All-in Signing Service

Reference Guide

Version: 2.15

Contents

1 Introduction 4
1.1 Terms and abbreviations 4
1.2 Referenced documents 6
2 Overview and main scenarios 8
1.3 Overview of Trusted Timestamps and Static CMS Signatures 9
1.4 Overview of On Demand CMS Signatures 9
1.5 Step-Up authentication 10
2 Preconditions and assumptions 11
2.1 Internet access 11
2.2 Certificate based client authentication 11
2.3 Request authorization 12
3 All-in Signing Service Introduction 13
3.1 Communication Modes 13
3.2 Type of Signatures 13
3.3 Signature Size 13
3.4 Adding Trusted Timestamps 13
3.5 Adding Revocation Information (long-term signature) 13
3.6 Declaration of Will (Step-Up Authentication) 15
3.7 Batch Processing 15
3.8 Detached Signature and Verification 15
4 All-in Signing Service Interface 16
4.1 Overview 16
4.2 Interface Description 16
4.3 HTTP/1.1 Header 16
4.4 Swisscom Basic Profile 16
4.5 Document Hash 16
4.6 Signing Options 17
4.7 On Demand Certificate Policy and Certification Practice Statement (CP/CPS) 26
4.8 Templating 27
4.9 Trusted Timestamp 28
4.10 Trusted Timestamp SignRequest 28
4.11 Trusted Timestamp SignResponse 29
4.12 CMS Signatures 31
4.13 CMS SignRequest for Static Signatures 31
4.14 CMS SignRequest for On Demand Signatures 33
4.15 CMS SignResponse 34
4.16 Asynchronous Mode 37
4.17 SignRequest 37
4.18 SignResponse 37
4.19 PendingRequest 38
4.20 PendingResponse 39
4.21 CMS On Demand Signatures with Step-Up Authentication 40
4.22 SignRequest 40
4.23 SignResponse 43
4.23.1 PendingRequest 45
4.23.2 PendingResponse 45
4.23.3 SignResponse (SUCCESS) 45
4.24 Step-up authentication with password-reset 46
4.25 App2App switch 47

4.26 Mobile ID Error Messages Responses 51
4.27 Static Plain Signatures (PKCS#1) 55
4.28 Fault Response Message 56
4.29 Wrong Digest Size (example) 56
4.30 Step-Up Authentication: Mobile ID User Account Problem (example) 57
4.31 Step-Up Authentication: User Cancel (example) 59
4.32 Step-Up Authentication: SerialNumber Mismatch (example) 60
4.33 Best Practices 61
4.34 On Demand Step-Up Pre-Signing Process 61
4.35 Signing Requests 62
4.36 Response handlings 62
4.37 Adobe PDF 62
4.38 Processing OCSP and CRL response elements in the REST API 63
4.39 Processing PDFs for PAdES LTV support 64
5 All-in Signing Service clients 69
5.1 PDFBox AIS client 69
5.2 iText Client 70
5.3 iText Dotnet Client 71
5.4 AIS React Flask 71
6 Appendix 72
6.1 Create self-signed certificate with openssl 72
6.2 Generate Key and CSR 72
6.3 Organization or Organizational Unit 72
6.4 Self-sign it and create your certificate 72
6.5 Convert into PKCS#12 (if needed) 72
6.6 Create self-signed certificate with Java keytool 72
6.7 Generate KeyStore & export the self-signed certificate 72
6.8 Root CA and Intermediate CA certificate import 72
6.9 Verification 72
6.10 Swisscom signed certificate 72
6.11 Result Major and ResultMinor list 74
6.12 Result Major Status Codes 74
6.13 Result Minor Status Codes 74
6.14 Swisscom CA Hierarchy 76
6.15 CMS Signature 76
6.16 Timestamp Signature 76

1 Introduction

The purpose of this document is to give advice and support to integrators, developers and customers who must implement the Swisscom All-in Signing Service, referred to as AIS, based on the OASIS Digital Signature Service (DSS) specification [OASIS DSS].

This manual assumes that you are familiar with general Web Services (SOAP, WSDL/WADL, XML, JSON, and Application Server) as well as with the digital signing topic itself.

1.1 Terms and abbreviations

Abbreviation	Definition
ℹ	Please note.
⚠	Be careful, important.
AIS	Swisscom All-in Signing Service
AP	Application Provider
AS	An Application Server (AS) is a server which provides software applications with services.
Authentication	An authentication process verifies who a person is.
Authorization	An authorization process verifies the access rights of a given person/system and grants access.
CA	Certificate Authority
CMP	Certificate Management Protocol, an Internet protocol for obtaining X.509 digital certificates in a public key infrastructure.
CMS	The Cryptographic Message Syntax (CMS) is a standard for cryptographically protected messages. It can be used to digitally sign, digest, authenticate or encrypt any form of digital data. CMS is based on the syntax of PKCS#7.
CP/CPS	Certificate Policy (CP) and Certification Practice Statement (CPS)
DSS	Digital Signature Service: Declared XML Namespace(s): urn:oasis:names:tc:dss:1.0:core:schema
ERP	Enterprise Resource Planning is a business management software.
Hash value	A mapping of an original document into a smaller one such as a fingerprint made of integer numbers.
HSM	Hardware security module
JSON	JavaScript Object Notation is a text-based open standard designed for human readable data interchange. Although derived from the JavaScript scripting language it is language independent. The JSON format is often used for serializing and transmitting structured data over a network connection, primarily between a server and a web application, as an alternative to XML.
LTV	Digitally signed documents may be used or archived for many years – even many decades. At any time in the future, when the CA will have no obligations to make revocation information available, it must still be possible to verify that the signature was valid at the time it was created – a concept known as Long-Term Validation (LTV).
OASIS	Organization for the Advancement of Structured Information Standards (OASIS), consortium for the development, convergence, and adoption of e-business and web service standards, www.oasis-open.org

Abbreviation	Definition
OCSP	Online Certificate Status Protocol is an Internet protocol used for obtaining the revocation status of a digital certificate.
RESTful	Representational State Transfer is a style of software architecture for distributed systems such as the World Wide Web. It is based on the existing design of HTTP/1.0. REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses.
RFC	A Request for Comments (RFC) is a publication of the Internet Engineering Task Force (IETF) and the Internet Society, the principal technical development, and standards-setting bodies for the Internet.
SOAP	Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information in the implementation of Web Services relying on Extensible Markup Language (XML)
TSA	Time Stamp Authority
WS	A Web Service (WS) is a method of communication between two electronic devices over the Web (Internet). The W3C defines a "Web service" as "a software system designed to support interoperable machine-to-machine interaction over a network". It has an interface described in a machine-processable format (specifically Web Services Description Language, known by the acronym WSDL).
WSDL	The Web Services Description Language (WSDL) is an XML-based language that is used for describing the functionality offered by a Web service. A WSDL description of a web service provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.
X.509	X.509 is a standard for a public key infrastructure. X.509 specifies, amongst other things, standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm.
XML	Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

1.2 Referenced documents

[CP/CPS]	Certification Practice Statement and Certificate Policy https://www.swisscom.ch/de/business/enterprise/angebot/security/digital_certificate_service.html
[MIDSOAP]	Mobile ID Client Reference Guide https://www.mobileid.ch/en/documents - see the technical documents
[OASISDSS]	OASIS DSS Standard http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-spec-v1.0-os.pdf
[PDFSIG]	Digital signatures in Acrobat https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/digisig_in_acrobat.pdf
[RFC6960]	X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP http://www.ietf.org/rfc/rfc6960.txt
[RFC2986]	Certification Request Syntax Specification http://www.ietf.org/rfc/rfc2986.txt
[RFC3161]	X.509 Public Key Infrastructure, Time-Stamp Protocol (TSP) http://www.ietf.org/rfc/rfc3161.txt
[RFC3369]	Cryptographic Message Syntax (CMS) http://www.ietf.org/rfc/rfc3369.txt Note: this RFC has been obsoleted by RFC 3852
[RFC5126]	CMS Advanced Electronic Signatures (CAdES) http://www.ietf.org/rfc/rfc5126.txt
[RFC5652]	Cryptographic Message Syntax (CMS) - obsoletes RFC3369 and RFC3852 http://www.ietf.org/rfc/rfc5652.txt
[RFC3447]	Public-Key Cryptography Standards (PKCS) #1 https://www.ietf.org/rfc/rfc3447.txt
[DCES]	Digital Certificates for Electronic Signatures https://www.swisscom.ch/en/business/enterprise/offer/security/digital_certificate_service.html?file=deutsch%2F002_CPS_SDCS_2_16_756_1_83_Zertifikatsprofile_de.pdf
[RASDN]	Use of evidence attributes in the DN https://github.com/SCS-CBU-CED-IAM/AIS/wiki/Distinguished-Name:-Use-of-Evidence-Attributes
[IFR]	SAS iFrame Embedding Guide https://github.com/SwisscomTrustServices/AIS/wiki/SAS-iFrame-Embedding-Guide
[ETSI TS 119 432]	Electronic Signatures and Infrastructures (ESI); Protocols for remote digital signature creation. https://www.etsi.org/deliver/etsi_ts/119400_119499/119432/01.02.01_60/ts_119432v010201p.pdf
[STSDW]	Swisscom Trust Services Developer Website https://dev.trustservices.swisscom.com/

[SignatureSize]	Signature sizes https://github.com/SwisscomTrustServices/AIS/wiki/Swisscom-CA-4

2 Overview and main scenarios

All-in Signing Service (AIS) allows customers' documents and files to be electronically signed: AIS itself cannot view the files and documents to be signed as only the hash values are transferred to the service. The signature types provided by AIS and applied to the hash values are **Trusted Timestamps** and **Cryptographic Message Syntax (CMS) Signatures**.

Trusted Timestamps

Trusted Timestamps applied to the hash values as signatures by AIS are qualified timestamps provided by a trusted third-party Time Stamp Authority (TSA), according to the [RFC3161] standard. Timestamp signatures are used to prove the existence of certain data at a certain point in time without a person or an organization behind them. This kind of signature is well suited for system transactions and log files.

Additional clarifications can be found in Wikipedia[1].

CMS Signatures

The CMS (PKCS#7) is a standard for cryptographically protected messages. The AIS is using this signature type when it comes to digitally sign the hash values with a customer certificate (X509). This certificate used during this signature process can be either **Static** or **On Demand**. Timestamps can be additionally applied to CMS Signatures to define a trusted point in time or adhere to local regulations. Swiss qualified signatures must include a qualified timestamp.

Static certificates are standard ones proposed and issued by any official Certificate Authority (CA) for the customer and are securely hosted at the AIS on its Hardware Security Module (HSM). After the certificate's registration process, the corresponding customer can address and use it in a secure and exclusive manner. Static certificates are well suited for any organization planning to sign many documents in its name in an automated manner, for example invoices, account listings, archives of documents.

On Demand certificates are context-based issued certificates that will contain the end user information collected at the customer's service side itself. The collected information can be set as attributes in the Distinguished Name (DN) of the short-lived certificate. Before issuing the certificate and using it only for one request, a declaration of will by the signer is enforced. On Demand certificates are well suited for signing documents interactively/online such as contracts, medical assessments, construction permits, tax declarations...

Static Plain (PKCS#1)

RSASSA-PKCS1 [RFC3447] signatures are also provided and can be used for special purposes like for example the signing of EDIFACT or XML documents.

[1] Trusted timestamping: http://en.wikipedia.org/wiki/Trusted_timestamping

1.3 Overview of Trusted Timestamps and Static CMS Signatures

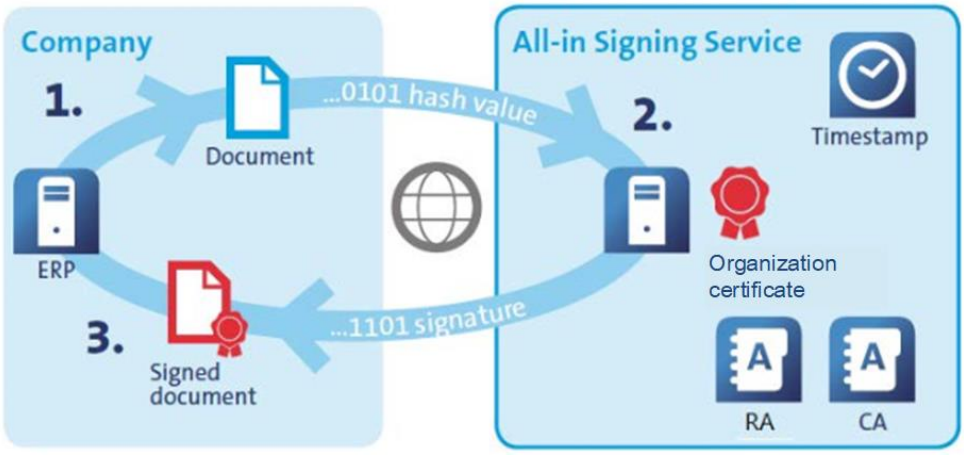

Figure 1. Three step static signature generation.

1. The customer (company) application server (Enterprise Resource Planning, ERP) calculates the hash value of the document to be timestamped/signed and sends it within a signature request to AIS.
2. AIS receives and validates the signature request. It timestamps or signs the hash value with the corresponding customer static certificate.
3. AIS sends the signature back to the customer. The signature is integrated in the document by the application server and stored.

1.4 Overview of On Demand CMS Signatures

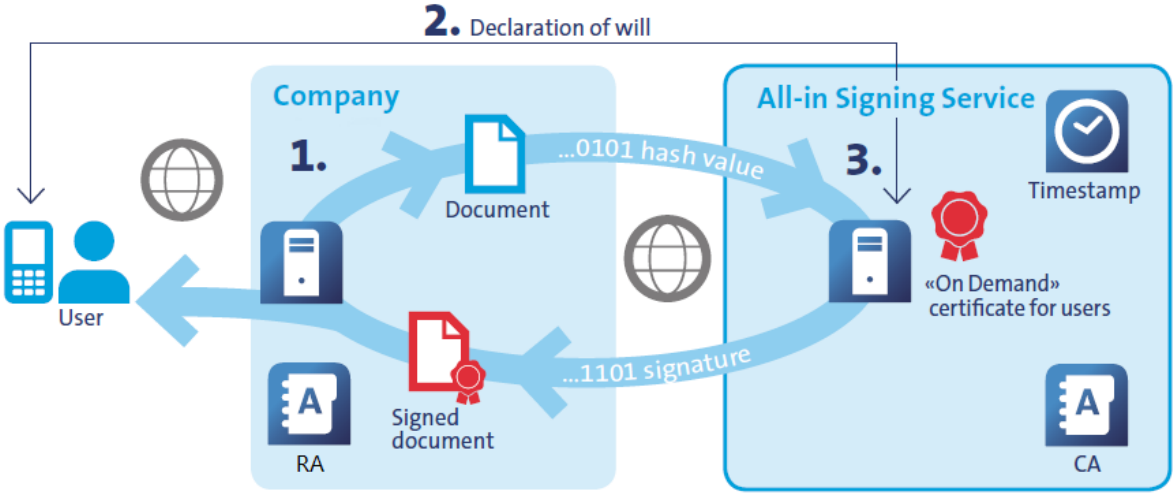

Figure 2. Three steps on demand signature generation.

The context is the following: an employee would like to sign documents interactively/online from the corporate intranet.

1. The customer (company) application server (Enterprise Resource Planning, ERP) calculates the hash value of the document to be signed and sends it within a signature request to AIS.
2. AIS receives and validates the signature request. Optionally, it first performs a declaration of will (step-up authentication: refer to the Chapter **Fehler! Verweisquelle konnte nicht gefunden werden.** f or more details). AIS requests a unique certificate issued by the CA with a short validity period and which will be used only once.
3. It signs the hash value with this certificate.
4. AIS sends the signature back to the customer. The signature itself is integrated by the application server in the related document and stored.

1.5 Step-Up authentication

Step-authentication is an optional client-based setting. If step-up authentication is defined for a client, a step-up authentication will be enforced where the end user must approve the signature request. Currently, there are two available methods:

1. Mobile ID: the end user shall approve the signature and prove sole control through Mobile ID authentication. (Remark: in case the Mobile-ID check returns an error – for example if the Mobile-ID of the user is not activated – the step-up process fallbacks to Password & SMS-Challenge. An enforcement of only MobileID without this fallback is still possible while using the old request payload for the definition of the Step-Up method.
2. Password & SMS-Challenge: the client application must show the content of a unique consent URL delivered by AIS in the signature response. On the site available under this URL, the user shall approve the signature and prove sole control through password and/or SMS-Challenge authentication.

