Smart Registration & Signing Guide

Reference Guide

Version: 1.3

swisscom

Content

1 Introduction

The purpose of this document is to give advice and support to integrators, developers and customers who must implement the Swisscom Signing Service, referred to as Signing Service, based on the ETSI RDSC [ETSI TS 119 432] specifications.

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.
Signing Service	Swisscom 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.
Access Token	Token in JWT format which grant access to signature resources.
CA	Certificate Authority
ACR	The Authentication Context Class Reference provided by third party provider to guarantee the requested authentication method. The value of this parameters is defined by the third-party identity provider.
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)
CIBA	Client Initiated Backchannel Authentication
DN	Distinguished Name. The Distinguished Name is a set of values entered during enrolment and the creation of a Certificate Signing Request (CSR). Here in special the Common Name, Surname, Last Name, Country, and Serial Number.
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

Abbreviation	Definition
IDToken	Identity Token in JWT format which contains identity information about the user.
IDP	Identify provider which can be an authorized external registration authority (like a bank), but which is also used in the scope of this Reference Guide for a party performing the authorization for signature approval (e.g., Mobile ID). The authentication can then be based on an identity registered in the same flow during registration.
JWT	JSON Web Token
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
OCSP	Online Certificate Status Protocol is an Internet protocol used for obtaining the revocation status of a digital certificate.
PAR	Pushed Authentication Request
PKCE	Extension of the Authorization Code Flow to prevent CSRF attacks.
OIDC	OpenID connect protocol
mTLS	Mutual TLS, mutual authentication, a client certificate is mandatory to get the token.
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.
Refresh Token	Token which is used to get a new Access Token if the current token expires.
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.
SCAL 1/2	Sole Control Assurance Level. Term of CEN 419 241-1 standard to determine the sole control of the signer in respect of the signing keys operated by the Trusted Service Provider.
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)
SP	Service provider

Abbreviation	Definition
SRS	Smart registration service Link: Smart Registration Service | Swisscom Trust Services
T&C	Terms and conditions
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 technical documents
[PDFSIG]	Digital signatures in Acrobat https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/digisig_in_acrobat.pdf
[CSC]	The CSC standard https://cloudsignatureconsortium.org/wp-content/uploads/2020/01/CSC_API_V1_1.0.4.0.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/Signing_Service/wiki/Distinguished-Name:-Use-of-Evidence-Attributes

[IFR] SAS iFrame Embedding Guide
https://github.com/SwisscomTrustServices/Signing_Service/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

[ETSI EN 319 122-1] Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 1: Building blocks and CAdES baseline signatures.

https://www.etsi.org/deliver/etsi_en/319100_319199/31912201/01.02.01_60/en_31912201v010201p.pdf

[ETSI EN 319 122-2] Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 2: Extended CAdES signatures

https://www.etsi.org/deliver/etsi_en/319100_319199/31912202/01.01.01_60/en_31912202v010101p.pdf

[ETSI EN 319 142-1] Electronic Signatures and Infrastructures (ESI); PAdES digital signatures; Part 1: Building blocks and PAdES baseline signatures.

https://www.etsi.org/deliver/etsi_en/319100_319199/31914201/01.01.01_60/en_31914201v010101p.pdf

[ETSI EN 319 142-2] Electronic Signatures and Infrastructures (ESI); PAdES digital signatures; Part 2: Additional PAdES signatures profiles

https://www.etsi.org/deliver/etsi_en/319100_319199/31914202/01.01.01_60/en_31914202v010101p.pdf

[SignatureSize] Signature sizes explained.

https://github.com/SwisscomTrustServices/Signing Service/wiki/Swisscom-CA-4

[SRSIdent] SRS identification link

https://srsident.trustservices.swisscom.com/

[STSDW] Swisscom Trust Services Developer Website

https://dev.trustservices.swisscom.com/

[ETSI Open API] Signing Service ETSI Open API interface specification

https://github.com/SwisscomTrustServices/Signing Service/blob/master/OpenAPI%20ETSI%20interface%20documentation.yaml

1.3 Document Outline

As businesses move more and more towards digital processes, the use of electronic signatures is becoming increasingly popular. However, one of the biggest obstacles to using an electronic signature today is identification and approval – it can be seen as a cumbersome process that slows down operations. The most important task of a Trust Service Provider (TSP) is to comply with the legal laws, legislation and ensure that for remote signing based on the

- target legal area,
- target level of signature (qualified or advanced) and
- type of signature (seal or personal signature)

the correct registration method was used. Further the TSP must assure that the signature is approved by the right method which was already checked during the registration process.

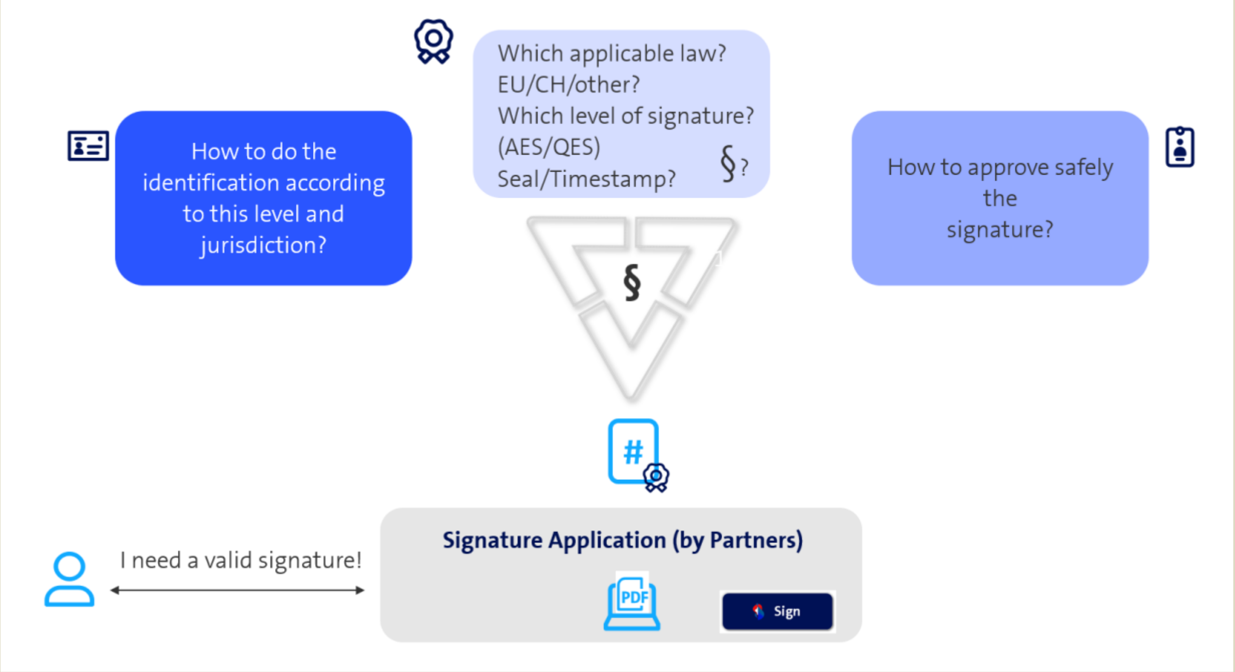

Which applicable law?
EU/CH/other?
Which level of signature?
(AES/QES)
Seal/Timestamp?
§?
How to do the identification according to this level and jurisdiction?
How to approve safely the signature?
§
#
I need a valid signature!
Signature Application (by Partners)
PDF
Sign

Figure 1: Challenges for a TSP to issue the certificate.

The tedious registration process can be sometimes avoided. Fortunately, many organizations have already collected data from us which has been verified for advanced or qualified electronic signatures. These organizations can serve as registration authorities and identity providers (IdPs) in this situation – a typical example being our house bank which holds our ID data and has also checked them against money laundering regulations.

Their apps are typically already installed on the mobile device and in daily use. It is not necessary to be re-identified or to install additional apps or remember additional passwords. This is not desirable for any user and does not match how we are used to in the digital world.

The law allows outsourcing identifications and signature approvals through Registration Authority Delegation (RA Delegation), provided certain rules are followed according to legislation requirements. These are usually verified by an initial and regular audit and released for the trust service.

By leveraging existing IdPs such as banks or other institutions who have already done much of the hard work required by verification processes, businesses can greatly reduce time spent on identifying customers via manual methods for signing. This makes adoption easier across all areas where digital signing is used, such as contracts, invoices, statements etc.

In this process, the IdP not only provides the identity as registration authority, but also offers a signature approval means or authentication means with which the person to be verified can confirm and authorize its signature. In the case of the house bank, this is often the app for online banking.

Users not identified by an IdP could use the standard methods of identity check and standard signature approval method offered by the Smart Registration Service of Swisscom. Beyond audited methods such as classic video identification, auto-identification, eID identification or bank identification (use of existing banks as IdPs) if offers the combination of different authentication means as signature approval means like Mobile ID (App), Password-One Time Code (via SMS) combination or a new Swisscom Signature Approval app. Other identification methods and signature approval methods – if passed by audit – could be easily added due to the built-in authentication broker which keeps track about the fact which user can approve by which authentication means its signature.

The new standard is built up to define the interaction between IdP and its authentication means and signature service backend and is based on the other standards for remote signature nowadays required in the laws.

It is now the task of a Multiple Authentication Broker to respond to the challenges imposed to the TSP. As decision making engine it will first ensure that the correct approval method is used in combination with an approved registration method according to the chosen jurisdiction, type and level of signature:

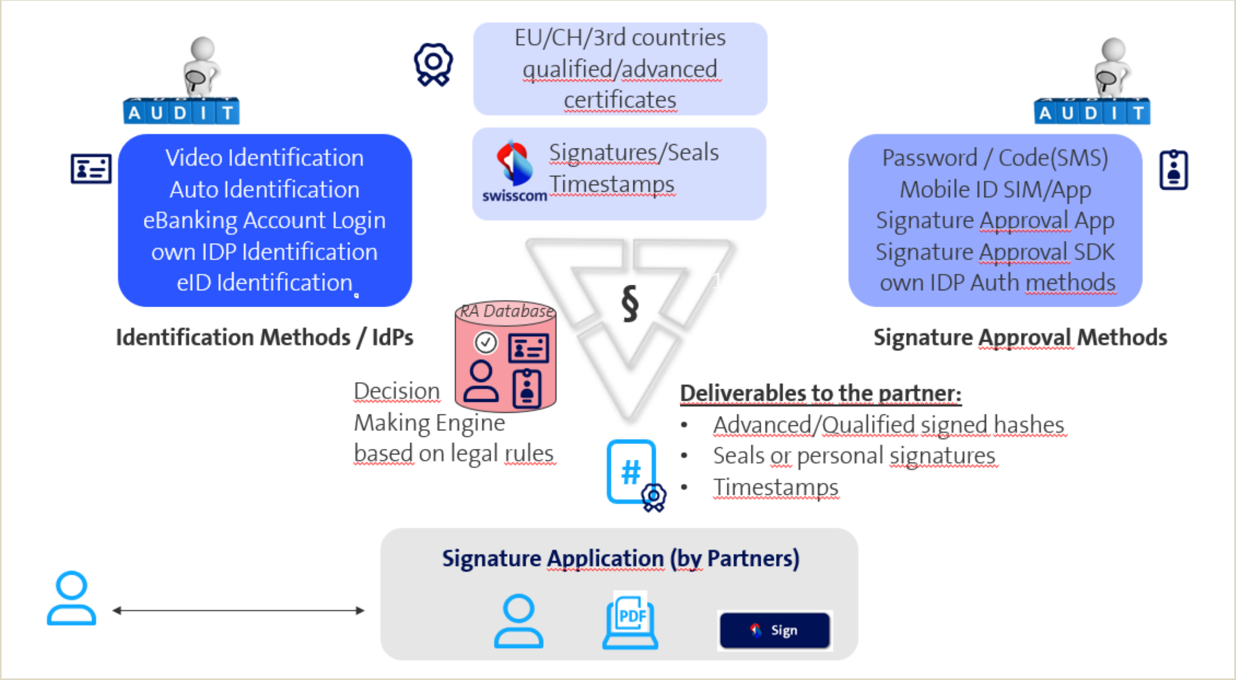


Figure 2: Multiple Authentication Broker as Decision Making Engine handling the correct approval and ID methods.

Typically, all registration/identification methods and signature approval methods require audits by the appropriate Conformity Assessment Body registered for the respective law in the chosen jurisdiction.

The Signing Service is offered for different types of signatures:

- Personal signatures for natural persons, which are based on short-term certificates and called “On Demand” certificates. Typically, each signature request requires an approval by a registered signature approval means handled by the Smart Registration Service. The nature of the short-term certificates allows to avoid the implementation of any revocation process since the certificate validity will elapse after a couple of minutes. To guarantee long term validation (LTV) later personal signatures will typically be combined with a qualified timestamp.

- Organization seals for organizations, which are based on long-term certificates and called “Static” certificates. In most situations also the seals will be combined with a qualified timestamp. For seals very often batch processes are necessary. The signature approval is based on a mutual TLS connection where the private key of this TLS certificate is managed and kept by a representative of the subject organization.
- Timestamps. They only supply integrity to the document and the current date and time.

2 Overview and Main Usage Scenarios

Signing Service allows customers' documents and files to be electronically signed: Signing Service 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 Signing Service 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 Signing Service 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].

2.1 CMS Signatures

The CMS (PKCS#7) is a standard for cryptographically protected messages. The Signing Service 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 Signing Service 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 and typically short-term 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-term certificate. Before issuing the certificate and using it only for one request, a signature approval as 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.

2.2 IdP Onboarding

Identity Providers (IdP) can be delegated part of the signature process for the registration of users and provision of authentication methods as signature approval methods. To do so they must undergo the necessary audit and be connected to the signing service via the Smart Registration Service.

Later, IdPs could use their onboarded signers and all other onboarded signers for the signature procedure itself based on the interface described in this guide.

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

2.3 Registration with Standard Identification Method and Signature Approval

Persons not registered via IdPs could use the standard identification partners of Swisscom to onboard in combination with standard signature approval means of Swisscom. During order the standard identification methods and standard signature approval methods can be selected and will be configured for the dedicated access (Claimed ID) to the signing service. Thus, only some of the methods or all methods will be shown to the signatory in case he or she is not registered, or it is not quite clear which approval method the signatory uses.

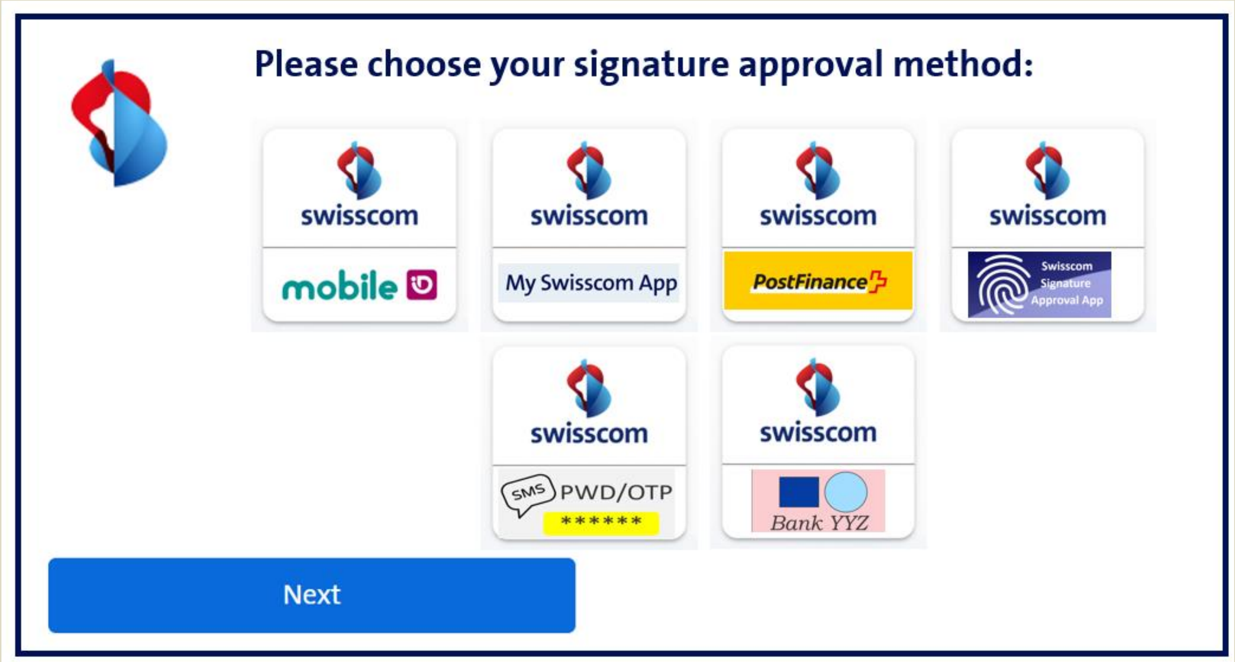


Figure 3: Selection of configured approval methods to be selected by the customer – either standard approval methods provided by Swisscom or provided by admitted and audited IDPs.

