Setting Up Your Digital Signing Solution With Entrust Remote Signing Engine

ENTRUST

SECURING A WORLD IN MOTION

Table of Contents

INTRODUCTION

Electronic signatures have become increasingly strategic as a key element of digitalization. They help save money, reduce the need for physical meetings, and reduce paper usage in daily transactions carried out between citizens, consumers, employees, companies, and administrations.

The following regulations have already established a common legal framework for the use of electronic signatures that made them legally equivalent to handwritten signatures:

- European directive 1999/93/EC on electronic signatures
- ESIGN Act approved in the U.S. in 2000
- International electronic commerce law UNCITRAL of 2005

1.1 New frameworks

The European regulation 910/2014 Electronic Identification and Trust Services for Electronic Transactions in the Internal Market (popularly known as eIDAS Regulation) helped define robust security and trust mechanisms for electronic signatures. Thanks to this framework, there's been a significant increase in the adoption of electronic signatures by both companies and individuals in Europe. This was much needed considering today's increasingly growing technological use of cloud services and mobile technology. The eIDAS regulation also guarantees interoperability between the Member States of the EU.

The eIDAS Regulation introduced remote signing as a use case. Remote signing centralizes keys and signing certificates on a server, while guaranteeing that the signer is the one who retains sole control of the signing key.

Some of the benefits of remote signing are:

- Ease of use
- Centralized control and auditing
- Reduced costs at different levels, especially in deployment and integration

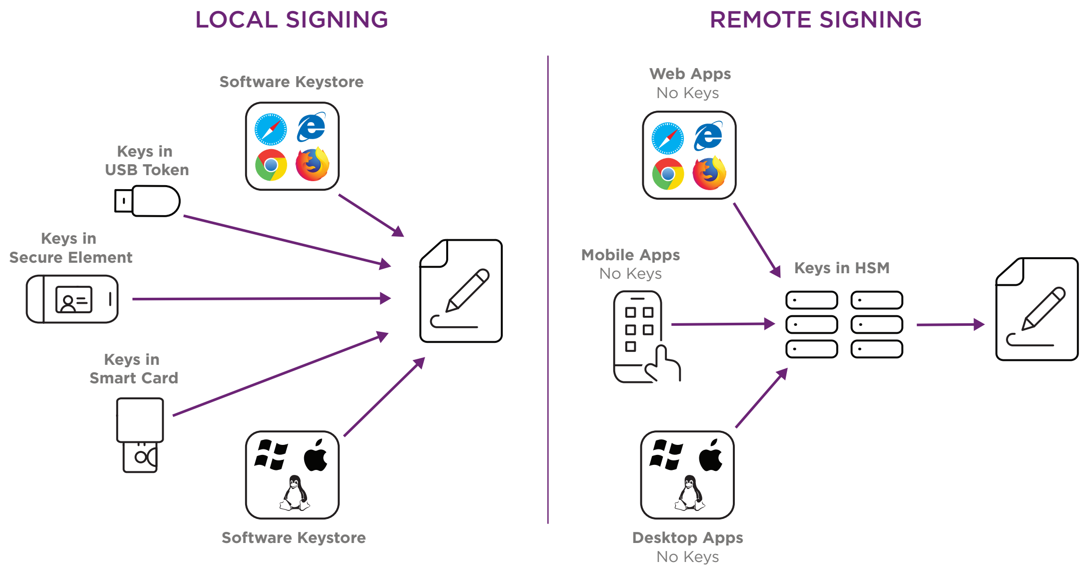


Figure 1. Local signing vs. remote signing

1.2 New standards

Although the scope of each regulation is specific to the jurisdiction (or grouping of jurisdictions) that defined it, it’s common for them to rely on a set of widely accepted technical standards. In this sense, it’s worth commending the standardization work carried out in recent years by organizations such as ETSI and CEN. Entrust is a member of both and has collaborated in their respective working groups.

ETSI is focused primarily on defining the requirements related to the operation of trust services, while CEN is mainly focused on the security requirements of the software and hardware products involved in the deployment of signing services. At the same time, thanks to the contributions made by the Cloud Signature Consortium, it has also been possible to cover the standardization of signature APIs. Entrust is a founding member of this consortium, which currently has more than 80 members.

You can find the list of technical regulations in Appendix A.

1.3 New signature types and formats

Regulations and technical standards have established all the legal and technical elements necessary to deploy electronic signatures as well as conditions of interoperability.

There are two types of electronic signatures:

- **Electronic signature:** made by individuals or natural persons
- **Electronic seal:** corresponding to the signature made by companies or legal persons

Interoperable formats are standardized:

- PAdES for signing PDF documents
- XAdES for signing XML documents
- CAdES for signing binary files

And the entities and elements involved in the signing process are specifically detailed, such as the role of the trust service provider (TSP) and the qualified signature creation device (QSCD).

Entrust electronic signature solutions are based on PKI technology and digital certificates. This technology is widely adopted by the industry and has numerous applications in the field of security. Plus, PKI technology is the basis for some authentication mechanisms used to gain access to the remote signature service, characterized by the high level of trust they offer and their contribution to fraud mitigation.

A signing solution can implement or accept different authentication methods depending on:

- The adequacy of the level of trust
- The ease of use (or user experience) that is sought
- The type of business process involved

Entrust Remote Signing Engine

The Entrust Remote Signing Engine is an on-premises software that enables the implementation of a remote signing service in accordance with the CEN EN 419 241 technical standards, specifically:

- CEN EN 419 241-1: Trustworthy Systems Supporting Server Signing Part 1: General Requirements
- CEN EN 419 241-2: Trustworthy Systems Supporting Server Signing Part 2: Protection Profile for QSCD for Server Signing
- CEN TS 419 221-5: Protection Profiles for TSP Cryptographic Modules

NOTE: The functions described in this document correspond to the Entrust Remote Signing Engine version 4.2 or higher.

The remote signing service can be operated by a TSP to service applications from multiple organizations. These applications are the service providers shown in Figure 2, since they are ultimately the ones that end up providing the services to the end-users.