⚠ If step-Up authentication is **not** enforced by Swisscom, the client is responsible to get the appropriate end user consent for the signature creation. The selected method must be documented and approved by Swisscom.

2 Preconditions and assumptions

Before using the All-in Signing Service some prerequisite steps are required.

In this reference guide we assume that:

1. The customer has an agreement with Swisscom and is already provisioned on AIS.
2. The customer has received from Swisscom its AIS Claimed Identities and the relevant CA certificates.

2.1 Internet access

The AIS interface is accessible through Internet. The max number of concurrent sessions is limited to 1500.

If not otherwise specified use the following default access configuration information:

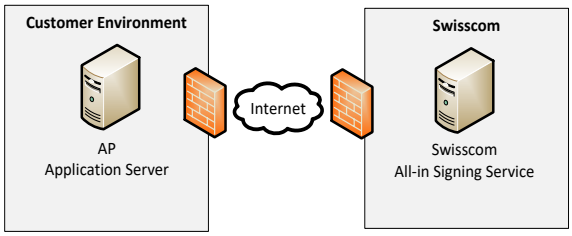

Figure 3. Internet based communication between customer app. server and Swisscom AIS service.

Base-URL

The APIs have a base URL to which the endpoint paths must be appended:

https://ais.swisscom.com

Service Endpoint(s):

SOAP: <**Base-URL**>/AIS-Server/ws

RESTful: <**Base-URL**>/AIS-Server/rs/v1.0/sign
<**Base-URL**>/AIS-Server/rs/v1.0/pending

2.2 Certificate based client authentication

The AIS requires a certificate-based authentication[2] to identify the client, referred as the Application Provider (AP), and grant access:

[2] http://en.wikipedia.org/wiki/Secure_Sockets_Layer#Client-authenticated_TLS_handshake

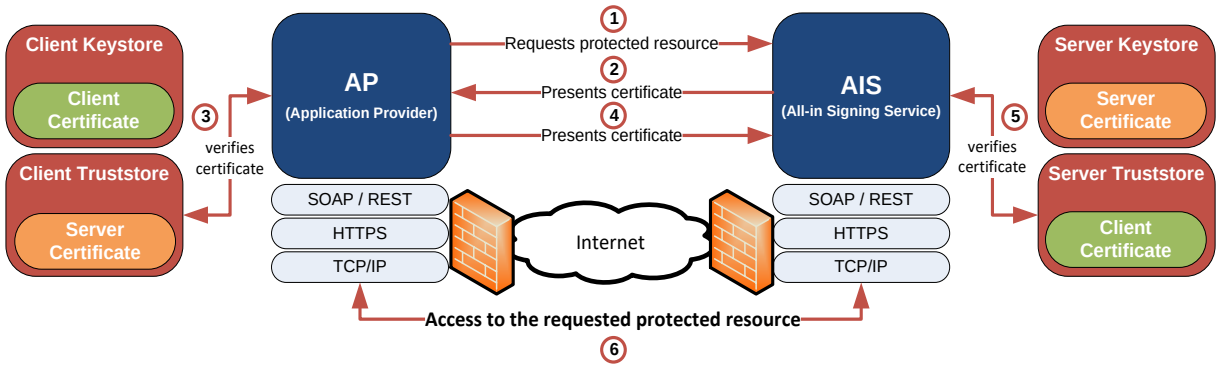

Figure 4. Certificate based client authentication steps and involved protocols.

1. The client AP requests access to a protected resource on AIS.
2. AIS presents its server certificate to the client AP.
3. The client AP optionally verifies the AIS server certificate.
4. If successful, the client AP sends its client certificate to AIS.
5. AIS verifies the AP client certificate.
6. If successful, AIS grants access to the protected resource requested by the client AP.

AIS side authentication does not do any validation of a client certificate chain or restrictions of the root CA. **The client shall send only its end entity certificate**. The authentication is denied in case the client sends the full certificate chain.

The client certificates must contain the value "Client Authentication" in the "Enhanced Key Usage attribute". Chapter 6 contains examples on how to create self-signed certificates.

2.3 Request authorization

AIS provides for each AP one or many Claimed Identities (see 4.6.1.9) to authorize the signing request. Each Claimed Identity must be used for the proper signature type mentioned in Chapter 0.

3 All-in Signing Service Introduction

The All-in Signing Service exposes its services over HTTPS in SOAP and RESTful in an [OASIS DSS] standard compliant form.

The following picture shows the available alternatives, from a protocol-stack point of view:

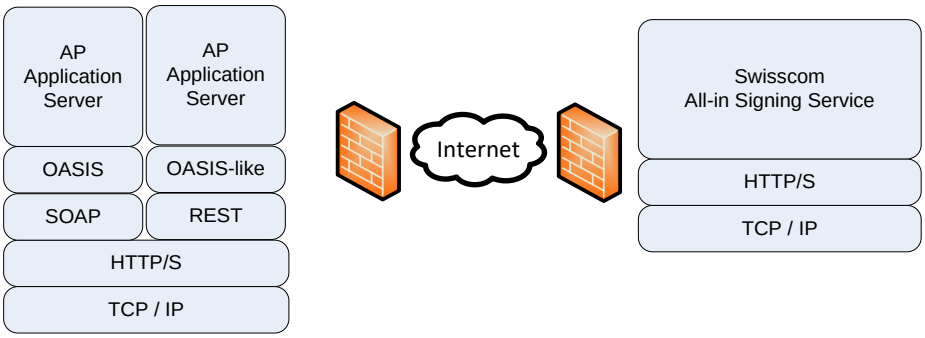

Figure 5. Protocol alternatives for the client w.r.t. the AIS server communication.

3.1 Communication Modes

The standard defines synchronous and asynchronous (client-server) modes and AIS supports both. When placing asynchronous requests, the result must be fetched within 15 minutes otherwise it will be dropped.

3.2 Type of Signatures

As defined in Chapter 0, for both types of supported signatures, the signature part in the response is Base64 encoded and represents either a [RFC3161] compliant Trusted Timestamp or a [RFC3369] / [RFC5652] compliant CMS Signature. See Chapter 4.6.1.1 for further details.

- [OASIS DSS] is using urn:ietf:rfc:3369 for the request definition. AIS is using this to be compliant to the standard itself, but the provided answer is [RFC5652] compliant.

3.3 Signature Size

Currently CMS signatures in binary format require a minimum reserved space of 30000 bytes to embed the signature including necessary information for long term validation. Please note that the size of PEM format can be larger. Timestamps require at least 15000 bytes.

See for more details here [SignatureSize].

3.4 Adding Trusted Timestamps

For CMS signatures, an additional [RFC3161] Trusted Timestamp may be requested and applied. By asking this, the response will additionally contain a Trusted Timestamp. This will define a trusted point in time for the signature. See Chapter 4.6.1.4 for further details.

3.5 Adding Revocation Information (long-term signature)

AIS supports the concept of long-term signature validation (LTV) which allows you to check the validity of a signature long time after the document was signed, when the CA will have no longer any obligations to make revocation information available. To achieve long-term validation, all the required revocation information for signature validation must be embedded in the signed document.

Without revocation information, a signature can be validated for only a limited time. This limitation occurs because certificates related to the signature eventually expire or are revoked. Once a certificate expires, the issuing authority is no longer responsible for providing revocation status on that certificate. Without conforming revocation information, the signature cannot be validated.

Revocation information may be requested for any type of Signature. By asking this, the response will additionally contain certificate status information (signed CRLs or OCSP responses, refer to Chapter 6.14) for the signing certificate chain. See Chapter 4.6.1.5 for further details.

3.6 Declaration of Will (Step-Up Authentication)

A declaration of will (subsequently referred to as *step-up authentication*) is an additional step in the signing process, which enforces the authorization of the requested signature through a method or device, which must remain under the sole control of the signature requestor. This step is mandatory for qualified signatures and optional for advanced ones.

AIS allows the generation of qualified signatures without Step-up authentication. In such a customer configuration, the AP himself is responsible of guaranteeing that the physical person the signing request has been submitted for has given his allowance according to the local regulations. The selected method must be documented and approved by Swisscom.

AIS supports Mobile ID as default consent method and a web-based solution as alternative.

See Chapter 4.21 for technical details regarding the signature request and response for this use case.

3.7 Batch Processing

AIS allows signing multiple hash values with a single request. Remarks:

- Batch signing has currently the limitation of 300 documents per batch. This limitation is since we currently use the Airlock WAF.
- There is no limitation of the number of hash values in a single request.
- If any error occurs during the processing, the entire batch request will fail.
- For On-Demand CMS Signatures, only one certificate is issued and used to sign all hashes.
- The step-up authentication takes place only once and is valid for the release of the entire batch.
- Each hash value can have its own digest algorithm.

A batch must always contain at least two documents. Do not use the batch functionality to send a batch containing one single element.

3.8 Detached Signature and Verification

Since only the document hash is provided to AIS, the returned signature itself is detached from the document per se. If the signature is not embedded into the document, then the signature is called a detached signature. For the verification process, both the detached signature and the document are required.

The verification proves the authentication, the integrity and non-repudiation of the signed hash value, respectively the document. Be aware that the verification process may require Internet connection to the CA online services (to an OCSP Responder or CRL source), unless you have requested revocation information to perform long-term signature validation (see Chapter 3.5 for more details).

There are document formats that support the integration of such detached signatures, e.g., Portable Document Format (PDF). The CMS Signature provided by AIS can be used as is for integrated signatures and to sample code referred in Chapter 5. The customer must perform the integration of the signature into the document by using libraries or partner solutions (referred on AIS website [3]).

[3] http://swisscom.ch/signing-service

4 All-in Signing Service Interface

4.1 Overview

4.2 Interface Description

A client who wants to connect to the All-in Signing Service can read the related **Web Service Description Language (WSDL)** file for SOAP or the **Web Application Description Language (WADL)** file for RESTful to determine what functions are available. Any special data types used are embedded in those files in the form of XML Schemas. It describes how the service can be called, what parameters it expects, and what data structures it returns.

WSDL: https://ais.swisscom.com/AIS-Server/ws/?wsdl

WADL: http://git.io/tPMdpQ

4.3 HTTP/1.1 Header

You can POST signature requests via HTTP/1.1 using the SOAP or RESTful interface. The RESTful interface supports two media types: XML and JavaScript Object Notation (JSON).

The header fields should be set as follows:

HEADER FIELD	SOAP	RESTFUL/XML	RESTFUL/JSON
Content-Type	text/xml;charset=UTF-8	application/xml;charset=UTF-8	application/json;charset=UTF-8
Accept	-	application/xml	application/json

4.4 Swisscom Basic Profile

The Swisscom Basic Profile describes AIS extensions based on the [OASIS DSS] interface. A version number ensures backward compatibility in case of future improvements.

The Profile URI must be specified as an attribute in every <SignRequest> element.

Example:

```
SOAP/XML:   <SignRequest Profile="http://ais.swisscom.ch/1.1">
JSON:       "SignRequest": {"@Profile": "http://ais.swisscom.ch/1.1"}
```

4.5 Document Hash

4.5.1.1 Digest Method

AIS supports the following digest algorithm URIs.

- SHA-256: http://www.w3.org/2001/04/xmlenc#sha256
- SHA-384: http://www.w3.org/2001/04/xmldsig-more#sha384
- SHA-512: http://www.w3.org/2001/04/xmlenc#sha512

Example:

```
SOAP/XML:   <dsig:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
JSON:       "dsig.DigestMethod": {"@Algorithm": "http://www.w3.org/2001/04/xmlenc#sha512"}
```

4.5.1.2 Digest Value

The document hash (digest) value shall be in a **Base64** encoded **binary form**.

Here is an example how to get the Base64 encoded binary SHA-256 digest of a document using OpenSSL:

```
$ openssl dgst -binary -sha256 <document> | openssl enc -base64
-A 1WON4H3Hrinf7LYRNmhV6Uf7apdUvuYEsmhxAklxumA=
```

4.6 Signing Options

With the element <OptionalInput>, it is possible to include additional parameters to each signing request.

- Signature Type, see 4.6.1.1
- Signature Standard, see 4.6.1.3
- Additional Profiles, see 4.6.1.3
- Add Timestamp, see 4.6.1.4
- Add Revocation Information (long-term validation), see 4.6.1.5
- On Demand Distinguished Name, see 4.6.1.6
- On Demand Step Up Authorisation, see 4.6.1.8
- Claimed Identity, see 4.6.1.9
- Templating, see 4.8

4.6.1.1 Signature Type

The element <SignatureType> defines the type of signature to be applied to the hash. There are currently three types of signature provided by AIS. Exactly one type of the following URN must be given in the request:

- **urn:ietf:rfc:3161**
  Creation of a Trusted Timestamp according to [RFC3161]. The response contains a Base64 encoded timestamp token. Optionally it may contain additional **revocation information** (see Chapter 4.6.1.5 for more details).
- **urn:ietf:rfc:3369**
  Creation of a CMS Signature according to [RFC5652]. The response contains a Base64 encoded signature. Optionally it may contain additional embedded **revocation information** (Chapter 4.6.1.5 for more details) and/or an additional **timestamp** (Chapter 4.6.1.4 for more details).
- **urn:ietf:3447 (http://ais.swisscom.ch/1.1/signaturetype/plain)**
  Creation of a static plain signature in PKCS#1 format (according to [RFC3447]).
- In all cases, the response contains the signer's certificate as well as the full certificate chain up to the root certificate.

Example:

```
SOAP/XML:   <SignatureType>urn:ietf:rfc:3161</SignatureType>
JSON:       "SignatureType": "urn:ietf:rfc:3161"
```

4.6.1.2 Signature Standard

The optional element <SignatureStandard> defines the standard of the signature and can take the values:

- "PAdES" (deprecated, please use "PDF")
- "PAdES-baseline" - PAdES compliant signatures, with the OCSP and CRL information returned as separate optional elements in the Sign Response. To get an LTV enabled PDF signature, the client is expected to embed the returned OCSP and CRL content in the final signed PDF. See 4.39 for more details on processing the OCSP and CRL content and for generating LTV enabled signatures, as well as provide PAdES Baseline B-LT(A) quality to the final PDF. Also see 5 for details on the support that the AIS clients provide for doing this embedding automatically for you.
- "PDF" - generates CMS signatures, having the OCSP and CRL information added as a separate revocation info archival attribute inside the signature dictionary.
- "PLAIN" - generates plain RFC 3477 signatures
- "CAdES" - generates CAdES compliant signatures

If not specified, the signature standard defaults to CAdES. The signing time is only included in the signature if the signature standard is CAdES.

For CMS signatures, if the type of the element <AddRevocationInfo> is not specified, the type of the revocation information will be selected based on the specified signature standard. For timestamps, the signature standard does not apply, so the revocation information type must be set explicitly.

CRLs are only available for the Root CAs. For the Issuing Cas and timestamping only OCSP is available.

4.6.1.3 Additional Profiles

The element <AdditionalProfile> enables the following profile enhancements.

mandatory: the profile must be set in the request
optional: the profile is supported and may be set
n/a: the profile is not supported and must not be set

ADDITIONAL PROFILE	URN:IETF:RFC:3161 TRUSTED TIMESTAMP	URN:IETF:RFC:3169 CMS SIGNATURE STATIC	URN:IETF:RFC:3369 CMS SIGNATURE ON DEMAND	URN:IETF:RFC: 3447 STATIC PLAIN PKCS#1
urn:oasis:names:tc:dss:1.0:profiles:timestamping The signature response will only contain a timestamp of the document hash	**Mandatory**	N/A	N/A	N/A
urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing The signature may be processed in asynchronous mode. The response may[4] be a **Pending**[5] result that contains the **ResponseID**. The client must be able to poll the result with a **PendingRequest** (see 4.16).	Optional	Optional	**Mandatory** with step-up auth., Optional otherwise	Optional
http://ais.swisscom.ch/1.1/profiles/redirect Required for step-up authentication. AIS receives an asynchronous call request. After receiving a "pending" response including a Consent URL, the client needs to redirect the user to this URL and to start polling the result with a PendingRequest.	N/A	N/A	**Mandatory** with step-up auth., Optional otherwise	N/A
http://ais.swisscom.ch/1.0/profiles/batchprocessing Required if a single signature request contains multiple document digests. Since you cannot rely on the order of the <DocumentHash> elements in the response, it's mandatory for a request to define a unique ID[6] attribute for each <DocumentHash>.	Optional	Optional	Optional	Optional
http://ais.swisscom.ch/1.0/profiles/ondemandcertificate Used to indicate the use of an On Demand certificate instead of a Static certificate. The signature request shall contain a subject **Distinguished Name** (see 4.6.1.6) and optionally a **Step-Up Authentication** (see 4.6.1.8)	N/A	N/A	**Mandatory**	N/A
http://ais.swisscom.ch/1.1/profiles/plainsignature Used to indicate the use of a static plain PKCS#1 signature.	N/A	N/A	N/A	**Mandatory**

[4] The term ‚may' is used because there is always the possibility of synchronous processing or an error response
[5] urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending
[6] DocumentHash ID is of type xsd:ID - refer to http://books.xmlschemata.org/relaxng/ch19-77151.html

Example:

```
SOAP/XML:   <AdditionalProfile>
                urn:oasis:names:tc:dss:1.0:profiles:timestamping
            </AdditionalProfile>
JSON:       "AdditionalProfile": "urn:oasis:names:tc:dss:1.0:profiles:timestamping"
```

4.6.1.4 Add Timestamp

The element <AddTimestamp Type="urn:ietf:rfc:3161"/> may be added to the signature request to include a timestamp information according to [RFC3161] to the signature response.