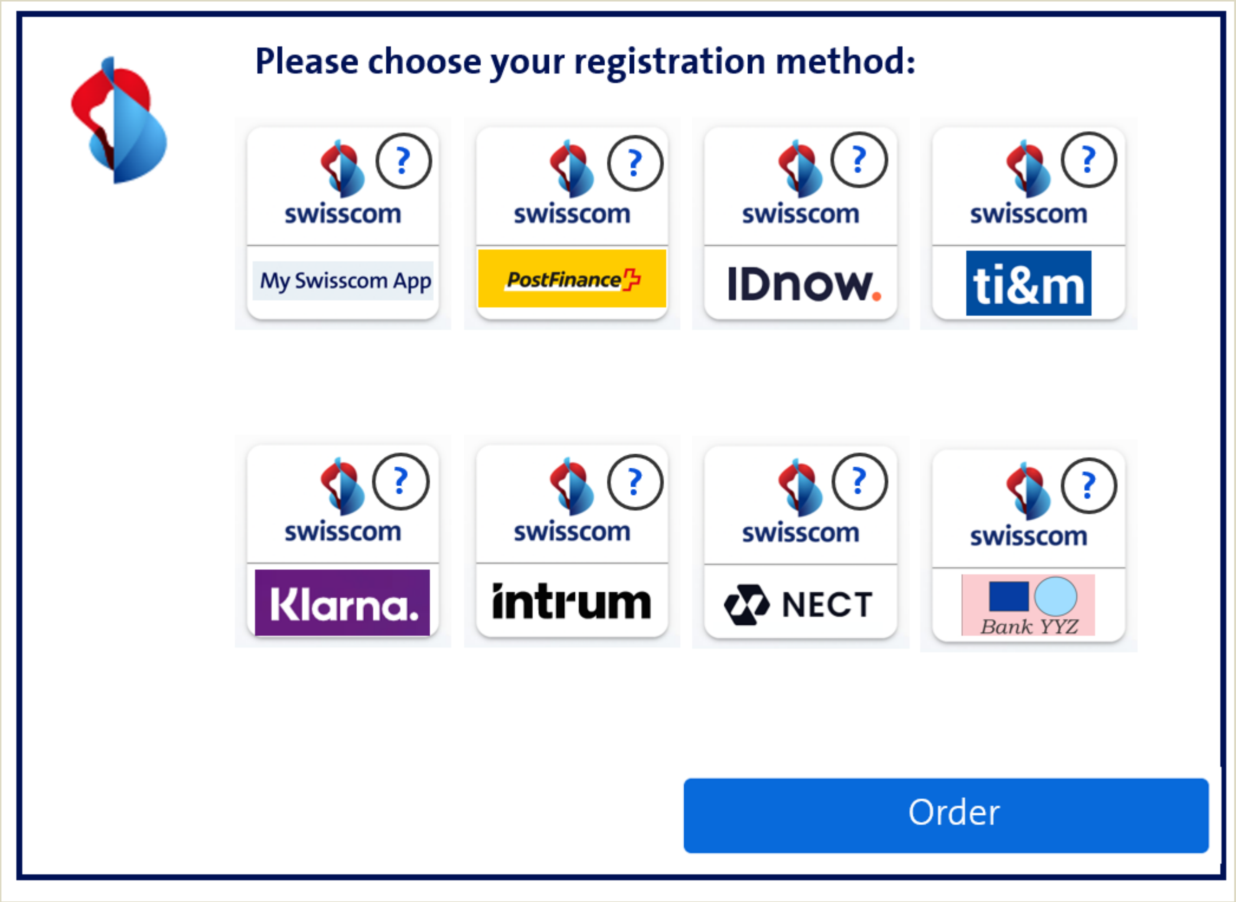


Figure 4: Configured registration methods – either standard identity proof methods or IDPs for signatories not already registered for the required service.

3 Overview Signing Service, Authentication Broker, and Interaction with IdPs

This section presents an overview of the important regulations, personal signatures, seals, and timestamps.

3.1 Regulation

The [TSI TS 119 432] standard is based on the EN 419 241-1 CEN and PP 419 241-2 standard for remote signature applications. The latter standards are now part of the Swiss Signature Act and required by supervisory authorities in Europe for trust services according the eIDAS regulation.

The signature application is the "Signer Interaction Component" which communicates with the "Service Signing Application" provided by the trust service. The Service Signing Application typically handles the signature activation by the "Signature Activation Module" and "Cryptographic Module" (HSM) both residing in a Tamper Protected Environment. The EN 419 241-1 standard outlines that the signer must have "sole control access" to the signature certificate kept in the Tamper Protected Environment of the Trust Service. The authentication may be delegated. It is distinguished between the

- "Sole Control Access Level 1" (SCAL1) necessary for all signatures on the level "advanced" and
- "Sole Control Access Level 2" (SCAL2) necessary for all signatures on the level "qualified" (eIDAS, Swiss Signature Act) or "regulated" (Swiss Signature Act)

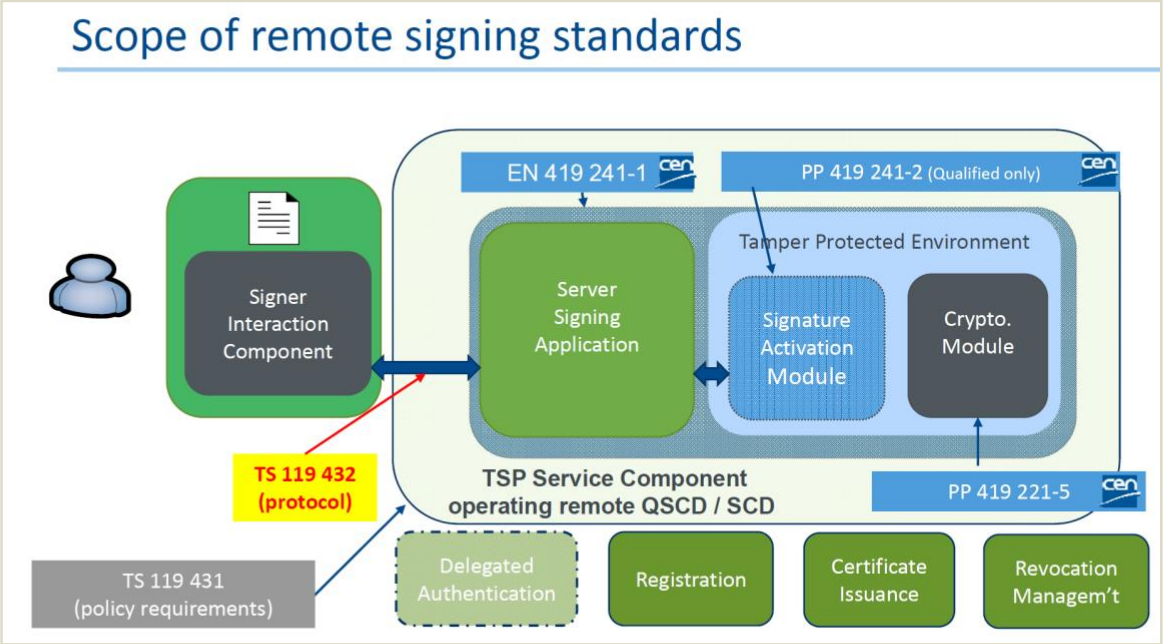
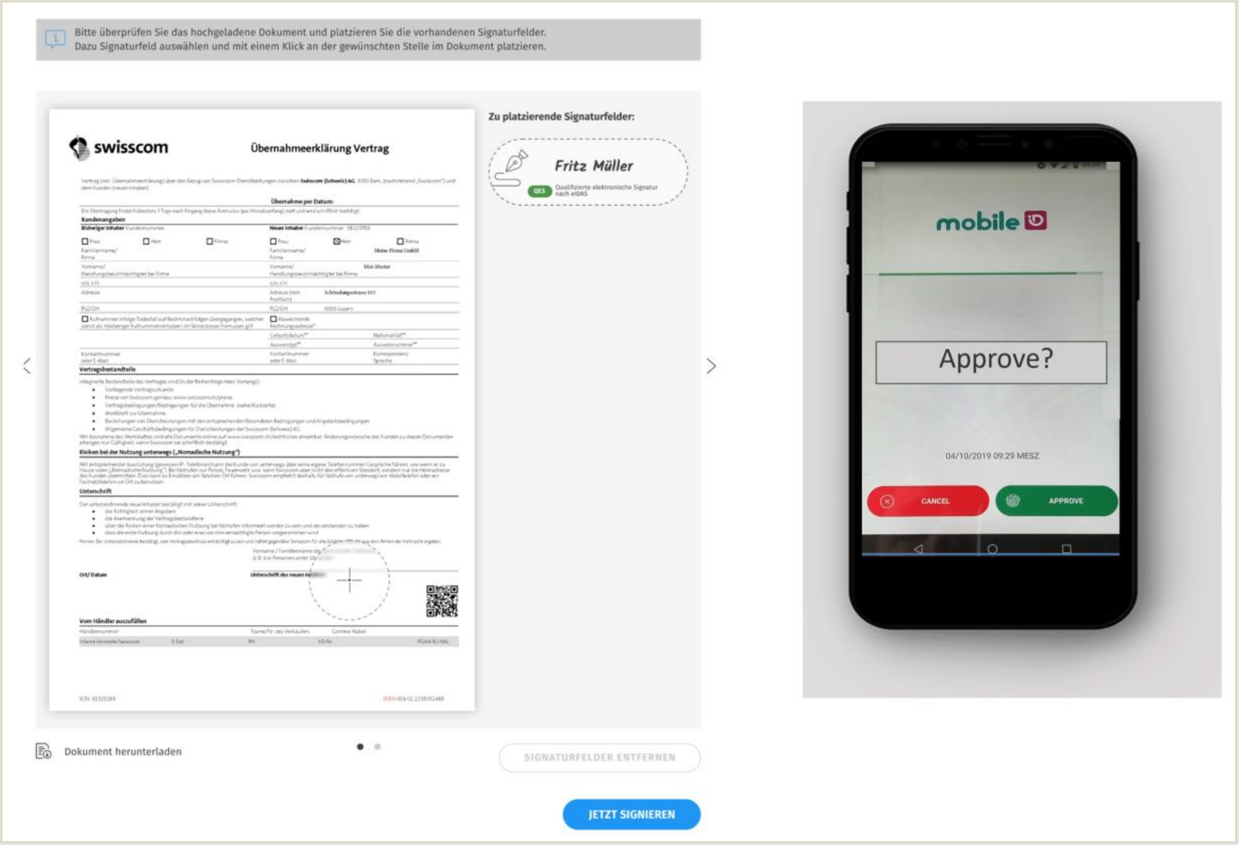

Scope of remote signing standards
EN 419 241-1
cen
PP 419 241-2 (Qualified only)
cen
Signer Interaction Component
Server Signing Application
Tamper Protected Environment
Signature Activation Module
Crypto. Module
TS 119 432 (protocol)
TSP Service Component operating remote QSCD / SCD
PP 419 221-5
cen
TS 119 431 (policy requirements)
Delegated Authentication
Registration
Certificate Issuance
Revocation Managem't

Figure 5. Standards describing the remote signing.

3.2 Personal Signatures

In case of personal signatures, it is the goal to eliminate the need for complex identification validation processes each time a signature needs approval; instead, registered persons can easily approve signatures over long periods of time without having to go through cumbersome identification procedures every single time they sign something. «Register once» - «Sign multiple» is the motto and people can after registration just confirm their signatures by fingerprint or face recognition if the identification is valid. A one-factor authentication is necessary for advanced signatures, a two-factor authentication is necessary for qualified signatures whereby the two factors should be part of two of three authentication categories: "possession", "biometrics" and "knowledge".



Bitte überprüfen Sie das hochgeladene Dokument und platzieren Sie die vorhandenen Signaturfelder.
Dazu Signaturfeld auswählen und mit einem Klick an der gewünschten Stelle im Dokument platzieren.

Figure 6. Typical signature process: Upload of a document, sign button and approval by an authentication means.

As the heart of the decision-making engine the Smart Registration Service database ensures that the user has been correctly identified. Based on the identification the user can use different signature approval methods: a standard signature approval means in combination with a standard identification method, or a specific authentication means of an IdP:

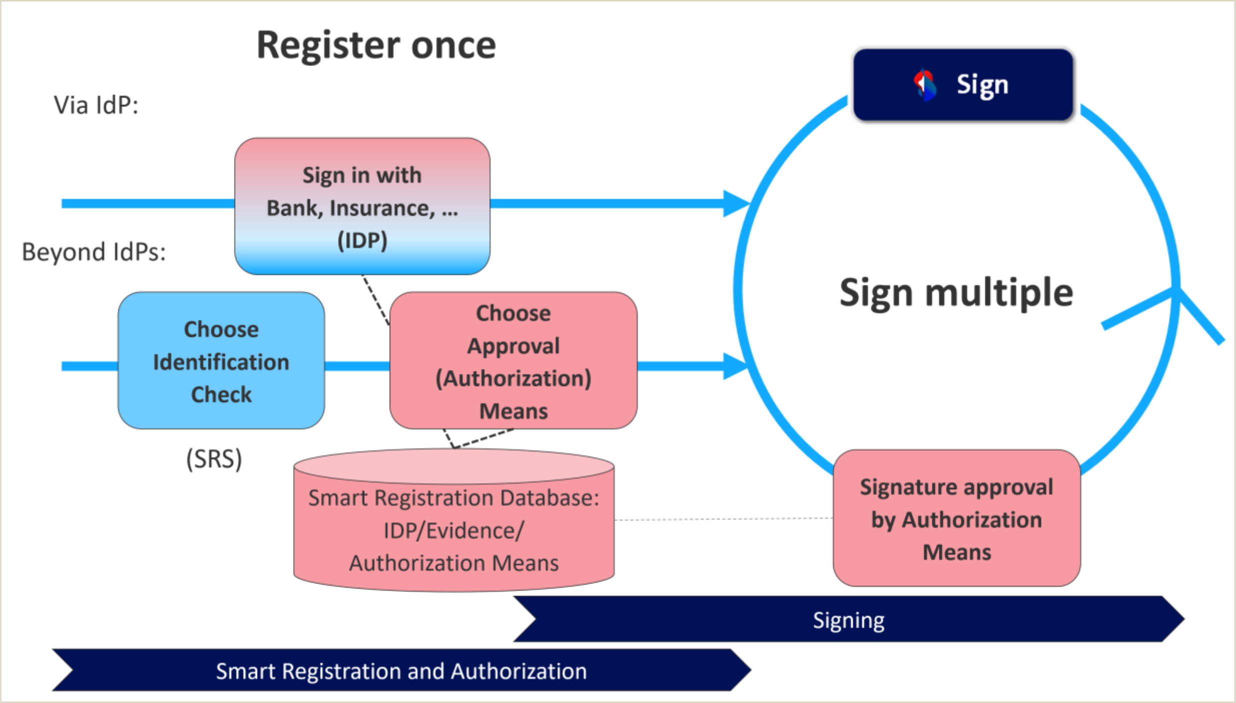

Register once
Via IdP:
Sign in with Bank, Insurance, ... (IDP)
Beyond IdPs:
Choose Identification Check
(SRS)
Choose Approval (Authorization) Means
Smart Registration Database: IDP/Evidence/ Authorization Means
Sign
Sign multiple
Signature approval by Authorization Means
Signing
Smart Registration and Authorization

Figure 7. Register once – sign multiple principle with the Smart Registration Service Database.

The Smart Registration Service database has different storing options: Based on a user ID (could be a UUID, E-Mail address, etc.) it can store evidence of the identification process and/or links to the user signature approval means. It could either store:

- The hint that the evidence is stored with an IdP (e.g., banks) on behalf of Swisscom and that the user uses the IdP own authentication means for signature approval.
- The full evidence of the identification process and that the user uses the IdP own authentication means for signature approval.
- The full evidence of the identification process and a standard signature approval means offered by Swisscom. Swisscom also offers an Authentication SDK which can be implemented in the customer flow thus it must not be a standalone authentication means.

The latter case is typically the onboarding in case the standard registration and authentication offered by Swisscom is used.

An authentication broker as part of the Smart Registration Service will handle the communication between the authentication service (which could be outsourced to an IdP), Signing Service and signature application based on the registered information in the Smart Registration Service Database.

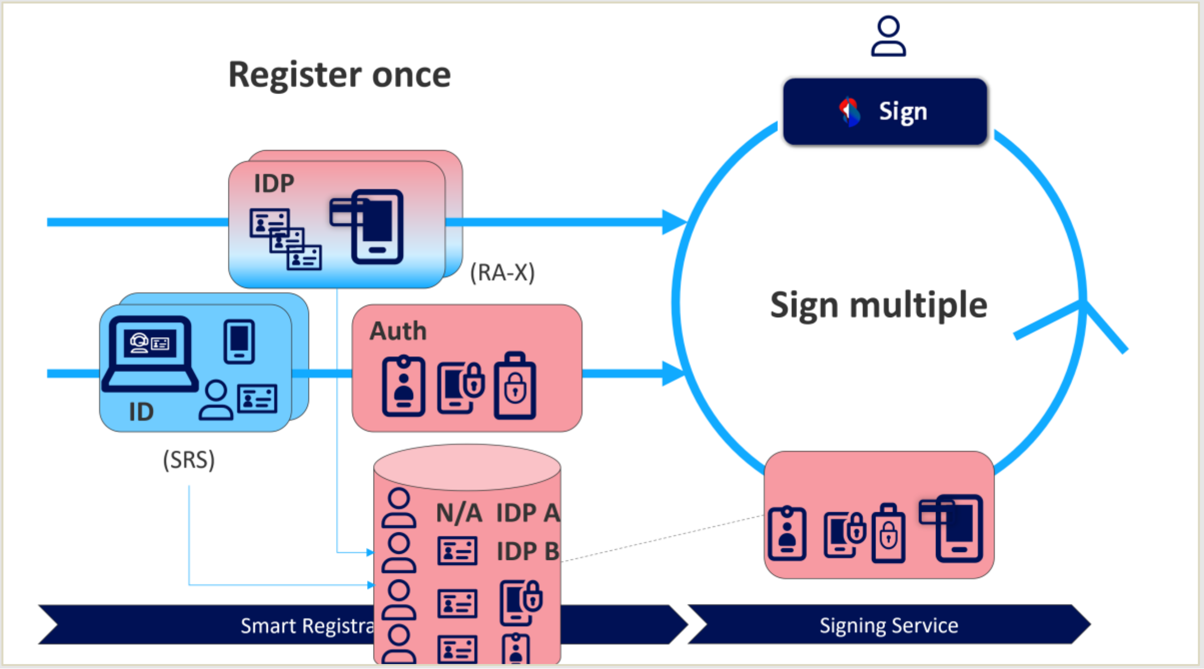


Figure 8. The different information stored in the Smart Registration Service Database concerning evidence and authentication means.

