ENTAKSISOLUTIONS

SISTEMA DI GESTIONE CERTIFICATO
ISO 9001 | ISO 20000-1 | ISO 22301 | ISO 37001
ISO 27001 | ISO 27017 | ISO 27018 | ISO 27035
SERVIZI FIDUCIARI QUALIFICATI
ETSI 319 401 | ETSI 319 411-1 e 2 | ETSI 319 421 | ETSI 119 511
FIRME E SIGILLI ELETTRONICI - MARCHE TEMPORALI
CONSERVAZIONE A LUNGO TERMINE

Trust Services

API Manual

UM 20240416 User Manual for Trust Services APIs
Version 1.0.0, 02/20/2026

Table of Contents

Document Information

Project	Entaksi TSP
Document ID	UM 20240416 User Manual for Trust Services APIs
Type	User Manual
Creation Date	04/16/2024
Last Revision	02/20/2026
Version	1.0.0
Author	Stefano Travelli
Status	Released
Classification	Confidential

Printed copies of this document are to be considered working copies not registered in the SIG.

Revisions

Date	Version	Name	Action
04/19/2024	0.0.1	Stefano Travelli	Draft documentation for the model and services.
07/08/2024	0.0.2	Stefano Travelli	Revision.
10/01/2024	0.0.3	Stefano Travelli	Definition of the process for ephemeral certificate signing.
01/09/2025	0.0.4	Luigi Ruocco	Specification of the hash signing service.
01/17/2025	0.0.5	Stefano Travelli	Added documentation for CSCv2 APIs.
02/20/2026	1.0.0	Ciro Sasso	Removed documentation related to the deprecated remote services.

Document Approval

Date	Person	Role	Signature
02/20/2026	Luigi Ruocco	DT	

1. Introduction

This manual describes how to use the APIs for accessing the eIDAS trust services provided by Entaksi Solutions SpA Irish Branch.

Entaksi Solutions SpA Irish Branch is a Trust Service Provider for the following services:

- `QCert for ESig` Qualified certificate for electronic signature
- `QCert for ESeal` Qualified certificate for electronic seal
- `QTimestamp` Qualified time stamp
- `QPres for QESig` Qualified preservation service for qualified electronic signature
- `QPres for QESeal` Qualified preservation service for qualified electronic seal

These services together provide a complete solution for applying qualified timestamps, electronic signatures, and seals, and for preserving them long-term with legal validity within the European Union.

The long-term preservation service for signatures and seals is implemented within the preservation system and has specific documentation related to document submission, storage, and distribution processes. The preservation service APIs, including those for the qualified long-term preservation of signatures and seals, are documented in a separate manual (see `UM 20150928 eDoc API User Manual`).

This document, therefore, focuses on the API documentation for the services related to the issuance of signature and seal certificates and the timestamping service.

Signature and seal certificates are typically issued to requesting entities along with the private key associated with the certificate. The cryptographic material is stored within a Qualified Signature Creation Device (QSCD). In this scenario, the private key is managed by the certificate holder and not by the Trust Service Provider. The application of signatures and seals is performed under the holder's control using the QSCD (e.g., a token or smart card) through dedicated software tools capable of using the QSCD to execute the signing process, such as eSIGN Desktop provided by Entaksi.

Entaksi provides services to maintain the validity of applied signatures over time by delivering validation services, particularly the distribution of the Certificate Revocation List (CRL) and the Online Certificate Status Protocol (OCSP) service.

The timestamping service is made available as an RFC 3161 endpoint, consisting of an HTTP service that can be queried to obtain a qualified timestamp to associate with a digitally signed or unsigned document. The qualified timestamp serves as proof of existence of the document at the specified date. This proof is legally binding and enforceable in the European Union.

The trust services APIs allow for management of the timestamping service authentication, as well as tracking of the number of timestamps issued and remaining in relation to those purchased.

In some scenarios, the private key associated with signature and seal certificates is managed directly by the Trust Service Provider. In these cases, the private keys managed by the Trust Service Provider are associated with:

- Qualified electronic seal certificates issued for automated document signing procedures where the seal owner guarantees the document's integrity.
- Qualified electronic signature certificates issued to individuals residing in the Republic of Italy, with a specific usage limitation for automatic signing procedures in accordance with the provisions established by the Agenzia per l'Italia Digitale.
- Qualified electronic signature certificates for which the private key managed by the Trust Service Provider is activated by the certificate holder through a Signature Activation Module (SAM), ensuring exclusive control of the key by the holder, in compliance with the EN 419 241-2:2019 security profile ("Trustworthy Systems Supporting Server Signing - Part 2: Protection Profile for QSCD for Server Signing").

For these scenarios, this manual provides documentation for accessing the remote signing service.