This option only applies to the **urn:ietf:rfc:3369** CMS Signature Type.

Example:

```
SOAP/XML:   <AddTimestamp Type="urn:ietf:rfc:3161"/>
JSON:       "AddTimestamp": {"@Type": "urn:ietf:rfc:3161"}
```

4.6.1.5 Add Revocation Information (long-term validation)

The element <AddRevocationInformation type="..."/> may be added to the signature request to include revocation information (RI) in the signature response. The attribute '**type**' supports the following values[7].

TYPE	DESCRIPTION	
Empty	For CMS signatures, if the attribute 'type' is not provided, the revocation information will match the defined SignatureStandard (CAdES or PAdES). (In case the SignatureStandard is not set, the default is CAdES). For timestamps, the signature standard does not apply so the revocation information type must be set explicitly.	
CAdES	RI will be **embedded** as an **unsigned** attribute with OID 1.2.840.113549.1.9.16.2.24	
PAdES	Deprecated, please use PDF. As behaviour, it is identical to PDF.	
PAdES-baseline	For both CMS signatures and timestamps, the OCSP and CRL content is provided as separate response elements in the Sign Response. It is up to the calling client to embed these elements in the final signed PDF. See 4.39 for more details on processing the OCSP and CRL content and for generating LTV enabled signatures, as well as provide PAdES Baseline B-LT(A) quality to the final PDF. Also see 5 for details on the support that the AIS clients provide for doing this embedding automatically for you.	
PDF	CMS Signatures:	RI will be **embedded** in the signature as a **signed** attribute with OID 1.2.840.113583.1.1.8
	Trusted Timestamps:	RI will be provided in the response as Base64 encoded OCSP responses or CRLs within the <OptionalOutputs>-Element
BOTH	Both RI types (CAdES,PAdES) will be provided	

AIS collects revocation information for every certificate in the signing certificate chain. It first tries to fetch an OCSP Response. If no OCSP Responder is available, it will try to fetch the CRL instead. An overview about the Swisscom CA Hierarchy and related OCSP or CRL can be found in Chapter 6.14.

Example:

SOAP/XML: <sc:AddRevocationInformation Type="BOTH"/>
JSON: "sc.AddRevocationInformation":{"@Type": "BOTH"}

The final signature response might contain (depending on the type of revocation information selected) the OCSP and CRL content that corresponds to the certificate used for signing. A sample response looks like the following (some parts are left out):

SOAP/XML Static/On Demand Certificate SignResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                xmlns:ais="http://service.ais.swisscom.com/"
                xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
              Profile="http://ais.swisscom.ch/1.1"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        ...
        <OptionalOutputs>
          <sc:RevocationInformation>
            <sc:CRLs><sc:CRL>CRL_#1</sc:CRL></sc:CRLs>
            <sc:OCSPs><sc:OCSP>OCSP_#1</sc:OCSP></sc:OCSPs>
          </sc:RevocationInformation>
```

[7] The values are case insensitive

```
        </OptionalOutputs>
        ...
      </SignResponse>
    </ais:signResponse>
  </soap:Body>
</soap:Envelope>
```

JSON Static/On Demand Certificate SignResponse

```
{ "SignResponse": {
   ...
   "OptionalOutputs": {
    "sc.RevocationInformation": {
     "sc.CRLs": { "sc.CRL": "CRL_#1" },
     "sc.OCSPs": { "sc.OCSP": "OCSP_#1" }
    }
   },
   ...
  }
}
```

4.6.1.6 On Demand Distinguished Name

The *subject name* of a certificate is a distinguished name (DN) that contains identifying information about the entity to which the certificate is issued. For On-Demand certificate signatures, you must provide a DN (of type string) in your request message.

The following table shows the AIS supported attributes.

You may use either the name or the abbreviation in the DN:

NAME	ABBREVIATION	DESCRIPTION	REQUIRED?[8]	COMMENTS
CommonName	CN	Common Name	**Mandatory**	
CountryName	C	Country Code (exactly two characters)	**Mandatory**	
EmailAddress	-	E-Mail Address	Optional	
GivenName	-	Given Name	Optional	Either a givenname/surname OR a pseudonym must be provided.
Surname	SN	Surname	Optional	
LocalityName	L	Locality	Optional	
OrganizationalUnitName	OU	Organizational Unit (multiple fields are allowed)	Optional	Mandatory only if O is present
OrganizationName	O	Organization	Optional	Only with permission of the customer
SerialNumber	-	Sequence of characters uniquely identifying the signatory.	**Mandatory**	
StateOrProvinceName	ST	State or Province	Optional	
Pseudonym		A name or key that uniquely identifies the user.	Optional	Either a givenname/surname OR a pseudonym must be provided.

Example:

SOAP/XML: <sc:DistinguishedName>cn=Jo Muster,GivenName=Johannes,sn=Muster,c=CH,serialNumber=1234567890</sc:DistinguishedName>

JSON: "sc.CertificateRequest": {"sc.DistinguishedName": "cn=Johannes Muster, givenname=Johannes, surname=Muster, c=CH, serialNumber=1234567890"}

In addition to the mandatory attributes defined above, the DN will be validated against the configured DN pattern. All attributes defined in the pattern will be enforced by AIS. See next section for more information regarding the DN pattern.

4.6.1.7 DN Pattern

For each On Demand customer, how the DN should look like must be agreed upon and defined contractually. To ensure that the DN provided in the sign request complies with the agreed format, for each On Demand

[8] "Required" in this case denotes if the attribute is technically enforced by AIS or not. The regulatory requirements for building the DN as described in [CP] and [CPS] are mandatory and must be agreed upon contractually. In general, either pseudonym or givenName+surname must be provided. The attribute enforcement reflected in the table is also independent of the configured DN pattern (see *5.1.5.7. DN Pattern*).

service a DN pattern is configured on the AIS backend. If the provided DN does not match against the configured value, the sign request will be rejected with the corresponding error message.

For example, for the claimed identity *MyCustomer:OnDemand-Qualified*, a DN format like the following is configured:

```
CN=<First name and last name of the customer>, pseudonym=<mobile number of the customer>, O=My Company AG, C=ch
```

For the DN pattern on the example, following DN examples will have following results:

- CN=John Doe,pseudonym=41791234567,O=My Company AG, C=ch -> OK
- CN=John Doe, O=My Company AG, C=ch -> FAIL (pseudonym missing)
- pseudonym=41791234567, CN=John Doe,O=My Company AG, C=ch -> FAIL (wrong sequence)

Swisscom Smart Registration Service customers can include variables in the DN referencing registered evidence attributes. The Smart Registration Service is out of the scope of this document. See [RASDN] for further information.

4.6.1.8 On Demand with Step-Up Authentication

Including an element <StepUpAuthorisation> in the signature request enforces step-up authentication for On Demand signatures. Although this element is optional, most customer certificate profiles require it. The parameter can only be omitted by special arrangement with Swisscom if it is replaced by a verified customer method.

The default method for step-up authorisation is "Mobile ID Only", which will only contact MobileID for authentication.

For the "Mobile ID and Password/OTP" authentication type, if MobileID is not supported by the user's mobile subscription (for example, for a non-Swiss number) an alternative method consisting of password and/or OTP verification will be invoked.

To invoke a Step-Up Authorisation, following data must be included within the <Phone> element.

ELEMENT	DESCRIPTION	REQUIRED?
<MSISDN>[9]	The Mobile phone number of the user. Example: <sc:MSISDN>491791234567</sc:MSISDN>	**Mandatory**
<Message>[10]	The text displayed to the user. Example: <sc:Message>Sign Contract.pdf? (#2Uk3Lv)</sc:Message>	**Mandatory**
<Language>[10]	The Language of the <Message> content. Example: <sc:Language>EN</sc:Language>	**Mandatory**
<SerialNumber>	This number is a unique serial number (ID) associated to the user's phone number by the backend step-up service. If provided in the request, AIS will only compute and return a signature if it equals the one returned by the step-up service Example: <sc:SerialNumber>SAS01E0D9GAI7OO1</sc:SerialNumber>	Optional

Example:

```
SOAP/XML:    <sc:StepUpAuthorisation>
                 <sc:Phone>
                     <sc:MSISDN>491791234567</sc:MSISDN>
                     <sc:Message> Sign Contract.pdf? (#2Uk3Lv)</sc:Message>
                     <sc:Language>EN</sc:Language>
        <sc:SerialNumber>SAS01E0D9GAI7OO1</sc:SerialNumber>
                 </sc:Phone>
             </sc:StepUpAuthorisation>
JSON:        "sc.StepUpAuthorisation": {
                 "sc.Phone": {
                     "sc.MSISDN": "491791234567",
                     "sc.Message": "Sign Contract.pdf? (#2Uk3Lv)",
                     "sc.Language": "EN",
        "sc.SerialNumber": "SAS01E0D9GAI7OO1"
                 }
             }
```

⚠ Step-up authentication is only supported in asynchronous mode!

4.6.1.9 Claimed Identity

[9] The value must be compliant with the format defined in [MIDSOAP] Chapter 5.1

The customer (your company) must have an agreement with Swisscom in order to receive his personal <customer name> and <key entity> values.

The <ClaimedIdentity> element must have a value constructed as **<customer name>:<key entity>**

<key entity> is only required for static certificate and On Demand certificate signature requests. Refer to your customer specific certificate profile configuration for further details.

Example:

SOAP/XML:	<ClaimedIdentity><Name>MyCustomer:MyKey</Name></ClaimedIdentity>
JSON:	"ClaimedIdentity": {"Name": "MyCustomer:MyKey"}

3.1.5.10. The Digital Signature Algorithm

The CP/CPS documentation contains the Digital Signing algorithm used by Swisscom. There you can find a chapter with the user algorithms and key lengths. The algorithms will be configured by Swisscom during setup. Further, you can read about the RSASSA-PSS algorithm in our CP/CPS [DCES] This is currently the only supported signing algorithm.

4.7 On Demand Certificate Policy and Certification Practice Statement (CP/CPS[10])

In the certificate creation process, the RSA key pair is generated on the HSM. The keys are always 3072 bits.

The AIS software creates a certificate signing request (CSR) according to [RFC2986]. The CSR contains the following extensions:

- Distinguished Name (DN) for the subject (see 4.6.1.6)
- Subject Alternative Name (SAN)
- Issuer Alternative Name (IAN)

All other fields and extensions are set by the CA.

Subject Alternative Name (SAN)
In case of success for On Demand signatures including step-up authentication, a DirectoryName entry in the SAN contains the related details:

- **Pseudonym** (OID 2.5.4.65)
  Contains the unique serial number linked to the user's mobile phone number in the Step-Up Authorisation System.
- **Description** (OID 2.5.4.13)
  May contain the Data to Be Signed (DTBS) string from the origin SignRequest
- **SerialNumber** (OID 2.5.4.5)
  Contains the transaction number which uniquely identifies the session established between AIS and the Step-up Authorisation System
- **Name** (OID 2.5.4.41)
  May contain the end user mobile phone number (MSISDN)

Issuer Alternative Name (IAN)
The IAN is used to enhance the certificate with the SignRequest details:

[10] Refer to [CP/CPS]

- **Description** (OID 2.5.4.13)
  Contains either the customer's name or the customer's unique ID.
- **SerialNumber** (OID 2.5.4.5)
  Contains the unique Request ID of the SignRequest

4.8 Templating

AIS allows the usage of pre-defined templates for the most usual distinguished names. For example, it is possible to specify following distinguished name:

JSON Request

"sc.CertificateRequest": {
"sc.DistinguishedName": "template:name"
}

SOAP XML Request
<sc:CertificateRequest> <sc:DistinguishedName>template:name</sc:DistinguishedName> </sc:CertificateRequest>

Currently there are two available templates:

1. Template "pseudonym", which translates to:

   cn=${given_name} ${family_name}, pseudonym=${evidence_id}, c=${country}, serialNumber=${evidence_id}

2. Template "name", which translates to:

   cn=${given_name} ${family_name}, givenname=${given_name}, surname=${family_name}, c=${country}, serialNumber=${evidence_id}

References to unknown templates result in an error.

Please take into consideration:

- Your configured DN pattern must exactly match the template for the feature to work. Otherwise, an "Invalid Distinguished Name" error would be returned by AIS in the SignResponse.
- The ${country} variable shall NOT be used if the distinguished name contains the organisation (O) attribute.

4.9 Trusted Timestamp

4.10 Trusted Timestamp SignRequest

Grey = SOAP specific; **Blue** = Optional batch processing; ***Italic*** = Optional but highly recommended

SOAP/XML Timestamp SignRequest

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ais="http://service.ais.swisscom.com/">
  <soap:Body>
    <ais:sign>
      <SignRequest RequestID="YOUR_UNIQUE_ID"
            Profile="http://ais.swisscom.ch/1.1"
            xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
            xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
            xmlns:sc="http://ais.swisscom.ch/1.0/schema">
        <OptionalInputs>
          <ClaimedIdentity>
            <Name>YOUR_CUSTOMER_NAME</Name>
          </ClaimedIdentity>
          <SignatureType>urn:ietf:rfc:3161</SignatureType>
          <sc:AddRevocationInformation Type="BOTH"/>
          <AdditionalProfile>urn:oasis:names:tc:dss:1.0:profiles:timestamping</AdditionalProfile>
          <AdditionalProfile>http://ais.swisscom.ch/1.0/profiles/batchprocessing</AdditionalProfile>
        </OptionalInputs>
        <InputDocuments>
          <DocumentHash ID="YOUR_DOCID_#1">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO_#1"/>
            <dsig:DigestValue>HASH_VALUE_#1</dsig:DigestValue>
          </DocumentHash>
          <DocumentHash ID="YOUR_DOCID_#2">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO_#2"/>
            <dsig:DigestValue>HASH_VALUE_#2</dsig:DigestValue>
          </DocumentHash>
        </InputDocuments>
      </SignRequest>
    </ais:sign>
  </soap:Body>
</soap:Envelope>
```

JSON Timestamp SignRequest

```
{ "SignRequest": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "OptionalInputs": {
    "ClaimedIdentity": {
     "Name": "YOUR_CUSTOMER_NAME"
    },
    "SignatureType": "urn:ietf:rfc:3161",
    "sc.AddRevocationInformation": {"@Type": "BOTH"},
    "AdditionalProfile": [
       "urn:oasis:names:tc:dss:1.0:profiles:timestamping",
       "http://ais.swisscom.ch/1.0/profiles/batchprocessing"
    ]
   },
   "InputDocuments": {
    "DocumentHash": [
    {
     "@ID": "YOUR_DOCID_#1",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO_#1" },
     "dsig.DigestValue": "HASH_VALUE_#1"
    },
    {
     "@ID": "YOUR_DOCID_#2",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO_#2" },
     "dsig.DigestValue": "HASH_VALUE_#2"
    }
    ]
```

```
  }
 }
}
```

4.11 Trusted Timestamp SignResponse

Grey = SOAP specific; **Blue** = Optional batch processing; ***Italic*** = Optional but highly recommended