The signature procedure follows based on OpenID connect standard and the following steps:

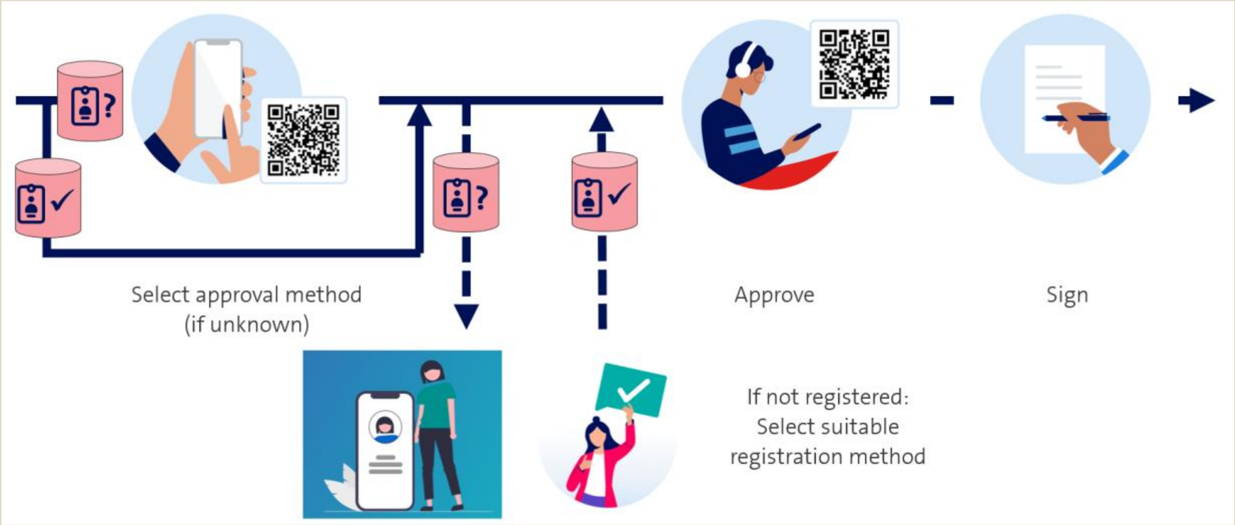


Figure 9: User signature flow.

- First, an authentication request is placed via an mTLS-protected connection with the authentication broker of the Smart Registration Service. A user characteristic as hint for the signature approval method can also be transferred. If it is missing the user must select the required approval method, he or she wants to apply or has chosen during registration.
- If the person has not been registered, the user will be automatically forwarded to the registration store where he or she can select an appropriate registration method.
- If the person has already been registered based on the selected signature approval method this method will be addressed. This can be a standard signature approval means from Swisscom (e.g.,

mobile ID, authentication app, or a Signature Approval SDK integrated in the customer app) or authentication by an IdP. Depending on the method e.g., a Mobile ID message pops up, a QR code is shown, or the user is forwarded to a login screen of an IdP.

- The signature application therefore provides an authorisation request (including ACR (Authorisation Context Reference) with the calculated document hash and the necessary information, e.g., jurisdiction (EU/CH) or signature level (advanced/qualified). This means that the document itself stays always with the signature application and is never transmitted to Swisscom.
- The authentication authority now checks the order, including the URL source, and, after successful approval, first transmits an authorisation code to the authentication broker, which, after further checking, transmits an authorisation code for the signature to the signature application.
- The signature application can now request an Access Token with the received Authorisation Code.
- The signature is then requested with this Access Token.
- The end-user short-term certificate for the signature is now generated, either with the information from the Smart Registration Service database or - if the data is exclusively kept with the delegated IdP - by requesting the first name, last name and country, or pseudonym and country from the IdP.
- The hash is signed, and the signature application receives the signed hash and can thus build up the signed document.

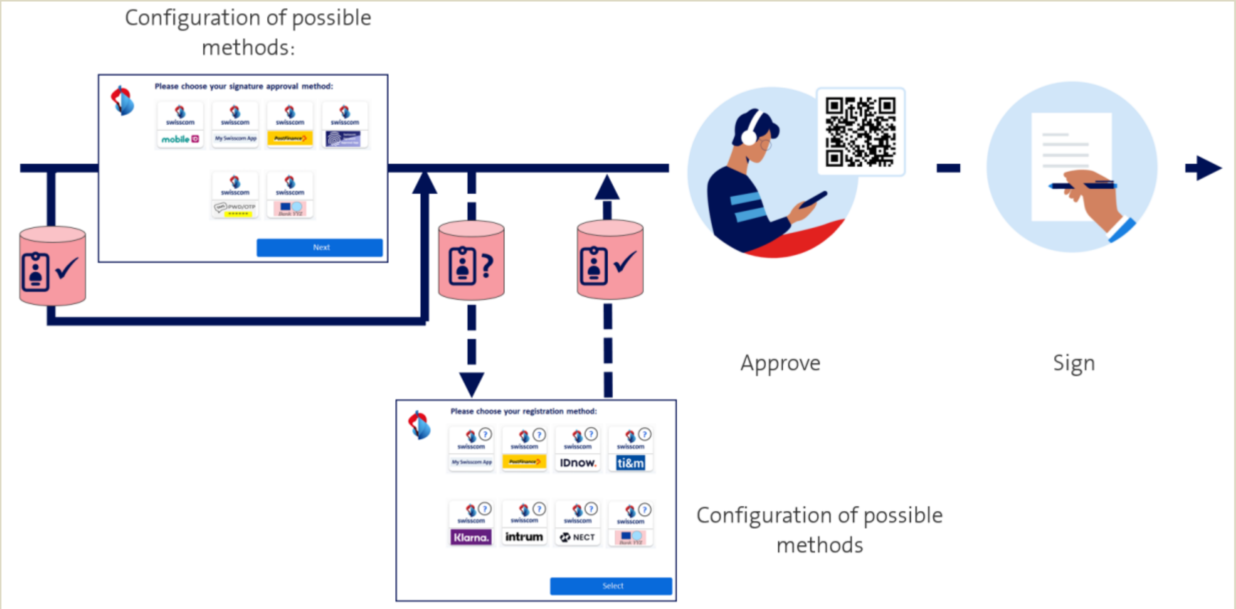

Configuration of possible methods:
Please choose your signature approval method:
swisscom
mobile ID
My Swisscom App
PostFinance
Next
Approve
Sign
Please choose your registration method:
IDnow
ti&m
Klarna.
intrum
NECT
Select
Configuration of possible methods

Figure 10: Signature flow for a user which must select first the approval methods and can select afterwards the suitable identification/registration method.

By use of the standard interface the Web browser shows the Swisscom designed views of approval and identification methods, which are configured for the customer according to the order sheet. In case of the selection of an IdP approval method only the corresponding IdP registration will be shown and no other registration possibilities.

By use of the "Pushed Authentication Request" (PAR) variant of interface the customer has the possibility at least to design its own style of "signature approval store" and "registration store":

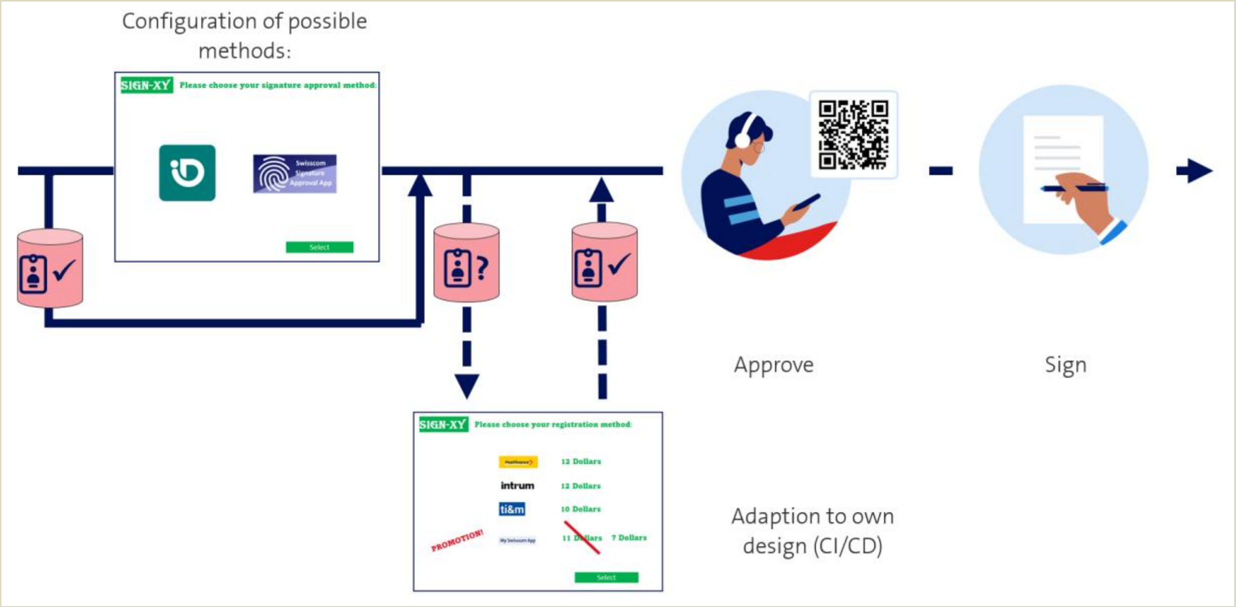

Configuration of possible methods:
SIGN-XY
Please choose your signature approval method:
Swisscom Signature Approval App
Select
SIGN-XY
Please choose your registration method:
intrum
ti&m
12 Dollars
12 Dollars
10 Dollars
PROMOTION!
My Swisscom App
11 Dollars
7 Dollars
Select
Approve
Sign
Adaption to own design (CI/CD)

Figure 11: Example of a customized view of signature approval and registration methods via PAR.

Full flexibility could even be reached by use of the CIBA interface e.g., for customizing the QR code for some approval activities.

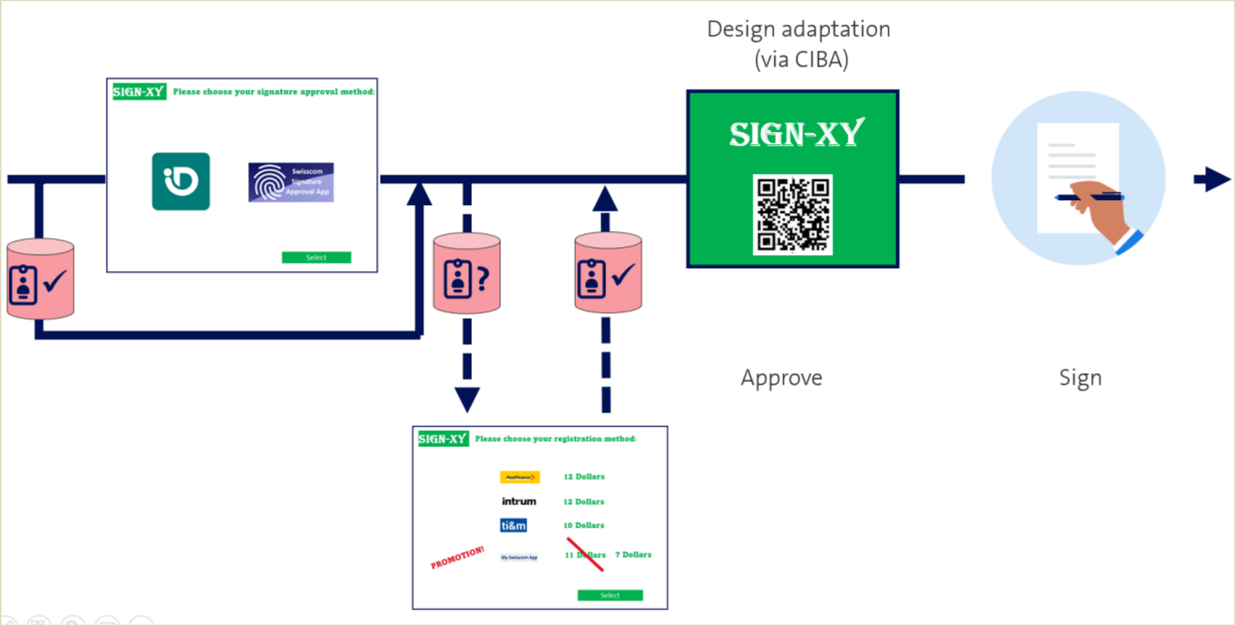

Design adaptation (via CIBA)
SIGN-XY
Please choose your signature approval method:
Swisscom Signature Approval App
Select
SIGN-XY
SIGN-XY
Please choose your registration method:
intrum
ti&m
12 Dollars
12 Dollars
10 Dollars
PROMOTION!
My Swisscom App
11 Dollars
7 Dollars
Select
Approve
Sign

Figure 12: Full customization by use of the CIBA interface.

All three signing flows will be presented in more detail in Section 5.2.2

3.3 Seals

In case of organizational seals an authenticated signature request as described before can also be possible approved by dedicated persons of an organization.

In most cases due to the very frequently used batch signing process the request-based signature approval is replaced by an authenticated connection between signature application and Signing Service of Swisscom. The connection is secured by a mTLS protocol, and the private key of the TLS certificate is controlled by a representative of the organization.

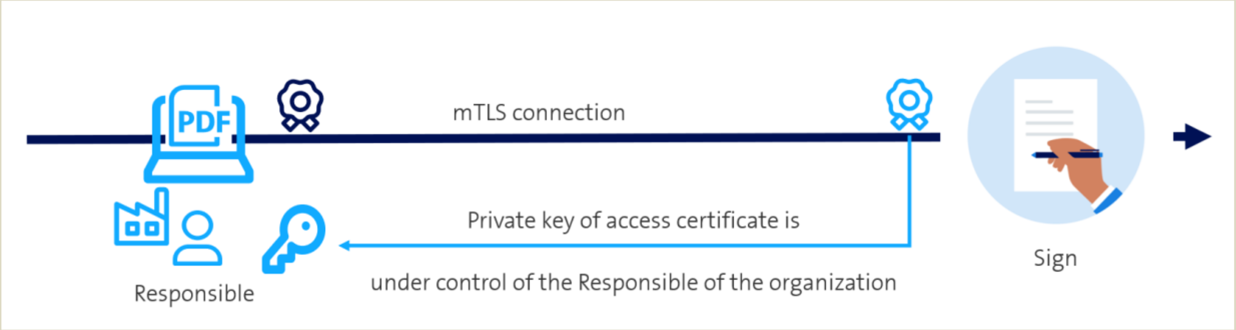

PDF
mTLS connection
Private key of access certificate is
under control of the Responsible of the organization
Responsible
Sign

Figure 13: Seal approval: mTLS connection with private key under control of the representative and responsible person for an organization

3.4 Timestamps

Timestamps do not need any authentication or approval. They can be directly requested based on the hash of the document.

4 Preconditions and Assumptions

Before using the 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 Signing Service.
2. The customer has received from Swisscom its Signing Service Claimed Identities and the relevant mTLS certificates.

4.1 Internet Access

The Signing Service 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:

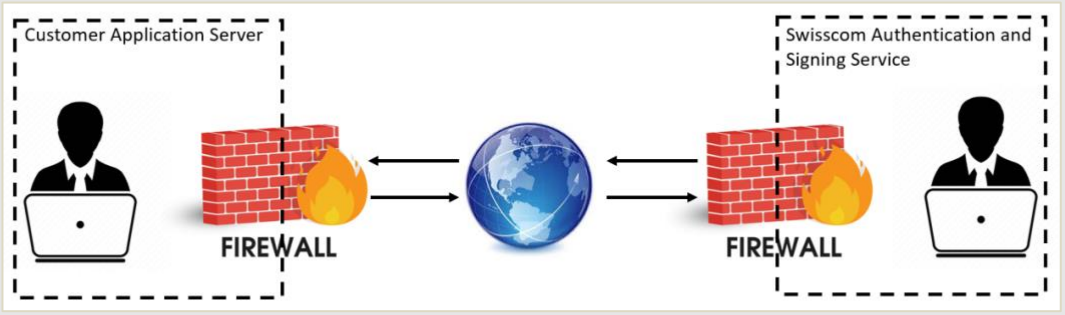

Customer Application Server
FIREWALL
FIREWALL
Swisscom Authentication and Signing Service

Figure 14. Internet based communication between customer application server and Swisscom authentication and signing service.

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

https://ais.swisscom.com

Signing Service ETSI Interface RESTful Endpoint:

https://<host>:<port>/<Signing Service-Server context>/etsi/standard/rdsc/v1/signatures/signDoc

For the ETSI interface we have only one endpoint. There is no pending endpoint. The pending endpoint can be added in a future update.

