueryInput"/>
        <output message="tns:MSS_ProfileQueryOutput"/>
    </operation>
</portType>
```

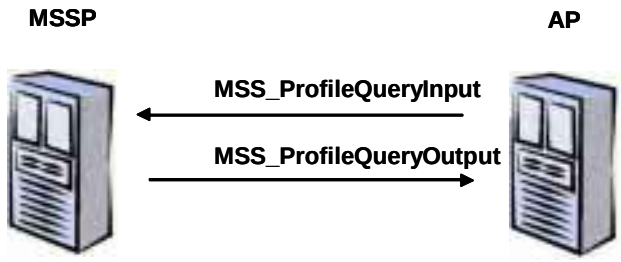


Figure 15: Mobile Signature profile query - RPC call

7.6 Mobile Signature notification method

In the case of the MSS_Notification operation the WSDL input message consists of an element of the type mss:MSS_StatusRespType while the WSDL output message consists of an element of the type mss:MSS_ReceiptReqType.

```
<message name="MSS_NotificationInput">
    <part name="MSS_StatusResp" type="mss:MSS_StatusRespType"/>
</message>
<message name="MSS_NotificationOutput">
    <part name="MSS_ReceiptReq" type="mss:MSS_ReceiptReqType"/>
</message>

<portType name="MSS_NotificationPortType">
    <operation name="MSS_Notification">
        <input message="tns:MSS_NotificationInput"/>
        <output message="tns:MSS_NotificationOutput"/>
    </operation>
</portType>
```

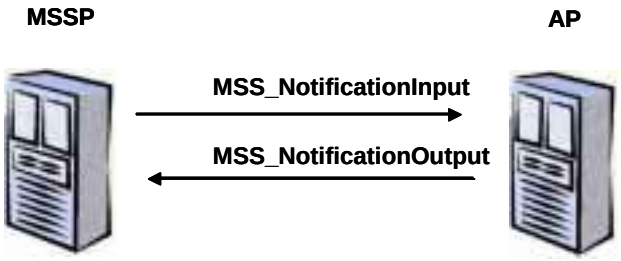


Figure 16: Mobile Signature notification - RPC Call

7.7 Mobile Signature handshake method

A MSS_Profile operation can be described exactly in the same way as the MSS_Signature operation using as part of the input message an element of the type mss: MSS_HandshakeReqType and as part of the output message an element of the type mss: MSS_HandshakeRespType.

```
<message name="MSS_HandshakeInput">
    <part name="MSS_HandshakeReq" type="mss:MSS_HandshakeReqType"/>
</message>
<message name="MSS_HandshakeOutput">
    <part name="MSS_HandshakeResp" type="mss:MSS_HandshakeRespType"/>
</message>

<portType name="MSS_HandshakePortType">
    <operation name="MSS_Handshake">
        <input message="tns:MSS_HandshakeInput"/>
        <output message="tns:MSS_HandshakeOutput"/>
    </operation>
</portType>
```

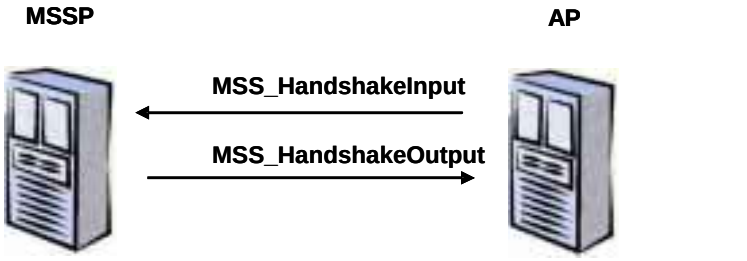


Figure 17: Mobile Signature handshake - RPC call

8 Message formats

In this clause the messages exchanged between the AP and the MSSP are presented. These messages are based upon the auxiliary types described in clause 9.

8.1 Message abstract type

This abstract type gathers all the parameters that will be systematically contained by all the MSS messages. The following parameters are only "Protocol" oriented.

Name		Format	Description	Mandatory
MajorVersion		Integer	Major version of the current interface, currently 1.	Y
MinorVersion		Integer	Minor version of the current interface, currently 1.	Y
AP_Info	AP_ID	anyURI	This identifier is given to the AP during his registration to the Mobile Signature Service. All APs, including MSSPs, are identified by means of URI-based identifier.	Y
	AP_TransID	NCName	Transaction number created by the AP toolkit on a new transaction. The AP fills this parameter in his requests. In messages from MSSP, the MSSP uses this parameter in order to indicate to the AP which transaction was handled.	Y
	AP_PWD	String	Password of the AP. When the AP registers as a Mobile Signature Service relying party, he gets a password that MUST be used in all the messages in order to be authenticated.	Y
	Instant	DateTime	The AP MUST mention the time and the date when the message was created.	Y
	AP_URL	anyURI	In the case where the messaging mode applied to this message is Asynchronous ServerServer, the MSSP needs to know the URL of the Mobile Signature Notification Method of the AP. The AP uses this parameter in order to indicate this URL to the MSSP in a Mobile Signature Request.	N
MSSP_Info	MSSP_ID	MeshMemberType, clause 9.2.1	All APs including the MSSP are identified by means of URI-based identifier.	Y
	Instant	DateTime	The MSSP MUST mention the time and date when the message was created. This parameter is not mandatory in the mandatory column, because this element is only used in the responses from the MSSP.	N

```
    <xs:complexType name="MessageAbstractType" abstract="true">
        <xs:sequence>
            <xs:element name="AP_Info">
                <xs:complexType>
                    <xs:attribute name="AP_ID" type="xs:anyURI" use="required"/>
                    <xs:attribute name="AP_TransID" type="xs:NCName" use="required"/>
                    <xs:attribute name="AP_PWD" type="xs:string" use="required"/>
                    <xs:attribute name="Instant" type="xs:dateTime" use="required"/>
                    <xs:attribute name="AP_URL" type="xs:anyURI" use="optional"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="MSSP_Info">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="MSSP_ID" type="mss:MeshMemberType"/>
                    </xs:sequence>
                    <xs:attribute name="Instant" type="xs:dateTime" use="optional"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="MajorVersion" type="xs:integer" use="required"/>
        <xs:attribute name="MinorVersion" type="xs:integer" use="required"/>
    </xs:complexType>
```

8.2 MSS Signature Request [SigREQ - STD]

Name	Format	Description	Mandatory
MobileUser	MobileUserType, clause 9.2.4	Any identifier that allows the MSSP to contact enduser's mobile device.	Yes
DataToBeSigned	DataType, clause 9.3.2	Data that the relying party wants the enduser to sign. Note that for confidentiality purpose the DataToBeSigned element may contain an XML encrypted element (see XML Encryption [16]).	Yes
DataToBeDisplayed	DataType, clause 9.3.2	The presence of this parameter indicates that the DataToBeSigned is non-display data (e.g. binary body of a PKCS#10, see PKCS#10 [24]). The contents of the display fields are to be presented to the user. Whether this option is permitted or not can be defined in the Signature Policy.	No
SignatureProfile	mssURIType, clause 9.2.3	The relying party is requesting this signature profile to be used in the processing of the transaction.	No
AdditionalServices	List of AdditionalServiceType, clause 9.3.4	List of added-value services upon Mobile Signature such as: Signature validation Time-stamping Archiving Others	No
MSS_Format	MssURI	The relying party requests a format for the signature. It may be : XML Signature, see RFC 3275 [7] CMS, see RFC 2630 [6] PKCS#7, see PKCS#7 [23] PKCS#10, see PKCS#10 [24] Others	No
KeyReference	KeyReferenceType, clause 9.3.3	In the case where the MSSP does not implicitly know the enduser's secret key or certificate that must be used for the signing of the transaction, the relying party is able to mention it.	No
SingatureProfileComparison	SingatureProfileComparisonType, clause 9.3.5	In this parameter, the relying party mentions how the MSSP must deal with the Signature Profile mentioned above.	No
ValidityDate	DateTime	Expiration date for the transaction. If this argument is not specified, a default TimeOut SHOULD be used.	No
TimeOut	positiveInteger	Represent the time in seconds the MSSP has to wait before responding to the AP. If this argument is not spcecified, a default TimeOut SHOULD be used.	No
MessagingMode	MessagingModeType, clause 9.3.1	Denotes the messaging mode either synchronous mode, or asynchronous ClientServer mode, or asynchronous ServerServer mode.	Y

```
    <xs:element name="MSS_SignatureReq" type="mss:MSS_SignatureReqType"/>
    <xs:complexType name="MSS_SignatureReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                    <xs:element name="DataToBeSigned" type="mss:DataType"/>
                    <xs:element name="DataToBeDisplayed" type="mss:DataType" minOccurs="0"/>
                    <xs:element name="SignatureProfile" type="mss:mssURIType" minOccurs="0"/>
                    <xs:element name="AdditionalServices" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Service" type="mss:AdditionalServiceType"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="MSS_Format" type="mss:mssURIType" minOccurs="0"/>
                    <xs:element name="KeyReference" type="mss:KeyReferenceType" minOccurs="0"/>
                    <xs:element name="SignatureProfileComparison"
type="mss:SignatureProfileComparisonType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="ValidityDate" type="xs:dateTime" use="optional"/>
                <xs:attribute name="TimeOut" type="xs:positiveInteger" use="optional"/>
                <xs:attribute name="MessagingMode" type="mss:MessagingModeType" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