SOAP/XML Timestamp SignResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                      xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
                      xmlns:ais="http://service.ais.swisscom.com/"
                      xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                      xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
                    Profile="http://ais.swisscom.ch/1.1"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</ResultMajor>
        </Result>
        <OptionalOutputs>
          <sc:RevocationInformation>
            <sc:CRLs><sc:CRL>CRL_#1</sc:CRL></sc:CRLs>
            <sc:OCSPs><sc:OCSP>OCSP_#1</sc:OCSP></sc:OCSPs>
          </sc:RevocationInformation>
        </OptionalOutputs>
        <SignatureObject>
          <Other>
            <sc:SignatureObjects>
              <sc:ExtendedSignatureObject WhichDocument="YOUR_DOCID_#1">
                <Timestamp>
                  <RFC3161TimeStampToken>TIMESTAMP_TOKEN_#1</RFC3161TimeStampToken>
                </Timestamp>
              </sc:ExtendedSignatureObject>
              <sc:ExtendedSignatureObject WhichDocument="YOUR_DOCID_#2">
                <Timestamp>
                  <RFC3161TimeStampToken>TIMESTAMP_TOKEN_#2</RFC3161TimeStampToken>
                </Timestamp>
              </sc:ExtendedSignatureObject>
            </sc:SignatureObjects>
          </Other>
        </SignatureObject>
      </SignResponse>
    </ais:signResponse>
  </soap:Body>
</soap:Envelope>
```

JSON Timestamp SignResponse


```
{ "SignResponse": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "Result": {
    "ResultMajor": "urn:oasis:names:tc:dss:1.0:resultmajor:Success"
   },
   "OptionalOutputs": {
    "sc.RevocationInformation": {
     "sc.CRLs": { "sc.CRL": "CRL_#1" },
     "sc.OCSPs": { "sc.OCSP": "OCSP_#1" }
    }
   },
   "SignatureObject": {
    "Other": {
     "sc.SignatureObjects": {
      "sc.ExtendedSignatureObject": [
       {
        "@WhichDocument": "YOUR_DOCID_#1",
        "Timestamp": {
         "RFC3161TimeStampToken": "TIMESTAMP_TOKEN_#1"
        }
       },
       {
        "@WhichDocument": "YOUR_DOCID_#2",
        "Timestamp": {
         "RFC3161TimeStampToken": "TIMESTAMP_TOKEN_#2"
        }
       }
      ]
     }
    }
   }
  }
}
```

4.12 CMS Signatures

This Chapter shows examples of signature requests and responses for static or On Demand signatures without additional step-up authentication.

4.13 CMS SignRequest for Static Signatures

Synchronous signature request for a static signature.

Grey = SOAP specific; **Blue** = Optional batch processing; ***Italic*** = Optional but highly recommended

SOAP/XML Static Certificate SignRequest

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ais="http://service.ais.swisscom.com/">
  <soap:Body>
    <ais:sign>
      <SignRequest RequestID="YOUR_UNIQUE_ID"
            Profile="http://ais.swisscom.ch/1.1"
            xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
            xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
            xmlns:sc="http://ais.swisscom.ch/1.0/schema">
        <OptionalInputs>
          <ClaimedIdentity>
            <Name>YOUR_CUSTOMER_NAME:YOUR_KEY_ENTITY</Name>
          </ClaimedIdentity>
          <SignatureType>urn:ietf:rfc:3369</SignatureType>
          <AdditionalProfile>http://ais.swisscom.ch/1.0/profiles/batchprocessing</AdditionalProfile>
          <AddTimestamp Type="urn:ietf:rfc:3161"/>
          <sc:AddRevocationInformation Type="BOTH"/>
        </OptionalInputs>
        <InputDocuments>
          <DocumentHash ID="YOUR_DOCID_#1">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO"/>
            <dsig:DigestValue>HASH_VALUE_#1</dsig:DigestValue>
          </DocumentHash>
          <DocumentHash ID="YOUR_DOCID_#2">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO"/>
            <dsig:DigestValue>HASH_VALUE_#2</dsig:DigestValue>
          </DocumentHash>
        </InputDocuments>
      </SignRequest>
    </ais:sign>
  </soap:Body>
</soap:Envelope>
```

JSON Static Certificate SignRequest

```
{ "SignRequest": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "OptionalInputs": {
    "ClaimedIdentity": {
     "Name": "YOUR_CUSTOMER_NAME:YOUR_KEY_ENTITY"
    },
    "SignatureType": "urn:ietf:rfc:3369",
    "AdditionalProfile": "http://ais.swisscom.ch/1.0/profiles/batchprocessing",
    "AddTimestamp": {"@Type": "urn:ietf:rfc:3161"},
    "sc.AddRevocationInformation": {"@Type": "BOTH"}
   },
   "InputDocuments": {
    "DocumentHash": [
    {
     "@ID": "YOUR_DOCID_#1",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO" },
     "dsig.DigestValue": "HASH_VALUE_#1"
    },
    {
     "@ID": "YOUR_DOCID_#2",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO" },
     "dsig.DigestValue": "HASH_VALUE_#2"
    }
    ]
   }
  }
}
```

4.14 CMS SignRequest for On Demand Signatures

Synchronous signature request for an On-Demand signature without additional step-up authentication.

Grey = SOAP specific; **Blue** = Optional batch processing; ***Italic*** = Optional but highly recommended
Orange = Optional On Demand certificate request (instead of a static certificate)

SOAP/XML On Demand Certificate SignRequest

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ais="http://service.ais.swisscom.com/">
  <soap:Body>
    <ais:sign>
      <SignRequest RequestID="YOUR_UNIQUE_ID"
            Profile="http://ais.swisscom.ch/1.1"
            xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
            xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
            xmlns:sc="http://ais.swisscom.ch/1.0/schema">
        <OptionalInputs>
          <ClaimedIdentity>
            <Name>YOUR_CUSTOMER_NAME:YOUR_KEY_ENTITY</Name>
          </ClaimedIdentity>
          <SignatureType>urn:ietf:rfc:3369</SignatureType>
          <AdditionalProfile>http://ais.swisscom.ch/1.0/profiles/batchprocessing</AdditionalProfile>
          <AdditionalProfile>http://ais.swisscom.ch/1.0/profiles/ondemandcertificate</AdditionalProfile>
          <sc:CertificateRequest>
            <sc:DistinguishedName>YOUR_DN</sc:DistinguishedName>
          </sc:CertificateRequest>
          <AddTimestamp Type="urn:ietf:rfc:3161"/>
          <sc:AddRevocationInformation Type="BOTH"/>
        </OptionalInputs>
        <InputDocuments>
          <DocumentHash ID="YOUR_DOCID_#1">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO"/>
            <dsig:DigestValue>HASH_VALUE_#1</dsig:DigestValue>
          </DocumentHash>
          <DocumentHash ID="YOUR_DOCID_#2">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO"/>
            <dsig:DigestValue>HASH_VALUE_#2</dsig:DigestValue>
          </DocumentHash>
        </InputDocuments>
      </SignRequest>
    </ais:sign>
  </soap:Body>
</soap:Envelope>
```

JSON On Demand Certificate SignRequest

```
{ "SignRequest": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "OptionalInputs": {
    "ClaimedIdentity": {
     "Name": "YOUR_CUSTOMER_NAME:YOUR_KEY_ENTITY"
    },
    "SignatureType": "urn:ietf:rfc:3369",
    "AdditionalProfile": [
       "http://ais.swisscom.ch/1.0/profiles/batchprocessing",
       "http://ais.swisscom.ch/1.0/profiles/ondemandcertificate"
    ],
    "sc.CertificateRequest": {
     "sc.DistinguishedName": "YOUR_DN",
    },
    "AddTimestamp": {"@Type": "urn:ietf:rfc:3161"},
    "sc.AddRevocationInformation": {"@Type": "BOTH"}
   },
   "InputDocuments": {
    "DocumentHash": [
    {
     "@ID": "YOUR_DOCID_#1",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO" },
     "dsig.DigestValue": "HASH_VALUE_#1"
    },
    {
     "@ID": "YOUR_DOCID_#2",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO" },
     "dsig.DigestValue": "HASH_VALUE_#2"
    }
    ]
   }
  }
}
```

4.15 CMS SignResponse

Signature response for synchronous signature requests for both static and On Demand signatures without additional step-up authentication (4.27 and 4.21).

Grey = SOAP specific; **Blue** = Optional batch processing; ***Italic*** = Optional but highly recommended

SOAP/XML Static/On Demand Certificate SignResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                      xmlns:ais="http://service.ais.swisscom.com/"
                      xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                      xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
                    Profile="http://ais.swisscom.ch/1.1"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</ResultMajor>
        </Result>
        <OptionalOutputs>
          <sc:RevocationInformation>
            <sc:CRLs><sc:CRL>CRL_#1</sc:CRL></sc:CRLs>
            <sc:OCSPs><sc:OCSP>OCSP_#1</sc:OCSP></sc:OCSPs>
          </sc:RevocationInformation>
        </OptionalOutputs>
        <SignatureObject>
          <Other>
```

```
          <sc:SignatureObjects>
            <sc:ExtendedSignatureObject WhichDocument="YOUR_DOCID_#1">
              <Base64Signature Type="urn:ietf:rfc:3369">SIGNATURE_RESPONSE_#1</Base64Signature>
            </sc:ExtendedSignatureObject>
            <sc:ExtendedSignatureObject WhichDocument="YOUR_DOCID_#2">
              <Base64Signature Type="urn:ietf:rfc:3369">SIGNATURE_RESPONSE_#2</Base64Signature>
            </sc:ExtendedSignatureObject>
          </sc:SignatureObjects>
        </Other>
      </SignatureObject>
    </SignResponse>
  </ais:signResponse>
 </soap:Body>
</soap:Envelope>
```

JSON Static/On Demand Certificate SignResponse


```
{ "SignResponse": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "Result": {
    "ResultMajor": "urn:oasis:names:tc:dss:1.0:resultmajor:Success"
   },
   "OptionalOutputs": {
    "sc.RevocationInformation": {
     "sc.CRLs": { "sc.CRL": "CRL_#1" },
     "sc.OCSPs": { "sc.OCSP": "OCSP_#1" }
    }
   },
   "SignatureObject": {
    "Other": {
     "sc.SignatureObjects": {
      "sc.ExtendedSignatureObject": [
       {
        "@WhichDocument": "YOUR_DOCID_#1",
        "Base64Signature": {
         "@Type": "urn:ietf:rfc:3369",
         "$": "SIGNATURE_RESPONSE_#1"
        }
       },
       {
        "@WhichDocument": "YOUR_DOCID_#2",
        "Base64Signature": {
         "@Type": "urn:ietf:rfc:3369",
         "$": "SIGNATURE_RESPONSE_#2"
        }
       }
      ]
     }
    }
   }
  }
}
```

4.16 Asynchronous Mode

Asynchronous mode is supported for both static and On Demand signatures and mandatory for On Demand certificate signature requests where step-up authentication is enforced.

4.17 SignRequest

To indicate that the process should take place asynchronously, the following additional profile must be included in the request:

```
SOAP/XML:    <AdditionalProfile>
               urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing
             </AdditionalProfile>
JSON:        "AdditionalProfile": "urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing"
```

The asynchronous scenario (flow):

1. Send a "signRequest" with the AdditionalProfile[11] to indicate support for asynchronous mode
2. Receive a "signResponse" (see Chapter 4.18 for more details) that contains the Response ID of the ongoing transaction
3. Poll the Response ID (from step 2) until you get a final response (signature or error)
    a. Send a "pendingRequest" (see Chapter 4.19 for more details) that contains the Response ID (from step 2)
    b. Receive a "pendingResponse" (see Chapter 4.20 for more details) and check the ResultMajor value
        i. The ResultMajor may indicate a pending[12] result (repeat step 3) or it may be the result: success (=signature) or an error.

4.18 SignResponse

Grey = SOAP specific

SOAP/XML SignResponse (asynchronous)

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                      xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
                      xmlns:ais="http://service.ais.swisscom.com/"
                      xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                      xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
                    Profile="http://ais.swisscom.ch/1.1"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>
            urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending
          </ResultMajor>
        </Result>
        <OptionalOutputs>
```

[11] urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing
[12] urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending

```
        <async:ResponseID>RESPONSE_ID</async:ResponseID>
      </OptionalOutputs>
    </SignResponse>
  </ais:signResponse>
 </soap:Body>
</soap:Envelope>
```

JSON SignResponse (asynchronous)

```
{ "SignResponse": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "Result": {
    "ResultMajor": "urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending"
   },
   "OptionalOutputs": {
    "async.ResponseID": "RESPONSE_ID"
   }
  }
}
```

4.19 PendingRequest

Grey = SOAP specific

SOAP/XML PendingRequest

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ais="http://service.ais.swisscom.com/">
 <soap:Body>
  <ais:pending>
   <async:PendingRequest Profile="http://ais.swisscom.ch/1.1"
              xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
              xmlns="urn:oasis:names:tc:dss:1.0:core:schema" >
    <OptionalInputs>
     <ClaimedIdentity>
      <Name>YOUR_CUSTOMER_NAME</Name>
     </ClaimedIdentity>
     <async:ResponseID>RESPONSE_ID</async:ResponseID>
    </OptionalInputs>
   </async:PendingRequest>
  </ais:pending>
 </soap:Body>
</soap:Envelope>
```

JSON PendingRequest

```
{ "async.PendingRequest": {
   "@Profile": "http://ais.swisscom.ch/1.1",
   "OptionalInputs": {
    "ClaimedIdentity": {
     "Name": "YOUR_CUSTOMER_NAME"
    },
    "async.ResponseID": "RESPONSE_ID"
   }
  }
}
```

4.20 PendingResponse

Grey = SOAP specific

SOAP/XML PendingResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:pendingResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
                xmlns:ais="http://service.ais.swisscom.com/"
                xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse  RequestID="YOUR_UNIQUE_ID"
              Profile="http://ais.swisscom.ch/1.1"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>
            urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending
          </ResultMajor>
        </Result>
        <OptionalOutputs>
          <async:ResponseID
                    xmlns:ns3="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
                    xmlns:ns2=http://www.w3.org/2000/09/xmldsig#
                    xmlns:ns4="http://ais.swisscom.ch/1.0/schema">
            RESPONSE_ID
          </async:ResponseID>
        </OptionalOutputs>
      </SignResponse>
    </ais:signResponse>
  </soap:Body>
</soap:Envelope>
```

JSON PendingResponse

```
{ "SignResponse": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "Result": {
    "ResultMajor": "urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending"
   },
   "OptionalOutputs": {
    "async.ResponseID": "RESPONSE_ID"
   }
  }
}
```

4.21 CMS On Demand Signatures with Step-Up Authentication

The following request and response samples show the On-Demand signature cases where a Step-Up method is defined.

4.22 SignRequest

Signature requests for On Demand signatures with step-up authentication must always be asynchronous.

Following elements must be included in the signature request to enforce the additional authorization step:

- Since step-up authentication is only supported for asynchronous requests, the *asynchronous* additional profile must be included in the signature request (see additional profiles in 4.6.1.3)
- In order to allow the aforementioned client redirect, a *redirect* additional profile must be included in the Signing Request (see additional profiles in 4.6.1.3).
- The <StepUpAuthentication> element including the required information to perform the step-up authentication.

Grey = SOAP specific; **Blue** = Optional batch processing; ***Italic*** = Optional but highly recommended
Orange = Additonal Profiles and Step-Up information

SOAP/XML On Demand Certificate SignRequest with Step-Up Authentication

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ais="http://service.ais.swisscom.com/">
  <soap:Body>
    <ais:sign>
      <SignRequest RequestID="YOUR_UNIQUE_ID"
            Profile="http://ais.swisscom.ch/1.1"
            xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
            xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
            xmlns:sc="http://ais.swisscom.ch/1.0/schema">
        <OptionalInputs>
          <ClaimedIdentity>
            <Name>YOUR_CUSTOMER_NAME:YOUR_KEY_ENTITY</Name>
          </ClaimedIdentity>
          <SignatureType>urn:ietf:rfc:3369</SignatureType>
          <AdditionalProfile>http://ais.swisscom.ch/1.0/profiles/batchprocessing</AdditionalProfile>
          <AdditionalProfile>http://ais.swisscom.ch/1.0/profiles/ondemandcertificate</AdditionalProfile>
          <AdditionalProfile>urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing</AdditionalProfile>
          <AdditionalProfile>http://ais.swisscom.ch/1.1/profiles/redirect</AdditionalProfile>
          <sc:CertificateRequest>
            <sc:DistinguishedName>YOUR_DN</sc:DistinguishedName>
            <sc:StepUpAuthorisation>
              <sc:Phone
                <sc:MSISDN>MOBILE_NUMBER</sc:MSISDN>
                <sc:Message>MESSAGE</sc:Message>
                <sc:Language>LANGUAGE_CODE</sc:Language>
                <sc:SerialNumber>UNIQUE_SERIAL_NUMBER</sc:SerialNumber>
              </sc:Phone>
            </sc:StepUpAuthorisation>
          </sc:CertificateRequest>
          <AddTimestamp Type="urn:ietf:rfc:3161"/>
          <sc:AddRevocationInformation Type="BOTH"/>
        </OptionalInputs>
        <InputDocuments>
          <DocumentHash ID="YOUR_DOCID_#1">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO"/>
            <dsig:DigestValue>HASH_VALUE_#1</dsig:DigestValue>
          </DocumentHash>
          <DocumentHash ID="YOUR_DOCID_#2">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO"/>
            <dsig:DigestValue>HASH_VALUE_#2</dsig:DigestValue>
          </DocumentHash>
        </InputDocuments>
      </SignRequest>
    </ais:sign>
```

```
  </soap:Body>
</soap:Envelope>
```