4.2 Certificate based Client Authentication

The Signing Service 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

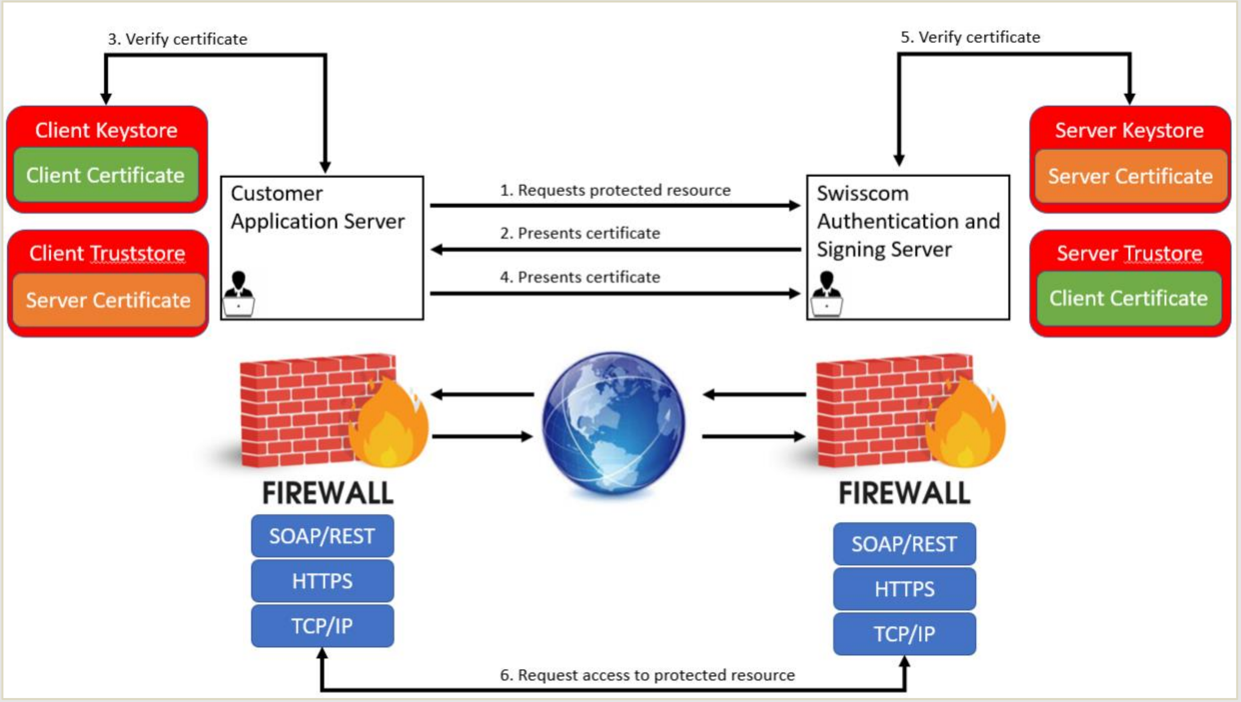


Figure 15. Certificate based client authentication steps.

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

ⓘ Signing Service 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 0 contains examples on how to create self-signed certificates.

4.3 mTLS Certificate for Signing Service Signing Service Usage

To sign the user must request and configured certificate as described in Section Section 5.2.5.3.1.

4.4 mTLS Certificate for the Broker and ETSI Sign Interface Usage

To be able to authenticate with the Broker the user must request a configured certificate as described in Section Section 5.2.5.3.1.Note that mTLS is required for the token endpoint of the broker but not for the auth endpoint. Further we need an mTLS based certificate for the sign request. We will use the same mTLS based certificate for both these requests. For all other requests no mTLS based certificate is needed.

The user either needs to have the SAS authentication service enabled, or he needs to be registered with one of the supported IDPs (MobileID App, Futurae, MySwisscom App, PostFinance App, etc.) in other to authenticate during the signing process.

4.5 Request Authorization

Signing Service provides for each AP one or many Claimed Identities to authorize the signing request. Each Claimed Identity must be used for the proper signature type.

4.5.1 User Identification

The user must be identified for QES for ZertES and/or EIDAS to be able to use the RA-Service claimed identities mentioned in the table below.

Here are the instructions on how the user can get identified by using the following link [SRSIdent].

4.5.2 Signature Type Selection

When the user gets onboarded for the 90 days trial account, he can use the following claimed identities:

Access to the test account jurisdiction CH (ZertES) with the following claimed ID:

```
<your test claimedID>:OnDemand-Advanced4
```

Access to the test account jurisdiction EU (eIDAS) with the following claimed ID:

```
<your test claimedID>:OnDemand-Advanced4.1-EU
```

The user can use all these claimed identities by requesting the Swisscom support for explicit access. The user needs to send to Swisscom support a TLS certificate (3078-bit) as described in the order form for the test account. After the customer certificate was added by the Swisscom support team then he can use all claimed identities mentioned above for 90 days.

4.6 Communication Modes

Please note that the asynchronous mode is not available for the ETSI interface.

4.7 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.

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

4.8 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]

4.9 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.

4.10 Adding Revocation Information (long-term signature).

Signing Service 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 7.11) for the signing certificate chain.

4.11 ETSI Interface and former Step-Up Authentication

Hint for users working already with older versions of the Swisscom Trust Services Signing Service:
Under the new introduced Multi-Authentication Broker the step-up authentication concept formerly used beforehand is no longer valid as the suited authentication method is selected from the beginning of the flow when the user wants to do the broker-based authentication such that he can than latter perform the signing operation.

By using the multi-Authentication broker, the following signature approval methods are currently available, and more and more methods will be added:

- Mobile ID
- Password – One Time Code via SMS (“PWD/OTP”)
- Swisscom Signature Approval App
- Swisscom Signature Approval SDK (based on Futurae)
- PostFinance App: the end user shall approve the signature and prove sole control through App authentication in case he uses one of the authentication apps.
- Others (like FIDO2 compatible methods) will follow.

4.12 Migration period and use of stand-alone SRS methods

The Signing Service via the Multiple Authentication Broker in the “new world” of signing allows a complete end2end process of selection of signature approval means – registration – signing. The historical model of separation of registration and separated signing process like it is offered via Swisscom Shop Registration, RA App registration or the standalone SRS registration will still work with some restrictions.

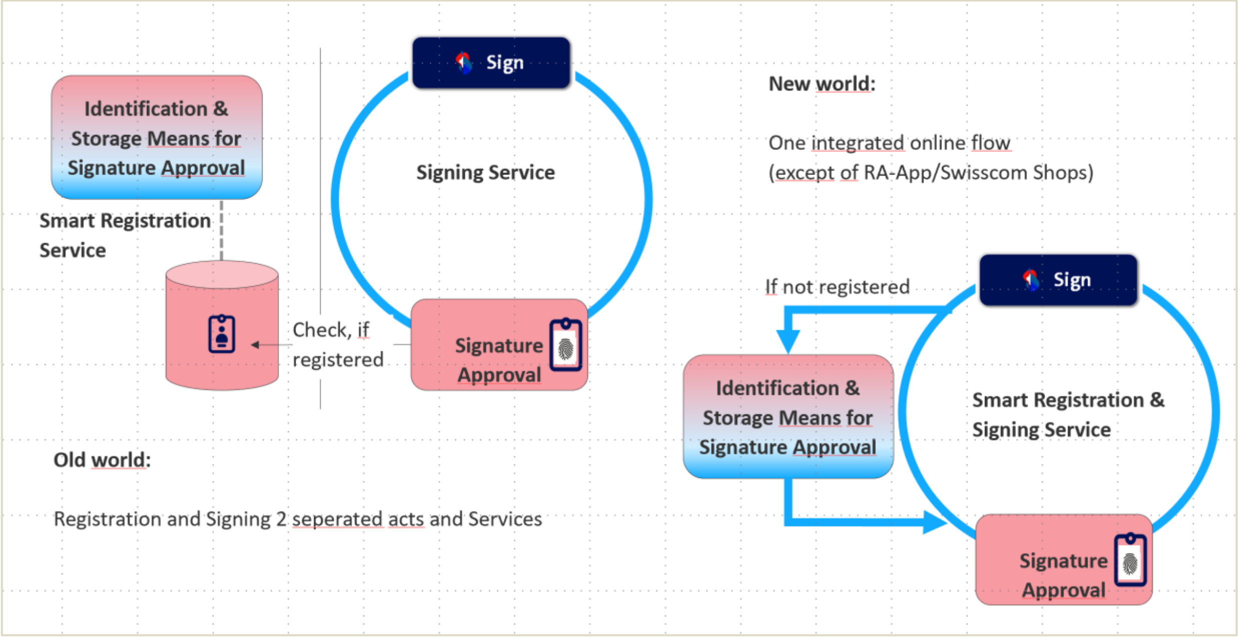

Sign
Identification & Storage Means for Signature Approval
Smart Registration Service
Signing Service
Check, if registered
Signature Approval
Old world:
Registration and Signing 2 seperated acts and Services
New world:
One integrated online flow (except of RA-App/Swisscom Shops)
If not registered
Sign
Identification & Storage Means for Signature Approval
Smart Registration & Signing Service
Signature Approval

Figure 16: Philosophy change between separated registration signing process and combined process.

Generally, all registration methods offered offline via SRS, RA App (including Swisscom Shops) and against vouchers or credit cards on the website of Swisscom Trust Services (https://srsident.trustservices.swisscom.com) are storing their evidences in the Registration Authority database which is also used by the Multiple Authentication Broker to decide which signature approval method can be chosen. RA App and SRS are (up to now) limited to the following three signature approval methods:

- Mobile ID App
- Mobile ID (SIM) – only usable by Swiss mobile providers
- Password-One Time Code via SMS ("PWD/OTP")

Nevertheless, these registration methods can be used in parallel to the registration methods already offered in the flow of the Multi-Authentication Broker. It allows to offer signatories also registration currently not available in the standalone SRS module and the other way round to use ETSI based signing:

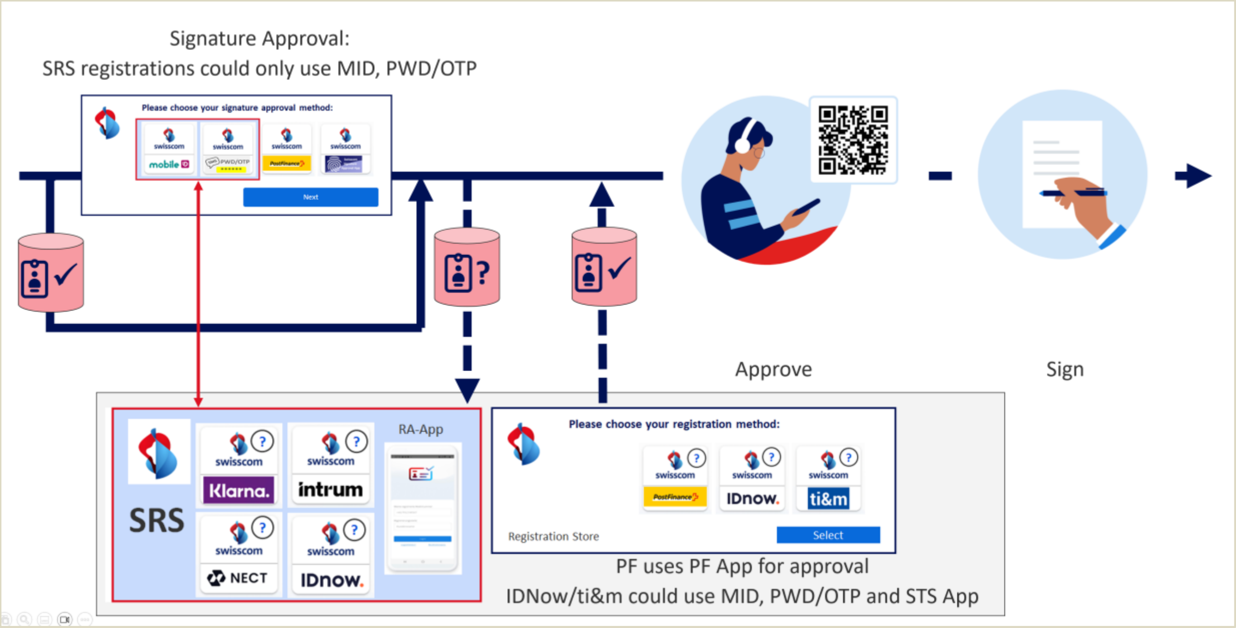

Signature Approval:
SRS registrations could only use MID, PWD/OTP
Please choose your signature approval method:
Next
Approve
Sign
RA-App
SRS
Klarna.
intrum
NECT
IDnow.
Please choose your registration method:
IDnow.
ti&m
Registration Store
Select
PF uses PF App for approval
IDNow/ti&m could use MID, PWD/OTP and STS App

Figure 17: Use of standalone SRS registrations and registrations within the Multi Authentication Broker Flow in parallel

4.13 Batch Processing

Signing Service allows signing multiple document hash values with a single request.

Next, we present the important remarks which must be taken into consideration:

- Batch signing has currently the limitation of **300** documents per batch. This limitation is since we currently use the Airlock WAF.
- We are not imposing any limitation on the number of hash values in a single request. It is recommended to use a reasonable number of around 10 hashes per sign 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.
- Each hash value can have its own digest algorithm.
- A batch must always contain at least two documents. Please do not use the batch functionality to send a batch containing one single element.

4.14 Detached Signature and Validation

Since only the document hash is provided to Signing Service, the returned signature itself is detached from the document. 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 which we strongly recommend, and which is required in most cases in eIDAS context.

There are document formats that support the integration of such detached signatures, e.g., Portable Document Format (PDF). The CMS Signature provided by Signing Service can be used as is for integrated signatures and to sample code referred in Chapter 0. The customer must perform the integration of the signature into the document by using libraries or partner solutions (referred on Signing Service website [3]).

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

5 Signing Service

5.1 Introduction

When integrating the signing service, the customers can use our ETSI RDSC interface. This interface is based on the ETSI standard for remote digital signature creation (RDSC) [ETSI TS 119 432].

The ETSI RDSC interface incorporates the latest identification and consent methods and defines the baseline for future extensions to the service offering.

We will present in more detail the contents of the next section. First, we present the whole user authentication and signing flow in a nutshell (Section 5.2.1). Second, we will provide the ETSI interface description. Third, we will describe and present the signing options (Section 5.2.4). Fourth, we will describe the broker-based user authentication (Section 5.2.5). Fifth, we will describe the claimed identities and how these are used within the broker authentication and signing flow (Section 5.2.6) using the Multi Authentication Broker. Sixth, we will describe the signing flow based on the ETSI interface (Section 5.2.7). Seventh, we will describe the errors supported by the ETSI interface (Section 5.2.8). Lastly, we will present several Postman samples used for authentication and signing (Section 5.2.9).

5.2 ETSI RDSC Interface

5.2.1 Interface Description

Creating one or multiple signatures using the ETSI RDSC interface is a procedure composed of up to two steps:

1. Request the authorization of a signature.
2. Request the creation of a signature.

Step 1 is conditionally required depending on the policies of the signing key being used. It is generally required for the creation of any qualified signature. The protocols in use build on a profile of OAuth 2.0 and rely on the use of the IDP Broker service.

Step 2 requires you to issue a signature request to the REST-style ETSI RDSC interface.

5.2.2 Signing Flow

This section describes the steps needed for authentication and signing.

This reference guide will describe the steps of the user signature approval as declaration of will to sign and the subsequent signature. It depends on the parameters of the authorization request if the calling application has a registered unique universal identifier (UUID) of the user like the mobile number or if the authentication means as signature approval means is unknown. In case the UUID is known Swisscom Trust Service will handle the authentication process directly with the corresponding IDP or will address the registered authentication means for this user otherwise a choice of possible authentication methods will be shown. For example, all registrations which took place before the signature process e.g., by using the RA-App, Swisscom Shop registration, credit card or voucher-based website registration on the following website.

https://srsident.trustservices.swisscom.com/ Another alternative is to use standalone SRS which can uses the mobile number as unique identifier that canbe passed as hint for the following authentication process.

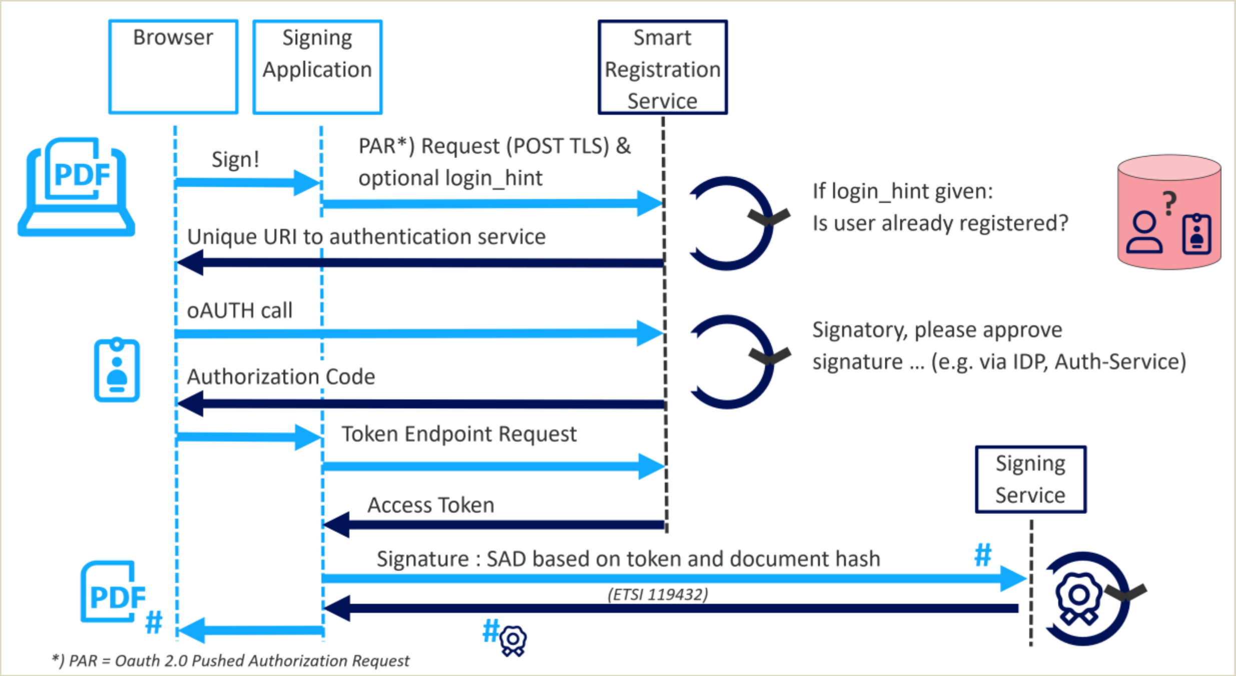

Browser
Signing Application
Smart Registration Service
Sign!
PAR*) Request (POST TLS) & optional login_hint
If login_hint given: Is user already registered?
Unique URI to authentication service
oAUTH call
Signatory, please approve signature … (e.g. via IDP, Auth-Service)
Authorization Code
Token Endpoint Request
Signing Service
Access Token
Signature : SAD based on token and document hash
(ETSI 119432)
PDF
#
*) PAR = Oauth 2.0 Pushed Authorization Request

Figure 18. the figure depicts the different systems involved during the signing process.