1.1. Reference Documentation

- RFC 7616 The 'Basic' HTTP Authentication Scheme.
- RFC 6749 The OAuth 2.0 Authorization Framework.
- MAN eIDAS 20230426 PKI Disclosure Statement EN 1.1.0.
- MAN eIDAS 20230426 Certificate Policy and Certification Practice Statement EN 1.1.0.

- MAN eIDAS 20230426 TSA Disclosure Statement EN 1.1.0.
- MAN eIDAS 20230426 TSA Policy and Practice Statement EN 1.1.0.
- UM 20150928 eDoc API User Manual.
- ETSI EN 319 122 Electronic Signatures and Infrastructures (ESI); CAdES digital signatures.
- ETSI EN 319 132 Electronic Signatures and Infrastructures (ESI); XAdES digital signatures.
- ETSI EN 319 142 Electronic Signatures and Infrastructures (ESI); PAdES digital signatures.

1.2. System Architecture

TBD

2. Access to the APIs

To access the trust service APIs, it is sufficient to perform HTTPS calls to the endpoints of the various services.

The published services are divided into three categories:

- APIs related to the management interface.
- APIs of the Certification Authority validation services.
- APIs for remote signing and timestamping services.

2.1. APIs Related to the Management Interface

The endpoints of the individual services and their detailed documentation are available in OpenAPI 3 format.

At the following address, a web page automatically interprets the API documentation organizing and displaying each service's features: https://entaksi.eu/tsp/api/v1-doc/

Access to the management interface APIs requires a JWT token obtained via the OpenID Connect authentication protocol.

All features of the management interface are also available in the user interface of Entaksi services, under the "Trust Services" section of the console accessible to authorized users at https://entaksi.eu/console.

2.2. Certification Authority Validation Service APIs

The Certification Authority provides the following services for certificate validation:

- Certificate Revocation Lists (CRLs)
- OCSP service

CRLs are updated every 8 hours, are valid for 24 hours, and are available at the following URLs:

For "Entaksi Qualified Electronic Signatures CA G1": http://va.entaksi.eu/crls/Entaksi_Qualified_Electronic_Signatures_CA_G1.crl

For "Entaksi Qualified Electronic Seals CA G1": http://va.entaksi.eu/crls/Entaksi_Qualified_Electronic_Seals_CA_G1.crl

For "Entaksi Qualified Time-stamps CA G1": http://va.entaksi.eu/crls/Entaksi_Qualified_Electronic_Seals_CA_G1.crl

The OCSP service endpoint is located at: http://va.entaksi.eu/ocsp

Normally, it is not necessary to manually use the URLs listed above, as the endpoints are already included in the certificate's Authority Information Access extension and are automatically detected by electronic signature validation tools.

The validation services are publicly accessible.

2.3. APIs for Timestamping and Automated Remote Signing Services

The RFC 3161 timestamping service is available at: https://tsa.entaksi.eu/tsu1.

The remote signing services are available at the following URLs:

- https://tsa.entaksi.eu/signer/{certId}/pdf, for applying PAdES signatures (PDF documents) in compliance with ETSI EN 319 142 ("Electronic Signatures and Infrastructures (ESI); PAdES digital signatures").
- https://tsa.entaksi.eu/signer/{certId}/xml, for applying XAdES signatures (XML documents) in compliance with ETSI EN 319 132 ("Electronic Signatures and Infrastructures (ESI); XAdES digital signatures").

- https://tsa.entaksi.eu/signer/{certId}/cms, for applying CAdES signatures (generic documents) in compliance with ETSI EN 319 122 ("Electronic Signatures and Infrastructures (ESI); CAdES digital signatures") and for signing processes with local hash calculation.
- https://tsa.entaksi.eu/signer/{certId}/plain, to obtain the DER-encoded binary signature of an arbitrary set of data.

Where `{certId}` is the identifier code of the certificate to be used.

Access to timestamping and remote signing services requires an access key associated with the signature and timestamp account, which must be generated by an authorized user via the management services.

3. Authentication

The authentication method for accessing the APIs is based on one of the following two mechanisms:

- Authentication via Access Key (for timestamping and remote signing services)
- Authentication via a JWT token obtained using the OpenID Connect protocol (for management system APIs)

3.1. Authentication via Access Key

The Access Key is generated by an authorized user through the management system APIs or via the dedicated function in the Entaksi services console.

The Access Key is associated with a timestamp or signature account, and its use results in the consumption of one credit unit from this account.

The Access Key consists of a code and a secret value, which can be considered as a username and password. To ensure the security of the authentication mechanism, the secret value is generated by the system with appropriate complexity, is not stored in plain text, and is displayed to the user only once during its generation.

The user must use the Access Key in the software interfacing with the service and store it securely. The user can revoke the Access Key at any time and may define an arbitrary number of Access Keys on the same account.

The Access Key — that is, the code and the secret — must be used as the username and password in the HTTP Basic authentication scheme. In practice, they must be used to set the value of the "Authorization" header as follows:

```
Authorization: Basic <authorization code>
```

Where `<authorization code>` is the Base64 representation of the Access Key's code and secret, separated by a colon (":").