8.3 MSS Signature response [SigRESP – STD]

Name	Format	Description	Mandatory
MobileUser	MobileUserType, clause 9.2.4	Any identifier of the enduser concerned by this transaction	Yes
Status	StatusType, clause 9.4.2	Current status of the transaction, see Annex B	Yes
SignatureProfile	mssURIType, clause 9.2.3	The MSSP should make a statement of the Signature Profile that was used.	No
MSS_Signature	SignatureType, clause 9.4.1	Mobile signature computed on the enduser's mobile device, plus optional additional service data.	No
MSSP_TransID	NCName	Transaction number created by the MSSP for this transaction	Yes

```
    <xs:element name="MSS_SignatureResp" type="mss:MSS_SignatureRespType"/>
    <xs:complexType name="MSS_SignatureRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                    <xs:element name="MSS_Signature" type="mss:SignatureType" minOccurs="0"/>
                    <xs:element name="SignatureProfile" type="mss:mssURIType" minOccurs="0"/>
                    <xs:element name="Status" type="mss:StatusType"/>
                </xs:sequence>
                <xs:attribute name="MSSP_TransID" type="xs:NCName" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Functional requirements:

If the request is valid AND accepted by the receiver for handling AND there are no further information, the receiver MUST then put the status code 100 REQUEST_OK in the response back to the sender:

- If the messaging mode is synchronous AND the MSSP got a Mobile Signature from the enduser, the MSSP MUST indicate the status code 500 SIGNATURE.
- If the signer's certificate is revoked (in the case of PKI signature validation), the MSSP MUST indicate the status code 501.

- If the signature is valid (in the case of PKI signature validation), the MSSP MUST indicate the status code 502.
- If the signature is invalid (in the case of PKI signature validation), the MSSP MUST indicate the status code 503.

8.4 MSS Status Request [StatREQ – STD]

Name	Format	Description	Mandatory
MSSP_Transaction_ID	NCName	In this parameter, the AP mentions the transaction number whose information he is looking for.	Yes

```
<xs:element name="MSS_StatusReq" type="mss:MSS_StatusReqType"/>
<xs:complexType name="MSS_StatusReqType">
    <xs:complexContent>
        <xs:extension base="mss:MessageAbstractType">
            <xs:attribute name="MSSP_TransID" type="xs:NCName" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

8.5 MSS status response [StatRESP - STD]

Name	Format	Description	Mandatory
MobileUser	MobileUserType, clause 9.2.4	Any identifier of the enduser concerned by this transaction	Yes
MSS_Signature	SignatureType, clause 9.4.1	If available, this parameter contains a Mobile signature computed on the enduser's mobile device.	No
Status	StatusType, clause 9.4.2	Current status of the transaction (see Annex B)	Yes

```
<xs:element name="MSS_StatusResp" type="mss:MSS_StatusRespType"/>
<xs:complexType name="MSS_StatusRespType">
    <xs:complexContent>
        <xs:extension base="mss:MessageAbstractType">
            <xs:sequence>
                <xs:element name="MobileUser" type="mss:MobileUserType"/>
                <xs:element name="MSS_Signature" type="mss:SignatureType" minOccurs="0"/>
                <xs:element name="Status" type="mss:StatusType"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Because the status method can be called by the AP at any time (even a long time after the completion of a transaction), the status value can take all the values among error codes and status codes of Annex B. In the case where the status method is called after a Mobile Signature request in an asynchronous ClientServer mode, the AP will rather get status codes than error codes. The typical status code for this method is the code 504 OUTSTANDING_TRANSACTION, which means that the AP will need to call again the status method.

8.6 MSS registration request [RegREQ – STD]

Name	Format	Description	Mandatory
MobileUser	MobileUserType, clause 9.2.4	At least the MSSP activates the Mobile signature Capability of this enduser if not active.	Yes
EncryptedData	xenc:EncryptedType	Loading of confidential data on the enduser's Mobile Signature Application (see XML Encryption [16])	No
EncryptResponseBy	anyURI	This element defines an individual encryption/decryption key together with an encryption method. For example: www.mssp-Companyy.com/securetransport/MSISDN=447766123456&Method=CBC3DES&Keyid=1 The AP specifies this element in order to specify the key and the method that will be used by the mobile equipment to encrypt confidential information. The AP will retrieve these encrypted information in the Mobile Signature Registration Response.	No
CertificateURI	anyURI	Store by the MSSP and download on the mobile signature application	No
X509Certificate	base64Binary	Store by the MSSP and download on the mobile signature application if possible	No
Any	Any	This parameter gives the opportunity to the MSSP to provide the AP with a bespoke registration method. The AP will use this parameter in order to indicate other relevant enduser's information coming from the registration phase.	No

```
    <xs:element name="MSS_RegistrationReq" type="mss:MSS_RegistrationReqType"/>
    <xs:complexType name="MSS_RegistrationReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                    <xs:element name="EncryptedData" type="xenc:EncryptedType" minOccurs="0"/>
                    <xs:element name="EncryptResponseBy" type="xs:anyURI" minOccurs="0"/>
                    <xs:element name="CertificateURI" type="xs:anyURI" minOccurs="0"/>
                    <xs:element name="X509Certificate" type="xs:base64Binary" minOccurs="0"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

8.7 MSS registration response [RegRESP – STD]

Name	Format	Description	Mandatory
Status	StatusType, clause 9.4.2	Current Status of the transaction (see Annex B). If the registration is successfully performed, the MSSP MUST then indicate the status code 408 REGISTRATION_OK.	Yes
EncryptedData	xenc:EncryptedType	Retrieving of confidential information from the registration phase on the enduser's mobile device. It can be a Signing PIN or a signing Key … The Encryption of this element is requested in the EncryptReponseBy element of the MSS Registration Request. (see XML Encryption [16])	No
CertificateURI	anyURI	Retrieving of public information	No
X509Certificate	base64Binary	Retrieving of public information	No
PublicKey	Base64Binary	Retrieving of public information	No

```
    <xs:element name="MSS_RegistrationResp" type="mss:MSS_RegistrationRespType"/>
    <xs:complexType name="MSS_RegistrationRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="Status" type="mss:StatusType"/>
                    <xs:element name="EncryptedData" type="xenc:EncryptedType" minOccurs="0"/>
                    <xs:element name="CertificateURI" type="xs:anyURI" minOccurs="0"/>
                    <xs:element name="X509Certificate" type="xs:base64Binary" minOccurs="0"/>
                    <xs:element name="PublicKey" type="xs:base64Binary" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

8.8 MSS profile request [ProfREQ – STD]

Name	Format	Description	Mandatory
MobileUser	MobileUserType, clause 9.2.4	If the MSSP implements different cryptographic techniques etc. for endusers, the MSSP may respond differently according to the enduser. The AP uses this parameter to indicate the enduser whose Mobile Signature Profile must be retrieved	No

```
    <xs:element name="MSS_ProfileReq" type="mss:MSS_ProfileReqType"/>
    <xs:complexType name="MSS_ProfileReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

8.9 MSS profile response [ProfRESP – STD]

Name	Format	Description	Mandatory
MobileSignatureProfile	mssURIType, clause 9.2.3	One or several Mobile Signature profiles supported by this MSSP for the enduser specified in the Mobile Signature Profile request	No
Status	StatusType, clause 9.4.2	Current Status of the transaction (see Annex B).	Yes

```
    <xs:element name="MSS_ProfileReq" type="mss:MSS_ProfileReqType"/>
    <xs:complexType name="MSS_ProfileReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Functional requirements:

If the enduser is unknown by the MSSP and the MSSP is not part of the Mesh of a Mobile Signature Roaming Service (see TS 102 207 [14]), then the MSSP MUST indicate the status code 105 UNKNOWN_CLIENT.

8.10 MSS receipt request [RecREQ – STD]

Name	Format	Description	Mandatory
MobileUser	MobileUserType, clause 9.2.4	Any identifier that allows the MSSP to contact enduser's mobile device.	Yes
Status	StatusType, clause 9.4.2	Status of the transaction	No
MSSP_Trans_ID	NCName	A receipt request MUST always be linked to a previous signature transaction. In this respect, this parameter MUST correspond to a transaction ID the MSSP has already handled.	Yes
Message	DataType, clause 9.3.2	A message that the relying party wants to send to the enduser.	No
SignedReceipt	XML Signature	Optionally the AP can provide the enduser with a digitally signed receipt.	No

```
    <xs:element name="MSS_ReceiptReq" type="mss:MSS_ReceiptReqType"/>
    <xs:complexType name="MSS_ReceiptReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                    <xs:element name="Status" type="mss:StatusType" minOccurs="0"/>
                    <xs:element name="Message" type="mss:DataType" minOccurs="0"/>
                    <xs:element name="SignedReceipt" type="ds:SignatureType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="MSSP_TransID" type="xs:NCName" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