JSON On Demand Certificate SignRequest with Step-Up Authentication

```
{ "SignRequest": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "OptionalInputs": {
    "ClaimedIdentity": {
     "Name": "YOUR_CUSTOMER_NAME:YOUR_KEY_ENTITY"
    },
    "SignatureType": "urn:ietf:rfc:3369",
    "AdditionalProfile": [
       "http://ais.swisscom.ch/1.0/profiles/batchprocessing",
      "http://ais.swisscom.ch/1.0/profiles/ondemandcertificate",
      "urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing",
      "http://ais.swisscom.ch/1.1/profiles/redirect"
    ],
    "sc.CertificateRequest": {
     "sc.DistinguishedName": "YOUR_DN",
     "sc.StepUpAuthorisation": {
      "sc.Phone": {
       "sc.MSISDN": "MOBILE_NUMBER",
       "sc.Message": "MESSAGE",
       "sc.Language": "LANGUAGE_CODE",
       "sc.SerialNumber": "UNIQUE_SERIAL_NUMBER"
       }
      }
     },
    "AddTimestamp": {"@Type": "urn:ietf:rfc:3161"},
    "sc.AddRevocationInformation": {"@Type": "BOTH"}
   },
   "InputDocuments": {
    "DocumentHash": [
    {
     "@ID": "YOUR_DOCID_#1",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO" },
     "dsig.DigestValue": "HASH_VALUE_#1"
    },
    {
     "@ID": "YOUR_DOCID_#2",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO" },
     "dsig.DigestValue": "HASH_VALUE_#2"
    }
    ]
   }
  }
}
```

4.23 SignResponse

If the enforced step-up authentication method requires redirection, the consent URL must be included among the "optional outputs" of the asynchronous signing response. Otherwise, the sign response does not differ from the one returned for On Demand asynchronous requests without step-up authentication (see 4.18).

After receiving the response, the application must make the following:

1) If the response contains a consent URL, open the provided URL, triggering the user password/OTP authentication. The consent URL will be included in the response:
    a. either if the Step-Up is defined with Mobile ID and the MobileID-check returns an error (fallback case)
    b. or if the Step-Up is defined with Password & SMS-Challenge is enforced by configuration
2) Start polling the AIS for a final signature response, including in each poll request the provided transaction ID.

⚠ The content of the website pointed by the consent URL can change over time and therefore the page must be displayed **as it is**. The recommended methods for showing the content hosted under the consent URL are:

- To embed an iFrame in the application (see [IFR] for guidelines)
- To send a SMS to the user with the consent URL so he can open it directly on his phone browser

Grey = SOAP specific
Orange = Additional consent URL for user redirection for non-Mobile ID users

SOAP/XML Asynchronous SignResponse with Consent URL (pending)

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                      xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
                      xmlns:ais="http://service.ais.swisscom.com/"
                      xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                      xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
                    Profile="http://ais.swisscom.ch/1.1"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>
            urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending
          </ResultMajor>
        </Result>
        <OptionalOutputs>
          <async:ResponseID>RESPONSE_ID</async:ResponseID>
          <sc:StepUpAuthorisationInfo>
            <sc:Result>
              <sc:ConsentURL>CONSENT_URL</sc:ConsentURL>
            </sc:Result>
          </sc:StepUpAuthorisationInfo>
        </OptionalOutputs>
      </SignResponse>
    </ais:signResponse>
  </soap:Body>
</soap:Envelope>
```

JSON Asynchronous SignResponse with Consent URL (pending)


```
{ "SignResponse": {
    "@RequestID": "YOUR_UNIQUE_ID",
    "@Profile": "http://ais.swisscom.ch/1.1",
    "Result": {
     "ResultMajor": "urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending"
      }
    "OptionalOutputs": {
     "async.ResponseID": "RESPONSE_ID"
     "sc.StepUpAuthorisationInfo": {
      "sc.Result": {
       "sc.ConsentURL":"CONSENT_URL"
      }
     }
    }
   }
}
```

4.23.1 PendingRequest

The pending request for On Demand signatures with step-up authentication does not differ from the ones of other asynchronous requests (see Chapter 4.19 for more details).

4.23.2 PendingResponse

Responses returned by AIS to PendingRequests before the signature has been computed will have the state "pending" as result major (see Chapter 4.20 for more details).

4.23.3 SignResponse (SUCCESS)

At soon as the signature is available, the next polling request from the application will return the final response containing the requested signature, assuming the step-up authentication succeeded. The response will include step-up information.

Grey = SOAP specific; **Blue** = Optional batch processing; ***Italic*** = Optional but highly recommended
Orange = Optional[13] information for step-up authentication

SOAP/XML Asynchronous SignResponse with Step-Up Information (SUCCESS)

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                      xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
                      xmlns:ais="http://service.ais.swisscom.com/"
                      xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                      xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
                    Profile="http://ais.swisscom.ch/1.1"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</ResultMajor>
        </Result>
        <OptionalOutputs>
          <sc:APTransID>AP_TRANSACTION_ID</sc:APTransID>
          <sc:StepUpAuthorisationInfo>
            <sc:Result>
              <sc:SerialNumber>UNIQUE_SERIAL_NUMBER</sc:SerialNumber>
            </sc:Result>
          </sc:StepUpAuthorisationInfo>
          <sc:RevocationInformation>
            <sc:CRLs><sc:CRL>CRL_#1</sc:CRL></sc:CRLs>
            <sc:OCSPs><sc:OCSP>OCSP_#1</sc:OCSP></sc:OCSPs>
          </sc:RevocationInformation>
        </OptionalOutputs>
        <SignatureObject>
          <Other>
            <sc:SignatureObjects>
              <sc:ExtendedSignatureObject WhichDocument="YOUR_DOCID_#1">
                <Base64Signature Type="urn:ietf:rfc:3369">SIGNATURE_RESPONSE_#1</Base64Signature>
              </sc:ExtendedSignatureObject>
              <sc:ExtendedSignatureObject WhichDocument="YOUR_DOCID_#2">
                <Base64Signature Type="urn:ietf:rfc:3369">SIGNATURE_RESPONSE_#2</Base64Signature>
              </sc:ExtendedSignatureObject>
            </sc:SignatureObjects>
          </Other>
        </SignatureObject>
      </SignResponse>
    </ais:signResponse>
  </soap:Body>
</soap:Envelope>
```

[13] Optional - but the customer specific certificate profile may require step up authorisation

JSON On Demand asynchronous SignResponse with Step-Up Information (SUCCESS)


```
{ "SignResponse": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "Result": {
    "ResultMajor": "urn:oasis:names:tc:dss:1.0:resultmajor:Success"
   },
   "OptionalOutputs": {
    "sc.APTransID ": "AP_TRANSACTION_ID",
    "sc.StepUpAuthorisationInfo":{
     "sc.Result":{
      "sc.SerialNumber":"UNIQUE_SERIAL_NUMBER"
     }
    }
    "sc.RevocationInformation": {
     "sc.CRLs": { "sc.CRL": "CRL_#1" },
     "sc.OCSPs": { "sc.OCSP": "OCSP_#1" }
    }
   },
   "SignatureObject": {
    "Other": {
     "sc.SignatureObjects": {
      "sc.ExtendedSignatureObject": [
       {
        "@WhichDocument": "YOUR_DOCID_#1",
        "Base64Signature": {
         "@Type": "urn:ietf:rfc:3369",
         "$": "SIGNATURE_RESPONSE_#1"
        }
       },
       {
        "@WhichDocument": "YOUR_DOCID_#2",
        "Base64Signature": {
         "@Type": "urn:ietf:rfc:3369",
         "$": "SIGNATURE_RESPONSE_#2"
        }
       }
      ]
     }
    }
   }
  }
}
```


4.24 Step-up authentication with password-reset

A new endpoint is used for authentication of users in order to sign. The new URL is easily obtained by replacing the "**pass**" with "**reset-enter**" into the actual received endpoint. Now, AIS responds with the following URL:

https://ais-sas.swisscom.com/sas/web/tk364c4c648c164659bee97c31d7424d6btx/**pass**?lang=en

The new endpoint after replacing "**pass**" with "**reset-enter**" looks like the above one:

https://ais-sas.swisscom.com/sas/web/tk364c4c648c164659bee97c31d7424d6btx/**reset-enter**?lang=en

When the new endpoint is called, a html page is returned for the user to set the password and authenticate itself. The CURL request for obtaining the HTML page:

curl --location --request GET 'https://ais-sas.swisscom.com/sas/web/tk75feb024df704e07bc58b9be6df06bdetx/**reset-enter**?lang=en'

Note: The URL also contains the token that is unique for every request. The token is retrieved as in the present. In the above URL this is the token: tk75feb024df704e07bc58b9be6df06bdetx.

This will return the following page, where the user can set the password. After that, the user will retrieve an OTP code that needs to be entered for authentication to be completed.

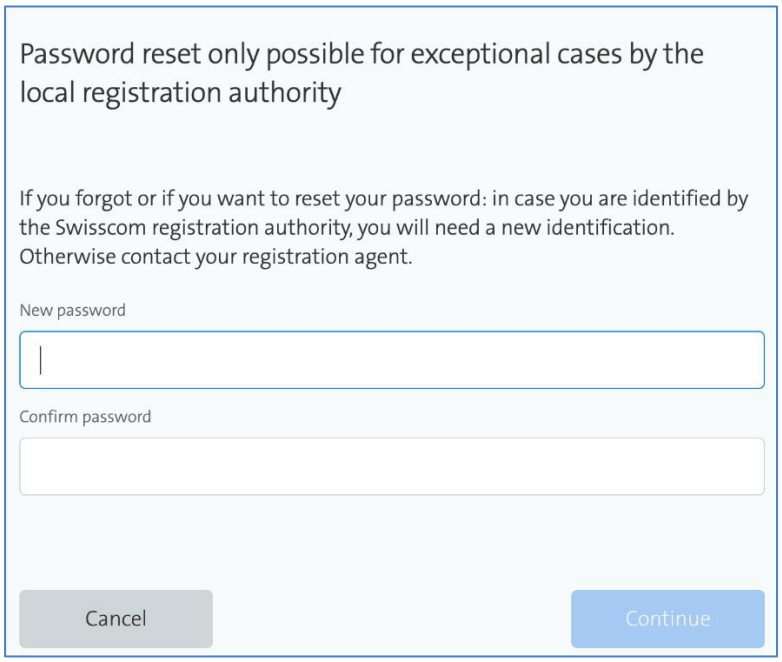

Figure 6. The picture depicts the SAS GUI where the user must introduce the new password and confirm it.

4.25 App2App switch

Mobile ID support App2App switch, as you can see in the link below – mobile-id-reference-guide, chapter 3.2.4.3. See here: https://github.com/MobileID-Strong-Authentication/mobileid-api/tree/main/doc

4.25.1.1 Description of the Flow

App2App flow description:

1. Open App X on my device and invokes an AIS signature request
2. Set the AIS "MobileIDOptions" property in the AIS request (see below)
3. AIS invokes Mobile ID signature request that open the mobile ID app and send the above property set
4. User login into the Mobile ID app
5. Mobile ID use "App2App switch" and reopen App X by using the "MobileIDOptions" Property
6. App X pool AIS to retrieve the signed document

Note: Add App2App parameters to the request if only the client application supports it and you want to trigger the Mobile ID app. Otherwise, the Mobile ID app will not show a notification.

As an AIS client, you must add the additionalprofile:
http://ais.swisscom.ch/1.1/profiles/mobileid/app2app
and the optional input MobileIDOptions with the redirect URI. For example:

JSON AIS Request with optional MobileIDOptions field

```
{
    "SignRequest": {
        "@Profile": "http://ais.swisscom.ch/1.1",
        "InputDocuments": {
            "DocumentHash": {
                "dsig.DigestMethod": {
                    "@Algorithm": "http://www.w3.org/2001/04/xmlenc#sha512"
                },
                "dsig.DigestValue":
"NA6wKClPbA+TYHW7GhPIiXn6gCGv9gSqOa508QzLGeJJYLjOfVD1tSD820M8btjEP49VhiAVK9xc/Y1z6h
x+6g=="
            }
        },
        "OptionalInputs": {
            "AdditionalProfile": [
                "http://ais.swisscom.ch/1.0/profiles/ondemandcertificate",
                "urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing",
                "http://ais.swisscom.ch/1.1/profiles/redirect",
                "http://ais.swisscom.ch/1.1/profiles/mobileid/app2app"
            ],
            "ClaimedIdentity": {
                "Name": "integrationtest-user:OnDemand-Qualified"
            },
            "SignatureType": "urn:ietf:rfc:3369",
            "sc.AddRevocationInformation": "",
            "sc.SignatureStandard": "PADES",
            "sc.CertificateRequest": {
                "sc.DistinguishedName": "cn=TEST integrationtest-user JUnit, C=DE",
                "sc.StepUpAuthorisation": {
                    "sc.Phone": {
                        "sc.Language": "en",
                        "sc.MSISDN": "49XXXXXXXXXXX",
                        "sc.Message": "Test: Please confirm"
                    }
                }
            },
            "sc.MobileIDOptions": {
                "@App2AppRedirectURI": "https://your_domain/app2app"
            }
        }
    }
}
```

On successful initiation of a Mobile ID confirmation, AIS will then add the optional output MobileIDResponse with the auth URI to its response. For example:

JSON AIS Successful response with optional MobileIDResponse field


```
{
    "SignResponse": {
        "@Profile": "http://ais.swisscom.ch/1.1",
        "Result": {
            "ResultMajor":
"urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending"
        },
        "OptionalOutputs": {
            "async.ResponseID": "14137dfb-813c-418e-9c02-2b57c8b1152d",
            "sc.MobileIDResponse": {
                "@App2AppAuthURI":
"mobileid://auth?mobile_auth_redirect_uri=https://your_domain&session_token=token&u
ser_id=***********"
            }
        }
    }
}
```


In the same manner, please find on the next page also the XML request for SOAP.