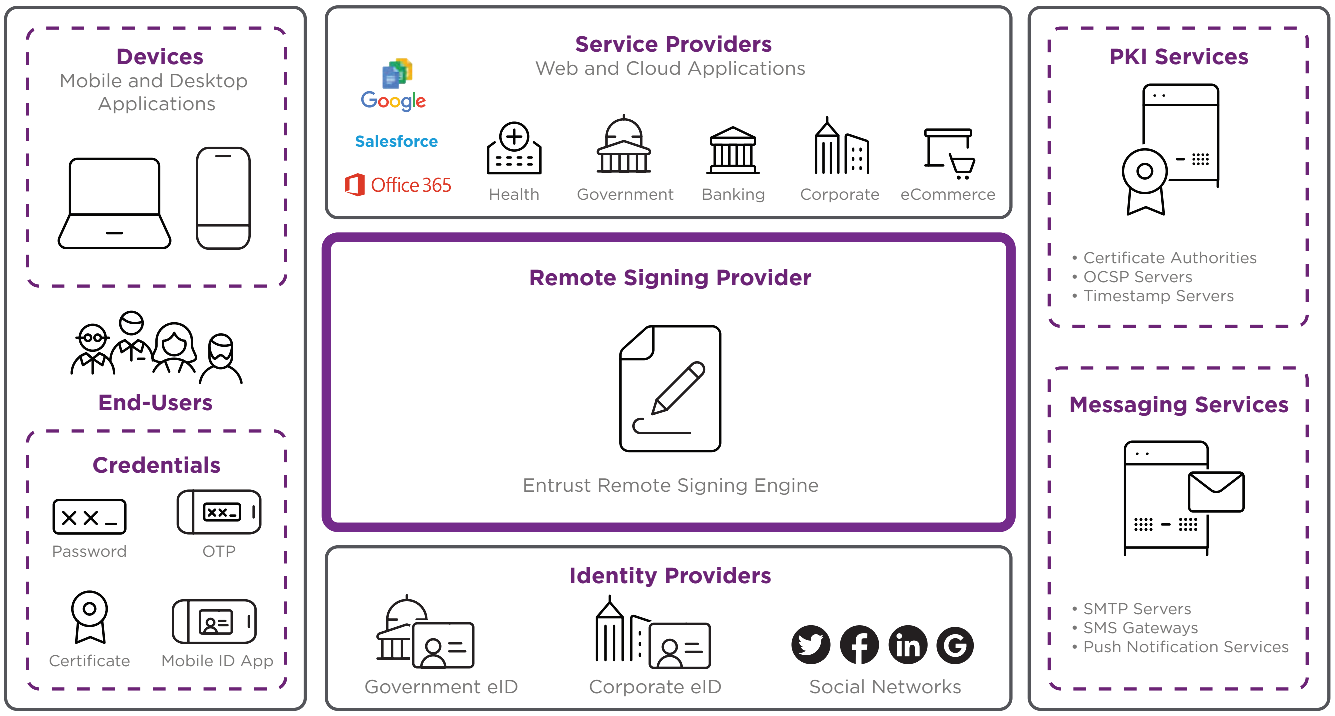


Figure 2. Relationship of the Entrust Signing Automation Engine eIDAS Platform with other systems

2.1 Use cases and compatibility

End-users consume the services offered by the service providers. In the field of public administration, end-users are citizens or public employees, and service providers are the applications of different ministries or municipalities. In the private enterprise environment, end-users are clients or employees of the company, and service providers are the corporate applications that they use.

Today it's common for end-users to access services from a variety of different devices, and in a variety of different ways:

- Mobile applications (e.g. devices using iOS or Android operating systems)
- Desktop applications (e.g. computers using Windows, Mac, or Linux operating systems)
- Directly from a browser without the need to install any software

Entrust Remote Signing Engine easily integrates the document signing processes with all three of these applications.

2.2 Integration requirements and options

2.2.1 Service providers

An important part of the remote signing process is the identity provider in charge of verifying the identity of the end-users through face-to-face or remote procedures. After their identity is verified, end-users are given a token and/or credentials they must use every time they need to authenticate themselves to carry out an electronic transaction, such as creating a remote signature.

An example of face-to-face processing is a citizen going to an office and presenting a physical document, such as a passport or a driver's license. The office employee verifies that the document is valid and visually compares it to the face of the user who is in front of them. After they verify the identity, they register the user's personal data in the IT system, including their mobile phone number. At the end of this process, the user typically receives a username and password as the first factor of authentication, and an SMS OTP (one-time password) as the second.

Entrust Remote Signing Engine can be integrated with any identity provider, generally using the SAML 2.0 and OpenID Connect standards, but custom connectors are also an option. For organizations that need an identity provider, Entrust offers Entrust Identity Enterprise (on-premises) or Entrust Identity as a Service (cloud).

2.2.2 PKI services

PKI services are required to issue signing certificates to end-users and to provide the extra elements required to create long-term signatures. They generally include a CA (certification authority), an OCSP server, and a TSA (timestamp authority).

Entrust Remote Signing Engine can be integrated with any PKI service, either by using one of the connectors available for third-party solutions, or the Entrust CA Gateway connector or through a custom connector. For organizations needing PKI services, Entrust offers Managed PKI solutions, Entrust Validation Authority, and Entrust Timestamping Authority.

Finally, messaging services may be necessary if the Entrust Remote Signing Engine manages any additional factor in user authentication beyond the factors that identity providers manage, or if users use our optional mobile app (Entrust Remote Signing Engine Mobile ID) for signature activation.

2.3 Benefits

Entrust Remote Signing Engine offers the following benefits:

- **Streamlined application development and mobility:** Helps develop use cases with widely used integration mechanisms, offering a complete set of REST APIs for developers. It also can be used for other use cases, such as cloud services or applications on mobile devices.
- **Improved user experience for end-users/citizens**: User-friendly features such as mobile applications and web applications don’t present barriers and don’t require software installation on the user’s computer.
- **Identity fraud protection:** Recognizes the level of assurance (LoA) of the identities verified by each identity provider, as recommended by the European regulation eIDAS, the OMB/NIST 800-63 recommendation in the U.S., or the ITU-T X.1254 and ISO standards/IEC 29115. The level can be stepped up by requesting the user to provide additional authentication factors when necessary, with the possibility to trigger the step-up for each signing operation.
- **Stronger branding:** Customizable user elements such as web interfaces and a mobile app for signature activation help support brand development.
- **Compliance with the most recent regulations:** Serves as a technological framework for deploying applications and services based on signature regulations, which, in some countries, have been recently reviewed and updated.

Operation

3.1 Grouping

Entrust Remote Signing Engine deployment can serve a large number of service providers, and these, in turn, can serve a large number of end-users. For example, in a nationwide deployment for citizens, the number of potential users may be millions. So a system must be able to support millions of users and the resulting workload, meaning the total number of remote signature transactions.

However, a single configuration may not meet the needs of all users. Perhaps there are differences between service providers depending on how user authentication is performed, or which CA issues which certificate. Entrust Remote Signing Engine can be configured to create different groups of service providers, grouping together those that require the same configuration.