For further details about the HTTP Basic authentication scheme, see RFC 7617.

4. Entaksi Trust Services Model

The trust services model related to the APIs consists of the following entities:

Entity	Description
Certification Authority	A Certification Authority that issues certificates for electronic signatures.
Certification Officer	Person representing the Certification Authority.
Registration Officer	Person appointed by the Certification Authority to perform identity verification tasks. The Registration Officer also collects requests for generating a certificate for signature or electronic seal.
Issued Certificate	Certificate generated and assigned to a subject as the result of an approved request.
Certificate Application	Request made by a subject, identified by a Registration Officer, to a Certification Authority to obtain a certificate.

Entity	Description
Subject	The individual or legal entity to whom a certificate is issued.
Identity Document	Identity document or other evidence that proves the identity of the subject.
Address	Subject's address.
Subscriber	Organization (customer) using trust services to obtain timestamps and certificates for one or more subjects.
Balance	Total available signing operations on a signature or timestamp account for a previously purchased credit package.
Account	Signature or timestamp account that can be recharged (Signature Package) to use the service from a specific Signing Unit.
Access Key	Access key used to submit signature or timestamp requests to a Signing Unit.
Signatures Package	Credit package purchased by a customer and assigned to an account to increase the balance.
Signature Request	Signature or timestamp request sent to a Signing Unit using an active Access Key on a connected account.
Order	Purchase order through which the customer adds credit packages to their accounts.
Signatures Transaction	Transaction that transfers a certain number of signature credits from one account to another, based on a purchased package.
Quality Of Service	Service quality level defined for an account.
QSCD	Qualified Signature Creation Device that stores one or more certificates together with the private key.
Signing Unit	Signature or timestamp service to which requests are sent using an Access Key.
User Role	Role assigned to a specific user within a customer account.
User	Service user.
Role	Role defined in a particular customer account to manage user permissions.

4.1. OpenAPI 3 Documentation

Further details on the model, including the properties defined for each entity, are available in the OpenAPI 3 documentation at the following address:

- https://entaksi.eu/tsp/api/v1-doc

4.2. Enumerations

4.2.1. `CertificateApplicationStatus`

Code	Description
`D`	Draft

Code	Description
I	Issued
A	Approved
R	Rejected
P	Provisioned
G	Waiting for the end of signature phase related the general terms documents
Y	Error status after the signature phase related the general terms documents
W	Waiting for the end of signature phase related the registration officer
E	Error status after the signature phase related the registration officer
C	Waiting for the end of signature phase related the certification officer
X	Error status after the signature phase related the certification officer

4.2.2. `CertificateStatus`

Code	Description
A	Certificate requested
I	Certificate issued
D	Certificate denied
R	Certificate revoked
S	Waiting for the end of signature phase related the physical acceptance of the certificate to the subject
E	Error status after the signature phase related the physical acceptance of the certificate to the subject
C	Certificate physically accepted by the subject

4.2.3. `ProvisioningMode`

Code	Description
LC	Certificate for local signing with a private key stored on a smart card QSCD.
LT	Certificate for local signing with a private key stored on a USB token QSCD.
RS	Certificate for remote signing activated by the user using a Signature Activation Module.
RA	Certificate for remote signing restricted to automated signing procedures.

Code	Description
RE	Certificate for remote signing with ephemeral certificate.
RO	Certificate for remote signing activated by the user using an OTP code.

4.2.4. GlobalRole

Code	Description
ADM	Backoffice user with administrative functions on the service.
AUD	Service auditor with read-only access.
SLS	Sales representative managing price lists and orders.
RAO	Registration Authority Officer.
CAO	Certification Authority Officer.
USR	User with a role in at least one subscriber.

4.2.5. IdentityDocumentStatus

Code	Description
V	Valid document.
I	Invalid document.
U	Document validity to be confirmed.

4.2.6. IdentityDocumentType

Code	Description
PP	Passport.
ID	National identity document.
SF	Signed form with a valid qualified electronic signature already held by the user.
EI	Electronically signed record proving an eIDAS login transaction confirming the user's identity.
OO	Other document.

4.2.7. OfficerStatus

Code	Description
G	Role assigned.
R	Role revoked.

4.2.8. OrderStatus

Code	Description
D	Draft.
I	Submitted.
W	Awaiting payment.
P	Paid and completed.
V	Canceled.

4.2.9. RateLimitType

Code	Description
F	Fixed window rate limit: time is divided into non-overlapping intervals, and access count is limited within each. Bursts are possible near interval transitions.
R	Rolling rate limit: continuously checks for access limit breaches. Exceeding the limit is not allowed at any time.
S	Sliding window rate limit: calculates the maximum number of calls over a time unit and ensures a uniform distribution. Allows short bursts if usage was below the limit.