XML AIS Request with optional MobileIDResponse field

```
<soap:Body>
    <ns6:sign xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns3="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
xmlns:ns4="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:ns5="http://ais.swisscom.ch/1.0/schema"
xmlns:ns6="http://service.ais.swisscom.com/">
        <SignRequest Profile="http://ais.swisscom.ch/1.1">
            <OptionalInputs>
<AdditionalProfile>urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing</Addi
tionalProfile>
<AdditionalProfile>http://ais.swisscom.ch/1.0/profiles/ondemandcertificate</Additio
nalProfile>
                <ClaimedIdentity>
                    <Name>integrationtest-user:OnDemand-Advanced</Name>
                </ClaimedIdentity>
                <ns5:CertificateRequest>
                    <ns5:DistinguishedName>CN=JUnit
Test,C=de</ns5:DistinguishedName>
                    <ns5:StepUpAuthorisation>
                        <ns5:Phone>
                            <ns5:MSISDN>YOUR-MSISDN</ns5:MSISDN>
                            <ns5:Message>Test: automated</ns5:Message>
                            <ns5:Language>en</ns5:Language>
                        </ns5:Phone>
                    </ns5:StepUpAuthorisation>
                </ns5:CertificateRequest>
<AdditionalProfile>http://ais.swisscom.ch/1.1/profiles/mobileid/app2app</Additional
Profile>
                <ns5:MobileIDOptions
App2AppRedirectURI="https://example.com/app2app"/>
            </OptionalInputs>
            <InputDocuments>
                <DocumentHash>
                    <ns2:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<ns2:DigestValue>BwcHBwcHBwcHBwcHBwcHBwcHBwcHBwcHBwcHBwc=</ns2:DigestValue>
                </DocumentHash>
            </InputDocuments>
        </SignRequest>
    </ns6:sign>
</soap:Body>
</soap:Envelope>
```

The above request will have the following response:

XML AIS Successful response with optional MobileIDResponse field

```
<soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
xmlns:ais="http://service.ais.swisscom.com/"
xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
xmlns:sc="http://ais.swisscom.ch/1.0/schema"
xmlns:sas="http://sas.swisscom.ch/1.0/schema"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
        <SignResponse Profile="http://ais.swisscom.ch/1.1">
            <Result>

<ResultMajor>urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor
:Pending</ResultMajor>
            </Result>
            <OptionalOutputs>
                <async:ResponseID>765f7383-1a20-4197-9d6d-
47e5839d4ad3</async:ResponseID>
                <sc:MobileIDResponse
App2AppAuthURI="mobileid://auth?mobile_auth_redirect_uri=https%3A%2F%2Fexample.com%
2Fapp2app&amp;session_token=6d1086e0-
0CxtJ4XjlT8NvPo0VG721DdrOcHZytxP6i2I9DUI_9Vbx&amp;user_id=6d1086e0-dea2-46bc-a25d-
03fde2f98b6b"/>
            </OptionalOutputs>
        </SignResponse>
    </ais:signResponse>
</soap:Body>
</soap:Envelope>
```

4.26 Mobile ID Error Messages Responses

During the signing request, AIS signing back-end will do a request to Mobile ID to identify the user. If by any reason (see table below) the Mobile ID responds with an Error Code, the user will have to use the consent URL to sign. This concept is known as “AIS fallback mechanism”. The AIS fallback mechanism is a feature of AIS used in case that Mobile ID reply with certain error code (please see Mobile ID technical reference guide), then AIS will trigger a certain fallback behavior.

As part of the new and improved AIS fallback mechanism we introduced a series of new error messages which are available in 4 languages. Please see the full list in this Mobile ID Client Reference Guide [MIDSOAP]at chapter 6.

To select a specific language the following field needs to be used (from the request example above): `OptionalInputs/CertificateRequest/StepUpAuthorisation/Phone/Language`

Languages which are not supported with automatically default to English.

Note: In contrast to the old error messages of AIS, the new error messages are now localized.

Error Code	Message English	Message French	Message Italian	Message German
101	Mobile ID error: Bad input parameter, please check consent message.	Erreur Mobile ID : Mauvais paramètre d'entrée, veuillez vérifier le message de consentement.	Errore Mobile ID: Parametro di input errato, si prega di controllare il messaggio di consenso.	Mobile-ID-Fehler: Ungültiger Eingabeparameter, bitte Zustimmungsnachricht prüfen.
102	Mobile ID error: Input parameter is missing, please check consent message.	Erreur Mobile ID: Paramètre d'entrée manquant, veuillez vérifier le message de consentement.	Errore Mobile ID: Parametro di input mancante, si prega di controllare il messaggio di consenso.	Mobile-ID-Fehler: Eingabeparameter fehlt, bitte Zustimmungsnachricht prüfen.
103	Mobile ID error: Bad input parameter, wrong consent message length.	Erreur Mobile ID: Mauvais paramètre d'entrée, longueur du message de consentement erronée.	Errore Mobile ID: Parametro di input errato, lunghezza messaggio di consenso errata.	Mobile-ID-Fehler: Text für die Einwilligung ist zu lang.
104	Mobile ID error: Internal error.	Erreur Mobile ID: Erreur interne.	Errore Mobile ID: Errore interno.	Mobile-ID-Fehler: Interner Fehler.
105	Mobile ID error: Mobile ID must be activated. Please visit https://mobileid.ch/activate.	Erreur Mobile ID: Mobile ID doit être activé. Veuillez visiter https://mobileid.ch/activate.	Errore Mobile ID: Mobile ID deve essere attivato. Si prega di visitare https://mobileid.ch/activate.	Mobile-ID-Fehler: Mobile ID muss aktiviert werden. Bitte besuchen Sie https://mobileid.ch/activate.
107	Mobile ID error: Bad input parameter.	Erreur Mobile ID: Mauvais paramètre d'entrée.	Errore Mobile ID: Parametro di input errato.	Mobile-ID-Fehler: Ungültiger Eingabeparameter.
108	Mobile ID error: Internal error.	Erreur Mobile ID: Erreur interne.	Errore Mobile ID: Errore interno.	Mobile-ID-Fehler: Interner Fehler.
109	Mobile ID error: Internal error.	Erreur Mobile ID: Erreur interne.	Errore Mobile ID: Errore interno.	Mobile-ID-Fehler: Interner Fehler.
208	Mobile ID error: Timeout. Please try again after 30s.	Erreur Mobile ID: Session expirée. Veuillez réessayer après 30 secondes.	Errore Mobile ID: Tempo scaduto. Si prega di riprovare dopo 30 secondi.	Mobile ID-Fehler: Zeitüberschreitung. Bitte versuchen Sie es nach 30 Sekunden erneut.
209	Mobile ID error: Timeout. Please	Erreur Mobile ID: Session expirée.	Errore Mobile ID: Tempo scaduto. Si	Mobile-ID-Fehler: Bitte versuchen Sie es nach 30 Sekunden erneut.

	try again after 30s.	Veuillez réessayer après 30 secondes.	prega di riprovare dopo 30 secondi.	
401	Mobile ID error: User canceled the request.	Erreur Mobile ID: L'utilisateur a annulé la demande.	Errore Mobile ID: L'utente ha annullato la richiesta.	Mobile-ID-Fehler: Benutzer hat Einwilligung abgebrochen.
402	Mobile ID error: The PIN has been blocked. Please visit https://mobileid.ch/reset to reset Mobile ID and re-identify. If you have a recovery code, please visit https://mobileid.ch/recovery to restore Mobile ID directly.	Erreur Mobile ID: Le code PIN a été bloqué. Veuillez visiter https://mobileid.ch/reset pour réinitialiser Mobile ID et vous identifier à nouveau. Si vous avez un code de récupération, veuillez visiter https://mobileid.ch/recovery pour restaurer directement Mobile ID.	Errore Mobile ID: Il PIN è stato bloccato. Visita https://mobileid.ch/reset per reimpostare Mobile ID e riidentificarlo. Se disponi di un codice di ripristino, visita https://mobileid.ch/recovery per ripristinare direttamente Mobile ID.	Mobile-ID-Fehler: Die PIN wurde gesperrt. Bitte besuchen Sie https://mobileid.ch/reset, um Mobile ID zurückzusetzen und sich neu zu identifizieren. Wenn Sie einen Wiederherstellungscode haben, besuchen Sie bitte https://mobileid.ch/recovery , um Mobile ID direkt wiederherzustellen.
403	Mobile ID error: User was suspended.	Erreur Mobile ID: L'utilisateur a été suspendu.	Errore Mobile ID: L'utente è stato sospeso.	Mobile-ID-Fehler: Benutzer wurde suspendiert.
404	Mobile ID error: Mobile ID must be activated. Please visit https://mobileid.ch/activate.	Erreur Mobile ID: Mobile ID doit être activé. Veuillez visiter https://mobileid.ch/activate.	Errore Mobile ID: Mobile ID deve essere attivato. Si prega di visitare https://mobileid.ch/activate.	Mobile-ID-Fehler: Mobile ID muss aktiviert werden. Bitte besuchen Sie https://mobileid.ch/activate.
406	Mobile ID error: Transaction ongoing.	Erreur Mobile ID: Transaction en cours.	Errore Mobile ID: Transazione in corso.	Mobile-ID-Fehler: Es ist bereits eine Signatur im Gang.
422	Mobile ID error: Mobile ID must be activated. Please visit https://mobileid.ch/activate.	Erreur Mobile ID: Mobile ID doit être activé. Veuillez visiter https://mobileid.ch/activate.	Errore Mobile ID: Mobile ID deve essere attivato. Si prega di visitare https://mobileid.ch/activate.	Mobile-ID-Fehler: Mobile ID muss aktiviert werden. Bitte besuchen Sie https://mobileid.ch/activate.
501	Mobile ID error: Certificate was revoked.	Erreur Mobile ID: Le certificat a été révoqué.	Errore Mobile ID: Il certificato è stato revocato.	Mobile ID-Fehler: Zertifikat wurde widerrufen.

503	Mobile ID error: Check SIM card.	Erreur Mobile ID: Vérifiez la carte SIM.	Errore Mobile ID: Controllare la scheda SIM.	Mobile ID-Fehler: Überprüfen Sie die SIM-Karte.
900	Mobile ID error: Internal error.	Erreur Mobile ID: Erreur interne.	Errore Mobile ID: Errore interno.	Mobile-ID-Fehler: Interner Fehler.

Table 1. The AIS Error Messages for the supported four languages.

The table depicts the different error messages the customer will receive based on the different error codes which are returned by MobileID.

4.27 Static Plain Signatures (PKCS#1)

Following 2 example requests for static plain signatures in PKCS#1 format.

Grey = SOAP specific; **Blue** = Optional batch processing; ***Italic*** = Optional but highly recommended
Orange = Additonal Profiles and Step-Up information

SOAP Static Plain PKCS#1 Certificate SignRequest

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ais="http://service.ais.swisscom.com/">
 <soap:Body>
  <ais:sign>
   <SignRequest RequestID="2017-04-13T13:50:25.655+0200" Profile="http://ais.swisscom.ch/1.1" xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" xmlns:sc="http://ais.swisscom.ch/1.0/schema">
    <OptionalInputs>
     <AdditionalProfile>http://ais.swisscom.ch/1.1/profiles/plainsignature</AdditionalProfile>
     <SignatureType>http://ais.swisscom.ch/1.1/signaturetype/plain</SignatureType>
     <ClaimedIdentity>
      <Name>IAM-Test:kp1-iam-signer</Name>
     </ClaimedIdentity>
     <AddTimestamp Type="urn:ietf:rfc:3161"/>
     <sc:AddRevocationInformation Type="PLAIN"/>
    </OptionalInputs>
    <InputDocuments>
     <DocumentHash>
      <dsig:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <dsig:DigestValue>BwcHBwcHBwcHBwcHBwcHBwcHBwcHBwcHBwcHBwc=</dsig:DigestValue>
     </DocumentHash>
    </InputDocuments>
   </SignRequest>
  </ais:sign>
 </soap:Body>
</soap:Envelope>
```

JSON Static Plain PKCS#1 Certificate SignRequest

```
{
 "SignRequest": {
  "@RequestID": "2017-04-13T13:50:28.030+0200",
  "@Profile": "http://ais.swisscom.ch/1.1",
  "OptionalInputs": {
   "ClaimedIdentity": {
    "Name": "IAM-Test:kp1-iam-signer"
   },
   "AdditionalProfile": "http://ais.swisscom.ch/1.1/profiles/plainsignature",
   "SignatureType": "http://ais.swisscom.ch/1.1/signaturetype/plain",
   "AddTimestamp": {"@Type": "urn:ietf:rfc:3161"},
   "sc.AddRevocationInformation": {"@Type": "PLAIN"}
  },
  "InputDocuments": {
   "DocumentHash": {
    "@ID": "256",
    "dsig.DigestMethod": { "@Algorithm": "http://www.w3.org/2001/04/xmlenc#sha256" },
    "dsig.DigestValue": "BwcHBwcHBwcHBwcHBwcHBwcHBwcHBwcHBwcHBwc="
   }
  }
 }
}
```

4.28 Fault Response Message

In case of an error a Fault SignResponse will be raised with the appropriate details. The details will contain a ResultMajor, ResultMinor and a ResultMessage. The list of codes can be found in Chapter 6.11.

4.29 Wrong Digest Size (example)

It follows an example SignResponse in case the SignRequest content was invalid due to a wrong digest size. The SignResponse has the related error details embedded.

Grey = SOAP specific

SOAP/XML Fault SignResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
                xmlns:ais="http://service.ais.swisscom.com/"
                xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
              Profile="http://ais.swisscom.ch/1.1"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError</ResultMajor>
          <ResultMinor>http://ais.swisscom.ch/1.0/resultminor/UnexpectedData</ResultMinor>
          <ResultMessage xml:lang="en">digest size doesn't match the expected size</ResultMessage>
        </Result>
      </SignResponse>
    </ais:signResponse>
  </soap:Body>
</soap:Envelope>
```

JSON Fault SignResponse

```
{ "SignResponse": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "Result": {
    "ResultMajor": "urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError",
    "ResultMinor": "http://ais.swisscom.ch/1.0/resultminor/UnexpectedData",
    "ResultMessage": {
     "@xml.lang": "en",
     "$": "digest size doesn't match the expected size"
    }
   }
  }
}
```

4.30 Step-Up Authentication: Mobile ID User Account Problem (example)

It follows an example SignResponse in case the user does not have an active Mobile ID account and when MobileID-StepUp is used without fallback to Password and SMS-Challenge.
The SignResponse contains a **Portal URL**[14] that the user should visit to resolve the problem.

Grey = SOAP specific; **Green** = Mobile ID Step Up authorization specific

SOAP/XML Fault SignResponse

```
..
   <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
             xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
             xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
             xmlns:ais="http://service.ais.swisscom.com/"
             xmlns:sc="http://ais.swisscom.ch/1.0/schema"
             xmlns:sas="http://sas.swisscom.ch/1.0/schema"
             xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
    <SignResponse RequestID="YOUR_UNIQUE_ID"
            Profile="http://ais.swisscom.ch/1.1"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <Result>
      <ResultMajor>http://ais.swisscom.ch/1.0/resultmajor/SubsystemError</ResultMajor>
      <ResultMinor>http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/service</ResultMinor>
      <ResultMessage xml:lang="en">mss:_404</ResultMessage>
     </Result>
     <OptionalOutputs>
      <sc:StepUpAuthorisationInfo>
       <sc:Result>
        <sc:MobileIDFault>
         <sc:Subcode xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#">mss:_404</sc:Subcode>
         <sc:Reason>NO_KEY_FOUND</sc:Reason>
         <sc:Detail>
          <ns1:detail xmlns:ns1="http://kiuru.methics.fi/mssp">
           Mobile user account needs to be activated
          </ns1:detail>
          <ns2:UserAssistance xmlns:ns2="http://www.swisscom.ch/TS102204/ext/v1.0.0">
           <ns2:PortalUrl xmlns="http://www.swisscom.ch/TS102204/ext/v1.0.0">
            https://.../MobileId?reactivateMobileId=true&amp;msisdn=41791234567
           </ns2:PortalUrl>
          </ns2:UserAssistance>
         </sc:Detail>
        </sc:MobileIDFault>
       <sc:Result>
      <sc:StepUpAuthorisationInfo>
     </OptionalOutputs>
    </SignResponse>
   </ais:signResponse>
..
```

JSON Fault SignResponse

[14] Refer to [MIDSOAP] Chapter 6.5 (Fault Code User Assistance)

```
{ "SignResponse": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "Result": {
    "ResultMajor": "http://ais.swisscom.ch/1.0/resultmajor/SubsystemError",
    "ResultMinor": "http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/service",
    "ResultMessage": {
     "@xml.lang": "en",
     "$": "mss:_404"
    }
   },
   "OptionalOutputs": {
    "sc.StepUpAuthorisationInfo": {
     "sc.Result": {
      "sc.MobileIDFault": {
       "sc.Subcode": "mss:_404",
       "sc.Reason": "NO_KEY_FOUND",
       "sc.Detail": {
        "ns1.detail": "Mobile user account needs to be activated",
        "ns2.UserAssistance": {
         "ns2.PortalUrl": "https://.../MobileId?reactivateMobileId=true&msisdn=41791234567"
        }
       }
      }
     }
    }
   }
}
```

4.31 Step-Up Authentication: User Cancel (example)

During a step-up authentication, the user presses cancel interrupting in this way the consent process.