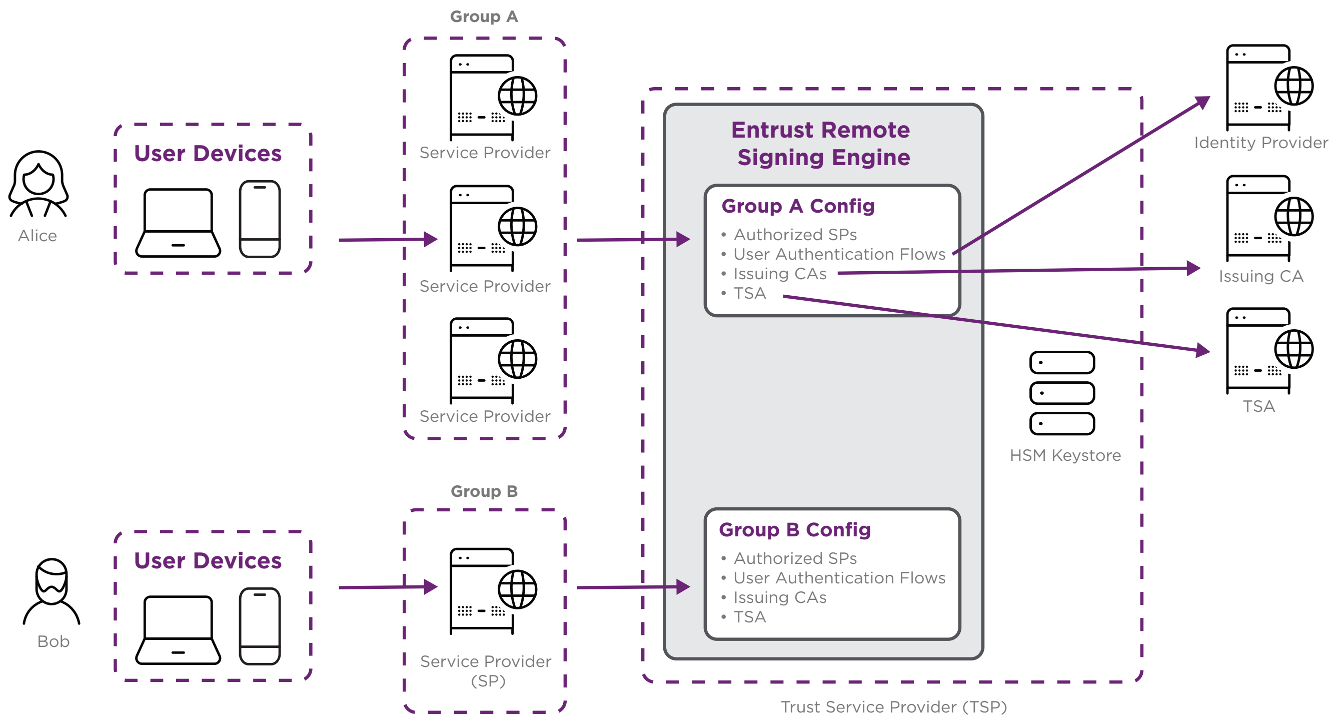


Figure 3. Different configurations for different groups of service providers

NOTE: Although service providers are grouped together, their consumption of services is still accounted for separately, identified by their unique API key.

3.2 Authorization process

Entrust Remote Signing Engine APIs follow the OAuth 2.0 standard, ensuring that:

1) The service provider is authorized to invoke the requested operation

2) If the operation requires the user's consent, an interface is displayed so the user can confirm if they'd like to proceed

As per signature regulations, this confirmation process must be triggered during two scenarios:

1) **During the creation of the signing keys:** Users must be informed that new keys (as well as the corresponding digital certificate) will be created for them and the documents that describe the policies and liabilities of the TSP must be provided so they can review and accept them.

2) **Before the creation of a document signature:** Users must be informed about the document they are about to sign, so they can review it and confirm that they agree with the content.

3.3 User authentication

In order for the user to give their consent, Entrust Remote Signing Engine needs to check that they have previously authenticated themselves with two factors. This may have happened much earlier, for example, if the user has already logged in to a web portal, since many systems are now integrated with the single sign-on feature. The only way to find out is to redirect to the corresponding identity provider.

If there is an active session and the user has already authenticated with two factors, nothing is requested from the user and the operation can continue directly.

If there is no active session, the identity provider displays a user interface requiring the user to authenticate before they can continue.

If there is an active session but the user has only authenticated with one factor, you can choose between having the identity provider display a user interface asking for a second factor, or having Entrust Remote Signing Engine do it.

These combinations are modeled with user authentication flows, and can be different for each group of service providers, as seen in Figure 3.

If you choose to have Entrust Remote Signing Engine perform the second-factor authentication, you have the following native authenticators at your disposal:

- **SMS/email OTP:** The advantage of these authenticators is the simplicity of implementation, since they only require verifying the user's mobile phone number or email address during the registration process.
- **Signing PIN:** This authenticator is the fastest, but the user experience is only good if it's a person who signs frequently; otherwise they are likely to forget the PIN and spend more time restoring it.
- **Entrust Remote Signing Engine Mobile ID:** This app offers the highest level of security, since cryptographic keys for authentication are created on the user's mobile device during the installation of the app. Every time remote signature keys are required to be activated, users receive a push notification on their mobile device and the app is launched so they can confirm the operation.

3.4 Creation of signing keys

Roughly speaking, we can distinguish two profiles of end-users: those who sign documents frequently, and those who only sign occasionally. Frequent users are usually employees of an organization (public or private), while occasional users are usually customers of an organization. Using the insurance industry as an example, customers may sign insurance policies and renewals only once a year, while employees responsible for reviewing and approving policies will sign several times daily.

3.4.1 For frequent users

For users who sign frequently, it's convenient to have a portal that allows them to carry out certain procedures on their signing keys, such as creating, renewing, or revoking them. Entrust Remote Signing Engine offers a series of APIs to implement these steps. Users can use the portal to view activity history and to change which authenticators they want to use for the first- and second-factor authentication. To implement some of these functions, it's usually required to use additional APIs from the identity provider or the reporting system, if available.

3.4.2 For occasional users

For users who only sign occasionally, it's not necessarily convenient to offer them a user portal. In this case, new signing keys (and the corresponding digital certificate) can be created for each new document to be signed, then destroyed immediately after. A very short validity period can be set so it's not necessary to implement the revocation process for these users.

As described in Figure 3, the configuration of the CAs that issue the certificates may be different for different groups of service providers. The characteristics of the certificate may also be different depending on the end-user. Keys are created in a hardware security module (HSM) so they have the best possible protection.

3.5 Signing of documents

End-users sign documents from a wide variety of applications. In general, each service provider has its own document workflow solution. For organizations that need to deploy a document workflow, they should analyze a third-party solution and integrate it with Entrust Remote Signing Engine.

3.5.1 Authentication and approval

Entrust Remote Signing Engine offers a series of APIs to integrate document workflows with the remote signing process. These APIs were standardized within the framework of the Cloud Signature Consortium (CSC). The current version of Entrust Remote Signing Engine implements V1.0.4.0 of the CSC specs, which is the one supported by most members of the Consortium. Future versions of the CSC converge on the ETSI TS 119 432 standard[1].