4.2.10. SubjectIdentifierType

Code	Description
PAS	Passport number.
IDC	National identity card number.
PNO	National personal identifier.
TIN	Tax Identification Number.
VAT	VAT number.
NTR	National trade register number.
PSD	National authorization number for a payment service provider under PSD2 Directive 2015/2366.

4.2.11. SubjectType

Code	Description
N	Natural person.
M	Natural person associated with a legal entity.
L	Legal entity.

4.2.12. SubscriberType

Code	Description
P	Natural person.
L	Legal entity.

4.2.13. AccessKeyStatus

Code	Description
A	Active access key.
R	Revoked access key.
S	Suspended access key.

4.2.14. AccountStatus

Code	Description
A	Active account.
S	Suspended account.
0	Account with no remaining credit.

4.2.15. ServiceType

Code	Description
RSA	Remote automatic signature service.
RSS	Remote signature service using a certificate whose private key is managed by the TSP and activated by the user via a Signature Activation Module.
TSU	Timestamping service.
QSC	Qualified certificate issuance service for electronic signature.

Code	Description
QSR	Qualified certificate renewal service for electronic signature.
QEC	Qualified certificate issuance service for electronic seal.
QER	Qualified certificate renewal service for electronic seal.

4.2.16. SigningUnitStatus

Code	Description
A	Active signing unit.
R	Revoked signing unit.

4.2.17. RequestStatus

Code	Description
ISS	Request completed.
WCR	Request denied: invalid credentials.
AKS	Request denied: suspended access key.
AKR	Request denied: revoked access key.
SUR	Request denied: signing unit revoked.
SUM	Request denied: accessed from an unauthorized IP for the signing unit.
AIN	Request denied: account not active.
OOC	Request denied: account out of credit.
ERR	Request denied due to internal error.

5. Signature and Timestamp Services

5.1. Time-stamp Unit for Applying Timestamps

The Time-stamp Unit service responds according to the RFC 3161 protocol at the following address:

- https://tsa.entaksi.eu/tsu1

The service must be called using HTTP Basic authentication as defined in RFC 7616, using an Access Key associated with a timestamp account.

The username is the code of an active Access Key on the signature account, and the password is the "secret" value of this Access Key.

5.2. Remote Signature Services CSCv2

The remote signature services with certificates managed by the TSP use the CSCv2 APIs. They are available in the following two scenarios:

- Automatic remote signature
- Remote signature with user approval

The first scenario involves signature authorization using access-key/secret-key credentials enabled on an account associated with the remote signature service.

The second scenario involves user confirmation for each signature applied through the remote service, and the authorization is performed via an OTP sent to the user's email address.

5.2.1. Certificate Information

Given the certificate code `certId`, the following API allows retrieving information about the certificate and the required authorization mode.

```
POST tsp/api/v1/csc/v2/credentials/info
```

The call requires authentication by a valid user with the CSC privilege on the subscriber where the certificate owner is registered.

The request body is a JSON object `CredentialInfoRequest`:

CredentialInfoRequest

```
{
  "credentialID": "214ECBD6-A470-4C40-83D5-A6A7F2C77FBA", ①
  "certificates": "single", ②
  "certInfo": "true", ③
  "authInfo": "true", ④
  "lang": "en", ⑤
  "clientData": "" ⑥
}
```

① The identifier code of the certificate.

② A value that can be `"single"`, `"chain"`, or `"none"`, to indicate whether the response should include the certificate, the full certificate chain, or nothing.

③ A boolean value to indicate if the response should include certificate information.

④ A boolean value to indicate if the response should include authorization mode information.

⑤ The language of the response in RFC5646 format.

⑥ Not used.

The response will be a `CredentialInfo` object:

CredentialInfo


```
{
  "credentialID": "214ECBD6-A470-4C40-83D5-A6A7F2C77FBA", ①
  "description": "Paolo Rossi", ②
  "signatureQualifier": "eu_eidas_qes", ③
  "keyInfo": { ④
    "status": "enabled",
    "algo": [
      "1.2.840.113549.1.1.1"
    ],
    "len": 3072,
    "curve": null
  },
  "cert": {
    "status": "valid",
    "certificates": [
      "RDdEOTc2ODYtMDVBRS00Nzk2LThENUMtNjZEMTI0Q0JERUVDCg==" ⑤
    ],
    "issuerDN": "cn=Entaksi Qualified eSignature CA G1",
    "serialNumber": "5AAC41CD8FA22B953640",
    "subjectDN": "cn=Paolo Rossi",
    "validFrom": "202401011000000Z",
    "validTo": "202501011000000Z"
  },
  "auth": {
    "mode": "explicit",
    "expression": "access-key AND secret-key", ⑥
    "objects": [
      {
        "type": "Password",
        "id": "OTP",
        "label": "One Time Password",
        "description": "OTP sent by email",
        "format": "A"
      },
      {
        "type": "Access Key",
        "id": "access-key",
        "label": "Access Key",
        "description": "An Access Key generated in a signing service account.",
        "format": "A"
      },
      {
        "type": "Secret Key",
        "id": "secret-key",
        "label": "Secret Key",
        "description": "A Secret Key generated in a signing service account.",
        "format": "A"
      }
    ]
  },
  "scal": "2", ⑦
  "multisign": 1,
  "lang": "en"
}
```


① The identifier code of the certificate.