8.11 MSS receipt response [RecRESP - STD]

Name	Format	Description	Mandatory
Status	StatusType, clause 9.4.2	Current Status of the transaction (see Annex B).	Yes

```
    <xs:element name="MSS_ReceiptResp" type="mss:MSS_ReceiptRespType"/>
    <xs:complexType name="MSS_ReceiptRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="Status" type="mss:StatusType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

8.12 MSS Handshake request [HShakeREQ – STD]

Name	Format	Description	Mandatory
SecureMethods	complexType	This element is a list of methods (listed in the present document) that the AP wants to be secured. For example, if the AP sets the value of the MSS_Signature element to TRUE, then the MSSP MUST include an XML signature in the Mobile Signature Response as specified in clause 11.3.	Yes
Certificates	complexType	By this element the AP indicates a list of certificates it can use for the generation of XML signatures. If this list is empty, the AP does not support XML signatures.	Yes
RootCAs	complexType	By this element, the AP indicates a list of supported root CAs that it can use for the purpose of XML signature verification. If this list is empty, then the AP does not mandate any Root CAs.	Yes
SignatureAlgList	complexType	This parameter indicates a list of supported signature algorithms by the AP. Each signature algorithm is represented as a URI, e.g. http://www.w3.org/2000/09/xmldsig#dsa-sha1	Yes

```
    <xs:element name="MSS_HandshakeReq" type="mss:MSS_HandshakeReqType"/>
    <xs:complexType name="MSS_HandshakeReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="SecureMethods">
                        <xs:complexType>
                            <xs:attribute name="MSS_Signature" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Registration" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Notification" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_ProfileQuery" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Receipt" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Status" type="xs:boolean" use="required"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="Certificates">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Certificate" type="xs:base64Binary" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="RootCAs">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="DN" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="SignatureAlgList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Algorithm" type="mss:mssURIType" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

8.13 MSS handshake response [HShakeRESP – STD]

Name	Format	Description	Mandatory
MSSP_TransID	NCName	Transaction number created by the MSSP for this handshake. Then, the AP will reuse this transaction number in the next method call.	Yes
SecureMethods	complexType	This element is a list of methods (listed in the present document) that the MSSP wants to be secured. For example, if the MSPP sets the value of the MSS_Signature element to TRUE, then the AP MUST include a XML signature in the Mobile Signature Request as specified in clause 11.3.	Yes
MatchingMSSPCertificates	complexType	The purpose of this element is to indicate to the AP the certificates of the MSSP that match the Root CAs mentioned by the AP in the Handshake request. If the list is empty, the MSSP cannot provide XML Signatures to the AP.	Yes
MatchingApCertificates	complexType	The purpose of this element is to indicate to the AP the certificates of the AP that the MSSP can understand and trust. Therefore, this element is a subset of the Certificates list provided in the Handshake request. If this list is empty, the MSSP cannot use the XML Signatures provided by the AP.	Yes
MatchingSigAlgList	complexType	This element denotes a list of Algorithms that both the AP and the MSSP support. The AP will then choose to use one of them. Each signature algorithm is represented as a URI, e.g. http://www.w3.org/2000/09/xmldsig#dsa-sha1	Yes

```
    <xs:element name="MSS_HandshakeResp" type="mss:MSS_HandshakeRespType"/>
    <xs:complexType name="MSS_HandshakeRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="SecureMethods">
                        <xs:complexType>
                            <xs:attribute name="MSS_Signature" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Registration" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Notification" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_ProfileQuery" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Receipt" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Status" type="xs:boolean" use="required"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="MatchingMSSPCertificates">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Certificate" type="xs:base64Binary" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="MatchingAPCertificates">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Certificate" type="xs:base64Binary" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="MatchingSigAlgList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Algorithm" type="mss:mssURIType" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="MSSP_TransID" type="xs:NCName" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

9 Auxiliary types

The message types specified in the previous section are based upon the XML data types provided in this clause. The XML namespace used in the present document and the prefixes used to denote other namespaces have already been defined in the introduction.

9.1 URI identifier

Uniform Resource Identifier according to RFC 2396 [5] are used in the context of the present document to uniquely identify:

- An Application Provider;
- A Mobile Signature Service Provider;
- Signature formats;
- Additional services as time stamping, signature validation and signature archiving;
- Signature Profiles;
- The namespace of the Mobile Signature Service XML-Schema.

All Application Provider including the Mobile Signature Service Provider are identified by means of URI-based identifier. Therefore the provider's URI-based identifier must be unique. For this reason it is RECOMMENDED that each provider uses a URL based on its own domain name for this identifier.

However, in some cases the AP is not server centric entity hosted by a company, but rather a mobile phone or a PC of the enduser (Instant messaging client on a PC or a mobile phone requesting that the mobile user signs an outgoing message). In those circumstances, the "thin" AP has a URI based on the MSSP's domain name or the Acquiring Entity's domain name for roaming scenarios (see TS 102 207 [14]).

The URI for the namespace used in the present document can be found in the introduction. To denote some signature formats URIs are specified in the present document, see section Signature Type and also to denote some requested additional services as time stamping, see section Additional Service Type. A provider of an additional service SHOULD specify and publish its own URI for its services. It is RECOMMENDED that this URI uses a URL based on its own domain name for this identifier.

Some URIs for Signature Profiles will be published in the namespace of the present document, please refer to TR 102 206 [13]. The parties involved in the Mobile Signature Service SHOULD publish further URIs according to their needs.

9.2 General auxiliary types

9.2.1 MeshMember

This type is used to denote an entity, e.g. an Identity Issuer or a Home MSSP. This can be done by means of a DNS-Name, an IP Address, a string that identifies the entity or a URI.

```
    <xs:complexType name="MeshMemberType">
        <xs:sequence>
            <xs:element name="DNSName" type="xs:string" minOccurs="0"/>
            <xs:element name="IPAddress" type="xs:string" minOccurs="0"/>
            <xs:element name="URI" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="IdentifierString" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

9.2.2 Digest alg and value

This type is simply a container that contains a hash value and denotes optionally the algorithm used to generate the hash value. In the present document, this type is only used in conjunction with mssURI type. For this purpose the types DigestMethodType and DigestMethodValue of the W3C XML-Signature, (see RFC 3275 [7]) are used.

```
    <xs:complexType name="DigestAlgAndValueType">
        <xs:sequence>
            <xs:element name="DigestMethod" type="ds:DigestMethodType" minOccurs="0"/>
            <xs:element name="DigestValue" type="ds:DigestValueType"/>
        </xs:sequence>
    </xs:complexType>
```

9.2.3 mssURI

This type is used in order to specify a URI and for security reasons also provide a digest value related only to this URI. E.g. in the case of a Signature Profile published by means of a URL, this type contains the URI and optionally the digest value of the Signature Profile as well as the digest algorithm. The any element provides the possibility to add further information.

```
    <xs:complexType name="mssURIType">
        <xs:sequence>
            <xs:element name="mssURI" type="xs:anyURI"/>
            <xs:element name="DigestAlgAndValue" type="mss:DigestAlgAndValueType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

The following URIs are used in the present document to specify different standard signature formats:

- http://uri.etsi.org/TS102204/v1.1.2#XML-Signature
- This URI denotes a W3C XML-Signature according to RFC 3275 [7].
- http://uri.etsi.org/TS102204/v1.1.2#PKCS7
- This URI denotes a PKCS7 signature according to PKCS#7 [23].
- http://uri.etsi.org/TS102204/v1.1.2#CMS-Signature
- This URI denotes a CMS-Signature according to RFC 2630 [6].
- http://uri.etsi.org/TS102204/v1.1.2#PKCS#10
- This URI denotes a PKCS#10 according to PKCS#10 [24].

9.2.4 Mobile user

This type specifies the enduser. Therefore an identifier (e.g. a pseudonym) specifying an enduser and the corresponding Identity Issuer, the MSISDN and the Home MSSP of the enduser can be specified in any combination. The purpose is to specify uniquely the enduser.

```
    <xs:complexType name="MobileUserType">
        <xs:sequence>
            <xs:element name="IdentityIssuer" type="mss:MeshMemberType" minOccurs="0"/>
            <xs:element name="UserIdentifier" type="xs:string" minOccurs="0"/>
            <xs:element name="HomeMSSP" type="mss:MeshMemberType" minOccurs="0"/>
            <xs:element name="MSISDN" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

9.3 AP auxiliary types

This section deals with types of data used by the AP when communicating with MSSP.

9.3.1 Messaging mode

As described above the MSSP provides application providers with various messaging modes. Three messaging modes are possible:

- Synchronous mode: that is the basic Request - Response functionality
- Asynchronous mode - ClientServer: the functionality implements several request-responses
- Asynchronous mode - ServerServer: this is the two-way messaging mechanism. It requires a special capability from the client side, i.e. the client must support also a server capability

The MSSP MUST support the three messaging modes. Basically, the AP imposes to the MSSP its messaging mode. The AP client MUST support at least one of these three messaging modes and SHOULD support all three.

```
    <xs:simpleType name="MessagingModeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="synch"/>
            <xs:enumeration value="asynchClientServer"/>
            <xs:enumeration value="asynchServerServer"/>
        </xs:restriction>
    </xs:simpleType>
```

9.3.2 Data

This type is used as a container for messages to be signed or displayed sent from the AP or RA to the enduser. The optional attributes specify the Mime Type and the encoding of the message according to RFC 2045 [1] and RFC 2046 [2].

```
    <xs:complexType name="DataType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="MimeType" type="xs:string" use="optional"/>
                <xs:attribute name="Encoding" type="xs:string" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

9.3.3 Key reference

This type specifies the key of an enduser that has to be used in order to generate a signature. It is possible to adress a key specifying:

- A URI that is linked to the certificate (e.g. the URL that is used to publish the certificate);
- A distinguished name of the certificate issuer;
- The hash value of the key (and the applied hash algorithm);
- The hash value of the public key of the CA that has issued the certificate;
- Or using any other method.