Before signing each document (or group of documents if batch signing is performed), a user interface is displayed so the user can review the content of the document(s) and give approval. Then the platform will again verify that the user has an active session and that they have provided two factors of authentication; if not, the platform will proceed with getting two-factor authentication done.

(1) More information can be found here.

3.5.2 Long-term validation

At the end of the signing process, the document contains the signature that has been created with the user's keys and the corresponding digital certificate. The next step is to ensure that this signature can be validated for several years by converting it into an LTV (long-term validation) format. Entrust Remote Signing Engine offers this operation in its API by:

1) Collecting evidence that the signer's certificate is valid at the present time – meaning, it has not expired and has not been revoked. This evidence is usually provided by an OCSP server that is part of the set of services offered by the CA issuing the certificates.
2) Collecting evidence of the validity of any other certificates that intervene in the certification chain up to the root.
3) Asking a TSA to create a timestamp that secures all the information collected.

From this point on, it no longer matters whether the signer's certificate has expired or has been revoked, since there is concrete proof that the signature was valid at the time evidence was collected. It is, however, important to remember that the timestamp also has a limited duration – typically around 10 years, which is much longer than normal signing certificates.

In cases where very long durations are required, it's necessary to define an archiving strategy.

3.5.3 Archiving

Some archiving solutions guarantee by their own means (through a combination of hardware and software protections) that the document has not been tampered with since it was archived. In these scenarios, the validity is guaranteed beyond the duration of the timestamp – for example, beyond the 10 years mentioned earlier.

Other archiving solutions constantly monitor documents and take care of requesting a new timestamp before the previous one expires. In this scenario, archiving solutions can use Entrust Remote Signing Engine API to query the TSA for the new timestamp.

In general, each service provider has its own archiving solution. For organizations seeking to deploy a new archiving system, we strongly recommend they verify the capability of the third-party solution to integrate with Entrust Remote Signing Engine APIs.

Architecture

Entrust Remote Signing Engine architecture follows the architecture defined in the CEN EN 419 241 standard. The elements that comprise it are described below, using the terminology used in the standard itself, as shown in Figure 4.

Entrust Remote Signing Engine includes the following five elements:

- Signature Creation Application
- Server Signing Application (SSA)
- Signature Activation Module (SAM)
- User Interface
- Signer's Interaction Component (SIC)

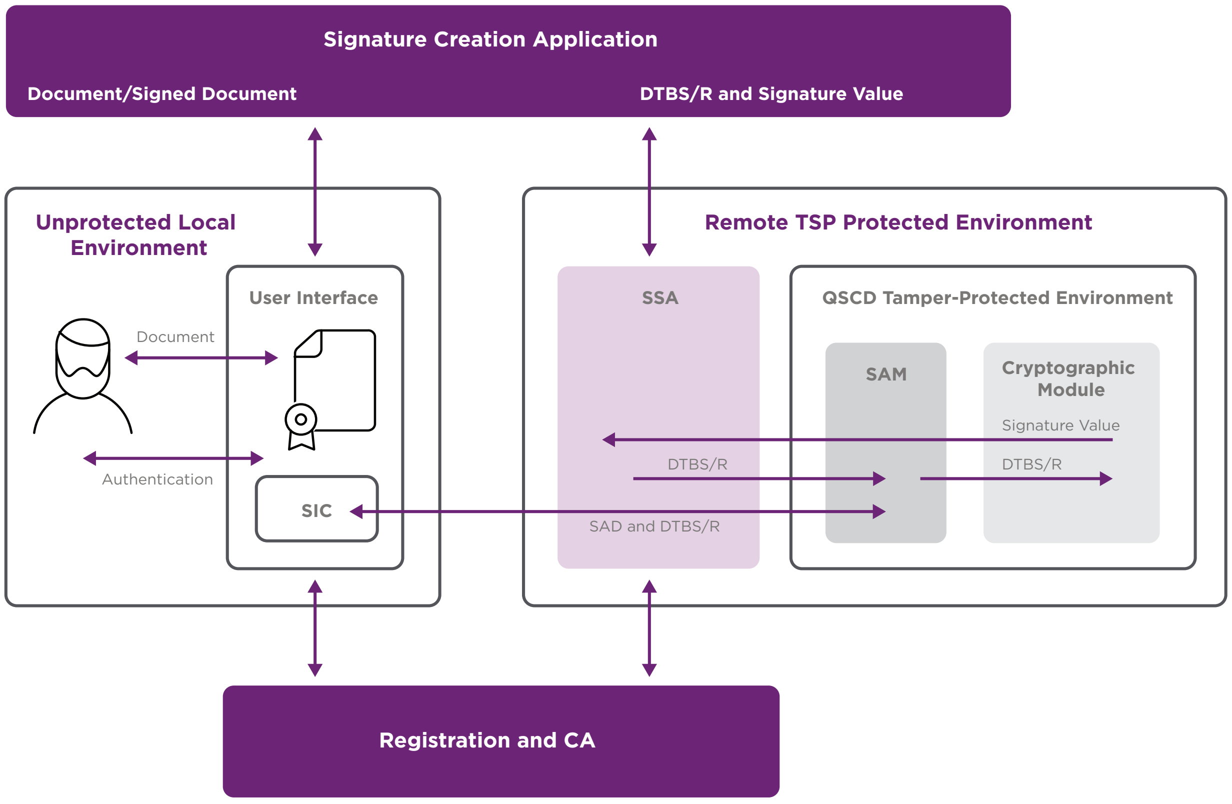

Signature Creation Application
Document/Signed Document
DTBS/R and Signature Value
Unprotected Local Environment
User Interface
Document
Authentication
SIC
Remote TSP Protected Environment
SSA
QSCD Tamper-Protected Environment
SAM
Cryptographic Module
Signature Value
DTBS/R
DTBS/R
SAD and DTBS/R
Registration and CA

Figure 4. Architecture defined in the CEN EN 419 241 standard

Figure 4 has four main components:

- The largest component, on the right, corresponds to elements that fall into the TSP's responsibility.
- To its left are the elements that are executed in the end-user environment; basically a user interface so the user can give their consent whenever required.
- The component at the top represents the service providers' applications. In the case of Entrust Remote Signing Engine, these elements are incorporated into the APIs to facilitate integration.
- The bottom component represents the services offered by the CA issuing the certificates to users.

Note that the architecture shown in Figure 4 does not include the elements of the service provider, such as business logic, document workflow, or the archiving system. The criterion followed in the standard was to specifically represent the elements that intervene at the time of signature creation, omitting any action that takes place as previous or subsequent steps. It is, however, worth mentioning that in the vast majority of cases, a service provider is in charge of providing the document to be signed; in other words, it's very uncommon to have a user providing the document directly. Typically, the document comes from a business logic/application workflow.