The figure above outlines the interactions between the customer´s browser and signing application (in light blue) and the Smart Registration & Authentication Service and the Signing Service of Swisscom in dark blue.

First the signatory will request a signature and the corresponding signature application will do an initial request and pass optionally also a hint about the authentication means used for the signature approval if they are known. The request is an oAuth call either without or with OAuth 2.0 Pushed Authorization Request (PAR) which standardizes a secure way of initiating an OIDC authorization flow using request objects. Without using the PAR, the standard authentication method selection view will be presented by Swisscom in case no hint is given. (See Figure 11)

The hint could beyond a mobile number, also an e-mail, a “sub” from the Signature Approval SDK or any other unique user identificatory. By support of this **login_hint** the Smart Registration & Authentication service will check if this user is already known. Now the decision-making engine process takes place and evaluates the given authentication based on the signing request (jurisdiction, level of signature AES/QES) and existing and valid registration based on this chosen authentication. If the authentication is not yet registered or no longer valid different registration possibilities could be offered either in a direct view offered by Swisscom or via PAR request in a way that the signing application can offer a view to the customer. In case the proper registration is found the oAuth Call will ask to activate the authentication process with the registered signature approval means. It returns an Authorization Code, allowing to request the token which is in the end necessary to pass the signature request based on the ETSI interface. Next the “Signature Activation Data” (SAD) with the corresponding hash of the document is passed. The Signing Service returns the signed hash of the document.

If we use a Password – One Time Code via SMS Service or Mobile ID service, we speak also of an IdP authentication. Even these are internal IdPs offered by Swisscom and will be enlarged by more IdP Services of Swisscom (like STS Signature Approval App) or third party IdPs (like PostFinance and others in future).

Since different signature approval methods are offered to a customer and different identification methods, we call them “authentication store” or “registration store”.

Next, we will describe the steps of the user signature approval and of the sign request based on the signing service ETSI interface in more detail. Note that the first part of the steps is used for authenticating the user (Section 5.2.3) in different ways and the last part of these steps are involving the signing service and the new ETSI signing interface.

5.2.3 Broker based Authentication and Signing

There are in total 3 different ways to integrate the authentication flow based on the requirements of the customer:

- Standard integration based on the registration and authentication store view offered by Swisscom to select the proper signature approval and identification method.
- PAR based integration to offer the authentication and registration store in an own CI/CD style and view.
- CIBA based integration for even more flexibility in styling for e.g., display of the QR code for the approval app in own CI/CD.

We will present in detail these 3 flows by depicting the differences between them and highlight how the user can integrate and use them.

5.2.3.1 Standard Broker Integration

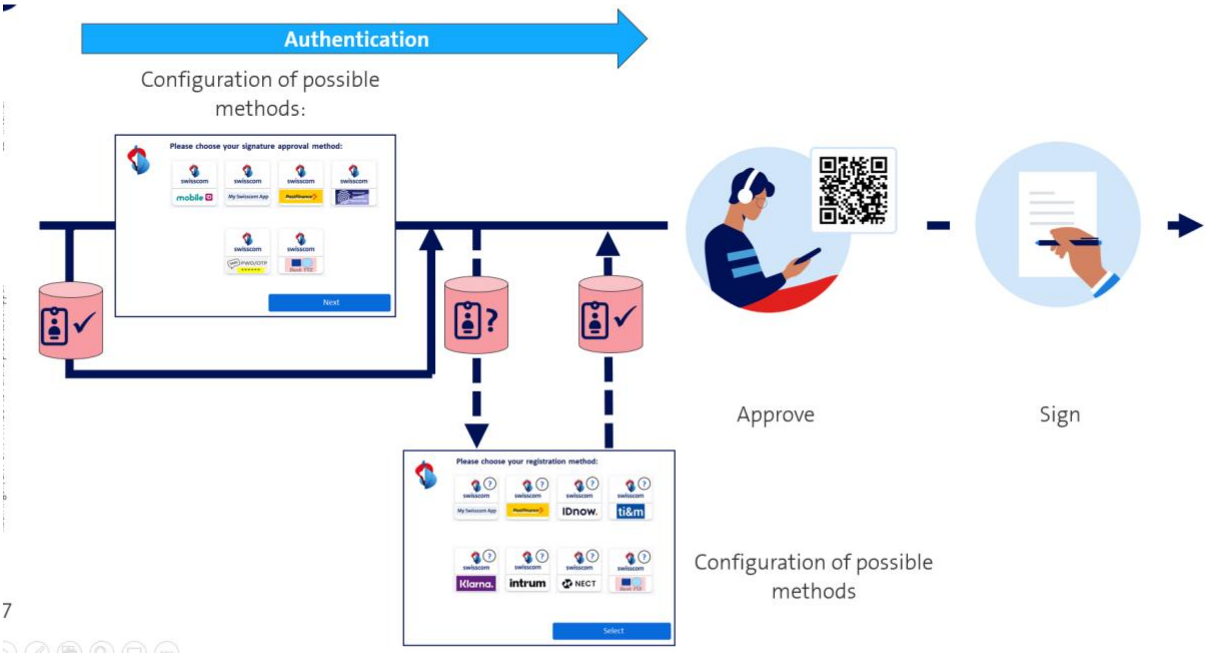

Authentication
Configuration of possible methods:
Please choose your signature approval method:
swisscom
mobile ID
My Swisscom App
PostFinance
PWD/OTP
Next
Approve
Sign
Please choose your registration method:
My Swisscom App
PostFinance
IDnow.
ti&m
Klarna.
intrum
NECT
Select
Configuration of possible methods
7

Figure 19. Standard broker integration with existing IDPs, standard authentication, registration store and signing.

Figure 19 depicts the standard IDPs which are offered in a browser view by Swisscom. It starts generally with a signing request by the signatory e.g., for a given PDF file. The signature application will trigger a signing request to Swisscom Trust Service based sending the hash of the PDF file. In the order of the signing access account the different methods for signature approval and registration in the scope of this signature application were chosen. The suitable methods based as well on the choice in the order and based on the signing request quality (e.g., jurisdiction etc.) itself will be shown. If the users signature approval method and thus the corresponding IDP is unknown (no *login_hint*) he can choose the appropriate IDP service. If not registered, the user can select the registration method from another view offered by Swisscom. The used evidence and authentication method (IDP based) will then be stored in the Registration Authority Database for the next time the user wants to sign a document. Given the successful signature approval the document hash will be signed.

5.2.3.1.1 Signing Flow

Authentication

- The client goes to the service provider website to sign a document.
- The client (i.e., user's browser) sends authentication Request containing:level of signature, jurisdiction, preferred IdP (optionally) to the Broker Authorization endpoint server
    - The Broker asks RA which IdP the user should be authenticated with (in case no *login_hint* is given)
    - The Broker displays the list of IdPs that the user can choose.
    - The Broker returns that the user is already registered in RA-DB. In case the user does not exist, the Broker will provide a list of identification methods.
- The Broker redirects client (i.e., user's browser) to a third-party identity provider with a new Authorization request (#2) including "SCAL2" or "SCAL1" ACR (Authorizaton Change Request) requested values including as the transaction message (incl. information the document(s) name(s)) and hash(s)
    - Third party identity provider checks the ACR value and Broker redirect URL (if the broker's redirect URL is not part of the whitelist, or if the ACR requested value is not supported, the third party IDP must block the transaction).
- SCAL2 authentication with optional transaction message (document hash and name)
- Third party identity provider returns an authorization code to broker.
- The broker uses the Authorization code to retrieve the IDToken by using the Authorization code on the third-party IdP Token endpoint.
    - The broker validates the OIDC Token endpoint certificate, if this one is not valid or not part of the Broker trust store, the flow is interrupted.
    - The broker validates the signature of the IDToken JWT token.
    - The broker validates the requested SCAL2 authentication (ACR Value) in the IDToken provided by the third-party identity provider.
    - The broker validates the sub claim (User unique ID) & ConsentSerialNumber (i.e., deviceID used for authentication) claims from the given IDToken.
    - The broker issues an Authorization code to the relying party, this is the signing platform. The service provider requests an access token using the previous authorization code at the Broker mTLS token endpoint.

Signing

- The Service provider performs a sign request with the access token to Signing Service 3.0 ETSI endpoint.
- The signing service validates the access token using an internal broker mTLS-based endpoint. Next, the broker validates the client certificate which was submitted by the signing service.
- The broker uses an internal endpoint to retrieve the user's identity information from the RA evidence database or from the IdP (first name, last name, country) using the DN lookup internal method. If the client certificate credentials are correct, then the broker replies with a JSON document with all requested claims from the "sign" scope and user evidence data.
- If the client certificate credentials are correct, then the broker replies with a JSON document with all requested claims from the "sign" scope and user evidence data.

5.2.3.1.2 Request and Response

Broker Request

See: https://openid.net/specs/openid-connect-core-1_0.html#CodeNotes

POST: https://auth.trustservices.swisscom.com/de/auth/realms/broker/protocol/openid-connect/

Parameter	Value	Description
auth_state	3528d31a-20a1-4a34-b78b-cb8e5de99d8c	Opaque value used to maintain state between the request and the callback. Typically, Cross-Site Request Forgery (CSRF, XSRF) mitigation.
nonce	fee0c0bc-6c74-4bde-82fa-d6a73a43b525	String value used to associate a client session with an ID Token, and to mitigate replay attacks.
response_type	Code	Informs the Authorization Server of the mechanism to be used for returning parameters from the Authorization Endpoint.
client_id	cfcda73d-858b-4f5f-b7df-xxxxxxx	OAuth 2.0 Client Identifier valid at the Authorization Server.
scope	Sign	OpenID Connect requests MUST contain the openid scope value. If the openid scope value is not present, the behaviour is entirely unspecified.
redirect_uri	https%3A%2F%2Fauth.trustservices.swisscom.com&claims=%7B%22credentialID%22%3A%22OnDemand-Advanced4%22%2C+%22documentDigests%22%3A%5B%7B%22hash%22%3A%22sTOgwOm%2B474gFj0q0x1iSNspKqbcse4IeiqlDg%2FHWuI%3D%22%2C+%22label%22%3A%22Leasing+agreement+Contract%22%7D+%5D%2C%22hashAlgorithmOID%22%3A%222.16.840.1.101.3.4.2.1%22+%7D	Redirection URI to which the response will be sent. This URI MUST exactly match one of the Redirection URI values for the Client pre-registered at the OpenID Provider, with the matching performed as described in Section 6.2.1 of **[RFC3986]** (Simple String Comparison).
code_challenge_method	S256	The method used to validate the token.

Broker Response

Parameter	Value	Description
State	3528d31a-20a1-4a34-b78b-cb8e5de99d8c&	If the state parameter is present in the Authorization Request. Clients MUST verify that the state value is equal to the value of state parameter in the Authorization Request.
Code	d80750a9-b24e-4a90-bcac-3db9daaa928f	When using the Authorization Code or Hybrid flows, an ID Token is returned from the Token Endpoint in response to a Token Request using an Authorization Code.

5.2.3.1.3 IDP Usage and Integration

- We offer standard authentication methods and standard ID methods which cover mostly all customer needs.
- Additional some customers may want to contribute additional ID Methods / Auth Methods.
- Note that these IDPs need to undergo a separate onboarding based on implementation concept and audit.
- Further, there are IDPs which use their IDP method without import of the evidence.
- Also, there are IDPs for which we support import of evidence into the RA DB.

5.2.3.1.4 QR based Signing Approval

The user can use the, for example, the MySwisscom app to scan a QR code to authorize the signing of the PDF file. See, for example, the Futurae API:

https://www.futurae.com/docs/api/auth/?json#authenticate-with-one-touch

Parameter	Value	Description
QR code content example	anonymous:9mDPnXyC6vm9TlCe84lyDUpF6tpv_Lqtxrg3XluJnUOg	PNG image of a scannable QR code containing the activation code in data URI scheme.
Registration QR code	d0hqWlI3Q29aNHVBT0k2TndteC1jN19rbTFreUp4UzNsUkM5RGc1ZlIxNWc6MDgyNWNhMDQtOWU2NC00NzNmLWI0MjUtZmMyMTMzM2I5ZWIwOmFwaS5mdXR1cmFlLmNvbQ	A short activation code is suitable for delivering to the user and requiring them to supply it

		during the FIDO registration process

5.2.3.2 PAR Based Broker Integration

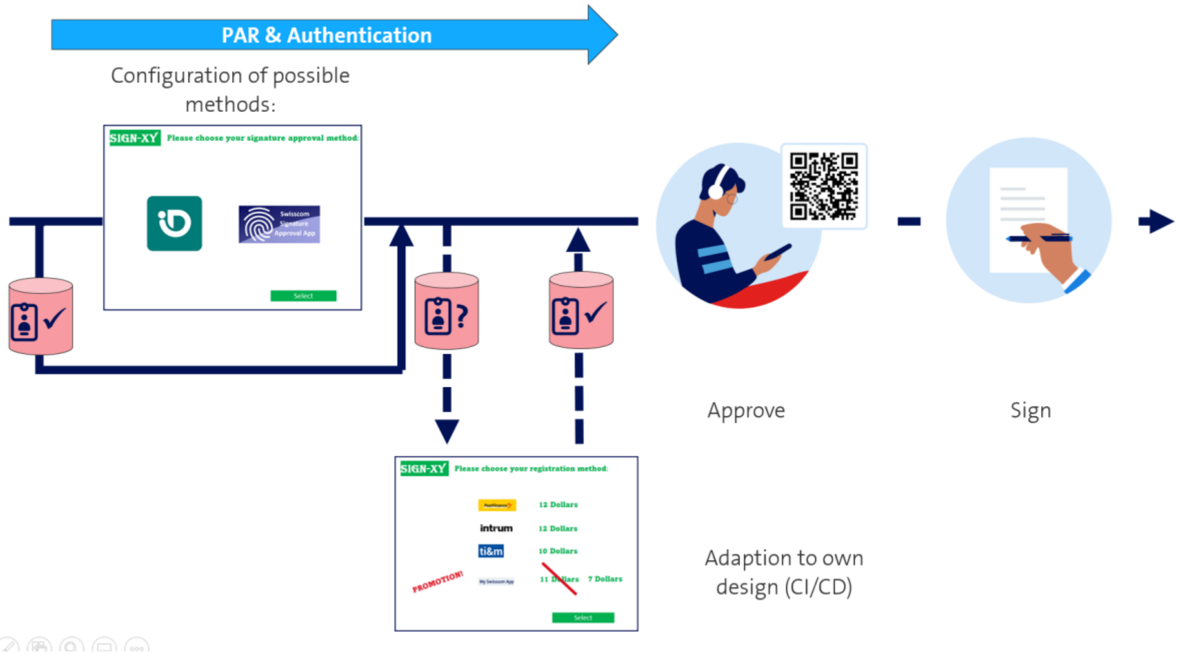

PAR & Authentication
Configuration of possible methods:
SIGN-XY
Please choose your signature approval method:
Swisscom Signature Approval App
Select
SIGN-XY
Please choose your registration method:
PostFinance
12 Dollars
intrum
12 Dollars
ti&m
10 Dollars
PROMOTION!
My Swisscom App
~~11 Dollars~~
7 Dollars
Select
Approve
Sign
Adaption to own design (CI/CD)

Figure 20. PAR based authentication and signing flow using the MySwisscom App.

Figure 20 presents the signing flow based on the PAR request. The signing flow is the same as described in the previous chapter. Based on the PAR request the user will not get a Swisscom designed view of selectable signature approval methods and registration methods but will get a view created by the signature application provider. For this the signature application provider gets a list of possible methods and can decide how to present them to the user. This is, for example, also the only possibility to include any pricing information concerning the different methods, if necessary. A view designed by Swisscom will not be able to show any pricing. The signing flow below describes the details.

5.2.3.2.1 Signing Flow

Authentication

- The client goes to the service provider website to sign a document.
- The signature platform needs to do a PAR request.
- The response will contain a `request_uri`.
- The client (i.e., user's browser) sends authentication request containing the previously obtained `request_uri`, level of signature, jurisdiction, preferred IdP to the broker Authorization endpoint server.
  - The broker asks RA which IdP the user should be authenticated with (in case no login_hint is given)
  - The broker returns that the user is already registered in RA-DB. In case the user does not exist in the RA-DB, the broker will provide a list of identification methods.