② A description of the certificate, corresponding to the cn of the subject.

③ The type of signature applied using the certificate.

④ Some details about the private key of the certificate.

⑤ The Base64 representation of the certificate.

⑥ An expression that defines the type of credentials required for authorization.

⑦ The level of exclusive access control by the certificate owner on signature operations.

Typically, the `expression` property will have one of the following values:

- OTP, for remote signature with user authorization
- `access-key AND secret-key`, for automatic remote signature

Accordingly, the `objects` list will contain either the element with `id` = OTP, or the two elements with `id` = `access-key` and `secret-key`.

The application handling the signature process will use one mode or the other. The authentication method for a certificate is not intended to change after creation.

5.2.2. OTP Sending Request

If the authorization mode requires sending a One-Time Password (OTP), this can be requested using the following API:

```
POST tsp/api/v1/csc/v2/credentials/sendOTP
```

The call requires authentication by a valid user with the CSC privilege on the subscriber where the certificate owner is registered.

The user will receive the OTP at the email address associated with the certificate. The application handling the signature process must prompt the user to enter the received value, which will then be used in the service call to request authorization to use the private key.

The content of the request is:

SendOTPRequest

```
{
  "credentialID": "214ECBD6-A470-4C40-83D5-A6A7F2C77FBA", ①
  "clientData": ""
}
```

① - The code that identifies the certificate.

If the authorization mode does not require sending an OTP, this service does not need to be used.

5.2.3. Authorization Request

To call the signing service, it is necessary to obtain authorization to use the private key associated with the certificate. This authorization takes the form of a data block called Signature Activation Data (SAD), which is obtained by calling the following API:

```
POST tsp/api/v1/csc/v2/credentials/authorize
```

For remote signing operations with user approval, the call requires authentication by a valid user with the CSC privilege on the subscriber where the certificate owner is registered.

For automatic remote signing operations, no authentication is required beyond the `access-key` and `secret-key` values included in the payload.

The content of the request is an `AuthorizeRequest` object:

AuthorizeRequest with access-key and secret-key

```
{
  "credentialID": "214ECBD6-A470-4C40-83D5-A6A7F2C77FBA", ①
  "numSignatures": 1, ②
  "hashes": [  ③
    "c1RPZ3dPbSs0NzRnRmowcTB4MWlTTnNwS3FiY3NlNEllaXFsRGcvSFd1ST0="
  ],
  "hashAlgorithmOID": "2.16.840.1.101.3.4.2.1",  ④
  "authData": [  ⑤
    {
      "id": "access-key",
      "value": "66B088E4-756D-48DF-9B58-A2018C292A00"
    },
    {
      "id": "secret-key",
      "value": "CA8F187D-2AB8-4196-9277-463D027A10EC"
    }
  ],
  "clientData": ""
}
```

① - The code that identifies the certificate.

② - The number of signature operations to authorize.

③ - If the Secure Certificate Access Level (SCAL) is equal to 1 (SCAL_1), the `hashes` property can be omitted. If instead it is equal to 2 (SCAL_2), the `hashes` property must include the hashes of the operations to authorize.

④ - The hash algorithm used (if hashes are provided).

⑤ - The authentication data, in this case "access-key" and "secret-key".

If the authorization mode is "OTP", the call must be preceded by a call to the `sendOTP` service described in the previous section. The value received by the user must be acquired by the application that manages the signing process and included in the `authorize` call as follows:

AuthorizeRequest with OTP

```
{
  "credentialID": "214ECBD6-A470-4C40-83D5-A6A7F2C77FBA",
  "numSignatures": 1,
  "hashes": [
    "c1RPZ3dPbSs0NzRnRmowcTB4MWlTTnNwS3FiY3NlNEllaXFsRGcvSFd1ST0="
  ],
  "hashAlgorithmOID": "2.16.840.1.101.3.4.2.1",
  "authData": [
    {
      "id": "OTP",
      "value": "93729209"
    }
  ],
  "clientData": ""
}
```

The response is an `AuthorizeResponse` object:

AuthorizeResponse

```
{
  "SAD": "RDQwRUExMDQtMEJGQS00ODU0LUE2N0MtRjE1MzBDN0ZERkE5Cg==", ①
  "expiresIn": 600 ②
}
```

① The value of the SAD (Signature Activation Data) that can be used to perform the next signature operations.