4.1 Signature workflow

4.1.1 Hash creation and user approval request

The Signature Creation Application element receives the document to be signed, calculates the document hash (technically named data to be signed/representation, or DTBS/R), and sends it to the server signing application (SSA). It requires user confirmation before continuing to create the signature, via the signature activation module (SAM). The main requirement for this element is that it must run in an environment equipped with tamper detection.

The architecture leaves open the option to have the SAM embedded within an HSM (as represented in Figure 4 under Cryptographic Module), taking advantage of the fact that some HSMs offer the option to run custom code. Although at first glance it may seem like a comfortable option, our laboratory tests determined that it does not provide enough stability to scale well, because:

- A percentage of the HSM's processing capacity is lost, which cannot be devoted to the crypto operations for which it has been mandated.
- It penalizes the performance of transactions by entrusting it with a job that it does not perform optimally.

It's much more efficient to run the SAM on a normal server like those found in any data center – and in this range, they all come equipped with tamper detection.

4.1.2 Signature activation

The SAM communicates with an element that is part of the user interface called Signer's Interaction Component (SIC). The user is asked for confirmation to sign the document, and if they accept it, a specific structure is created called signature activation data (SAD). This structure links the hash of the document with the user's signing key and with the proof that the user has previously provided two factors of authentication. Indirectly, this means that the identity provider is also taking part in the process (indicated by the Authentication arrow in Figure 4). In the end, the SAD represents something that can only be used once to sign. In order to sign another document, a new SAD is needed and the same process is repeated.

4.1.3 Signing operation

Upon user confirmation, the SAM sends the hash of the document to the cryptographic module to create the signature using the user's signing key (technically speaking, this is the standard PKCS#1 format). The signature value travels through the SSA to the signature creation application, and finally, is embedded in the document to form the signed document.

NOTE: Although it's not represented in the architecture, the signed document is returned to the service provider, and is integrated back into the business workflow.

4.2 Network architecture

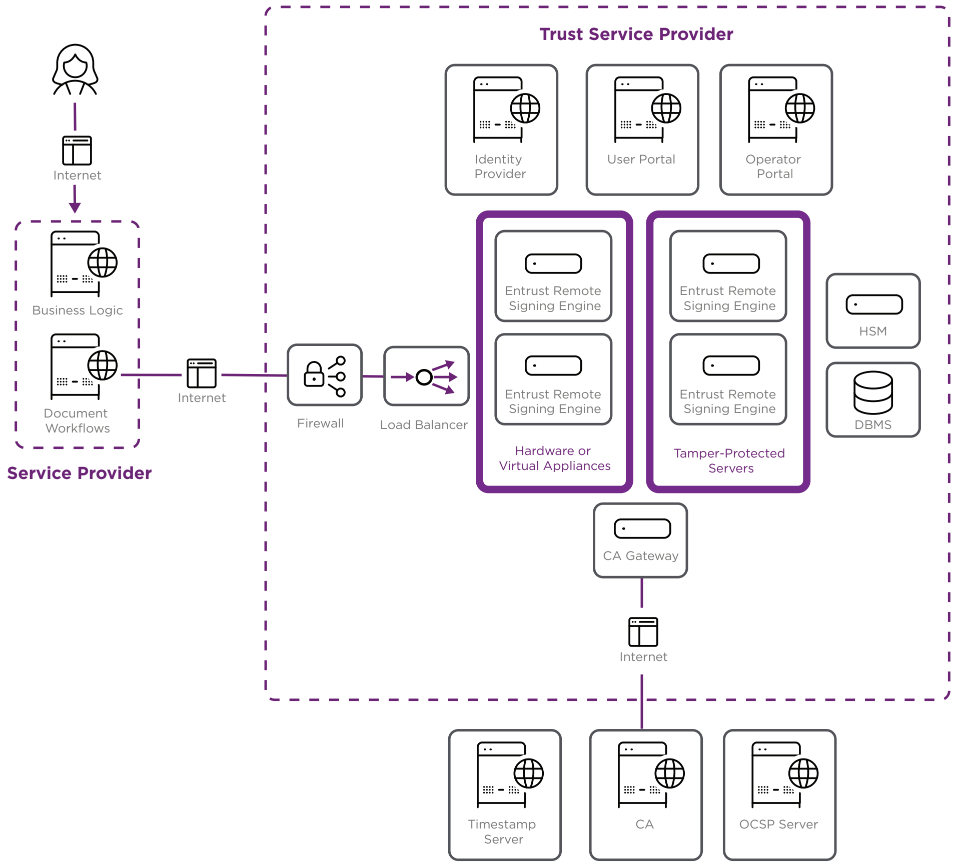


Figure 5. Complete architecture, including the main elements of network infrastructure, once all the systems involved in the service have been integrated

4.2.1 High availability

In Figure 5, Entrust Remote Signing Engine and Entrust SAM elements are represented twice, since this is the minimum number of nodes that are required to guarantee the high availability of the service. It's possible to incorporate more Entrust Remote Signing Engine nodes to achieve greater system scalability, or to increase the number of cores of existing instances, or to do both at the same time. The same applies to Entrust SAM, although since it has less work to do, it's unlikely that more than two nodes will ever be needed.

To simplify the figure, the high availability of the other systems has not been represented. However, it would be reasonable to expect that, similarly, there will be two load balancers, two HSMs, and so on.

4.2.2 Appliance

Regarding the deployment format, Entrust Remote Signing Engine can be deployed as a hardware appliance or a virtual (software) appliance. The appliance format means that the product comes with its own hardened operating system, specifically Linux-based.

- If the hardware appliance is chosen, the platform must be on a dedicated server, but no other software element is required.
- If the virtual appliance is chosen, the platform can run on one of the approved hypervisors and the dedicated server is no longer needed.

The SAM element is not always required, nor must it run on its own dedicated server. In fact, the CEN EN 419 241 standard defines two possible scenarios: one with a SAM, which is considered to offer the greatest possible security; and one without a SAM, which presents a less complex architecture. This detail may be relevant for compliance with signature regulations in some countries. For example, it's expected that the eIDAS regulation will soon require the presence of a SAM for a TSP to offer qualified signatures. Without a SAM, the TSP will only be able to offer advanced signatures.

Due to the requirements of CEN EN 419 241, our SAM runs in a tamper-protected environment. In very large deployments where three or four Remote Signing Engine nodes may be required, we recommend that the SAM pair of nodes run on their own dedicated servers. In this scenario, Entrust Remote Signing Engine nodes can be virtual appliances and run on a hypervisor.

In medium-to-large deployments, you can opt for a more compact integration by deploying the Entrust Remote Signing Engine as a hardware appliance (using a tamper-protected server) and embed the SAM within it.

The User Portal and Operator Portal elements are custom developments, which run on a highly available web server and invoke the Entrust Remote Signing Engine APIs.