- The broker redirects the client (i.e., user's browser) to a third-party identity provider with a new authorization request (#2) including "SCAL2" or "SCAL1" ACR (Authorization Change Request) requested values including as the transaction message (incl. information the document(s) name(s)) and hash(s)
    - Third party identity provider checks the ACR value and broker redirect URL (if the broker's redirect URL is not part of the whitelist, or if the ACR requested value is not supported, the third party IdP must block the transaction)
- SCAL2 authentication with optional transaction message (document hash and name) is performed.
- Third party identity provider returns an Authorization code to Broker.
- The broker uses the authorization code to retrieve the IDToken by using the authorization code on the third-party IDP token endpoint.
    - The broker validates the OIDC Token endpoint certificate, if this one is not valid or not part of the broker trust store, the flow is interrupted.
    - The broker validates the signature of the IDToken JWT token.
    - The broker validates the requested SCAL2 authentication (ACR Value) in the IDToken provided by the third-party identity provider.
    - The Broker validates the sub claim (User unique ID) & ConsentSerialNumber (i.e., deviceID used for authentication) claims from the given IDToken.
    - The broker issues an Authorization code to the Service provider.
    - The service provider requests an access token using the previous authorization code at the broker mTLS token endpoint.

Signing

- The Service provider performs a sign request with the access token to Signing Service 3.0 ETSI endpoint.
- The signing service validates the access token using an internal broker mTLS-based endpoint. Next, the broker validates the client certificate which was submitted by the signing service.
- The broker uses an internal endpoint to retrieve the user's identity information from the RA evidence database or from the IdP (first name, last name, country) using the DN lookup internal method.
- If the client certificate credentials are correct, then the broker replies with a JSON document with all requested claims from the "sign" scope and user evidence data.

5.2.3.2.2 Swisscom Signature Approval App

PAR request

POST: .....api/auth/realms/broker/protocol/openid-connect/ext/par/request

Parameter	Value	Description
client_id	23bn32423asa.xxy........	Client Identifier valid at the Authorization Server.
client_secret	23mm324icsq5555.......	Clients that have received a client secret from the Authorization Server to authenticate with the Authorization Server in accordance with Section 2.3.1 of **OAuth 2.0** [RFC6749] using the HTTP Basic authentication scheme.

login_hint	Composed of identifier and namespace	Hint to the Authorization Server about the login identifier the End-User might use to log in (if necessary).
identifier	……	sub identifier type, locally unique and never reassigned identifier within the Issuer for the End-User, which is intended to be consumed by the client.
namespace	Different types of names: MSA, STS, PFM, MID, PWD-OTP	These are namespaces used to identify different IDPs.
client_session_id	test_session_id	Broker session ID bound to the browser cookie
redirect_uri	https://rax-preprod-blue.scapp.swisscom.com	Redirection URI to which the response will be sent. This URI MUST exactly match one of the Redirection URI values for the Client pre-registered at the OpenID Provider, with the matching performed as described in Section 6.2.1 of **[RFC3986]** (Simple String Comparison).

claims	Composed of credentialID and documentDigests. The content of the claims is proprietary. Example:
	"claims": {
	"credentialID": "OnDemand-Qualified4.1-EU",
	"documentDigests": [
	{
	"hash": "sTOgwOm+474gFj0q0x1iSNspKqbcse4IeiqlDg/HWuI=",
	"label": "Leasing agreement Contract 1"
	}
	],
	"hashAlgorithmOID": "2.16.840.1.101.3.4.2.1"
	}
		The JSON example contains the name of the document to be signed, hash of the document to be signed, and credential ID.

credentialID	OnDemand-Qualified4.1-EU	This is one of the parameters from the claimes above. Note that the credentialID can have different values.
documentDigests	composed of hash and label	This is one of the parameters from the claimes above
hash	sTOgwOm+474gFj0q0x1iSNspKqbcse4IeiqlDg/HWuI=	This is one of the parameters from the claimes above

label	Leasing agreement Contract 1	This is one of the parameters from the claimes above
hashAlgorithmOID	2.16.840.1.101.3.4.2.1	This is one of the parameters from the claimes above

PAR Response

Parameter	Value	Description
request_uri	urn:example:bwc4JK-ESC0w8acc191e-Y1LTC2	This parameter enables OpenID Connect requests to be passed by reference, rather than by value. The request_uri value is a URL using the https scheme referencing a resource containing a Request Object value, which is a JWT containing the request parameters.
expires_in	90	The configured lifetime of the request URI, in seconds.

5.2.3.2.3 Authorize Request and Response

Authorization Request

GET: https://auth.trustservices.swisscom.com/de/auth/realms/broker/protocol/openid-connect/

Parameter	Value	Description
auth_state	3528d31a-20a1-4a34-b78b-cb8e5de99d8c	Opaque value used to maintain state between the request and the callback. Typically, Cross-Site Request Forgery (CSRF, XSRF) mitigation.
nonce	fee0c0bc-6c74-4bde-82fa-d6a73a43b525	String value used to associate a client session with an ID Token, and to mitigate replay attacks.
response_type	code	Informs the Authorization Server of the mechanism to be used for returning parameters from the Authorization Endpoint.
client_id	cfcda73d-858b-4f5f-b7d-xxxxxx	OAuth 2.0 Client Identifier valid at the Authorization Server.
scope	sign	OpenID Connect requests MUST contain the openid scope value. If the openid scope value is not present, the behaviour is entirely unspecified.
redirect_uri	https%3A%2F%2Fauth.trustservices.swisscom.com&claims=%7B%22credentialID%22%3A%22OnDemand-Advanced4%22%2C+%22documentDigests%	Redirection URI to which the response will be sent. This URI MUST exactly match one of the Redirection URI values for the Client pre-registered at the OpenID Provider, with

	22%3A%5B%7B%22hash%22%3A%22sTOgwOm%2B474gFj0q0x1iSNspKqbcse4IeiqlDg%2FHWuI%3D%22%2C+%22label%22%3A%22Leasing+agreement+Contract%22%7D+%5D%2C%22hashAlgorithmOID%22%3A%222.16.840.1.101.3.4.2.1%22+%7D	the matching performed as described in Section 6.2.1 of **[RFC3986]** (Simple String Comparison).
code_challenge_method	S256	The method used to validate the token.

Authorization Response

Parameter	Value	Description
state	3528d31a-20a1-4a34-b78b-cb8e5de99d8c	If the state parameter is present in the Authorization Request. Clients MUST verify that the state value is equal to the value of state parameter in the Authorization Request.
code	d80750a9-b24e-4a90-bcac-3db9daaa928f	When using the Authorization Code or Hybrid flows, an ID Token is returned from the Token Endpoint in response to a Token Request using an Authorization Code.

POST: ..... auth/realms/broker/protocol/openid-connect/token

Parameter	Value	Description
grant_type	authorization_code	A Client makes a Token Request by presenting its authorization grant (in the form of an Authorization Code) to the Token Endpoint using the grant type value authorization code , as described in Section 4.1.3 of **OAuth 2.0** [RFC6749].
code	d716e1d1-1018-4b24-bb33-d1899907582c	As described above. The code is part of the grant type the user makes.
client_id	x87362vn.......	Client ID value for the Client, which in this case contains the redirect_uri value of the Client. Since the Client's redirect_uri URI value is communicated as the Client ID, a redirect_uri parameter is NOT REQUIRED to also be included in the request.

5.2.3.2.4 Registration Method Selection

- The user has the possibility to select his desired user registration method. For example, Intrum, PostFinance, etc.

5.2.3.2.5 User Identity RA-DB Import

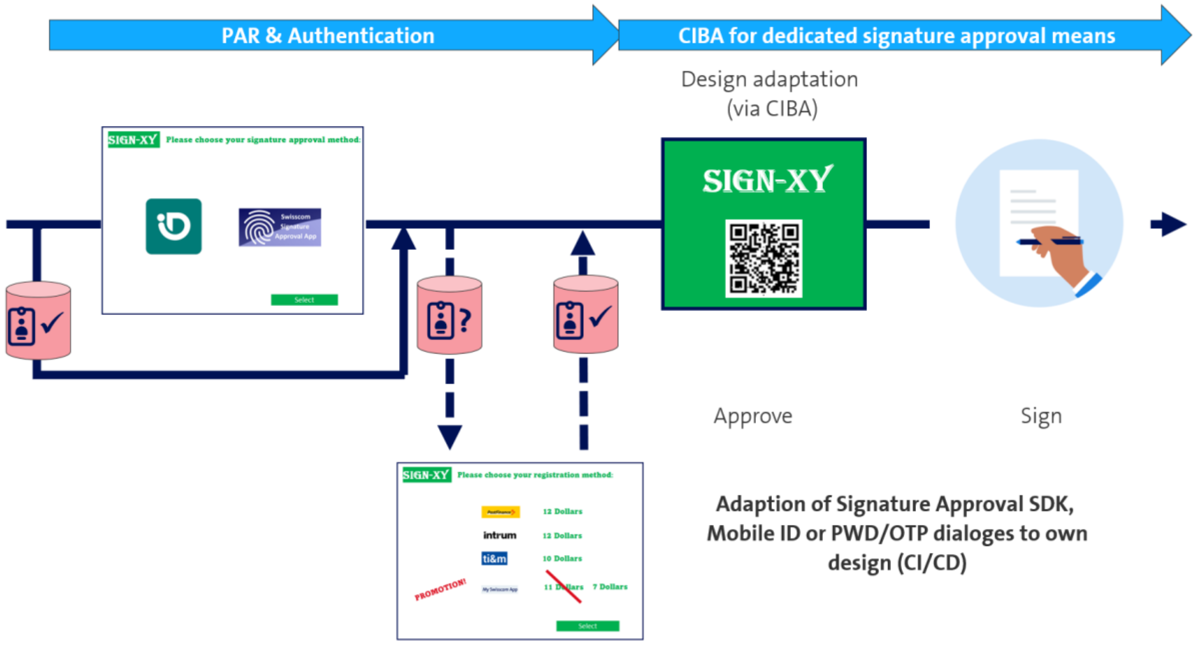
- In case the user identity is not present in the RA-DB than this will be imported into the RA-DB based on information provided by the IDP which was selected by the user for authentication. This is done once when the user authenticates for the first time with the selected IDP. This import will be done only once.

5.2.3.2.6 Signature Approval based on My Swisscom App

- The My Swisscom App will be used to scan the QR code and authorize the signature. This is the last step before the signature can be created by the signing back-end.

5.2.3.3 CIBA Based Broker Integration

The CIBA based authentication flow works only with the applications which support the Futurae SDK. CIBA should be used in case the customer wants to customize the GUI of his application.



Adaption of Signature Approval SDK, Mobile ID or PWD/OTP dialoges to own design (CI/CD)

Figure 21. CIBA based authentication, registration, authorization and signing flow.

Figure 21 depicts the signing flow based on the CIBA based backchannel calls. In detail, if the selection of some methods – such as the Swisscom Trust Services Approval app – is done the specific method will be started. It could now have views which are designed by Swisscom like the request to scan a QR code with the smartphone the appropriate app is installed. If even here the CI/CD of Swisscom is not wanted the CIBA interface could be a solution. It must be carefully handled in interaction with the Swisscom team since the signature application provider is now more dealing with the signature approval process itself and this could be seen by the auditor as risk of manipulation. In questionable situations at least a walkthrough by the auditor could be necessary. The following technical flow is hereby applied:

5.2.3.3.1 Signing Flow

Authentication

- The client goes to the service provider website to sign a document.
- The signing platform performs a POST auth backchannel POST request.
- In the CIBA Post request, we put the following information:
  - signature level, jurisdiction, preferred IdP
- The signing platform obtains the response and inside the response there is the auth_req_id
- The user is authenticated with a compatible Futurae App
- The broker gets a successful call-back about the successful authentication.
- The Broker uses the call-back payload to retrieve the IDToken.
  - The Broker validates the OIDC Token endpoint certificate, if this one is not valid or not part of the Broker trust store, the flow is interrupted.
  - The Broker validates the signature of the IDToken JWT token.
  - The Broker validates the requested SCAL2 authentication (ACR Value) in the IDToken provided by the third-party identity provider.
  - The Broker validates the sub claim (User unique ID) & ConsentSerialNumber (i.e., deviceID from Futurae used for authentication) claims from the given IDToken.
- The previous obtained request Id is then used for the token request.
- The token response gives an access token if the user was authenticated.

Signing

- The service provider performs a sign request with the access token to Signing Service 3.0 ETSI endpoint.
- The signing service validates the access token using an internal broker mTLS-based endpoint. Next, the broker validates the client certificate which was submitted by the signing service.
- The broker uses an internal endpoint to retrieve the user's identity information from the RA evidence database or from the IdP (first name, last name, country) using the DN lookup internal method.
- If the client certificate credentials are correct, then the broker replies with a JSON document with all requested claims from the "sign" scope and user evidence data.

5.2.3.3.2 CIBA Request and Response

There are in total 2 CIBA backchannel calls.

See the CIBA API: https://curity.io/resources/learn/ciba-flow/

First CIBA auth request

POST:……/api/auth/realms/broker/protocol/openid-connect/oauth-authorize HTTP/1.1

Parameter	Value	Description
scope	sign	OpenID Connect requests MUST contain the openid scope value. If the openid scope value is not present, the behaviour is entirely unspecified. Other scope values MAY be present. Scope values used that are not understood by an implementation SHOULD be ignored.

client_id	84ecf111-f0c0-457f-9e0c-caf8f1axxxxxxxxx	OAuth 2.0 Client Identifier valid at the Authorization Server.
client_secret	56b96882af-c84xxxxxxx	Clients that have received a client secret from the Authorization Server authenticate with the Authorization Server in accordance with Section 2.3.1 of **OAuth 2.0** [RFC6749] using the HTTP Basic authentication scheme.
login_hint_token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.xxx	Hint to the Authorization Server about the login identifier the End-User might use to log in (if necessary).

claims_token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjcmVkZW50xxxxx
	Example:
	"claims": {
	"credentialID": "OnDemand-Qualified4.1-EU",
	"documentDigests": [
	{
	"hash": "sTOgwOm+474gFj0q0x1iSNspKqbcse4IeiqlDg/HWuI=",
	"label": "Leasing agreement Contract 1"
	}
	],
	"hashAlgorithmOID": "2.16.840.1.101.3.4.2.1"
	}
		Contain the names of the document, hash to be signed, and credential ID.

First CIBA Response

Parameter	Value	Description
auth_req_id	bspuw6ea-scst-u5hn-p3nt-37khzwY4g	This is the unique ID of the authentication request. It is used in the token requests to identify the transaction started by the client's authentication request. Thus, the client must keep this parameter.
expires_in	900	The number of seconds, since the authentication request

		was received, that the authentication request will be valid for.
interval	5	The number of seconds that the client must wait between two polling requests. The default value is 5.

Second CIBA token request

- POST: …./oauth/v2/oauth-token

Parameter	Value	Description
grant_type	urn%3Aopenid%3Aparams%3Agrant-type%3Aciba	The flow to run to retrieve tokens. For CIBA, the value must be urn:openid:params:grant-type:ciba.
auth_req_id	bspuw6ea-scst-u5hn-p3nt-37khzwY4g	The unique identifier for a transaction initiated by the client. The Authorization Server uses this identifier to retrieve the status of the authentication and issue tokens if authentication was successful and the user gave consent.
client_id	consumer_device	When using mutual TLS for client authentication, the client must include its client_id in the requests.

Second CIBA response

Parameter	Value	Description
access_token	5ca69968-00ca-4535-82e3-c7badf90b51b	If the token request is valid, and if the user has been authenticated and has authorized the request, the Authorization Server will return a successful response. The response will contain the access_token.

5.2.3.3.3 Registration Method Selection

- Based on the customer needs he can select between different methods for registration as depicted in the previous picture.

5.2.3.3.4 Design Adaptation

- This part of the authentication flow is under customer control. The customer can choose his own application GUI design. In this way the user achieves his own look and feel of the application.

5.2.4 Signing Options

There are two major parameters which generally control the flavor and scope of the signing data produced in a signature request:

- conformanceLevel
- signatureFormat

The following sections provide more details on their use.

5.2.4.1 Parameter Conformance Level

The **conformanceLevel** parameter controls the profile of the CMS signature created as result of the signing process and the scope of data conveyed within or in addition to the signature.

By requesting a signature of level AdES-B-B, a signature is created which neither encompasses a signature timestamp nor revocation information.

By requesting a signature of level AdES-B-T, a signature is created which contains a signature timestamp but no revocation information.

By requesting a signature of level AdES-B-LT, a signature is created which encompasses both a signature timestamp and revocation information. This is the baseline for the creation of LTV-enabled signatures.

5.2.4.2 Parameter Signature Format

The **signatureFormat** parameter controls the flavor of the CMS signature created as result of the signing process and the method by which eventual revocation information is conveyed (embedded or detached).

By requesting a PAdES signature, Signing Service ensures that the CMS is suitable for integration into a PDF, resulting in a CMS conforming to one of the profiles defined in [ETSI EN 119 142-1] and [ETSI EN 119 142-2]. It is up to the client to integrate the received data into a PDF signature object. If the **conformanceLevel** requires the provisioning of revocation information, it is provided in detached form in the resulting JSON data structure.

By requesting a CadES signature, Signing Service ensures that the CMS is suitable for use as a self-contained detached signature, resulting in a CMS conforming to one of the profiles defined in [ETSI EN 119 122-1] and [ETSI EN 119 122-2]. It is up to the client to integrate the received data into a PDF signature object. If the **conformanceLevel** requires the provisioning of revocation information, it is provided in embedded form within the resulting CMS.

5.2.5 Authentication Service

The broker is based on the OpenID connect protocol. The goal of the broker is to bundle the authentication methods we provide for customers, with the goal to delegate sign requests to the Signing Service back-end.