```
    <xs:complexType name="KeyReferenceType">
        <xs:sequence>
            <xs:element name="CertificateURL" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="CertificateIssuerDN" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="HashOfUsersPublicKey" type="mss:DigestAlgAndValueType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="HashOfCAPublicKey" type="mss:DigestAlgAndValueType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax"/>
        </xs:sequence>
    </xs:complexType>
```

9.3.4 Additional service

The AdditionalServiceType is used to indicate that the SP requests not only a signature of the enduser but also a validation of this signature, a timestamp or an archiving of this signature or anything else. Therefore the corresponding procedure is specified using an URI. An entity in charge of performing the additional service can also be specified.

```
    <xs:complexType name="AdditionalServiceType">
        <xs:sequence>
            <xs:element name="Description" type="mss:mssURIType"/>
            <xs:element name="Entity" type="mss:MeshMemberType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

The following standard URIs are used in the present document to specify these additional service types by means of the mssURI type:

- http://uri.etsi.org/TS102204/v1.1.2#validate
- http://uri.etsi.org/TS102204/v1.1.2#timestamp
- http://uri.etsi.org/TS102204/v1.1.2#archive

NOTE: These URIs do not define a specified procedure with respect to the validation of a signature, the archiving of a signature or a format of a timestamp. The URIs defined above can be used to request the "standard" service of an additional service provider. A provider of an additional service SHOULD specify and publish a URI for its service.

9.3.5 Signature profile comparison

The SignatureProfileComparison element specifies a behavior of the MSSP according to Mobile Signature Profiles provided in a Mobile Signature Request:

- If set to "exact", then the MSSP is asked to match at least one of the specified <SignatureProfile> elements of the MSS_SignatureReq message exactly;
- If set to "minimum", the MSSP is asked to use a profile that he feels is at least as good as any specified in the <SignatureProfile> element;
- If set to "better", the MSSP is asked to use any profile better than any that were supplied.

If not specified, this is assumed to be exact.

```
<xs:simpleType name="SignatureProfileComparisonType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="exact"/>
        <xs:enumeration value="minimum"/>
        <xs:enumeration value="better"/>
    </xs:restriction>
</xs:simpleType>
```

9.4 MSSP auxiliary types

This clause deals with types of data used by the MSSP when communicating back with the AP.

9.4.1 Signature

The SignatureType acts as a container for the signature provided by the enduser. This can either be an XML-Signature according to RFC 3275 [7] using the ds:SignatureType or a base64 encoded signature according to the PKCS#7 [23] standard or a CMS-Signature according to RFC 2630 [6]. The any element supports the use of other formats.

This type acts also as a container for the signature provided by the enduser plus additional information (validation data, verification data, timestamps, signature assertions etc.) provided by the MSSP or another entity. If e.g. the AP requests an XML Advanced Electronic Signature, see TS 101 903 [20], (by means of the mssURI) there is also a signature of type ds:SignatureType returned.

```
<xs:complexType name="SignatureType">
    <xs:choice>
        <xs:element name="XMLSignature" type="ds:SignatureType"/>
        <xs:element name="Base64Signature" type="xs:base64Binary"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
        <!-- this can also be an advanced XML Signature-->
    </xs:choice>
</xs:complexType>
```

9.4.2 Status

The StatusType indicates the status of a response. Therefore a mandatory status code of type StatusCodeType described below is used as well as an optional message of type string and an optional element StatusDetail of type StatusDetailType described below.

```
<xs:complexType name="StatusType">
    <xs:sequence>
        <xs:element name="StatusCode" type="mss:StatusCodeType"/>
        <xs:element name="StatusMessage" type="xs:string" minOccurs="0"/>
        <xs:element name="StatusDetail" type="mss:StatusDetailType" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

9.4.3 Status code

The StatusCodeType contains the status of the corresponding response message. Therefore one or more nested status codes can be used. The top-level status code must be one of the values specified in Annex B. Subsequent nested status codes are optional and can be implementation specific.

```
<xs:complexType name="StatusCodeType">
    <xs:sequence>
        <xs:element name="StatusCode" type="mss:StatusCodeType" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Value" type="xs:integer" use="required"/>
</xs:complexType>
```

9.4.4 Status Detail

The StatusDetailType may be used to provide further information with respect to an error.

```
<xs:complexType name="StatusDetailType">
    <xs:sequence>
        <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

10 Communication Protocol Binding

This clause defines the protocol binding for the Mobile Signature Service in communication protocols. The mapping from the Mobile Signature Service request-response message exchange mechanism into standard communication protocols is referred as protocol binding. In the context of the present document only the binding to the SOAP protocol, see SOAP 1.2 [8], is treated, because the Mobile Signature Web Service MUST be implemented over SOAP 1.2 [8] over HTTP. The intent of the specified protocol binding is to ensure that implementations are interoperable.

SOAP is a protocol intended for exchanging structured information in a decentralized, distributed environment. The protocol is based upon XML and HTTP and consists of three parts: First an envelope that wraps XML data and contains processing information. Second a set of encoding rules in order to declare application specific data types and third a convention for representing remote procedure calls and their responses. The SOAP encoding rules are optional. The protocol provides further a binding for SOAP messages to the HTTP protocol.

10.1 Encoding rules

The SOAP encoding rules are optional. These encoding rules are not used in the SOAP binding of the Mobile Signature Service, but the MSS XML-Schema encoding is used.

10.2 SOAP header

The SOAP envelope comprises two SOAP child elements, an optional header and a mandatory body. The SOAP header is subject of the present document as we define a MSS_MessageSignature Header block in clause 11.3. TS 102 207 [14] defines also additional header blocks for the purpose of routing of Mobile Signature messages.

10.3 SOAP body

The message parts (using the WSDL [15] terminology) of the Mobile Signature Service operations are ordered in an RPC style. i.e. each part is a parameter and appears inside a wrapper element within the body. The name of this wrapper element is identical to the name of the operation. Every method parameter, i.e. WSDL message part is presented by an accessor of this wrapper. The name of the method parameter and the accessor are identical. The order is of importance: the parts appear in the same order as the parameters of the call.

10.4 SOAP over the HTTP protocol

The binding of the Mobile Signature Service MUST be implemented over SOAP 1.2 [8] over HTTP. SOAP 1.2 over HTTP requires a SOAPAction entry in the HTTP header. TLS RFC 2246 [2] is also used in order to secure the channel (see clause 11).

10.5 WSDL Description

In this clause the protocol binding of the Mobile Signature Service is summarized formally by means of the Web Service Description Language (WSDL) introduced already in the previous chapter.

In the following WSDL example the soap:binding element has two attributes. The value "rpc" of the attribute style indicates that the operation is used as a remote procedure call with the implications described in the section SOAP Body. The value of the attribute transport denotes that SOAP over HTTP is used.

If SOAP over HTTP is used a SOAPAction entry in the HTTP header is required. Therefore the soap:operation element requires a mandatory attribute soapAction that takes a URI as value which is implementation dependant. The attribute style indicates again that the data contained in the SOAP body has to be arranged as a SOAP remote procedure call.

The value "literal" of the attribute use of the soap:body element indicates that each WSDL part references a concrete schema definition namely the MSS XML-Schema. The type referenced by the part becomes the schema type of the enclosing element, i.e. the part accessor element.

The service element combines a group of related ports.