Finally, the CA Gateway or the PKI connector allows integrating with a variety of CAs to issue user certificates.

Integration

Entrust Remote Signing Engine comes with a complete set of modern APIs based on best practice models. Operations use JSON/REST formats, and applications must obtain authorization using the OAuth 2.0 standard in order to invoke the operations.

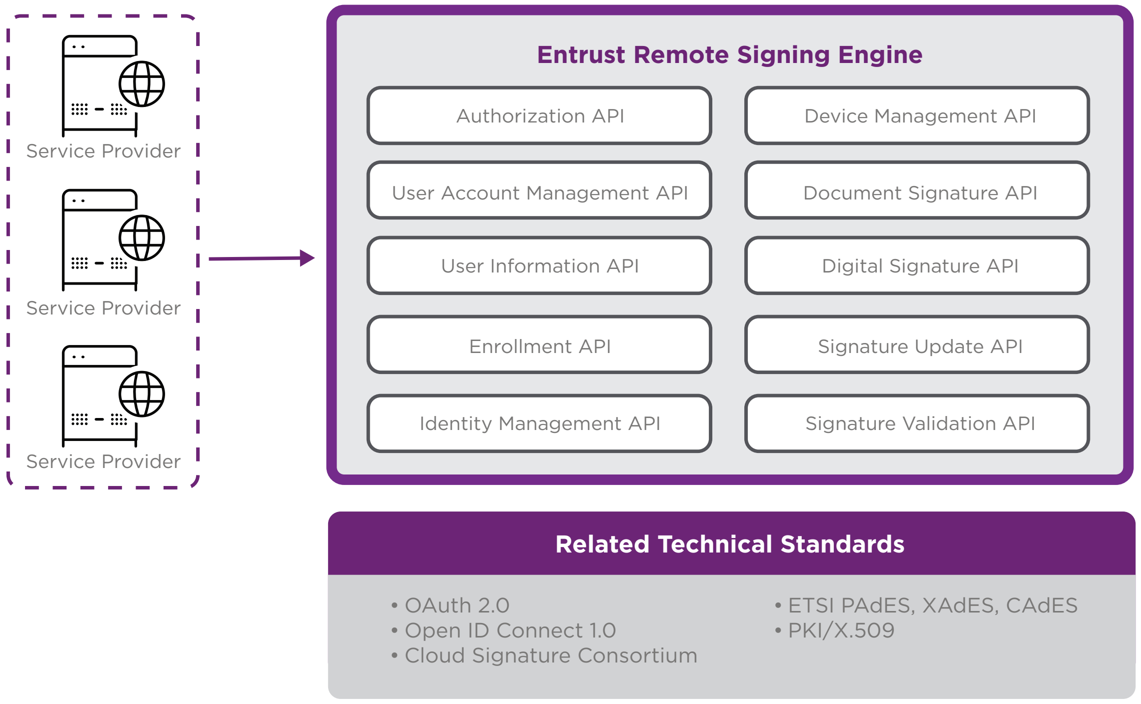


Figure 6. Set of APIs and their relationship with technical standards

5.1 API list

Here are the main APIs for Entrust Remote Signing Engine, and some of the operations that each offers:

- **Authorization API:** Allows applications to obtain the access tokens required to invoke the rest of the APIs. It offers different operations to obtain the token, depending on whether the user's consent is required or not, because certain administrative operations can be carried out without having to ask the user.
- **User Account Management API:** Allows you to create user accounts and link them to other existing account(s) from federated identity providers. If the user does not have an active account in Entrust Remote Signing Engine, they cannot use the remote signature service, even if the identity provider indicates that they have an active session.

- **User Information API:** Retrieves the personal data of a user after they've been authenticated in one of the federated identity providers. Some applications can obtain more data than others, depending on the permissions they've been granted through Entrust Remote Signing Engine administration console, and whether the user consents. It also allows obtaining information on the level of authentication, which depends on how many factors the user has provided and the type of factors.
- **Enrollment API:** Manages the lifecycle of the user's signature certificates, and coordinates the communication with the PKI service(s) previously configured through Entrust Remote Signing Engine administration console. It performs registration, issuance, and revocation functions.
- **Identity Management API:** Allows you to create, list, and delete the signing keys of a user.
- **Device Management API:** Allows you to register, list, and delete a user's mobile device(s). This API is only required when the Entrust Remote Signing Engine is used to activate signatures.
- **Document Signature API:** Allows you to create an advanced or qualified digital signature, in accordance with the PAdES, XAdES, and CAdES standards. The application sends the document (e.g. PDF file) and receives the signed document in response.
- **Digital Signature API:** Allows you to create a digital signature in accordance with the PKCS#1 standard. The application does not need to send the document; only a hash of it. This is useful in cases where an organization's security policies do not allow documents to go outside the organization's domains for privacy and/or confidentiality reasons.
- **Signature Update API:** Converts a signature into an LTV (long-term validation) signature, incorporating evidence of the validity of the signer's certificate and the rest of the chain's certificates up to the root, plus a timestamp that covers all the information collected. This API can be invoked more than once, extending the validity period by adding a new timestamp every time.
- **Signature Validation API:** Validates a signature by verifying that the integrity of the document is intact, that the signer's certificate is not expired or revoked at the present time, and that none of the certificates in the chain are missing to the root. If the signature is rejected, the errors found are listed. In the case of LTV signatures, the verification is based on the analysis of the evidence collected as well as the validity of the timestamp.

5.2 Signing integration into web applications

It's possible for users to sign remotely from a browser, without the need for any additional software installed on their computer. The advantages of this scenario are ease of use and a uniform experience across all the devices that the user handles.

The user accesses a web application that, at some point, will ask them to sign a document. If the web application is used frequently, the user most likely has a login. Otherwise, they will have to authenticate themselves to start the signing process.

The user experience is defined by:

- The service provider (which operates the application)
- The TSP (operated by Entrust Remote Signing Engine)
- Which authenticators are available for authentication and for signature activation
- Whether the user can choose between several signing keys
- Whether the user must indicate some characteristic of the visual aspect of the signature

In most cases, the user only has to confirm that they agree with the content of the document, and the application is in charge of choosing the rest of the options. Also, the application can display the signed document in the same browser, without the user having to download it.

Figure 7 illustrates the interaction that occurs in a remote signing process with this type of integration.