② The expiration time of the SAD in seconds.

5.2.4. Hash Signature Request

Using the SAD, the application that manages the signing process can proceed with the signature via:

```
POST tsp/api/v1/csc/v2/signatures/signHash
```

The request does not require any authentication other than including a valid SAD.

The request will contain the hashes to be signed:

Hash Signature - Request

```
{
  "credentialID": "8F45ACFE-F075-470D-9E7D-C3F737EB87DE", ①
  "SAD": "dGhlIHNpZ25hdHVyZSBhdXRob3JpemF0aW9uIGRhdGEK", ②
  "hashes": [
    "Y2lhbyBjaWFvIGNpYW8gaGFzaAo=" ③
  ],
  "hashAlgorithmOID": "2.16.840.1.101.3.4.2.1", ④
  "signAlgo": "1.2.840.113549.1.1.11", ⑤
  "operationMode": "S", ⑥
  "clientData": "" ⑦
}
```

① The unique code identifying the certificate.

② The SAD value obtained from the authorization call.

③ The hash to be signed.

④ The OID of the algorithm used to compute the hash (`2.16.840.1.101.3.4.2.1` for SHA-256).

⑤ The algorithm used to sign the document hash (`1.2.840.113549.1.1.11` for SHA256WithRSA).

⑥ S (Synchronous), the execution mode of the operation.

⑦ Not used.

Hash Signature - Response

```
{
  "signatures": [
    "ZmlybWExZmlybWExZmlybWExCg==" ①
  ],
  "responseID": "" ②
}
```

① The base64-encoded signature of the hash.

② Not used.

5.2.5. Document Signature Request

The application managing the signing process, having obtained the SAD, can call the following service to sign one or more documents.

```
POST tsp/api/v1/csc/v2/signatures/signDoc
```

The request does not require any authentication other than including a valid SAD.

The request contains the following data:

Document Signature - Request

```
{
  "credentialID": "8F45ACFE-F075-470D-9E7D-C3F737EB87DE", ①
  "signatureQualifier": "eu_eidas_qes", ②
  "SAD": "dGhlIHNpZ25hdHVyZSBhdXRob3JpemF0aW9uIGRhdGEK", ③
  "documents": [ ④
    {
      "document": "ZmlybWExZmlybWExZmlybWExCg==", ⑤
      "signature_format": "C", ⑥
      "conformance_level": "Ades-B-B", ⑦
      "signAlgo": "1.2.840.113549.1.1.13", ⑧
      "signAlgoParams": null, ⑨
      "signed_props": null, ⑩
      "signed_envelop_property": "Detached" ⑪
    }
  ],
  "documentDigests": [ ⑫
    {
      "hashes": "ZmlybWExZmlybWExZmlybWExCg==", ⑬
      "hashAlgorithmOID": null, ⑭
      "signature_format": "C",
      "conformance_level": "Ades-B-B",
      "signAlgo": "1.2.840.113549.1.1.13",
      "signAlgoParams": null,
      "signed_props": null,
      "signed_envelope_property": "Detached"
    }
  ],
  "operationMode": "S", ⑮
  "validity_period": null, ⑯
  "response_uri": null, ⑰
  "clientData": null, ⑱
  "returnValidationInfo": null ⑲
}
```

① The unique code identifying the certificate.

② The identifier of the signature type to be created, `eu_eidas_qes` is the only supported value at the moment.

③ The SAD value obtained from the authorization call.

④ The list of documents to be signed, mutually exclusive with the list in point (12).

⑤ The document to be signed, encoded in base64 format.

⑥ The signature format to apply to the single document. Possible values:

- C, CAdES (CMS Advanced Electronic Signatures).
- X, XAdES (XML Advanced Electronic Signatures).
- P, PAdES (PDF Advanced Electronic Signatures).
- J, JAdES (JSON Advanced Electronic Signatures).

⑦ The compliance level of the issued signature. Possible values:

- Ades-B-B (default value).
- Ades-B-T.
- Ades-B-LT.
- Ades-B-LTA.

⑧ The OID of the algorithm used to sign the document. Supported values:

- `1.2.840.113549.1.1.11` for SHA256WithRSA.
- `1.2.840.113549.1.1.13` for SHA512WithRSA.

⑨ Additional parameters for specific types of signing algorithms. Currently supported signing algorithms do not require further parameters.

⑩ Not used.

⑪ The type of package with which the signature is generated. Supported values:

- XAdES.
  a. Enveloped.
  b. Enveloping.
  c. Detached.
- CAdES.
  a. Enveloping.
  b. Detached.
- PAdES.
  a. Enveloped.
- JAdES.
  a. Enveloping.
  b. Detached.