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:tns="http://new.webservice.namespace"
targetNamespace="http://new.webservice.namespace">

<import namespace="http://uri.etsi.org/2003/v1.1.2#" location="MSS.xsd"/>
    <message name="MSS_SignatureInput">
        <part name="MSS_SignatureReq" type="mss:MSS_SignatureReqType"/>
    </message>
    <message name="MSS_SignatureOutput">
        <part name="MSS_SignatureResp" type="mss:MSS_SignatureRespType"/>
    </message>

<portType name="MSS_SignaturePortType">
        <operation name="MSS_Signature">
            <input message="tns:MSS_SignatureInput"/>
            <output message="tns:MSS_SignatureOutput"/>
        </operation>
    </portType>

<binding name="MSS_SignatureBinding" type="tns:MSS_SignaturePortType">
        <soap:binding style="rpc"
                                                transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="MSS_Signature">
            <soap:operation soapAction="uri" style="rpc"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>

<service name="MSS_SignatureService">
        <port name="MSS_SignaturePort" binding="tns:MSS_SignatureBinding">
            <soap:address location="http://www.Example-MSSP.com"/>
        </port>
    </service>

</definitions>
```

10.6 Error handling

SOAP 1.2 specification has standardized a generic SOAP Fault message (see clause 5.4 of SOAP 1.2 Part1 [9]) in order to carry error information. When an error occurs after receiving the input message of a SOAP method, the receiver sends back a SOAP Fault message instead of the output message that correspond to the input message.

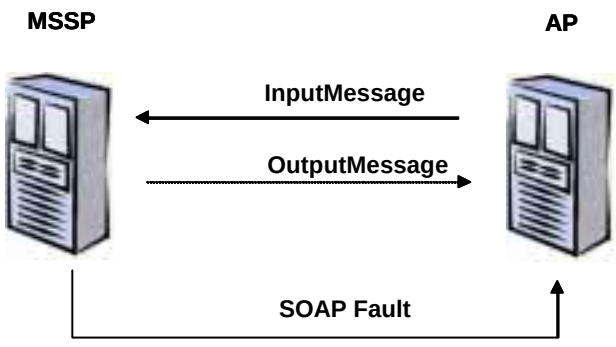


Figure 18: Error handling

The SOAP Fault message is a classic SOAP message (with a SOAP envelop…). The SOAP Body contains a Fault XML element, which contains itself the following elements:

- **Code** element has a value, which can be among five SOAP standard codes, and contains application specific subcodes.
- **Reason** element provides a human readable explanation of the fault.
- **Node** element provides information about which SOAP node on the SOAP message path caused the fault to happen (see TS 102 207 [14]).
- **Role** element identifies the role the node was operating in at the point the error occurred (see TS 102 207 [14]).
- **Detail** is intended for carrying application specific error.

The SOAP standard codes are:

- VersionMismatch.
- MustUnderstand.
- DataEncodingUnknown.
- Sender.
- Receiver.

The Mobile Signature Web Service specifies its own bespoke error subcodes, reasons and details in Annex B:

- Codes 101 to 108 are subcodes of Sender.
- The other codes are under Receiver.

Example 1:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:m="http://www.example.org/timeouts"
              xmlns:xml="http://www.w3.org/XML/1998/namespace">
 <env:Body>
  <env:Fault>
   <env:Code>
     <env:Value>env:Receiver</env:Value>
     <env:Subcode>
      <env:Value>208</env:Value>
     </env:Subcode>
   </env:Code>
   <env:Reason>
     <env:Text xml:lang="en">EXPIRED_TRANSACTION</env:Text>
   </env:Reason>
  </env:Fault>
 </env:Body>
</env:Envelope>
```

Example 2:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:m="http://www.example.org/timeouts"
              xmlns:xml="http://www.w3.org/XML/1998/namespace">
 <env:Body>
  <env:Fault>
   <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
      <env:Value>101</env:Value>
     </env:Subcode>
   </env:Code>
   <env:Reason>
     <env:Text xml:lang="en">WRONG_PARAM</env:Text>
   </env:Reason>
  </env:Fault>
 </env:Body>
</env:Envelope>
```

11 Web Service: Security and Privacy Considerations

We reused in this clause RFC 3552 [21] and draft-iab-sec-mech although it is still in a "work in progress" stage (see bibliography) because they provide a very good guidance for security considerations and mechanisms for internet protocols. A web service implemented over SOAP over HTTP can be considered as an Internet protocol, and the security considerations in this clause are limited to the scope of this protocol. The reader will find in TS 102 206 [13] a further analysis of the overall Mobile Signature System itself.

11.1 Handshake

Security considerations lead to a good understanding of the potential countermeasures that should be implemented according to a threat model. However, the purpose of the present document is to ensure interoperability between different implementations of our protocol. Therefore, we must identify mandatory security mechanisms.

It is important to understand that mandatory mechanisms can be mandatory to **implement** or mandatory to **use** (which includes also the **implement** feature). It is not specially mandatory that endusers actually use these mechanisms. If the parties know that they are deploying a protocol over a "secure" network, then they may choose to disable security mechanisms that they believe are adding insufficient value as compared to their performance cost. On the contrary, some basic security mechanisms must be systematically used and will therefore be mentioned as mandatory to use.

Thus, it is up to the parties involved in the processing of a Mobile Signature transaction to negotiate the use of these mechanisms taking into considerations factors that are not only security oriented. This is the purpose of the Handshake method we have specified in the sections above, and the security considerations below aim at explaining the content of this method.

11.2 Security and privacy

11.2.1 Purposes

Confidentiality, integrity and non-repudiation are the means to protect a web service against:

- Unauthorized usage.
- Inappropriate usage.
- Denial of Service attack.

Some privacy and confidentiality issues may also be raised by the AP or the signer, e.g. on the Data To Be Signed or the signer's identifiers etc. In that case, the means or the target is "End to End Security" between the signer and the AP (the Relying party), which want to protect themselves from MSSP eavesdropping.

There are not so many solutions to achieve end to end security. The AP needs to encrypt the Data To Be Signed with the signer's secret key (in case of SKI) or with the signer's public key (in case of PKI). On the way back, the signer should encrypt the Mobile Signature or omit the plain text data.

However, the reader should keep in mind that commercial agreements are assumed to rule relationships between the AP, MSSP and Signers. That is the reason why we only deal with confidentiality for the signer and the AP. We assume that the MSSP is also interested in the integrity of the data received from the AP, because of the quality of service relies on it. Anyway, there should be a trust relationship with privacy rules between the Signer, the MSSP and the AP.

Moreover, as described in TS 102 207 [14], the messages of this Mobile Web Service Interface can be routed through several routing entities before reaching the appropriate ultimate receiver. In that case, there may be no commercial agreements between the ultimate receiver (the Home MSSP) and the initial sender (the AP). In that case, the non-repudiation techniques such as mobile signatures must be considered.

11.2.2 Simplified threat model for Mobile Signature Web Service

As described in RFC 3552 [21] the Internet Threat model is nowadays fairly well understood and we assume that the end-systems engaging in a protocol exchange have not themselves been compromised. By contrast, we assume that the attacker has nearly complete control of the communications channel over which the end-systems communicate.

Classic attacks occurring on the internet protocols are: Confidentiality violations, password sniffing, offline cryptographic attacks, replay attacks, message insertion, message deletion, message modification, Man-In-The-Middle.

11.2.3 Security framework

The security framework is a set of countermeasures related to the attacks described above:

- TLS RFC 2246 [4]: prevents from confidentiality violations, message insertion, message modification, password sniffing, and Man-In-The-Middle attacks
  - Server Authentication: MSSP has a X509v3 Certificate.
  - Mutual Authentication: Both MSSP and AP have a X509v3 Certificate.
- Digital signature: XML signatures are further described in clause 11.3. Broadly speaking, they provide Object (in opposition to channel security provided by TLS) security through a Mesh (see TS 102 207 [14]), and they aim at providing non-repudiation between the MSSP and the AP.
- AP and MSSP identifiers allow mutual recognition.
- AP password provides AP authentication in case where only TLS Server authentication is used.
- Instant and transaction number mechanisms prevent from replay attack.

Table 1: Security mechanisms

Security mechanisms	Mandatory to implement		Mandatory to use	
	MSSP	AP	MSSP	AP
TLS RFC 2246 [4] server authentication	Y	Y	Y	Y
TLS RFC 2246 [4] mutual authentication	Y	N	N	N
XML Signatures	Y	N	N	N
AP and MSSP identifiers	Y	Y	Y	Y
AP password	Y	Y	Y	Y
Instant	Y	Y	Y	Y
Transaction number	Y	Y	Y	Y

11.3 XML Signatures

In order to bring digital XML signatures as a security mechanism in our protocol, which is specified over SOAP over HTTP, we reuse the "SOAP Extensibility Model" of SOAP 1.2 [8]. This is just a matter of putting an XML signature in a header block within the Header of the SOAP Envelope.

In effect, the following header block is added in the SOAP message:

```
<xs:element name="MSS_MessageSignature">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="ds:Signature"/>
    </xs:sequence>
    <xs:attribute ref="env:role" use="required"/>
    <xs:attribute ref="env:mustUnderstand" use="required"/>
</xs:complexType>
</xs:element>
```

The MSS_MessageSignature element has a child that is a XML signature as described in the W3C Recommendation [19], and two attributes are added:

- Role attribute is used to indicate the SOAP node to which MSS_MessageSignature header block is targeted. Roles with respect to the Mobile Signature (Roaming) Service are specified in TS 102 207 [14].
- MustUnderstand attribute is used to indicate whether the processing of the MSS_MessageSignature header block is mandatory or optional.

The XML signature is a digital signature of the SOAP Body.

When a conforming MSSP or AP (or a Mesh member, see TS 102 207 [14]) receives a SOAP message containing a MSS_MessageSignature header block, and the role of the receiver matches the role attribute, then the receiver MUST try to validate the signature using the processing model of XML Signature [19].

One way to create a MSS_MessageSignature header entry is as follows:

1) Prepare the target SOAP envelope with the body and necessary headers.
2) Create a template of a <ds:Signature> element. The template is assumed to contain empty contents for <ds:DigestValue> and <ds:SignatureValue> elements.
3) Create a new Header entry MSS_MessageSignature and add the template to this entry.
4) Add the header entry MSS_MessageSignature to the SOAP Header.
5) Add the SOAP "Role" and "mustUnderstand" attributes to the entry.
6) Calculate the <ds:DigestValue> and <ds:SignatureValue> elements of the SOAP Body.

Note that we use XML Signatures in the SOAP messages in order to achieve non-repudiation between the MSSP and the AP. However, the easiest way for the signing party to repudiate the message is by claiming that his private key has been compromised and that some attacker (though not necessarily the relying party) signed the disputed message. In order to defend against this attack the relying party needs to demonstrate that the signing party's key had not been compromised at the time of the signature. This requires substantial infrastructure, including archival storage of certificate revocation information and timestamp servers to establish the time that the message was signed. All this is outside the scope of the present document.

Therefore, the specification of an additional header block in the SOAP header of SOAP messages is just a step forward non-repudiation, and it must not be considered as the last one.

11.4 Mobile signatures

The Web interface we specify in the present document can carry various types of digital signatures. The inappropriate and unauthorized usage of a digital signature is a security threat for the signer. However, it is up to the MSSP to define the level of security of the digital signatures format it can provide. It is up to the AP, which is the Relying Party of the Mobile Signature, to ask for a particular Signature Policy. Once again, commercial agreements between MSSP, AP and signers should cover this kind of threats.

11.5 Security protocols

As described previously, TLS RFC 2246 [4] is used in the present document in order to provide channel security. In effect, the Mobile Signature Web Service Interface is specified over SOAP over HTTP over TLS. TLS provides an encrypted, authenticated channel that runs on top of TCP. Generally the server side is always authenticated by a certificate. Clients may possess certificates too, providing mutual authentication. These different levels of TLS are used according to the needs of the AP and the MSSP (see Table 1).

Furthermore, TLS uses different cryptographic algorithms for, broadly speaking, signature and encryption. Accordingly, there is a negotiation between the client and the server about these algorithms that will be used for the channel security. This is called the Cipher suite. Therefore, it is susceptible to downgrade attacks. In a downgrade attack, an active attacker tampers with the negotiation in order to force the parties to negotiate weaker protection than they otherwise would. In order to prevent this kind of attack, the present document mandates a list of negotiable cipher suites:

- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA

Annex A (normative): XML Schema

```
<xs:schema targetNamespace="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:env="http://www.w3.org/2003/05/soap-envelope" elementFormDefault="qualified">
    <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"/>
    <xs:import namespace="http://www.w3.org/2001/04/xmlenc#"/>
    <xs:import namespace="http://www.w3.org/2003/05/soap-envelope"/>

    <xs:complexType name="MessageAbstractType" abstract="true">
        <xs:sequence>
            <xs:element name="AP_Info">
                <xs:complexType>
                    <xs:attribute name="AP_ID" type="xs:anyURI" use="required"/>
                    <xs:attribute name="AP_TransID" type="xs:NCName" use="required"/>
                    <xs:attribute name="AP_PWD" type="xs:string" use="required"/>
                    <xs:attribute name="Instant" type="xs:dateTime" use="required"/>
                    <xs:attribute name="AP_URL" type="xs:anyURI" use="optional"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="MSSP_Info">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="MSSP_ID" type="mss:MeshMemberType"/>
                    </xs:sequence>
                    <xs:attribute name="Instant" type="xs:dateTime" use="optional"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="MajorVersion" type="xs:integer" use="required"/>
        <xs:attribute name="MinorVersion" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:element name="MSS_SignatureReq" type="mss:MSS_SignatureReqType"/>
    <xs:complexType name="MSS_SignatureReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                    <xs:element name="DataToBeSigned" type="mss:DataType"/>
                    <xs:element name="DataToBeDisplayed" type="mss:DataType" minOccurs="0"/>
                    <xs:element name="SignatureProfile" type="mss:mssURIType" minOccurs="0"/>
                    <xs:element name="AdditionalServices" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Service" type="mss:AdditionalServiceType"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="MSS_Format" type="mss:mssURIType" minOccurs="0"/>
                    <xs:element name="KeyReference" type="mss:KeyReferenceType" minOccurs="0"/>
                    <xs:element name="SignatureProfileComparison"
type="mss:SignatureProfileComparisonType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="ValidityDate" type="xs:dateTime" use="optional"/>
                <xs:attribute name="TimeOut" type="xs:positiveInteger" use="optional"/>
                <xs:attribute name="MessagingMode" type="mss:MessagingModeType" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_SignatureResp" type="mss:MSS_SignatureRespType"/>
    <xs:complexType name="MSS_SignatureRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                    <xs:element name="MSS_Signature" type="mss:SignatureType" minOccurs="0"/>
                    <xs:element name="SignatureProfile" type="mss:mssURIType" minOccurs="0"/>
```

```
                    <xs:element name="Status" type="mss:StatusType"/>
                </xs:sequence>
                <xs:attribute name="MSSP_TransID" type="xs:NCName" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_StatusReq" type="mss:MSS_StatusReqType"/>
    <xs:complexType name="MSS_StatusReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:attribute name="MSSP_TransID" type="xs:NCName" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_StatusResp" type="mss:MSS_StatusRespType"/>
    <xs:complexType name="MSS_StatusRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                    <xs:element name="MSS_Signature" type="mss:SignatureType" minOccurs="0"/>
                    <xs:element name="Status" type="mss:StatusType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_RegistrationReq" type="mss:MSS_RegistrationReqType"/>
    <xs:complexType name="MSS_RegistrationReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                    <xs:element name="EncryptedData" type="xenc:EncryptedType" minOccurs="0"/>
                    <xs:element name="EncryptResponseBy" type="xs:anyURI" minOccurs="0"/>
                    <xs:element name="CertificateURI" type="xs:anyURI" minOccurs="0"/>
                    <xs:element name="X509Certificate" type="xs:base64Binary" minOccurs="0"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_RegistrationResp" type="mss:MSS_RegistrationRespType"/>
    <xs:complexType name="MSS_RegistrationRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="Status" type="mss:StatusType"/>
                    <xs:element name="EncryptedData" type="xenc:EncryptedType" minOccurs="0"/>
                    <xs:element name="CertificateURI" type="xs:anyURI" minOccurs="0"/>
                    <xs:element name="X509Certificate" type="xs:base64Binary" minOccurs="0"/>
                    <xs:element name="PublicKey" type="xs:base64Binary" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_ProfileReq" type="mss:MSS_ProfileReqType"/>
    <xs:complexType name="MSS_ProfileReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_ProfileResp" type="mss:MSS_ProfileRespType"/>
    <xs:complexType name="MSS_ProfileRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
```