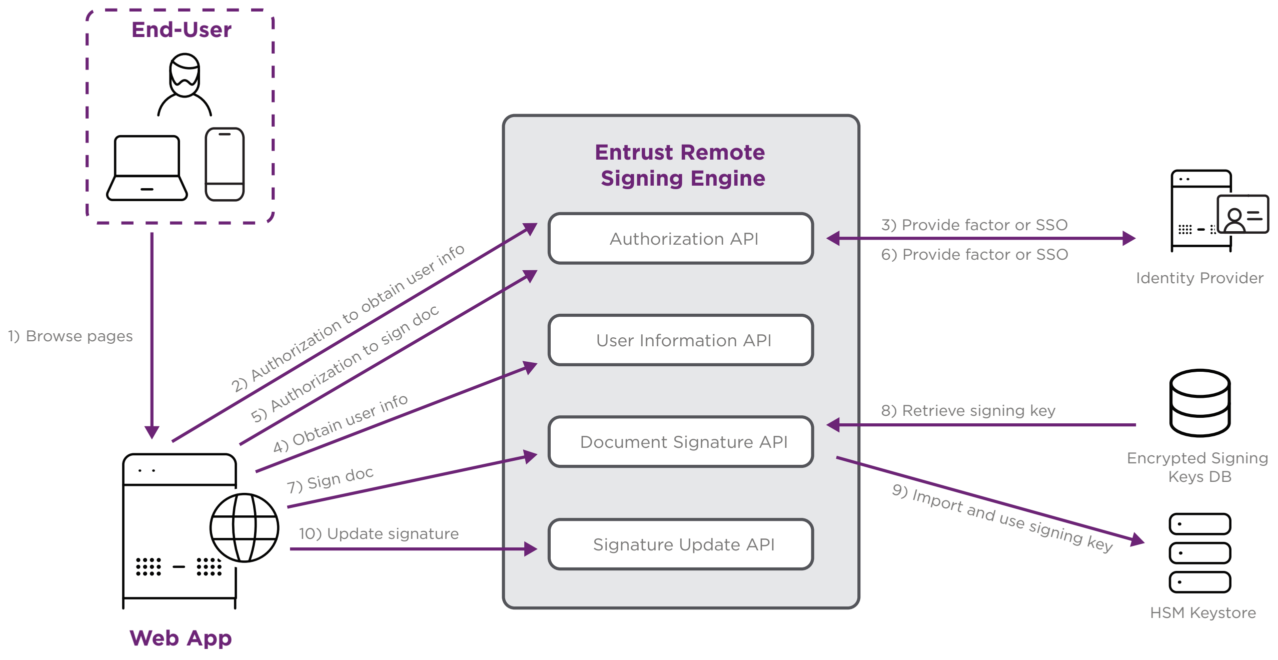


Figure 7. Invocations of APIs from a web application

Details of the integration illustrated in Figure 7 are summarized below, assuming that both the account and the user's signing key have been previously created.

1. The user accesses the web application using their browser. An administrator must first register the application in Entrust Remote Signing Engine so it can invoke the APIs that follow.
2. The application requests authorization to obtain some personal data from the user. Integration is done through the authorization API that uses the OAuth 2.0 and OpenID Connect 1.0 standards. This call triggers several actions, the first being user authentication (see step 3). The user is then asked if they wish to grant access to their personal data. Once the user approves, the application gets an access token allowing it to invoke the User Information API.
3. Entrust Remote Signing Engine redirects the user's browser to the identity provider. The identity provider checks for an active user session, in which case it applies single sign-on (SSO). Otherwise, the user is asked to authenticate by providing one or two factors, as configured (more factors can be provided later).
4. The application invokes the user information API to obtain the personal data for which it had requested authorization.
5. The application asks for authorization to sign a document. Again, this call triggers several actions, the first being user authentication (see step 6). The user is then asked if they want to activate their signing key to sign that specific document. Once the user approves, the application gets an access token allowing it to invoke the document signing API.
6. Entrust Remote Signing Engine redirects the user's browser to the identity provider. Typically, there will be an active user session due to step 3 (unless a long time has passed and the session has expired). In that case, the user will have already provided one or two factors, so they are only asked to provide an additional factor, as configured.
7. The application invokes the document signing API to sign the document for which it had requested authorization. The signature is created with the help of the HSM (see steps 8 and 9) and then the signed document is delivered to the application.
8. Entrust Remote Signing Engine stores large volumes of encrypted user keys in a database, since no HSM would have enough storage capacity. In this step, the user's signing key is retrieved. If there is more than one, the user is asked which one they want to use.
9. The key is imported into the HSM, decrypted, and used to create the signature. The key is then deleted from the HSM to free up resources.
10. The application invokes the signature update API to convert the signature into an LTV signature. As explained above, in this step, Entrust Remote Signing Engine communicates with several OCSP servers to collect validity evidence of the signer's certificate and the rest of the chain's certificates to the root, and communicates with a TSA to request a timestamp be created.

From the user/web application perspective, the complexity of the signing process is masked by simple web redirects, which makes it a smooth experience.

5.3 Signing integration into desktop applications

In other use cases, desktop applications are already designed to access signature functions through the MS-CAPI or PKCS#11 cryptographic interfaces, such as Adobe Acrobat Reader or some Microsoft Office applications. These interfaces isolate the applications from the particularities of the cryptographic provider, regardless of whether it's implemented in software format or hardware format (e.g. a smart card), and regardless of who the application publisher is. This means there's no need to modify the application if the provider changes.

In desktop scenarios, we can provide an MS-CAPI or PKCS#11 interface that acts as a bridge to Entrust Remote Signing Engine, in order to replace local signing keys with remote signing keys.

5.3.1 Virtual smart card

Entrust Remote Signing Engine Desktop Virtual Card (also called Desktop VC) is a plugin for user desktops that implements the functions of MS-CAPI and PKCS#11 interfaces, making use of the Entrust Remote Signing Engine to carry out the creation of signatures. The steps it performs are similar to those in Figure 7, with the only difference being that it ends up invoking the digital signature API instead of the document signature API. That's because the MS-CAPI and PKCS#11 interfaces do not provide the complete document, but only the hash of the document.

A virtual smart card provides the same functionality as a physical smart card, but with fewer costs (no need for physical smart cards and the corresponding readers) and less hassle (no problems derived from installing the drivers in different versions of operating systems). Just like a physical smart card, it provides strong protection for signing keys. And since it's centrally stored in Entrust Remote Signing Engine, it's isolated from the operating system from which it's used.