Grey = SOAP specific

SOAP/XML Fault SignResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
                xmlns:ais="http://service.ais.swisscom.com/"
                xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
              Profile="http://ais.swisscom.ch/1.1"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>http://ais.swisscom.ch/1.0/resultmajor/SubsystemError</ResultMajor>
          <ResultMinor>
            http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/cancel
          </ResultMinor>
          <ResultMessage/>
        </Result>f
      </SignResponse>
    </ais:signResponse>
  </soap:Body>
</soap:Envelope>
```

JSON Fault SignResponse

```
{ "SignResponse": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "Result": {
    "ResultMajor": "http://ais.swisscom.ch/1.0/resultmajor/SubsystemError",
    "ResultMinor": "http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/cancel",
    "ResultMessage": {
     "@xml.lang": "en",
     "$":
    }
   }
  }
}
```

4.32 Step-Up Authentication: SerialNumber Mismatch (example)

It follows an example of the case where the serial number included in the signature request does not match the serial number registered to the provided mobile phone number.

Grey = SOAP specific

SOAP/XML Fault SignResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                      xmlns:async="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:1.0"
                      xmlns:ais="http://service.ais.swisscom.com/"
                      xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                      xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
                    Profile="http://ais.swisscom.ch/1.1"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>http://ais.swisscom.ch/1.0/resultmajor/SubsystemError</ResultMajor>
          <ResultMinor>
            http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/SerialNumberMismatch
          </ResultMinor>
          <ResultMessage xml:lang="en">
            SerialNumber mismatch. We strongly advise to go through the Pre-Signing Process in order
            to retrieve the actual SerialNumber
          </ResultMessage>
        </Result>
      </SignResponse>
    </ais:signResponse>
  </soap:Body>
</soap:Envelope>
```

JSON Fault SignResponse

```
{ "SignResponse": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "Result": {
    "ResultMajor": "http://ais.swisscom.ch/1.0/resultmajor/SubsystemError",
    "ResultMinor": "http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/SerialNumberMismatch",
    "ResultMessage": {
     "@xml.lang": "en",
     "$": "SerialNumber mismatch. We strongly advise to go through the Pre-Signing Process in order to
          retrieve the actual SerialNumber"
    }
   }
  }
}
```

4.33 Best Practices

4.34 On Demand Step-Up Pre-Signing Process

Step-up authentication is mandatory for qualified signatures and most advanced signatures. Every on Demand Signature with step-up authentication of will contains a unique serial number in the signature response.

- The serial number is a unique attribute associated to a user's mobile phone number. The Application Provider (AP) can rely to have the same physical body behind the step-up authentication if the serial number remains the same.

4.34.1.1 Pre-Signing Process

If an AP would like to track the serial number, the Pre-Signing process could look like as follows:

1. A user is authenticated to a portal or application of the AP
2. The AP displays an "Enrolment Document" that the user must sign and whose signature must be approved through step-up authentication.
3. The user invokes the signature, and the AP sends an AIS On Demand Signature with Step Up authentication where there is no serial number is provided in the Request.
4. The user confirms the Signing request on the corresponding device
5. The AP receives the AIS On Demand Signature Response that contains the serial number
6. The AP stores the serial number and uniquely binds it to the user's identity.

Once the Pre-Signing process has been completed successfully, the AP should always set the known Serial Number in every Step-Up authentication as described in the Chapter 4.34.1.2 Signing Process).

4.34.1.2 Signing Process

1. The AP provides the serial number in the Step-Up authentication
2. After a successful Step-Up authentication, AIS checks whether the serial number provided by the AP and the one being returned by the Step-up Authentication System are equal.
    - In case they match the signature, the process proceeds as usual.
    - In case they mismatch the signature, process is aborted, and a fault response is returned (see 0). It is strongly advised to go through the Pre-Signing process again to retrieve the correct (actual) serial number.

4.34.1.3 Serial Number Mismatch

1. Each User Evidence evidence has a serial number associated to it.
2. This serial number together with the customer evidence are stored in in the RA Service Data Base linked to the user evidence and verified within the identification process.
3. A serial mismatch can appear due to for example when the user resets his password or the MobileID app was reinstalled, and the user did not use a recovery code for activation.
4. This means that the link between serial number and the evidence is lost.
5. Remediation consists in a re-identification of the user:
    a. https://srsident.trustservices.swisscom.com/en/

b. Directly though SRS API if the Partner/Customer has integrated SRS.

4.35 Signing Requests

4.35.1.1 General

Those major aspects need to be considered when any signing request is being constructed:

1) Define a unique RequestID (type string) attribute for every <SignRequest> element. Note that from the DSS Standard it is optional – but we highly recommend to always define it.
2) On Demand certificate signature requests will take more time compared to static certificate requests because it takes time to generate a key pair for each request.
3) When signing multiple hashes consider using the batch mode. That because the AIS authorization (which may include the optional step-up authorization) is applied only once per request.

4.35.1.2 Step Up Authorisation

1) Refer to [MIDSOAP] Chapter 5.1 to properly define the Step-Up content.
2) Since a step up can take significant longer time due to the user interaction, you may consider handling the AIS signature requests in asynchronous mode. In case of synchronous mode, ensure that the client (AP) is waiting long enough for the response.
3) If step-up authentication must be enforced, asynchronous mode is the only option available.
4) The Serial Number is an optional element to increase the security of the authorization. In case of a SerialNumber mismatch fault response, we strongly advise you go through the pre-signing process again (see 4.34.1.1).

4.36 Response handlings

It is under the responsibility of the AP to verify the response of the All-in Signing Service (AIS). The following points are crucial:

1. Verify that the RequestID attribute of the response message matches the RequestID attribute of the request message
2. Verify the signature.
3. Ensure proper exception handling for error response messages (see 6.11).
4. If Step-Up authentication is enforced, ensure proper handling of status and fault codes as described in [MIDSOAP], Chapter 6.

4.37 Adobe PDF

Please refer also to the iText and PHP code examples on GitHub at http://git.io/Fcp7

4.37.1.1 Validation with On Demand Certificates

Usually, a PDF Signature is based on three main steps:

1) A new PDF document is created, and the signing time (local time) is set.
2) A signature is requested to the All-in Signing Service.
3) The signature is embedded in the new PDF document which has been created in step 1)

For a successful signature validation, it is important that you either have a trusted timestamp information in the signature (see 4.6.1.4) or the signing time set within the 10 Minutes validity period of the On Demand certificate (see 1.4).

For the latter case we recommend adding +3 Minutes to the local time when setting the signing time in step 1).

To have LTV-enabled signatures, the CMS signature must include the timestamp and the revocation information. You also need to add to the PDF document the timestamp revocation information, which for the PAdES Signature Standard is delivered separately in the OptionalOutputs element of the SignResponse (see 4.6.1.4).

4.37.1.2 Estimating the size of the signature content

In some cases, you may need to estimate the size of the signature content, i.e., when you want to embed a digital signature into a PDF document.

With the All-in Signing Service, it is relatively easy to make an educated guess. The size of the document to be signed has no impact on the signature size. Only the following request specific options have an impact on the size of the signature content.

Options that have a fixed length:

- Length of the customer's name
- Digest Algorithm (the length of the hash value)
- Additional signing options such as Revocation Information or Timestamp

Options that have a variable length:

- Use and length of the Distinguished Name (DN)
- Use and length of the Step-Up message

We recommend sending a few examples Signature Requests with your preferred options first, to get the actual size of the signature content. This should give you a good indication for the estimation of the signature size.

Please take into consideration that the size of the signature might change due to different factors which are not always easy to foresee. Make sure to let some margin when estimating the signature size.

4.38 Processing OCSP and CRL response elements in the REST API

When using the All-in Signing Service REST API and requesting revocation information to be included in the Sign Response, the returned response will look something like this (some parts are left out):

JSON Static/On Demand Certificate SignResponse

```
{ "SignResponse": {
   . . .
   "OptionalOutputs": {
    "sc.RevocationInformation": {
     "sc.CRLs": { "sc.CRL": "CRL_#1" },
     "sc.OCSPs": { "sc.OCSP": "OCSP_#1" }
    }
   },
   . . .
  }
}
```

The service can return zero, one or more OCSP elements and zero, one or more CRL elements. For current version of the REST API, special care must be taken on the client side for handling the response:

- If no OCSP and no CRL elements are available, the *sc.RevocationInformation* node is entirely missing
- If only one OCSP or one CRL element is returned, the element is returned as a string value for the *sc.OCSP* and *sc.CRL* nodes, respectively.
- If more than one OCSP or more than one CRL element are returned, the elements are returned as a string array for the *sc.OCSP* and *sc.CRL* nodes, respectively.

Therefore, for one OCSP and one CRL, the response looks like this:

```
{ "SignResponse": {
   . . .
  "OptionalOutputs": {
   "sc.RevocationInformation": {
    "sc.CRLs": { "sc.CRL": "CRL_#1" },
    "sc.OCSPs": { "sc.OCSP": "OCSP_#1" }
   }
  },
  . . .
 }
}
```

For more than one OCSP and more than one CRL, the response looks like this:

```
{ "SignResponse": {
   . . .
  "OptionalOutputs": {
   "sc.RevocationInformation": {
    "sc.CRLs": { "sc.CRL": [ "CRL_#1", CRL_#2", CRL_#3" ]},
    "sc.OCSPs": { "sc.OCSP": [ "OCSP_#1", "OCSP_#2", "OCSP_#3" ]}
   }
  },
  . . .
 }
}
```

Depending on the JSON parsing library that you use, this might come as built-in support, or you might have to parse the response as a JSON document model to extract the correct OCSP and CRL information out of it.

As an example, for the Jackson library in Java world, there is built-in support:

```
ObjectMapper jsonMapper = new ObjectMapper();
jacksonMapper.configure(DeserializationFeature.ACCEPT_SINGLE_VALUE_AS_ARRAY, true;
```

4.39 Processing PDFs for PAdES LTV support

The PDF documents signed with the All-in Signing Service can be further enriched to ensure the resulting document has PAdES LTV quality. PAdES (PDF Advanced Electronic Signatures) and the LTV (Long Term Validation) variant provide conformance to the signature according to the ETSI standards for digital signatures (AES and QES).

As general guidelines for this processing:

- The signature is included in a data structure in the PDF as a CMS binary encoded object

- The PDF is enriched with a validation data, necessary to validate the electronic signature. This data contains the CA certificate(s), OCSP and CRL information
- An LTV signature is valid after the signing certificate has expired and even after the validation data (certificate, OCSP and CRL) is not available online anymore.

When requesting a signature on a PDF document to the All-in Signing Service, you must construct the Signing Request with the following parameters:

- The *SignatureStandard* element must be set to *PAdES-baseline* for AIS to correctly process and embed in the signature object the corresponding attributes as defined by the PAdES standard
- the *AddTimestamp* element must be present, for the timestamp to be included in the signature
- the *AddRevocationInformation* element must be present so the validation information is delivered by the service in the Signing Response
- for CMS signatures (both static and on-demand) the type of the *AddRevocationInformation* element is not necessary, since it will automatically match the defined signature standard

Here is an example of a Signing Request to trigger a PAdES LTV signature:

SOAP/XML SignRequest

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ais="http://service.ais.swisscom.com/">
  <soap:Body>
    <ais:sign>
      <SignRequest RequestID="YOUR_UNIQUE_ID"
            Profile="http://ais.swisscom.ch/1.1"
            xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
            xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
            xmlns:sc="http://ais.swisscom.ch/1.0/schema">
        <OptionalInputs>
          <ClaimedIdentity>
            <Name>YOUR_CUSTOMER_NAME:YOUR_KEY_ENTITY</Name>
          </ClaimedIdentity>
          <SignatureType>urn:ietf:rfc:3369</SignatureType>
          <AdditionalProfile>http://ais.swisscom.ch/1.0/profiles/batchprocessing</AdditionalProfile>
          <AddTimestamp Type="urn:ietf:rfc:3161"/>
          <sc:AddRevocationInformation/>
          <sc:SignatureStandard>PAdES-baseline</sc:SignatureStandard>
          ...
        </OptionalInputs>
        <InputDocuments>
          <DocumentHash ID="YOUR_DOCID_#1">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO"/>
            <dsig:DigestValue>HASH_VALUE_#1</dsig:DigestValue>
          </DocumentHash>
          <DocumentHash ID="YOUR_DOCID_#2">
            <dsig:DigestMethod Algorithm="DIGEST_ALGO"/>
            <dsig:DigestValue>HASH_VALUE_#2</dsig:DigestValue>
          </DocumentHash>
        </InputDocuments>
      </SignRequest>
    </ais:sign>
  </soap:Body>
</soap:Envelope>
```

JSON SignRequest

```
{ "SignRequest": {
   "@RequestID": "YOUR_UNIQUE_ID",
   "@Profile": "http://ais.swisscom.ch/1.1",
   "OptionalInputs": {
    "ClaimedIdentity": {
     "Name": "YOUR_CUSTOMER_NAME:YOUR_KEY_ENTITY"
    },
    "SignatureType": "urn:ietf:rfc:3369",
    "AdditionalProfile": "http://ais.swisscom.ch/1.0/profiles/batchprocessing",
    "AddTimestamp": {"@Type": "urn:ietf:rfc:3161"},
    "sc.AddRevocationInformation": {},
    "sc:SignatureStandard": "PAdES-Baseline"
    ...
   },
   "InputDocuments": {
    "DocumentHash": [
    {
     "@ID": "YOUR_DOCID_#1",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO" },
     "dsig.DigestValue": "HASH_VALUE_#1"
    },
    {
     "@ID": "YOUR_DOCID_#2",
     "dsig.DigestMethod": { "@Algorithm": "DIGEST_ALGO" },
     "dsig.DigestValue": "HASH_VALUE_#2"
    }
    ]
   }
  }
}
```

After a successful signature, the returned response looks like this:

SOAP/XML SignResponse

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ais:signResponse xmlns="urn:oasis:names:tc:dss:1.0:core:schema"
                xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
                xmlns:ais="http://service.ais.swisscom.com/"
                xmlns:sc="http://ais.swisscom.ch/1.0/schema"
                xmlns:sas="http://sas.swisscom.ch/1.0/schema"
                xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      <SignResponse RequestID="YOUR_UNIQUE_ID"
              Profile="http://ais.swisscom.ch/1.1"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Result>
          <ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</ResultMajor>
        </Result>
        <OptionalOutputs>
          <sc:RevocationInformation>
            <sc:CRLs><sc:CRL>CRL_#1</sc:CRL></sc:CRLs>
            <sc:OCSPs><sc:OCSP>OCSP_#1</sc:OCSP></sc:OCSPs>
          </sc:RevocationInformation>
        </OptionalOutputs>
        <SignatureObject>
          <Other>
            <sc:SignatureObjects>
              <sc:ExtendedSignatureObject WhichDocument="YOUR_DOCID_#1">
                <Base64Signature Type="urn:ietf:rfc:3369">SIGNATURE_RESPONSE_#1</Base64Signature>
              </sc:ExtendedSignatureObject>
              <sc:ExtendedSignatureObject WhichDocument="YOUR_DOCID_#2">
                <Base64Signature Type="urn:ietf:rfc:3369">SIGNATURE_RESPONSE_#2</Base64Signature>
              </sc:ExtendedSignatureObject>
            </sc:SignatureObjects>
          </Other>
        </SignatureObject>
      </SignResponse>
    </ais:signResponse>
```

```
  </soap:Body>
</soap:Envelope>
```

JSON Static/On Demand Certificate SignResponse


```
{"SignResponse": {
  "@RequestID": "YOUR_UNIQUE_ID",
  "@Profile": "http://ais.swisscom.ch/1.1",
  "Result": {
   "ResultMajor": "urn:oasis:names:tc:dss:1.0:resultmajor:Success"
  },
  "OptionalOutputs": {
   "sc.RevocationInformation": {
    "sc.CRLs": { "sc.CRL": "CRL_#1" },
    "sc.OCSPs": { "sc.OCSP": "OCSP_#1" }
   }
  },
  "SignatureObject": {
   "Other": {
    "sc.SignatureObjects": {
     "sc.ExtendedSignatureObject": [
      {
       "@WhichDocument": "YOUR_DOCID_#1",
       "Base64Signature": {
        "@Type": "urn:ietf:rfc:3369",
        "$": "SIGNATURE_RESPONSE_#1"
       }
      },
      {
       "@WhichDocument": "YOUR_DOCID_#2",
       "Base64Signature": {
        "@Type": "urn:ietf:rfc:3369",
        "$": "SIGNATURE_RESPONSE_#2"
       }
      }
     ]
    }
   }
  }
 }
}
```


Please note the *sc.RevocationInformation* node from the Signing Response. It contains the revocation information (OCSP and CRL content) that needs to be added to the PDF document, together with the digital signature, to ensure LTV (Long Term Validation). The server might return one OCSP and one CRL content, like in the example above, or multiple entries for each one of them.