```
                    <xs:element name="SignatureProfile" type="mss:mssURIType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="Status" type="mss:StatusType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_ReceiptReq" type="mss:MSS_ReceiptReqType"/>
    <xs:complexType name="MSS_ReceiptReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="MobileUser" type="mss:MobileUserType"/>
                    <xs:element name="Status" type="mss:StatusType" minOccurs="0"/>
                    <xs:element name="Message" type="mss:DataType" minOccurs="0"/>
                    <xs:element name="SignedReceipt" type="ds:SignatureType" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="MSSP_TransID" type="xs:NCName" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_ReceiptResp" type="mss:MSS_ReceiptRespType"/>
    <xs:complexType name="MSS_ReceiptRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="Status" type="mss:StatusType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="MSS_HandshakeReq" type="mss:MSS_HandshakeReqType"/>
    <xs:complexType name="MSS_HandshakeReqType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="SecureMethods">
                        <xs:complexType>
                            <xs:attribute name="MSS_Signature" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Registration" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Notification" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_ProfileQuery" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Receipt" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Status" type="xs:boolean" use="required"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="Certificates">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Certificate" type="xs:base64Binary" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="RootCAs">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="DN" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="SignatureAlgList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Algorithm" type="mss:mssURIType" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

```
    <xs:element name="MSS_HandshakeResp" type="mss:MSS_HandshakeRespType"/>
    <xs:complexType name="MSS_HandshakeRespType">
        <xs:complexContent>
            <xs:extension base="mss:MessageAbstractType">
                <xs:sequence>
                    <xs:element name="SecureMethods">
                        <xs:complexType>
                            <xs:attribute name="MSS_Signature" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Registration" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Notification" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_ProfileQuery" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Receipt" type="xs:boolean" use="required"/>
                            <xs:attribute name="MSS_Status" type="xs:boolean" use="required"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="MatchingMSSPCertificates">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Certificate" type="xs:base64Binary" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="MatchingAPCertificates">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Certificate" type="xs:base64Binary" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="MatchingSigAlgList">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Algorithm" type="mss:mssURIType" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="MSSP_TransID" type="xs:NCName" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="MobileUserType">
        <xs:sequence>
            <xs:element name="IdentityIssuer" type="mss:MeshMemberType" minOccurs="0"/>
            <xs:element name="UserIdentifier" type="xs:string" minOccurs="0"/>
            <xs:element name="HomeMSSP" type="mss:MeshMemberType" minOccurs="0"/>
            <xs:element name="MSISDN" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="DataType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="MimeType" type="xs:string" use="optional"/>
                <xs:attribute name="Encoding" type="xs:string" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:simpleType name="SignatureProfileComparisonType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="exact"/>
            <xs:enumeration value="minimum"/>
            <xs:enumeration value="better"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="MessagingModeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="synch"/>
            <xs:enumeration value="asynchClientServer"/>
            <xs:enumeration value="asynchServerServer"/>
        </xs:restriction>
    </xs:simpleType>
```

```
    <xs:complexType name="DigestAlgAndValueType">
        <xs:sequence>
            <xs:element name="DigestMethod" type="ds:DigestMethodType" minOccurs="0"/>
            <xs:element name="DigestValue" type="ds:DigestValueType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="mssURIType">
        <xs:sequence>
            <xs:element name="mssURI" type="xs:anyURI"/>
            <xs:element name="DigestAlgAndValue" type="mss:DigestAlgAndValueType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MeshMemberType">
        <xs:sequence>
            <xs:element name="DNSName" type="xs:string" minOccurs="0"/>
            <xs:element name="IPAddress" type="xs:string" minOccurs="0"/>
            <xs:element name="URI" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="IdentifierString" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="KeyReferenceType">
        <xs:sequence>
            <xs:element name="CertificateURL" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="CertificateIssuerDN" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="HashOfUsersPublicKey" type="mss:DigestAlgAndValueType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="HashOfCAPublicKey" type="mss:DigestAlgAndValueType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SignatureType">
        <xs:choice>
            <xs:element name="XMLSignature" type="ds:SignatureType"/>
            <xs:element name="Base64Signature" type="xs:base64Binary"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"/>
            <!-- this can also be an advanced XML Signature-->
        </xs:choice>
    </xs:complexType>

    <xs:element name="MSS_MessageSignature">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="ds:Signature"/>
        </xs:sequence>
        <xs:attribute ref="env:role" use="required"/>
        <xs:attribute ref="env:mustUnderstand" use="required"/>
    </xs:complexType>
    </xs:element>

    <xs:complexType name="AdditionalServiceType">
        <xs:sequence>
            <xs:element name="Description" type="mss:mssURIType"/>
            <xs:element name="Entity" type="mss:MeshMemberType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="StatusType">
        <xs:sequence>
            <xs:element name="StatusCode" type="mss:StatusCodeType"/>
            <xs:element name="StatusMessage" type="xs:string" minOccurs="0"/>
            <xs:element name="StatusDetail" type="mss:StatusDetailType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="StatusCodeType">
        <xs:sequence>
            <xs:element name="StatusCode" type="mss:StatusCodeType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="Value" type="xs:integer" use="required"/>
```

```
    </xs:complexType>

    <xs:complexType name="StatusDetailType">
        <xs:sequence>
            <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Annex B (normative): SOAP FAULT Subcodes

Subcode	Reason	Detail
101	WRONG_PARAM	Error among the arguments of the request
102	MISSING_PARAM	An argument in the request is missing
103	WRONG_DATA_LENGTH	The DataToBeSigned are too large. Limitations are due to the Mobile Signature technology implemented by the MSSP
104	UNAUTHORIZED_ACCESS	The AP is unknown or the password is wrong or the AP asks for an additional service for which it has not subscribed
105	UNKNOWN_CLIENT	The enduser targeted by the AP is unknown from the MSSP
106	HANDSHAKE_REQUIRED	The MSSP wants prior to negotiate with the AP the use of XML signatures in the messages
107	INAPPROPRIATE_DATA	MSSP cannot handle given MIME Type or encoding style of the DataToBeSigned or DataToBeDisplayed
108	INCOMPATIBLE_INTERFACE	the minor version and/or major version parameters are inappropriate for the receiver of the message
109	UNSUPPORTED_PROFILE	The AP has specified a Mobile Signature Profile that the MSSP does not support
208	EXPIRED_TRANSACTION	Transaction Expiry date has been reached or Time out has elapsed
209	OTA_ERROR	The MSSP has not succeeded to contact the enduser's mobile equipment (Bad connection…)
401	USER_CANCEL	The client has cancelled the transaction
402	PIN_NR_BLOCKED	Error during the Mobile Signature process on the Mobile equipment
403	CARD_BLOCKED	
404	NO_KEY_FOUND	
405	NO_URL_FOUND	
406	PB_SIGNATURE_PROCESS	
407	REGISTRATION_NOK	
422	NO_CERT_FOUND	No certificate has been found for this msisdn
423	CRL_PB	Error during the Certificate verification. The platform does not assume if the certificate is revoked or not
424	CRL_EXPIRED	
425	ERROR_CERTIFICATE	
900	INTERNAL_ERROR	Unknown Error

Annex C (normative): MSS Status Codes

Value	Message	Detail
100	REQUEST_OK	Request accepted by the receiver (MSSP or AP)
400	USER_SIGN	The enduser has received the Mobile Signature request on his Mobile equipment
408	REGISTRATION_OK	
500	SIGNATURE	A Mobile Signature has been successfully constructed and is available
501	REVOKED_ CERTIFICATE	A Mobile Signature has been successfully constructed. But, the signer's certificate is revoked
502	VALID_SIGNATURE	A Mobile Signature has been successfully constructed and the signature is valid
503	INVALID_SIGNATURE	A XML Signature has been successfully constructed and the signature is not valid
504	OUSTANDING_TRANSACTION	
600	OK with PUSH confirmation	The message has been received and is going to be processed. A confirmation must be sent to the enduser (text SMS)
601	OK without PUSH confirmation	The message has been received and is going to be processed. No confirmation to the enduser
602	NOK with PUSH Information	Problem when receiving the message. Information must be sent to the enduser
603	NOK without PUSH information	Problem when receiving the message. No Information sent to the enduser

Annex D (informative): Examples

D.1 Mobile Signature request - Response in synchronous mode without XML Signatures

The use-case corresponds to "Wireless LAN Access Control". The enduser is waiting for his flight in the lounge of an airport. The enduser wants to use his laptop and connect to the airport lounge WLAN. When his laptop tries to connect to the WLAN a window opens and asks for enduser's mobile phone number for Mobile Signature Processing. Then, the enduser receives a Signature request on his mobile phone with the following text "Confirm access to the (airport name) WLAN".

For the example, we will use the information below:

- AP information:
  - ID: http://www.Example-AP.com
  - TransID: 203
  - Password: 1AP-PWD2
  - DateTime: 2003-06-24T21:32:00Z
  - URL: http://www.Example-AP.com/
- MSSP information:
  - ID: http://www.Example-MSSP.com
- Signature transaction information:
  - Mobile User ID: MSISDN : +3243934598
  - DataToBeSigned: "`Confirm access to the (airport name) WLAN`"
  - MSS_Format: XML Signature
  - KeyReference: Hash of the CA public key that the airport trusts : j6lwx3rvEPO0vKtMup4NbeVu8nk=
  - ValidityDate: 2003-06-25T21:32:00Z

HTTP POST Request to the MSSP SOAP Method MSS-Signature:

```
POST /MSS_Signature HTTP/1.1
Host: www.Example-MSSP.com
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
        <MSS_SignatureReq xmlns="http://uri.etsi.org/TS102204/v1.1.2#" MajorVersion="1"
MinorVersion="1" MessagingMode="synch" ValidityDate="2003-06-25T21:32:00Z">
            <AP_Info AP_ID="http://www.Example-AP.com" AP_TransID="A203" AP_PWD="1AP-PWD2"
Instant="2003-06-24T21:32:00Z"/>
            <MSSP_Info>
                <MSSP_ID>
                    <URI>
                    http://www.Example-MSSP.com
                </URI>
                </MSSP_ID>
            </MSSP_Info>
            <MobileUser>
```

```
            <MSISDN>
        +3243934598
        </MSISDN>
        </MobileUser>
        <DataToBeSigned MimeType="text/plain" Encoding="8bit">
    Confirm access to the (airport name) WLAN
    </DataToBeSigned>
        <MSS_Format>
            <mssURI>
    http://uri.etsi.org/TS102204/v1.1.2#XMLSignature
    </mssURI>
        </MSS_Format>
        <KeyReference>
            <HashOfCAPublicKey>
                <DigestValue>
            j6lwx3rvEPO0vKtMup4NbeVu8nk=
        </DigestValue>
            </HashOfCAPublicKey>
        </KeyReference>
    </MSS_SignatureReq>
</env:Body>
</env:Envelope
```

HTTP Response back to the AP:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
        <MSS_SignatureResp xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
MajorVersion="1" MinorVersion="1" MSSP_TransID="B653">
            <AP_Info AP_ID="http://www.Example-AP.com" AP_TransID="A203" AP_PWD="1AP-PWD2"
Instant="2003-06-24T21:32:00Z"/>
            <MSSP_Info Instant="2003-06-24T21:33:00Z">
                <MSSP_ID>
                    <URI>
            http://www.Example-MSSP.com
        </URI>
                </MSSP_ID>
            </MSSP_Info>
            <MobileUser>
                <UserIdentifier>
            (User's Alias)
            </UserIdentifier>
                <MSISDN>
            +3243934598
        </MSISDN>
            </MobileUser>
            <MSS_Signature>
                <XMLSignature>
                    <ds:SignedInfo>
                        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026">
            </ds:CanonicalizationMethod>
                        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                        <ds:Reference URI="#Body">
                            <ds:Transforms>
                                <ds:Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026"/>
                            </ds:Transforms>
                            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                            <ds:DigestValue>c7kab5rvABO0vKtNVq5McfUu8nk=</ds:DigestValue>
                        </ds:Reference>
                    </ds:SignedInfo>
                    <ds:SignatureValue>NDPaEEgRVLtRlk=</ds:SignatureValue>
                </XMLSignature>
            </MSS_Signature>
            <Status>
                <StatusCode Value="500"/>
                <StatusMessage>
            SIGNATURE
            </StatusMessage>
            </Status>
```

```
        </MSS_SignatureResp>
    </env:Body>
</env:Envelope>
```