⑫ The list of hashes of the documents to be signed is mutually exclusive with the list in point (4).
⑬ The base64-encoded hash of the single document to be signed.
⑭ The OID of the algorithm used to compute the document hash. If omitted, it will be implicitly derived from the signing algorithm.
⑮ S (Synchronous), the execution mode of the operation.
⑯ Not used.
⑰ Not used.
⑱ Not used.
⑲ Not used.

Document Signature - Response

```
{
  "responseID": "string", ①
  "DocumentWithSignature": [
    "string" ②
  ],
  "SignatureObject": [
    "string" ③
  ],
  "validationInfo": null
}
```

① Not used
② The document signature in base64 format, in the cases: Enveloped, Enveloping.
③ The document signature in base64 format, in the case: Detached.

5.3. Signature with Ephemeral Certificate

An ephemeral certificate (or "one-shot" or "short-term") is a certificate whose validity is shorter than the time required to record a possible revocation request. The certificate and its private key are used for a single signing operation, after which the private key is destroyed.

The issuance and use of an ephemeral certificate consists of three phases that occur sequentially in a limited time period:

- Subject Registration
- Request Definition and Validation
- Document Signature

Prerequisites

1. The operations are managed by a Signing Application that uses the eTRUST APIs. Access to the APIs requires an account **with Registration Officer permissions** (profile responsible for registering subjects and their certificate issuance requests).
2. The certificate issuance request must be validated by registering a link with a digital identity of the subject to whom the certificate is issued. Currently, linking with an SPID identity held by the requesting subject is supported.

The following sequence diagram summarizes the three phases and the interaction among the various components.

The diagram defines the following actors:

- **User**: The user interacting with a UI that needs to sign a document.
- **UserBrowser**: The frontend application used by the user.
- **SigningApplication**: The backend application used by the user that must integrate eTRUST signing services.
- **eTRUST**: The Entaksi service, provider of the signing services.
- **IdentityProviderService**: The eID service provider (SPID).