5.2.5.1 User Authentication to the Broker Service

After a successful onboarding process the service provider can access the broker using OIDC authentication standard. The following describes the required authentication steps for each endpoint.

#	Endpoints	Authentication
1	Authorization	Service provider needs a valid client identifier (ClientID)* – This is a UUID format where Service provider needs to have at least one registered valid redirect URL
2	Token	Service provider needs a valid ClientID, Client Secret and a valid mTLS Client Certificate* to access this endpoint

*All these credentials are given during the onboarding process.

5.2.5.2 OIDC Token Endpoint

Given the Authorization code, the service provider can use the following URL to retrieve the Access Token. Using the following **curl** request, replace the provided **client-cert.pfx** file, the associated p12 password file, the **Client_ID**+Value, the **client_id, client_secret**, the PKCE verifier value and the authorization CODE with your own values.

Parameter Name	Description
grant_type	Authorization code
code	The authorization code generated by the broker after the user was successfully authenticated
client_id	The id of the client, note that this is different for each client.

Request:

The request is URL encoded.

TYPE: x-www-form-urlencoded

POST: https://<host>:<port>/auth/realms/broker/protocol/openid-connect/token

The response is a JWT Access Token

Parameter Name	Description
access_token	This is the JWT token
expires_in	The lifetime of the token in milliseconds
token_type	The of the token, in this case bearer
session_state	The of the transaction which is also present in the JWT token
scope	There is only one scope, sign

Response:

```
{
   "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NzY0NzIzMjMsImlhdCI6MTY3NjM
4NTkyMywibmJmIjoxNjc2Mzg1OTIzLCJhdXRoX3RpbWUiOjE2NzYzODU5MjMsImp0aSI6ImYzOWY3NzBlLTE
2MGQtNDJjZS1iMDI3LTI4OWE4YTdmYjk3MSIsImlzcyI6Imh0dHBzOi8vcmF4LXByZXByb2QtZ3JlZW4uc2Nhc
HAuc3dpc3Njb20uY29tL2F1dGgvcmVhbG1zL2Jyb2tlciIsImF1ZCI6Imh0dHBzOi8vcmF4LXByZXByb2QtZ3JlZ
W4uc2NhcHAuc3dpc3Njb20uY29tL2F1dGgvcmVhbG1zL2Jyb2tlciIsImF6cCI6IjYyMmM3ZGRiLWNkZTgtNDA
wNS1iZWQxLWQwZTllZWM3MjNiNyIsInN1YiI6InNaMW1oQ3kvdGpBZTNzQjdOQVNFaTQvK1JKajNmRDhh
cVRSTzRUcTlnKzg9IiwidHlwIjoiQmVhcmVyIiwic2Vzc2lvbl9zdGF0ZSI6Ijk0NjQzMGY1LWMzYTYtNDVhNy1h
MjJiLTFlMTgwOWI5OGY2NyIsImFjciI6ImFwcGlkIiwibm9uY2UiOiI5aE0vVGNIMG5ZeVFVUDhWIiwic3RydW
N0dXJlZF9zY29wZSI6eyJzaWduIjp7ImNyZWRlbnRpYWxJRCI6Ik9uRGVtYW5kLUFkdmFuY2VkNCIsImRvY3Vt
ZW50RGlnZXN0cyI6W3siaGFzaCI6InNUT2d3T20rNDc0Z0ZqMHEweDFpU05zcEtxYmNzZTRJZWlxbERnL0h
XdUk9IiwibGFiZWwiOiJMZWFzaW5nIGFncmVlbWVudCBDb250cmFjdCJ9XSwiaGFzaEFsZ29yaXRobU9JRCI
6IjIuMTYuODQwLjEuMTAxLjMuNC4yLjEifX19.Y7frDIHyrATEWU5Y4JObIgmrpPTFaKAMWP3gqVbcLbtgX-
pUtktxRt9Lg6FhLGzCQL57yYWactdMz7rzUYvoF6kv2uekFikRdLICKUTIbo1LdyFbuO8g4cmEtCTD40FzN8PIjA
SHQu_9H3XLdEWWAOkZy-t1lWoVbjT5VQgv601KfVANOKSDhFPJj-KG-xxxxxxxxxxxxxxxxxxxxxxxxxxxx",

   "expires_in": "86400",

   "token_type": "Bearer",

   "session_state": "946430f5-c3a6-45a7-a22b-1e1809b98f67",

   "scope": "sign"

}
```

5.2.5.3 mTLS Broker Certificate Ordering and Configuration

This section describes the mTLS certificate ordering and configuration process such that it can be used to authenticate with the Broker.

5.2.5.3.1 Signing Account Order Process

- In parallel to the order the customer or at least the subscriber operating the signature application must fill out the Declaration of Acceptance for the signature application. Here the setup process is described, and important duties are confirmed to be fulfilled (e.g., to show the document to be signed to the signatory, to handle securely the hash information and to protect the application against attacks).
- It is also possible to request only a test account. A special test account order from will be provided with detailed information about the next steps.
- Part of the setup is the access certificate to connect mTLS wise the signature application with the Smart Registration and Signing Service. For this a Certificate Signing Request (CSR) must be placed containing the following common name (CN):
    - CN:<UUID – see below>
    - O= Name of the organisation
    - OU= <optional> name of the suborganization
    - L= locality of the organization

  - C=country of the organization
  - Email=E-Mail address for our support
- The UUID can be determined via UUID v4 Online - live generator (uuidonline.com), it is important to choose version 4!
- Key length shall be 4096 bit and the validity period must be exactly one year.
- To place the CSR and the private key you could use the OpenSSL library like: openssl req -new -newkey rsa:4096 -nodes -keyout <yourNameForYourPrivateKey>.key -out <yourNameForTheCSRrequ-est>.csr
  In this way the private key will always stay with the customer. The certificate must later include by the user in the sign request as an **ssl_client_cert** header.
- Normally the same certificate is used for the Signing Service ETSI signing interface and the Multiple Authentication Broker.
- In case of seals the CSR for the access certificate as authentication means is created in a manner described in an implementation concept for seals. For example, it could be created in a common process hosted by staff from Swisscom and the customer or based on a personal signature issued in a special solution for creating CSRs for seals.
- Note that in the productive trial environment all signatures are only advanced electronic signatures and signature certificates show the “TEST” string in the common name or even name fields of the signing certificate.
- The Swisscom setup team will now generate the certificate and announce the access parameters, the “ClaimedID”. It will also configure in case of personal signatures the possible authentication and registration methods allowed for this signature application based on the choice made in the order sheet.

5.2.6 Claimed Identity

Signing Service Service

The claimed identity provided to you in the partner registration process consists of two parts:

<customer name>:<key entity>

Please be aware that only the <key entity> is to be used as credential ID within the ETSI protocol. The customer's name is implicitly deduced from the service authorization.

The credential ID is conveyed as parameter “credentialID” within the /signDoc call.
Further resources for this can be found here:

- The signDoc call, see Section 5.2.7.4 for more details.
- Sample request, see Section 5.2.7.5.1 for more details.
- the OpenAPI doc, see Section 5.2.7.1 for more details.

The same value is passed as “credentialID” to the brokers /authorize endpoint.

Multiple Authorization Broker and Smart Registration Service

The credential ID is used in the broker authorization call. As the credential ID is provided to the broker it must be checked. More exactly, the broker checks that the customer LoA level matches the given credential ID.

5.2.7 Signing Service Interface

This section describes the Signing Service ETSI standard based signing interface.

5.2.7.1 Interface Description

The ETSI interface conforms to the ETSI standard [ETSI Open API]. It adds additional data structures from the CSC v2 specification for the provisioning of validation information. For more details we will provide the YAML specification as open-source documentation. We provide a YAML file specification with the ETSI interface description. [ETSI Open API]ETSI Open API

ETSI Open APIETSI Open APIETSI Open APIETSI Open API

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	Header Value
Content-Type	application/json;charset=UTF-8
Accept	application/json

5.2.7.2 Profile Description

Signing Service provides a subset of the features defined by ETSI standard and adds custom features derived from the CSC v2 specification. This section highlights the major profile elements to be considered.

- **Endpoints**

  The interface provides the following ETSI RDSC endpoints: info signatures/signDoc

  The endpoints are available below by using the following base URI:

https://<host>:<port>/<Signing Service-Server context>/etsi/standard/rdsc/v1/signatures/signDoc

- **Service Authorization**

  To access the new ETSI endpoint, a mTLS certificate is required. The mTLS certificate can be for example generated by using OpenSSL.

5.2.7.3 Credential Authorization

For signatures based on static certificates, credential authorization is implicit, solely relying on the service authorization. The SAD needed for the creation of a signature is to be provided as an empty string.

For signatures based on short-lived certificates, OAuth 2.0 Authorization Code Grant is used. The authorization protocol does not match the specification in the ETSI / CSC standard. The access token resulting from the authorization flow is used as SAD input to signatures/signDoc.

The token can be obtained by calling the following endpoint:

This is the RAX PROD Endpoint:

- POST: https://auth-trustservices.mtls-scapp.swisscom.com/api/auth/realms/broker/protocol/openid-connect/token

- Note: "-green" keyword from the URL above will be removed in the release from April 2023.

The body (x-www-from-urlencoded) needed is described in the table below:

Parameter Name	Description
grant_type	e.g., authorization_code
Code	Obtained code in the previous step
client_id	The secret client_id

5.2.7.4 Signature Creation

Endpoint:

https://<host>:<port>/<Signing Service-Server context>/etsi/standard/rdsc/v1/signatures/signDoc

Supported Endpoint Parameters

Name	Reference	Profile / remarks	Description
Profile	[ETSI TS 119 432] v1.2.1. standard, section 7.15	Must be "http://uri.etsi.org/19432/v1.1.1#/creationprofile#"	This is the ETSI profile used for signing
requestID	[ETSI TS 119 432] v1.2.1. standard, section 7.3	-	Identifier for the request
credentialID	[ETSI TS 119 432] v1.2.1. standard, section 7.8	Different credential IDs can be used based on the jurisdiction and the needs of the user.	Example of credential ID based on the LoA for the customer who wants to use when signing: OnDemand-Advanced4 / OnDemand-Qualified4 / OnDemand-Qualified4.1-EU / static-diamant4-1-eu / static-saphir4-ch / static-saphir4-1-eu
signatureFormat	[ETSI TS 119 432] v1.2.1. standard, section 7.16	-	The supported signature formats P, C, and X.
conformanceLevel	[ETSI TS 119 432] v1.2.1. standard, section 7.16	Supported values: AdES-B-B, AdES-B-T, AdES-B-LT	We support 3 types of signature conformance levels.
documentDigests	[ETSI TS 119 432] v1.2.1. standard, section 7.19	Check that the digest is correct.	Array which contains the hash algorithm and the hashes of the document to be signed

SAD	[ETSI TS 119 432] v1.2.1. standard, section 7.4.2	For on-demand signature: Value of the access token issued by the IDP For static signature: "" (empty string)	The JWT token which is generated by RA service. It contains the identifiable customer data, as this is contained in the RA service. The RA service passes this JWT token to the signing endpoint.
hashAlgorithmOID	-	-	OID which identifies the hashing algorithm
Hashes	-	-	The hashes of the documents to be signed

Supported response parameters.

Name	Reference	Profile / remarks
responseID	[ETSI TS 119 432] v1.2.1. standard, section, 7.26	-
signatureObject	[ETSI TS 119 432] v1.2.1. standard, section, 7.21	Array of base64-encoded CMS signatures
validationInfo	none	Validation information to be embedded into the resulting signed document to achieve AdES-B-LT level. Only returned when **signatureFormat** is 'P' and **conformanceLevel** is 'AdES-B-LT'. This structure is derived from CSC specification v2.

5.2.7.5 Sample Requests

5.2.7.5.1 On-demand Qualified Request

Note that … means that the text was cropped due to extensive length.

Sign Request Payload

```
POST /Signing Service-Server/etsi/standard/rdsc/v1/signatures/signDoc HTTP/1.1

Host: ais.intarsys.de

Content-Type: application/json

Content-Length: 1863

{

  "SAD": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJwSnZFWVFSM1lxTDlqdU4wMzBjV2lK…

YRolwkkin936BnJdiZVjYKf6dhZ2LDIH7xKah8I9JWFC51mWtJlcMT1gzctlRYNjWqEjxa8cDCSqSsRg7…
qWyRTTe0PHCkIJVEviQ377LcqfHySAjYEq-hZNFNa_5tNINKo5edpUuR_fEvDUncKcqq9rsPHC-
JUizgUmREkupC_fV065fk5ExurWjUatZNYD-fyDcxmEt_g9XHeBidWz9jfkaV6qMJk18w",
  "requestID": "f1ef942a-01ed-4515-83fc-e136d774393b",
  "credentialID": "OnDemand-Qualified",
  "profile": "http://uri.etsi.org/19432/v1.1.1#/creationprofile#",
  "signatureFormat": "P",
  "conformanceLevel": "AdES-B-LT",
  "documentDigests": {
    "hashAlgorithmOID": "2.16.840.1.101.3.4.2.1",
    "hashes": [
      "HLNTuE2+zWOo+p1VfQdjdEjDC9xcLfVdqdHYX2gwTFM=",
      "sHS3ei9wNyR/rGu5ghto/v0+h22wmdlD3TGxRyO/sgM="
    ]
  }
}
```

Sign Request Response (Success Case) HTTP 200

```
{
 "validationInfo": {
  "ocsp": [
   "MII...AGk="
  ],
  "crl": [
   "MII...sNrl="
  ]
 },
 "responseID": "fdf41e6a-382a-4512-afe9-fd2a9bab30d7",
 "SignatureObject": [
  "MII2...23w4=",
  "MII2...m5c="
 ]
}
```

Sign Request Response (Failure Case) HTTP 4xx/5xx

```
{
 "error": "invalid_request",
  "error_description": "Required parameter `SAD` is missing"
}
```

5.2.7.5.2 Static Request

Note that … means that the text was cropped due to extensive length.

Sign Request Payload

```
POST /Signing Service-Server/etsi/standard/rdsc/v1/signatures/signDoc HTTP/1.1
Host: ais.intarsys.de
Content-Type: application/json
Content-Length: 498
{
  "SAD": "",
  "requestID": "f1ef942a-01ed-4515-83fc-e136d774393b",
  "credentialID": "static-key-pair",
  "profile": "http://uri.etsi.org/19432/v1.1.1#/creationprofile#",
  "signatureFormat": "P",
  "conformanceLevel": "AdES-B-LT",
  "documentDigests": {
    "hashAlgorithmOID": "2.16.840.1.101.3.4.2.1",
    "hashes": [
      "HLNTuE2+zWOo+p1VfQdjdEjDC9xcLfVdqdHYX2gwTFM=",
      "sHS3ei9wNyR/rGu5ghto/v0+h22wmdlD3TGxRyO/sgM="
    ]
  }
}
```

Sign Request Response

```
  {
"responsetID": "f1ef942a-01ed-4515-83fc-e136d774393b", "SignatureObject": [
"MIIwlQYJKoZIhvcNAQcCoIIwhjCCMIICAQExDzANBglghkgBZQMEAgEFADALBgkqhkiG9w0BBwGgggwO
MIIF/TCCA+WgAwIBAgIQeKpq…
"MIIwlQYJKoZIhvcNAQcCoIIwhjCCMIICAQExDzANBglghkgBZQMEAgEFADALBgkqhkiG9w0BBwGgggwO
MIIF/TCCA+WgAwIBAgIQeKpq…
], "validationInfo": { "ocsp": [
"MIIJ0woBAKCCCcwwggnIBgkrBgEFBQcwAQEEggm5MIIJtTCBnqIWBBR3PdEPHRznyHUVfuMc3c0FpEAp
MRgPMjAyMTA2MjUyMTQ0MTF…
], "crl": [
"MIIC4zCBzAIBATANBgkqhkiG9w0BAQsFADBpMQswCQYDVQQGEwJjaDERMA8GA1UEChMIU3dpc3Njb20xJTAjBgNVBAsTHERpZ2l0YWw…
]
}
```

5.2.8 Supported Errors

In this section, we list the errors and their descriptions as supported by the ETSI interface.

Status Code	Description
200 OK	Response to a successful API method request.
204 No Content	Response to a successful API method request in case no content is returned.
302 Found	Response used to redirect the user to an OAuth 2.0 authorization endpoint.
400 Bad Request	Returned due to unsupported, invalid, or missing required parameters.
401 Unauthorized	Returned when a bad or expired authorization token is used
429 Too Many Requests	Returned when a request is rejected due to rate limiting
500 Internal Server Error	Returned when the server encounters an unexpected condition.
501 Not Implemented	Returned when an unimplemented method is requested.
503 Service Unavailable	Returned when the server is currently unable to handle the request due to temporary overloading or maintenance conditions.

Note that the status codes 429 and 50x are applicable to the remote service overall and are not specific to any API methods. For this reason, they are not mentioned in the error tables for each method specifically.

Further, about status und error codes see in the ETSI-Specification [ETSI TS 119 432] in the section 7.24.2. Further, this section is referencing section 10 of the [CSC] standard.

Error	Error Description	Status Code Mapping
invalid_request	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed.	400
unauthorized_client	The client is not authorized to use this method.	401
access_denied	The user, authorization server or remote service denied the request.	401
unsupported_response_type	The authorization server does not support obtaining an authorization code using this method.	400
invalid_scope	The requested scope is invalid, unknown, or malformed.	400

server_error	The authorization server encountered an unexpected condition that prevented it from fulfilling the request	500
temporarily_unavailable	The authorization server is currently unable to handle the request due to a temporary overloading or maintenance of the server	503
expired_token	The access or refresh token is expired or has been revoked.	401
invalid_token	The token provided is not a valid OAuth access or refresh token.	401

Predefined common errors messages and mapping on the status codes.