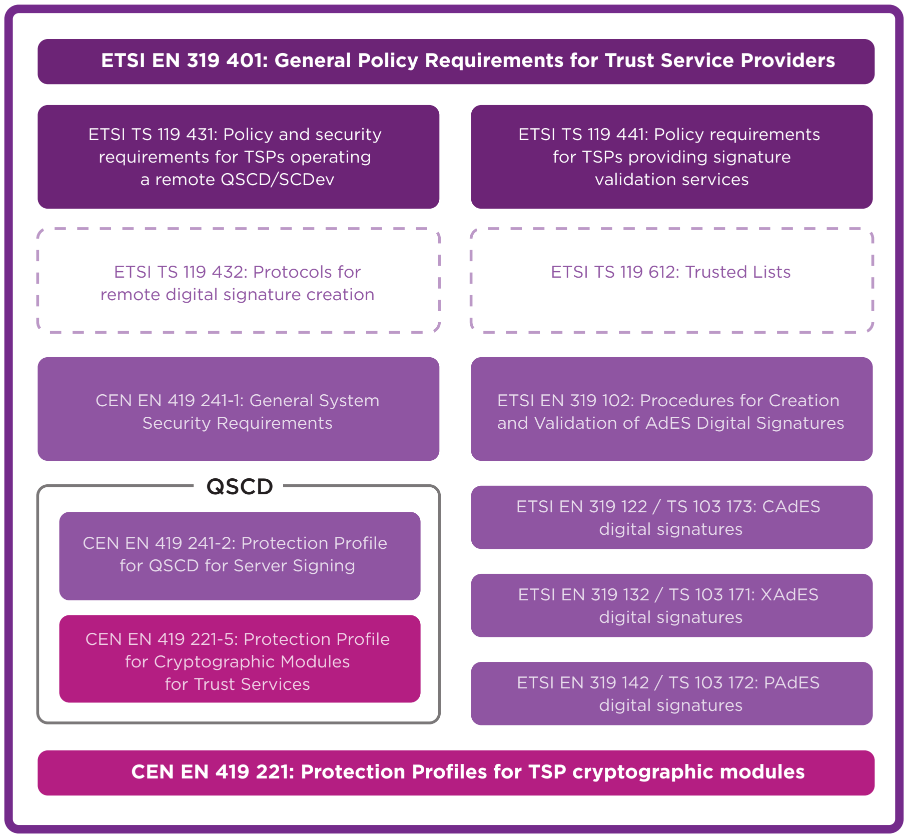
It also provides advantages in terms of access control to keys. Unlike physical smart cards that usually protect the keys with a static PIN, a virtual smart card allows for multi-factor authentication. And it also allows for a centralized key usage log that can be audited at any time.

It's worth noting that LTV signatures are not created by Entrust Remote Signing Engine, but by the desktop applications themselves. For example, Adobe Acrobat Reader obtains the URLs of the OCSP servers from the information contained in the digital certificates, while the URL of the TSA has to be indicated in its settings.

Regarding the subsequent archiving of documents, the same options are available as previously described.

Appendix A – Technical Standards

Figure 8 below shows the main technical standards that a trust service provider (TSP) must comply with to provide the remote signing service, when the objective includes being able to create qualified signatures.


ETSI EN 319 401: General Policy Requirements for Trust Service Providers
ETSI TS 119 431: Policy and security requirements for TSPs operating a remote QSCD/SCDev
ETSI TS 119 441: Policy requirements for TSPs providing signature validation services
ETSI TS 119 432: Protocols for remote digital signature creation
ETSI TS 119 612: Trusted Lists
CEN EN 419 241-1: General System Security Requirements
ETSI EN 319 102: Procedures for Creation and Validation of AdES Digital Signatures
QSCD
CEN EN 419 241-2: Protection Profile for QSCD for Server Signing
CEN EN 419 221-5: Protection Profile for Cryptographic Modules for Trust Services
ETSI EN 319 122 / TS 103 173: CAdES digital signatures
ETSI EN 319 132 / TS 103 171: XAdES digital signatures
ETSI EN 319 142 / TS 103 172: PAdES digital signatures
CEN EN 419 221: Protection Profiles for TSP cryptographic modules

- (Dark purple): Scope of the TSP
- (Light purple): Scope of the signing product
- (Pink): Scope of the HSM product
- (Dashed lines): Functionality not available in Entrust Signing Automation Engine 4.2 - please consult us for more information

Figure 8. Technical standards for a TSP creating qualified signatures

Note that, in Figure 8, the two standards in the QSCD (qualified signature creation device) box are both Protection Profiles, which means that the product must first implement the requirements specified in the document, and then undergo a conformity assessment to obtain a Common Criteria EAL4+ certification.

Entrust currently offers a Signature Activation Module (SAM) which is common criteria certified (v1.0.3) and certified as a QSCD by A-SIT (v 1.0.3 and v1.0.4). Entrust has also started the common criteria certifcation of v1.1.0, which is expected to be completed by mid-2025.

In Europe, the eIDAS regulation does not yet accept the Common Criteria certification of the SAM module for the creation of qualified signatures. As a transitional measure, the so-called QSCD Accreditation is required and must be granted by an eIDAS auditor.

Appendix B – Supported Standards and Algorithms

Standards

REFERENCE	STANDARD
[OIDC]	OpenID Connect Core 1.0
[OAUTH]	OAuth 2.0 Authorization Framework, IETF RFC 6749
[OCSP]	Online Certificate Status Protocol, IETF RFC 2560
[LDAP]	Lightweight Directory Access Protocol
[PAdES]	PDF Advanced Electronic Signature Profiles. ETSI EN 319 142, ETSI TS 102 778 PAdES Baseline Profile. ETSI TS 103 172
[PDFRef]	PDF Reference (sixth edition). Adobe Portable Document Format Version 1.7 or above
[PKCS#1]	PKCS #1 v2.1: RSA Cryptography Standard, version 2.1. IETF RFC 3447
[SAML]	Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS SSTC
[TWS]	Security Requirements for Trustworthy Systems Supporting Server Signing, CEN EN 419 241
[TLS/SSL]	Transport Layer Security/Secure Socket Layer
[TSP]	Time-Stamp Protocol, IETF RFC 3161
[X509]	ITU-T Recommendation X509v3
[XAdES]	XML Advanced Electronic Signatures. ETSI EN 319 132, ETSI TS 101 903 XAdES Baseline Profile. ETSI TS 103 171
[XML-DSig]	D. Eastlake et al. XML Signature Syntax and Processing. W3C Recommendation
[CAdES]	CMS Advanced Electronic Signatures. ETSI EN 319 122, ETSI TS 101 733 CAdES Baseline Profile. ETSI TS 103 173

Signature algorithms

Entrust Remote Signing Engine supports the following digital signature algorithms:

- RSA-SHA1
- RSA-SHA256
- RSA-SHA384
- RSA-SHA512

For RSA signatures, the supported key sizes are:

- Up to 4096, in supported HSMs (depending on models)

For more information

888.690.2424
+44 (0) 118 953 3000
sales@entrust.com
entrust.com

ABOUT ENTRUST CORPORATION

Entrust keeps the world moving safely by enabling strong identities, secure payments, and protected data. We offer an unmatched breadth of solutions that are critical to the future of secure enterprises, governments, the people they serve, and the data and transactions associated with them. With our experts serving customers in more than 150 countries and a network of global partners, it's no wonder the world's most trusted organizations trust us.

Learn more at
entrust.com

ENTRUST

U.S. Toll-Free Phone: 888 690 2424
International Phone: +1 952 933 1223
+44 (0) 118 953 3000
info@entrust.com