D.2 Mobile Signature Request - Response with an error

HTTP POST Request to the MSSP SOAP Method MSS-Signature:

```
POST /MSS_Signature HTTP/1.1
Host: www.Example-MSSP.com
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
        <MSS_SignatureReq xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" MajorVersion="1" MinorVersion="1"
MessagingMode="synch" ValidityDate="2003-06-25T21:32:00Z">
            <AP_Info AP_ID="http://www.Example-AP.com" AP_TransID="A203" AP_PWD="1AP-PWD2"
Instant="2003-06-24T21:32:00Z"/>
            <MSSP_Info>
                <MSSP_ID>
                    <URI>
                    http://www.Example-MSSP.com
                </URI>
                </MSSP_ID>
            </MSSP_Info>
            <MobileUser>
                <MSISDN>
            +3243934598
            </MSISDN>
            </MobileUser>
            <DataToBeSigned MimeType="text/plain" Encoding="8bit">
        Confirm access to the (airport name) WLAN
        </DataToBeSigned>
            <MSS_Format>
                <mssURI>http://uri.etsi.org/TS102204/v1.1.2#XMLSignature</mssURI>
            </MSS_Format>
            <KeyReference>
                <HashOfCAPublicKey>
                    <DigestValue>
                    j6lwx3rvEPO0vKtMup4NbeVu8nk=
            </DigestValue>
                </HashOfCAPublicKey>
            </KeyReference>
        </MSS_SignatureReq>
    </env:Body>
</env:Envelope>
```

HTTP Response back to the AP:

The following message is a SOAP Fault message, because the MobileUser information is missing in the request above.

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
        <env:Fault>
            <env:Code>
                <env:Value>env:Sender</env:Value>
                <env:Subcode>
                    <env:Value>102</env:Value>
                </env:Subcode>
            </env:Code>
            <env:Reason>
                <env:Text xml:lang="en">MISSING_PARAM</env:Text>
            </env:Reason>
        </env:Fault>
```

```
    </env:Body>
</env:Envelope>
```

D.3 Mobile Signature Request - Response in Asynchronous Client-Server mode with XML Signatures

HTTP POST Request to the MSSP SOAP Method MSS-Signature:

```
POST /MSS_Signature HTTP/1.1
Host: www.Example-MSSP.com
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <MSS_MessageSignature xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" env:role="http://www.w3.org/2003/05/soap-
envelope/role/ultimateReceiver" env:mustUnderstand="true">
            <XMLSignature>
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026">
                </ds:CanonicalizationMethod>
                    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#Body">
                        <ds:Transforms>
                            <ds:Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026"/>
                        </ds:Transforms>
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>a6lwb3rv4jO4iKtGsd4NbeVu8nk=</ds:DigestValue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>AFRC5erVLtLjk=...</ds:SignatureValue>
            </XMLSignature>
        </MSS_MessageSignature>
    </env:Header>
    <env:Body>
        <MSS_SignatureReq xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" MajorVersion="1" MinorVersion="1"
MessagingMode="asynchClientServer" ValidityDate="2003-06-25T21:32:00Z">
            <AP_Info AP_ID="http://www.Example-AP.com" AP_TransID="A203" AP_PWD="1AP-PWD2"
Instant="2003-06-24T21:32:00Z"/>
            <MSSP_Info>
                <MSSP_ID>
                    <URI>http://www.Example-MSSP.com</URI>
                </MSSP_ID>
            </MSSP_Info>
            <MobileUser>
                <MSISDN>
            +3243934598
            </MSISDN>
            </MobileUser>
            <DataToBeSigned MimeType="text/plain" Encoding="8bit">
        Confirm access to the (airport name) WLAN
        </DataToBeSigned>
            <MSS_Format>
                <mssURI>http://uri.etsi.org/TS102204/v1.1.2#XMLSignature</mssURI>
            </MSS_Format>
            <KeyReference>
                <HashOfCAPublicKey>
                    <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
                </HashOfCAPublicKey>
            </KeyReference>
        </MSS_SignatureReq>
    </env:Body>
</env:Envelope>
```

HTTP Response back to the AP:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <MSS_MessageSignature xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" env:role="http://www.w3.org/2003/05/soap-
envelope/role/ultimateReceiver" env:mustUnderstand="true">
            <XMLSignature>
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026">
                </ds:CanonicalizationMethod>
                    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#Body">
                        <ds:Transforms>
                            <ds:Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026"/>
                        </ds:Transforms>
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>MC0CFFrVLtRlk=</ds:SignatureValue>
            </XMLSignature>
        </MSS_MessageSignature>
    </env:Header>
    <env:Body>
        <MSS_SignatureResp xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" MajorVersion="1" MinorVersion="1" MSSP_TransID="B653">
            <AP_Info AP_ID="http://www.Example-AP.com" AP_TransID="A203" AP_PWD="1AP-PWD2"
Instant="2003-06-24T21:32:00Z"/>
            <MSSP_Info Instant="2003-06-24T21:32:02Z">
                <MSSP_ID>
                    <URI>http://www.Example-MSSP.com</URI>
                </MSSP_ID>
            </MSSP_Info>
            <MobileUser>
                <MSISDN>
            +3243934598
            </MSISDN>
            </MobileUser>
            <Status>
                <StatusCode Value="100"/>
                <StatusMessage>
            REQUEST_OK
            </StatusMessage>
            </Status>
        </MSS_SignatureResp>
    </env:Body>
</env:Envelope>
```

HTTP POST Request to the MSSP SOAP Method MSS-Status:

```
POST /MSS_Status HTTP/1.1
Host: www.Example-MSSP.com
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <MSS_MessageSignature xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"env:role="http://www.w3.org/2003/05/soap-
envelope/role/ultimateReceiver" env:mustUnderstand="true">
            <XMLSignature>
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026">
                </ds:CanonicalizationMethod>
                    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#Body">
                        <ds:Transforms>
```

```
                            <ds:Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026"/>
                        </ds:Transforms>
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>MC0CFFrVLtRlk=</ds:SignatureValue>
            </XMLSignature>
        </MSS_MessageSignature>
    </env:Header>
    <env:Body>
        <MSS_StatusReq xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" MajorVersion="1" MinorVersion="1" MSSP_TransID="B653">
            <AP_Info AP_ID="http://www.Example-AP.com" AP_TransID="A203" AP_PWD="1AP-PWD2"
Instant="2003-06-24T21:33:32Z"/>
            <MSSP_Info>
                <MSSP_ID>
                    <URI>http://www.Example-MSSP.com</URI>
                </MSSP_ID>
            </MSSP_Info>
        </MSS_StatusReq>
    </env:Body>
</env:Envelope>
```

HTTP Response back to the AP:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>

        <MSS_MessageSignature xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" env:role="http://www.w3.org/2003/05/soap-
envelope/role/ultimateReceiver" env:mustUnderstand="true">

            <XMLSignature>
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026">
                </ds:CanonicalizationMethod>
                    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                    <ds:Reference URI="#Body">
                        <ds:Transforms>
                            <ds:Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026"/>
                        </ds:Transforms>
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>MC0CFFrVLtRlk=</ds:SignatureValue>
            </XMLSignature>
        </MSS_MessageSignature>
    </env:Header>
    <env:Body>
        <MSS_StatusResp xmlns="http://uri.etsi.org/TS102204/v1.1.2#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" MajorVersion="1" MinorVersion="1">
            <AP_Info AP_ID="http://www.Example-AP.com" AP_TransID="A203" AP_PWD="1AP-PWD2"
Instant="2003-06-24T21:32:32Z"/>
            <MSSP_Info Instant="2003-06-24T21:32:34Z">
                <MSSP_ID><URI>http://www.Example-MSSP.com</URI>

            </MSSP_ID>
            </MSSP_Info>
            <MobileUser>
                <MSISDN>
            +3243934598
            </MSISDN>
            </MobileUser>

            <MSS_Signature>
                <XMLSignature>
                    <ds:SignedInfo>
```

```
                        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026">
            </ds:CanonicalizationMethod>
                        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                        <ds:Reference URI="#Body">
                            <ds:Transforms>
                                <ds:Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-
20001026"/>
                            </ds:Transforms>
                            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                            <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
                        </ds:Reference>
                    </ds:SignedInfo>
                    <ds:SignatureValue>MC0CFFrVLtRlk=</ds:SignatureValue>
                </XMLSignature>
            </MSS_Signature>
            <Status>
                <StatusCode Value="500"/>
                <StatusMessage>
            SIGNATURE
            </StatusMessage>
            </Status>
        </MSS_StatusResp>
    </env:Body>
    </env:Envelope>
```

Annex E (informative): Bibliography

"Security Mechanisms for the Internet" <draft-iab-secmech-03.txt> July 2003 "Work in progress".

EESSI: http://www.ictsb.org/eessi/EESSI-homepage.htm.

History

Document history		
V1.1.4	August 2003	Publication