5.2.9 Postman Samples

In this section we describe the Postman samples which can be used in the context of RAX. We will focus on only the Signing Service ETSI interface. Please note that in the GitHub repo we also provide the old versions of these samples.

These samples are in our GitHub Web page available in the RAX folder. There are also sample videos describing how to use them.

- https://github.com/SwisscomTrustServices/Signing Service-Postman-Samples

We will list each of the Postman samples and describe them one by one and reference them such that the flows within signing service can be more easily understood.

5.2.9.1 Description

The GitHub repo documents in 2 sections the details related to the Signing Service ETSI based signing.

Signing Service-Postman-Samples/README.md at main · SwisscomTrustServices/Signing Service-Postman-Samples · GitHub

The section **ETSI Postman Samples Description** presents all the Postman samples which can be used together with the ETSI interface.

- https://auth.trustservices.swisscom.com/ **Broker Authentication Flow**: To generate an auth_code for the /token endpoint, you will need to identify via broker using an appropriate IdP depending on your evidence (MID, PF or PWDOTP). After successful authentication you will receive the code in the URL.

- https://auth.trustservices.swisscom.com/api/auth/realms/broker/protocol/openid-connect/token **Broker Token Generation**: This request is used to generate a JWT token for document signing for the upcoming requests by using the code from the previous step and client credentials provided by support after successful trial account onboarding.

- https://ais.swisscom.com/Signing Service-Server/etsi/standard/rdsc/v1/signatures/signDoc **ETSI Signing (OnDemand) eIDAS:** This call will sign a document hash using ETSI (OnDemand), eIDAS, and the JWT token generated in the previous step, prompting Signing Service to respond with the signatureObject after a successful signing.

- https://ais.swisscom.com/Signing Service-Server/etsi/standard/rdsc/v1/signatures/signDoc **ETSI Signing (OnDemand) ZertES**: This call will sign a document hash using ETSI (OnDemand), ZertES, and the JWT token generated in the previous step, prompting Signing Service to respond with the signatureObject after a successful signing.

- https://ais.swisscom.com/Signing Service-Server/etsi/standard/rdsc/v1/signatures/signDoc **ETSI Signing (static) eIDAS:** This call will sign a document hash using ETSI (static seal), eIDAS, and the JWT token generated in the previous step, prompting Signing Service to respond with the signatureObject after a successful signing.

  https://ais.swisscom.com/Signing Service-Server/etsi/standard/rdsc/v1/signatures/signDoc **ETSI Signing (static) ZertES:** This call will sign a document hash using ETSI (static seal), ZertES and the JWT token generated in the previous step, prompting Signing Service to respond with the signatureObject after a successful signing.

The section **ETSI Postman Sample Videos** describes based on video recording on how to configure and use the Postman sample.

Signing based on the ETSI interface and ZertES. https://youtu.be/Pbl1kJmAnNI

5.2.9.2 Token Request

TYPE: x-www-form-urlencoded

POST: https://<host>:<port>/auth/realms/broker/protocol/openid-connect/token

Parameter name	Description
grant_type	e.g., authorization_code
code	This is the code obtained in authentication step
client_id	This is the secret client_id
client_secret	This is the secret associated with the client_id

5.2.9.3 Signing

TYPE: application/json

POST: **Error! Hyperlink reference not valid.**

Sign Request Payload

```
POST /Signing Service-Server/etsi/standard/rdsc/v1/signatures/signDoc HTTP/1.1

Host: ais.intarsys.de

Content-Type: application/json

Content-Length: 1863

{

   "SAD": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJwSnZFWVFSM1lxTDlqdU4wMzBjV2lK...

YRolwkkin936BnJdiZVjYKf6dhZ2LDIH7xKah8I9JWFC51mWtJlcMT1gzctlRYNjWqEjxa8cDCSqSsRg7...

qWyRTTe0PHCkIJVEviQ377LcqfHySAjYEq-hZNFNa_5tNINKo5edpUuR_fEvDUncKcqq9rsPHC-
JUizgUmREkupC_fV065fk5ExurWjUatZNYD-fyDcxmEt_g9XHeBidWz9jfkaV6qMJk18w",

   "requestID": "f1ef942a-01ed-4515-83fc-e136d774393b",
   "credentialID": "OnDemand-Qualified",
   "profile": "http://uri.etsi.org/19432/v1.1.1#/creationprofile#",
   "signatureFormat": "P",
   "documentDigests": {
      "hashAlgorithmOID": "2.16.840.1.101.3.4.2.1",
      "hashes": [
         "HLNTuE2+zWOo+p1VfQdjdEjDC9xcLfVdqdHYX2gwTFM=",
         "sHS3ei9wNyR/rGu5ghto/v0+h22wmdlD3TGxRyO/sgM="
      ]
   }
}
```

Sign Request Response

```
{
"policy": null,
"signaturePolicyLocations": null,
"requestID": "f1ef942a-01ed-4515-83fc-e136d774393b",
"DocumentWithSignature": null,
"SignatureObject":
["MII2swYJKoZIhvcNAQcCoII2pDCCNqACAQExDzANBglghkgBZQMEAgEFADALBgkqhkiG9w0BBwGgghKj
MIIFujCCA6KgAwIBAgIQW+f...
"MII2tQYJKoZIhvcNAQcCoII2pjCCNqICAQExDzANBglghkgBZQMEAgEFADALBgkqhkiG9w0BBwGgghKjMII
FujCCA6KgAwIBAgIQW+...
],
"signaturePolicyID": null,
"revocationInfo": { "ocsp": [
"MIIJggoBAKCCCXswggl3BgkrBgEFBQcwAQEEgglOMIIJZDCBnqIWBBRhv7ed9UZvCuYanha+P8zJKWU9Lx
gPMjAyMTA2MjUwOTU3Mj...
],
"crl": [
"MIIC4zCBzAIBATANBgkqhkiG9w0BAQsFADBpMQswCQYDVQQGEwJjaDERMA8GA1UEChMIU3dpc3Njb
20xJTAjBgNVBAsTHERpZ2l0YWw...
]
}
```

5.3 User Authentication, Identification and Registration

The user can be authenticated based on existing IDPs or he can provide his own IDP which has to be audited. Further the user can be identified using existing methods such as SRS or by providing his own method, which as for IDPs, must be audited accordingly.

We distinguish between two main cases.

- The IDP wants to identify and register the users and imports the data into the Swisscom RA-DB.
- The IDP keeps the identification and registration data (some privacy concerned organizations prefer this approach) and will audit their archival process as delegated process.

In case you want not rely on the standard methods please find more details in the "RA Evidence Import Guide" and additional information shared during an onboarding workshop as IDP.

5.4 Standard Authentication, Identification and Registration

- The user can use the standard authentication and identification methods offered by Swisscom and integrate these into his signing flow.

5.4.1 Standard IDP based Authentication and Integration

- The user can use and integrate existing IDPs, such as PostFinance and MySwisscom App, etc.

5.4.2 Standard Identification and Registration Integration

- The user can use existing identification and registration methods such as SRS, etc.

5.5 Custom Authentication, Identification and Registration

- The user can use his own authentication and identification methods. In case the user decides to use his own IDP then this must undergo a separate onboarding process which is based on auditing his implementation. Custom IDP based Authentication Integration
- The user can provide its own IDP. In this case we differentiate between evidence ID import and without IDP evidence ID import into the RA-DB.

5.5.1.1 Without IDP Evidence Import

- The authentication and signing flow support third party IDPs without import of the evidence in the RA-DB. In this case there is no need to import the user digital evidence into the RA-DB but rather a link to the IDP service.

5.5.1.2 With IDP Evidence Import

- The authentication and signing flow support third party IDPs with import of the evidence into the RA-DB. In this case the evidence ID must be imported into the RA-DB.

5.6 Validation with On Demand Certificates

The following chapter describes topics to be regarded in case a successful validation of the signature should be guaranteed later on.

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 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 have a trusted timestamp information in the signature, or the signing time set within the 10 Minutes validity period of the On Demand certificate.

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.

5.6.1 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 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) (set by Multiple Authentication Broker)
- Use and length of the Step-Up message. (historically used in old interface)

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.

Please see further details about the signature size in here:
Swisscom CA 4 · SwisscomTrustServices/Signing Service Wiki (github.com)

Processing OCSP and CRL Response Elements in the REST API

When using the 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 Sign Response

```
{ "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;
```

5.7 Processing PDFs for PAdES LTV Support

The PDF documents signed with the 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 Signing Service, you must use the ***conformanceLevel*** parameter which controls if revocation information is added or not to the resulting signature. See more details about the ***conformanceLevel*** parameter in Section 5.2.4.1 and Section 5.2.4.1.

Here is an example of a Signing Request to trigger a PAdES B-LT signature using the ETSI interface:

JSON Sign Request

```
POST /Signing Service-Server/etsi/standard/rdsc/v1/signatures/signDoc HTTP/1.1
Host: ais.intarsys.de
Content-Type: application/json
Content-Length: 1863
{
  "SAD": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJwSnZFWVFSM1lxTDlqdU4wMzBjV2lK...
YRolwkkin936BnJdiZVjYKf6dhZ2LDIH7xKah8I9JWFC51mWtJlcMT1gzctlRYNjWqEjxa8cDCSqSsRg7...
qWyRTTe0PHCkIJVEviQ377LcqfHySAjYEq-hZNFNa_5tNINKo5edpUuR_fEvDUncKcqq9rsPHC-
JUizgUmREkupC_fV065fk5ExurWjUatZNYD-fyDcxmEt_g9XHeBidWz9jfkaV6qMJk18w",
  "requestID": "f1ef942a-01ed-4515-83fc-e136d774393b",
  "credentialID": "OnDemand-Qualified",
  "profile": "http://uri.etsi.org/19432/v1.1.1#/creationprofile#",
  "signatureFormat": "P",
  "conformanceLevel": "AdES-B-LT",
  "documentDigests": {
    "hashAlgorithmOID": "2.16.840.1.101.3.4.2.1",
    "hashes": [
      "HLNTuE2+zWOo+p1VfQdjdEjDC9xcLfVdqdHYX2gwTFM=",
      "sHS3ei9wNyR/rGu5ghto/v0+h22wmdlD3TGxRyO/sgM="
    ]
  }
}
```

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

JSON Sign Response - Static/On Demand Certificate

```
{
 "validationInfo": {
  "ocsp": [
   "MII...AGk="
  ],
  "crl": [
   "MII...sNrI="
  ]
 },
 "responseID": "fdf41e6a-382a-4512-afe9-fd2a9bab30d7",
 "SignatureObject": [
  "MII2...23w4=",
  "MII2...m5c="
 ]
}
```

*The revocation information is part of the “validationInfo” result property.

Please note the ***validationInfo*** 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.

The main considerations are as follows:

- 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.

6 Signing Service Source Code Clients

6.1 Adobe PDF Signing – Java/.Net Clients

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

Repositories of interest:

- Signing Service
  - **shell** - A set of shell scripts that use the Signing Service 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 Signing Service from PHP and signing PDFs that way.
- PDFBox Signing Service
  - 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 Signing Service
  - Signing Service client written in Java and using the iText library for processing PDFs. Similar to PDFBox Signing Service, 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 Signing Service
  - Docker setup that will provide a container to automatically sign PDF documents located in each directory, using the Signing Service.
- iText Dotnet Signing Service
  - A .NET Standard client library and a CLI wrapper for using the Swisscom Signing Service (Aigning Service)Swisscom Signing Service (Aigning Service) to sign and/or timestamp PDF documents. The library (Aigning Service 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.
- Signing Service React Flask
  - This client is based on JavaScript, React and uses Swisscom Signing Service (Aigning Service) to sign and/or timestamp PDF documents. *This client is based on JavaScript, React and uses Swisscom Signing Service (Aigning Service) to sign and/or timestamp PDF documents*. The client has the same functionalities for PDF files processing as our iText7 client.

6.2 Configure the iText/PDFBox Clients for the ETSI Interface

The following samples can be run with the test instance. For testing, use the client TLS certificate which you get after you request a 90-day trial account. Sample Signing Service REST Endpoints.

# The Signing Service server REST URL for sending the Signature requests

server.rest.signUrl = https://<host>:<port>/<Signing Service-Server context>/etsi/standard/rdsc/v1/signatures/signDoc

# The Signing Service server REST URL for sending the Signature status poll requests (Pending requests)

- For the ETSI interface there is no pending endpoint to be configured.

6.3 PDFBox Java Client

While you can use the 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 Signing Service 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/Signing Service-Server/rs/v1.0/sign");
restConfig.setRestServicePendingUrl("https://ais.swisscom.com/Signing Service-
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 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!
}
```

6.4 iText Java Client

To use the Signing Service 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 Signing Service configuration documentation.

6.5 iText .Net Client

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

To start using the Swisscom Signing Service and this client library, you will need the following:

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

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

6.6 Signing Service React Flask

To start using the Swisscom Signing Service and this client library, do the following:

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

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

7 Appendix

7.1 Create Self-signed Certificate with OpenSSL

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

7.2 Generate Key and CSR

```
$  openssl req -new -newkey rsa:4096 -nodes -rand /dev/urandom -keyout mycert.key -out mycert.csr -
sha256 -subj "/CN=ais.company.ch/C=CH"
```

7.3 Organization or Organizational Unit

This information can be also optionally provided.

```
$  openssl req -new -newkey rsa:4096 -nodes -rand /dev/urandom -keyout mycert.key -out mycert.csr -
sha256 -subj '/CN=ais.company.ch/O=Company/OU=OrganizationalUnit/C=CH'
```

7.4 Self-sign it and Create your Certificate

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

7.5 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.

7.6 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 3 years.

7.7 Generate KeyStore & Export the Self-signed Certificate

```
$ keytool -genkey -alias <alias-name> -keyalg RSA -keysize 4096  -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
```

7.8 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
```

7.9 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.

7.10 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 Signing Service can be found here https://github.com/SwisscomTrustServices/Signing Service/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

7.11 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.

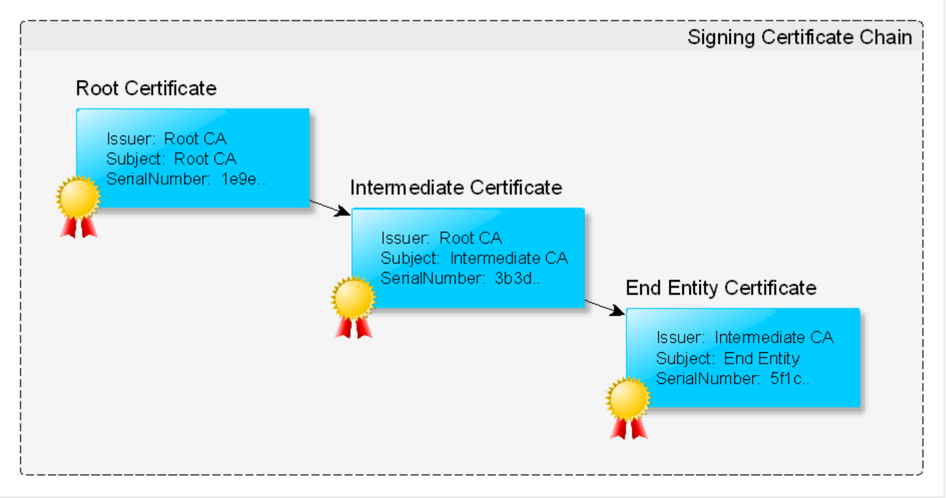


Figure 9. Root certificate hierarchy.

In the sub-chapter below, you will find an overview about the certificate hierarchy related to the Swisscom Signing Service and its available revocation information.

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

Name	Fingerprint Algorithm	Fingerprint
Swisscom Root CA 4	SHA1	545A671F6B34F99E5940A25F417A3EC108C61513

7.12 CMS Signature

Certificate Subject	Issuer	Available Returned Revocation Info RI

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

EE Cert	CN = <Username>	ICA: CN = Swisscom Saphir CA 4	OCSP	OCSP-Response
EE Cert	CN = <username>	ICA: CN = Swisscom Saphir EU CA 4.1		OCSP-Response
EE Cert	CN = <username>	ICA: CN = Swisscom Diamant CA 4		OCSP-Response
EE Cert	CN = <username>	ICA: CN = Swisscom Diamant EU CA 4.1		OCSP-Response
ICA Cert	CN = Swisscom Saphir CA 4	RCA: CN = Swisscom Root CA 4	CRL	http://aia.swissdigicert.ch/sdcs-root4.crt
ICA Cert	CN = Swisscom Saphir CA 4	CN = Swisscom Saphir EU CA 4.1		http://aia.swissdigicert.ch/sdcs-root4.crt
ICA Cert	CN = Swisscom Diamant CA 4	RCA: CN = Swisscom Root CA 4	CRL	http://aia.swissdigicert.ch/sdcs-root4.crt
ICA Cert	CN = Swisscom Diamant EU CA 4.1	RCA: CN = Swisscom Root CA 4		http://aia.swissdigicert.ch/sdcs-root4.crt
RCA Cert	CN = Swisscom Root CA 4	RCA: CN = Swisscom Root CA 4	-	-

7.13 Timestamp Signature

Certificate	Subject	Issuer	Available RI	Returned Revocation Info
TSA Cert	CN = Swisscom TSU 4.1	ICA: CN = Swisscom TSS CA 4.1	OCSP	OCSP-Response
ICA Cert	CN = Swisscom Saphir EU CA 4.1	RCA: CN = Swisscom Root CA 4	CRL	http://aia.swissdigicert.ch/sdcs-root4.crt
RCA Cert	CN = Swisscom Root CA 4	RCA: CN = Swisscom Root CA 4	-	-