To ensure the signature is LTV enabled, you must ensure that the validation information is included in the document. Considerations:

- The signature Validation Information must be available in the document
- The timestamp Validation Information must also be available in the document
- For PAdES signatures, the Validation Information is embedded in the signature object as an unauthenticated attribute
- The validation information for both the signature and the timestamp are delivered as separated objects in the OptionalOutputs element
- It's up to the signing application (i.e., the one invoking the service) to embed this information in the PDF. The delivered OCSP and CRL content (see example above) must be included in the *DSS* dictionary object. See the PDF specification for further information.

To ensure your signed PDF is PAdES B-T compliant, you must set the subfilter as the PAdES-defined "*ETSI.CAdES.detached*" (and not "*adbe.pkcs7.detached*").

To ensure your signed PDF is PAdES LTA compliant, the document must include two timestamps:

- The one included in the signature; this is already ensured by the All-in Signing Service if the *AddTimestamp* element is included in the Signing Request as described above
- An additional one, issued by timestamping the already signed document sending an additional sign request with timestamp as signature type

5 All-in Signing Service clients

Swisscom has published a set of libraries, tools and scripts on the Swisscom TrustServices space on GitHub at https://github.com/SwisscomTrustServices.

Repositories of interest:

- AIS
  - **shell** - A set of shell scripts that use the AIS SOAP and RESTful interface
  - **services** - Schemas and WSDL/WADL service description files
  - **soapui** - A sample project for SoapUI that contains example of requests and a test suite
  - **php** - Code for calling the All-In Signing Service from PHP and signing PDFs that way.
- PDFBox AIS
  - All-in Signing Service client written in Java and using Apache PDFBox for processing PDFs. The library provides complete support for On Demand (with Step Up), Static, Plain, Timestamp signatures, with LTV and PAdES B-LT(A) support. Can be used as project library (in your own projects) or as a command line tool. See the library documentation for more details.
- iText AIS
  - All-in Signing Service client written in Java and using the iText library for processing PDFs. Similar to PDFBox AIS, it provides support for all types of signatures, LTV and PAdES B-LT(A). Can be used only as a command line tool.
- Folder signer AIS
  - Docker setup that will provide a container to automatically sign PDF documents located in a given directory, using the All-in Signing Service.
- iText Dotnet AIS
  - A .NET Standard client library and a CLI wrapper for using the Swisscom All-in Signing Service (AIS) to sign and/or timestamp PDF documents. The library(AIS project) can be used as a project dependency. You can also use the CLI wrapper as a command-line tool for batch operations. It relies on the iText library for PDF processing.
- AIS React Flask
  - This client is based on JavaScript, React and uses Swisscom All-in Signing Service (AIS) to sign and/or timestamp PDF documents. The client has the same functionalities for PDF files processing as our iText7 client.

5.1 PDFBox AIS client

While you can use the All-in Signing Service SOAP or REST API directly, there is a clear benefit for you to use a client of the service that provides built-in support for most scenarios and most specialized PDF processing operations. The PDFBox AIS client is one such client, implemented in Java, that you can use either as a dependency for your project or from the command line.

For example, to use the client as a command line tool for signing a PDF with an On Demand with Step Up signature, you could run:

```
/bin/ais-client.sh -type ondemand-stepup -input sample.pdf -output sample-signed.pdf
```

And to use the client as a project dependency, you would configure it first (this is done once per entire application lifecycle):

```
RestClientConfiguration restConfig = new RestClientConfiguration();
restConfig.setRestServiceSignUrl("https://ais.swisscom.com/AIS-Server/rs/v1.0/sign");
restConfig.setRestServicePendingUrl("https://ais.swisscom.com/AIS-Server/rs/v1.0/pending");
restConfig.setServerCertificateFile("/home/user/ais-server.crt");
restConfig.setClientKeyFile("/home/user/ais-client.key");
restConfig.setClientKeyPassword("secret");
restConfig.setClientCertificateFile("/home/user/ais-client.crt");

RestClientImpl restClient = new RestClientImpl();
restClient.setConfiguration(restConfig);

AisClientConfiguration aisConfig = new AisClientConfiguration();
aisConfig.setSignaturePollingIntervalInSeconds(10);
aisConfig.setSignaturePollingRounds(10);
```

Then you need to prepare the details for the signature (this is done once per signing user):

```
UserData userData = new UserData();
userData.setClaimedIdentityName("ais-90days-trial");
userData.setClaimedIdentityKey("keyEntity");
userData.setDistinguishedName("cn=TEST User, givenname=Max, surname=Maximus, c=US, serialnumber=RAS62b1992011a589293800ca4b");

userData.setStepUpLanguage("en");
userData.setStepUpMessage("Please confirm the signing of the document");
userData.setStepUpMsisdn("40799999999");

userData.setSignatureReason("For testing purposes");
userData.setSignatureLocation("Topeka, Kansas");
userData.setSignatureContactInfo("test@test.com");

userData.setAddRevocationInformation(RevocationInformation.PADES);
userData.setSignatureStandard(SignatureStandard.PADES);

userData.setConsentUrlCallback((consentUrl, userData1) ->
                    System.out.println("Consent URL: " + consentUrl));
```

Finally call the All-in Signing Service for acquiring the signature (this is done for each signature):

```
PdfHandle document = new PdfHandle();
document.setInputFromFile("/home/user/input.pdf");
document.setOutputToFile("/home/user/signed-output.pdf");
document.setDigestAlgorithm(DigestAlgorithm.SHA256);

SignatureResult result = aisClient.signWithOnDemandCertificateAndStepUp(Collections.singletonList(document), userData);
if (result == SignatureResult.SUCCESS) {
   // all good!
}
```

5.2 iText Client

To use the AIS client, you first have to obtain it (or build it), then you have to configure it. The way you configure the client depends a lot on how you plan to use the client and integrate it in your project/setup.

For configuration details, please check the AIS configuration documentation.

5.3 iText Dotnet Client

The standalone client library is available as a nuget package to reference in your projects.

To start using the Swisscom AIS service and this client library, you will need the following:

1. Acquire an iText license
2. Get authentication details to use with the AIS client.
3. Build or download the AIS client binary package
4. Configure the AIS client for your use case
5. Use the AIS client, either programmatically or from the command line

For more information, please check the itext-dotnet-ais-client documentation available here

5.4 AIS React Flask

To start using the Swisscom AIS service and this client library, do the following:

1. Acquire an iText license
2. Get authentication details to use with the AIS client.
3. Configure the AIS client for your use case
4. Use the AIS client from the command line

For more information, please check the ais-react-flask-client documentation available here.

6 Appendix

6.1 Create self-signed certificate with openssl

Below are some examples on how to create a self-signed certificate with OpenSSL, valid for 3 years.

6.2 Generate Key and CSR

```
$ openssl req -new -newkey rsa:3072 –nodes -rand /dev/urandom -keyout mycert.key
 -out mycert.csr –sha256 -subj '/CN=ais.company.ch/ C=CH'
```

6.3 Organization or Organizational Unit

This information can be also optionally provided.

```
$ openssl req -new -newkey rsa:3072 –nodes -rand /dev/urandom -keyout mycert.key
 -out mycert.csr –sha256 -subj '/CN=ais.company.ch/O=Company/OU=OrganizationalUnit/C=CH'
```

6.4 Self-sign it and create your certificate

```
$ openssl x509 -req -days 1095 –sha256 -in mycert.csr -signkey mycert.key -out mycert.crt
```

6.5 Convert into PKCS#12 (if needed)

If you need a PKCS#12 file you can convert the Key and Certificate with this command:

```
$ openssl pkcs12 -export -in mycert.crt -inkey mycert.key -out mycert.p12
```

Provide the **mycert.crt** file to Swisscom and keep your *.key file securely stored.

6.6 Create self-signed certificate with Java keytool

Below are some examples on how to create a self-signed certificate with the Java Keytool, valid for 5 years.

6.7 Generate KeyStore & export the self-signed certificate

```
$ keytool -genkey -alias <alias-name> -keyalg RSA -keysize 3072 –validity 1095
 -dname 'CN=ais.company.ch,O=Company,C=CH' -keystore mycert.jks

$ keytool -export -alias <alias-name> -keystore mycert.jks -file mycert.crt
```

6.8 Root CA and Intermediate CA certificate import

```
$ keytool -keystore truststore.jks -import -file Swisscom_Root_CA_2_der.crt
$ keytool -keystore truststore.jks -import -file Swisscom_Rubin_CA_2_der.crt
```

6.9 Verification

```
$ keytool -printcert -v -file mycert.crt
$ keytool -list -v -keystore keystore.jks
$ keytool -list -v -keystore keystore.jks -alias <alias-name>
```

Provide the **mycert.crt** file to Swisscom and keep your *.jks file securely stored.

6.10 Swisscom signed certificate

Any client certificate issued by an official CA like Swisscom SDCS http://www.swissdigicert.ch may be used as an alternative to the self-signed certificate.

An example of a Swisscom CA Certificate file for the AIS service can be found here https://github.com/SwisscomTrustServices/AIS/blob/master/shell/ais-ca-signature.crt

The Swisscom CA Certificates can be downloaded from the Swisscom SDCS website: https://www.swisscom.ch/en/business/enterprise/offer/security/digital_certificate_service.html?node=download_ca#tab-ca-zertifikate

6.11 Result Major and ResultMinor list

The list of ResultMajor and ResultMinor status codes that might appear. Additionally, details will be set in the ResultMessage. Example:

```
<Result>
 <ResultMajor>http://ais.swisscom.ch/1.0/resultmajor/SubsystemError</ResultMajor>
 <ResultMinor>http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/service</ResultMinor>
 <ResultMessage xml:lang="en">mss:_401</ResultMessage>
</Result>
```

This list may not be complete and additional status codes may be returned by the platform.

6.12 Result Major Status Codes

URN / URI	Description
http://ais.swisscom.ch/1.0/resultmajor/SubsystemError	Some subsystems of the server produced an error. Details are included in the minor status code.
urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing: resultmajor:Pending	Asynchronous request was accepted. It is pending now.
urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError	It is assumed that the caller has made a mistake. Details are included in the minor status code.
urn:oasis:names:tc:dss:1.0:resultmajor:ResponderError	The server could not process the request. Details are included in the minor status code.
urn:oasis:names:tc:dss:1.0:resultmajor:Success	Request was successfully executed

6.13 Result Minor Status Codes

URN / URI	Description
http://ais.swisscom.ch/1.0/resultminor/AuthenticationFailed	Request authentication failed. For example, the customer used an unknown certificate.
http://ais.swisscom.ch/1.0/resultminor/CantServeTimely	The request could not be processed on time. The subsystem might be overloaded.
http://ais.swisscom.ch/1.0/resultminor/InsufficientData	The request could not be completed, because some information is missing.
http://ais.swisscom.ch/1.0/resultminor/ServiceInactive	The requested service is inactive (or not defined at all).
http://ais.swisscom.ch/1.0/resultminor/SignatureError	An error occurred, while creating a signature.
http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/ SerialNumberMismatch	During a step-up authentication, the optional unique serial number was provided in the request but did not match the one of the user's mobile number.
http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/service	A service error occurred during the step-up authentication. Error details are included in the error message.
http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/status	An unknown status code of the step-up subsystem was included in the fault response.
http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/timeout	The transaction expired before the step-up authorization was completed.
http://ais.swisscom.ch/1.1/resultminor/subsystem/StepUp/cancel	The user canceled the step-up authorization.
http://ais.swisscom.ch/1.0/resultminor/TimestampError	An error occurred, while creating a timestamp.
http://ais.swisscom.ch/1.0/resultminor/UnexpectedData	The request contains unexpected (wrong or misleading) data.
http://ais.swisscom.ch/1.0/resultminor/UnknownCustomer	The customer is unknown.
http://ais.swisscom.ch/1.0/resultminor/UnknownServiceEntity	The service entity (static key pair or the On-Demand CA server) could not be found. Maybe the customer does not have access to it.
http://ais.swisscom.ch/1.0/resultminor/UnsupportedDigestAlgorithm	The request contains a document hashed with unsupported or weak digest algorithms.
http://ais.swisscom.ch/1.0/resultminor/UnsupportedProfile	The request contained unknown profile URI.
http://ais.swisscom.ch/1.1/resultminor:subsystem/StepUp/transport	A subsystem transport error occurred.

urn:oasis:names:tc:dss:1.0:resultminor:GeneralError	A general internal error occurred.

6.14 Swisscom CA Hierarchy

The certificate chain provides a way to verify that all certificates related to the certificate being validated are trustworthy. A certificate-validation software walks through the signing certificate's chain starting with the end entity (EE) certificate, through the intermediate CA (ICA) certificate, until it finds a trusted Root CA (RCA) certificate. The Root CA certificate is the trust anchor.

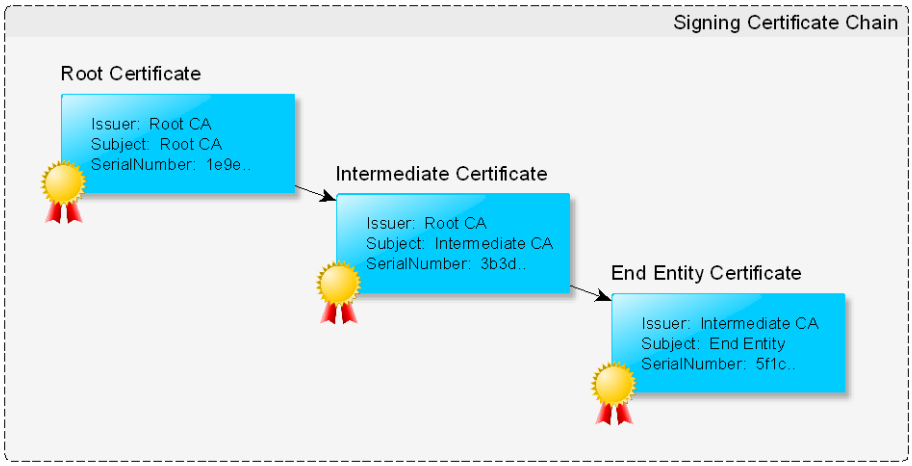

Figure 7. Root certificate hierarchy.

In the sub-chapter below, you will find an overview about the certificate hierarchy related to the Swisscom All-in Signing Service and its available revocation information (see Chapter 4.6.1.5 for more details).

The Swisscom Root CA 4 certificate is included in the Adobe Approved Trust List (AATL)[15]

NAME	FINGERPRINT ALGORITHM	FINGERPRINT
Swisscom Root CA 2	SHA1	77 47 4f c6 30 e4 0f 4c 47 64 3f 84 ba b8 c6 95 4a 8a 41 ec

6.15 CMS Signature

CERTIFICATE	SUBJECT	ISSUER	AVAILABLE RI	RETURNED REVOCATION INFO
EE Cert	CN = <Username>	ICA: CN = Swisscom Saphir CA 4	OCSP	OCSP-Response
ICA Cert	CN = Swisscom Saphir CA 4	RCA: CN = Swisscom Root CA 4	CRL	http://crl.swissdigicert.ch/sdcs-root2.crl
ICA Cert	CN = Swisscom Diamant CA 4	RCA: CN = Swisscom Root CA 4	CRL	http://crl.swissdigicert.ch/scds-root2.crl
RCA Cert	CN = Swisscom Root CA 4	RCA: CN = Swisscom Root CA 4	-	-

6.16 Timestamp Signature

CERTIFICATE	SUBJECT	ISSUER	AVAILABLE RI	RETURNED REVOCATION INFO
TSA Cert	CN = Swisscom TSA 3	ICA: CN = Swisscom TSS CA 4	OCSP	OCSP-Response
ICA Cert	CN = Swisscom TSS CA 4	RCA: CN = Swisscom Root CA 4	CRL	http://crl.swissdigicert.ch/sdcs-root2.crl
RCA Cert	CN = Swisscom Root CA 4	RCA: CN = Swisscom Root CA 4	-	-

[15] http://helpx.adobe.com/acrobat/kb/approved-trust-list2.html