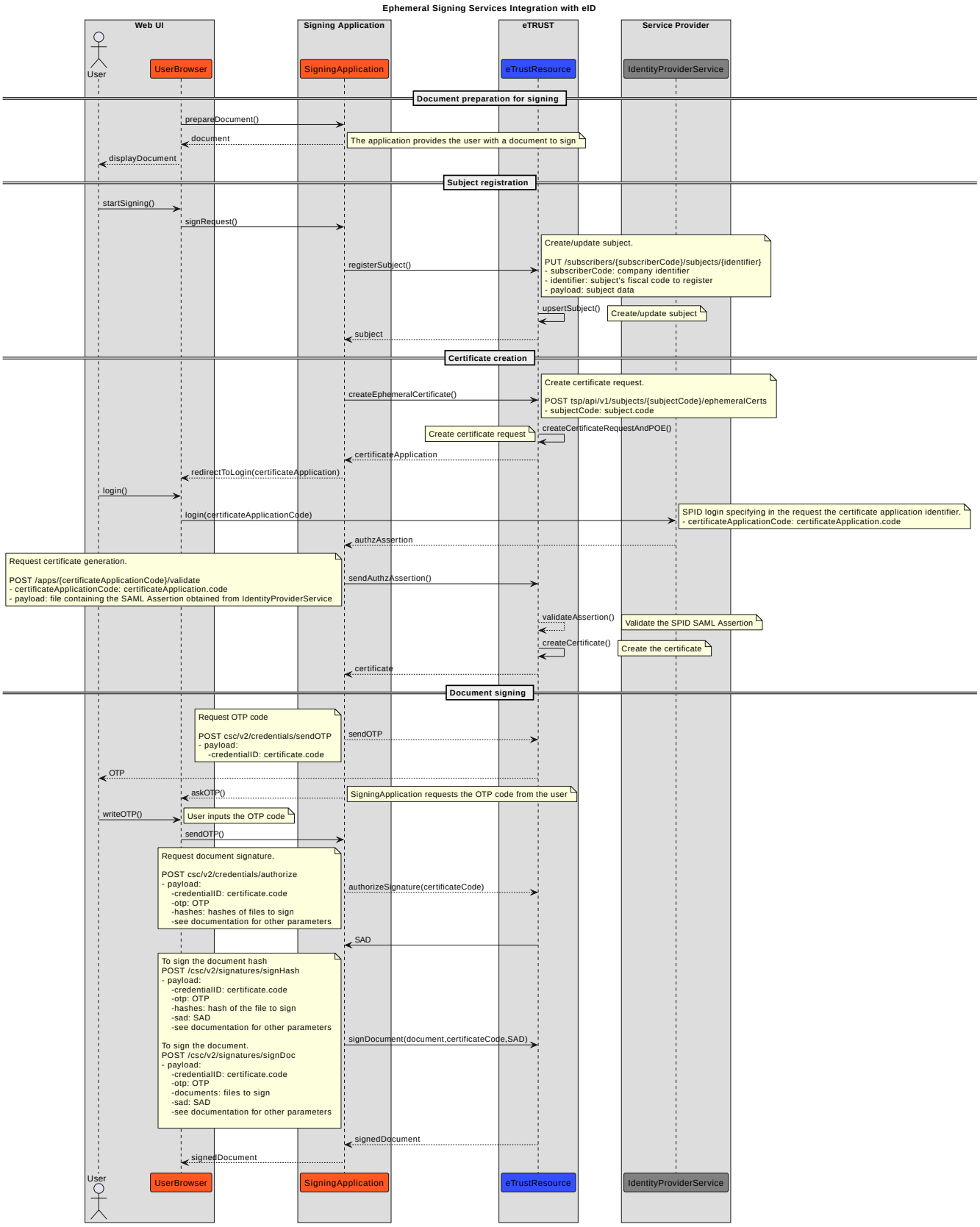

Figure 1. Sequence diagram for integrating ephemeral signing services with SPID identification.

5.3.1. Subject Registration

The following API allows registering a subject with the fiscal code specified in `identifier`. If the subject does not exist, it will be created; otherwise, it will be updated.

```
PUT tsp/api/v1/subscribers/{subscriberCode}/subjects/{identifier}
```

- `subscriberCode`: the company identifier
- `identifier`: the fiscal code of the subject to register
- `payload`: subject data

Subject Registration - Request

```
{
  "type": "N", ①
  "first_name": "Paul", ②
  "last_name": "Reds", ③
  "id_type": "TIN", ④
  "identifier": "RDSPLA17A41H501W", ⑤
  "email": "paul.reds@example.eu", ⑥
  "country": "IT" ⑦
}
```

① `N`, fixed value that defines the subject as a natural person.
② Subject's first name.
③ Subject's last name.
④ `TIN` (Tax Identification Number), fixed value identifying the fiscal code.
⑤ Subject's fiscal code.
⑥ Subject's email address.
⑦ `IT`, fixed value indicating the subject's country (SPID authentication implies the subject is resident in Italy).

Subject Registration - Response

```
{
  "code": "22C13C1B-76B9-4659-8D9F-5BE4156EA1DF",
  "subscriber": {
    "code": "E8836C90-5ABF-4567-9644-9BFF14D1B029",
    "description": "Acme Ltd"
  },
  "type": "N",
  "first_name": "Paul",
  "last_name": "Reds",
  "id_type": "TIN",
  "identifier": "RDSPLA17A41H501W",
  "org_name": null,
  "org_unit": null,
  "title": null,
  "org": null,
  "email": "paul.reds@example.com",
  "country": "IT"
}
```

5.3.2. Request Definition and Validation

The following API is used to create a request for an ephemeral certificate:

```
POST tsp/api/v1/subjects/{subjectCode}/ephemeralCerts
```

Creation of ephemeral certificate request - Request

```
{
  "notes": "Created by the signing application" ①
}
```

① Optional text.

Creation of ephemeral certificate request - Response

```
{
  "code": "494E729C-A56D-42F5-8A32-D6C3C064D767", ①
  "subscriber": {
    "code": "E8836C90-5ABF-4567-9644-9BFF14D1B029", ②
    "description": "Acme Ltd"
  },
  "subject": {
    "code": "22C13C1B-76B9-4659-8D9F-5BE4156EA1DF", ③
    "description": "Paul Reds"
  },
  "ca": {
    "code": "629CE2B6-6C59-4A3F-98FA-B3B7E6A80132", ④
    "description": "Entaksi Qualified eSignature CA G1"
  },
  "type": {
    "code": "EF872B18-6579-4C26-A6F6-4742C9DA184B", ⑤
    "description": "Ephemeral Certificate authenticated with SPID"
  },
  "status": "I", ⑥
  "ro": {
    "code": "2960C7C6-D35E-4FA9-8659-58F6B932D04B", ⑦
    "description": "The Signing Application"
  },
  "app_time": "2022-03-10T12:15:50-04:00", ⑧
  "approval_time": null, ⑨
  "co": null, ⑩
  "notes": "Created by the signing application" ⑪
}
```

① Unique code representing the certificate request.
② The subscriber of the subject, i.e., the entity managing the SigningApplication.
③ The subject associated with the request.
④ The Certification Authority to which the certificate issuance is requested.
⑤ The type of certificate requested, predetermined by the way the request was created.
⑥ `I` (ISSUED), indicates that the request has been sent to the Certification Authority.
⑦ The reference Registration Officer corresponding to the credentials used by the SigningApplication to invoke the services.
⑧ The date and time of the request.
⑨ The date and time of the request approval.
⑩ The Certification Officer who approves the request.
⑪ Optional text.

Once the `certificateApplicationCode` is obtained, the SigningApplication must redirect the user to the SPID service, so that this code is specified in the authentication request.

The SPID authentication request will contain the `certificateApplicationCode` in the `ID` attribute, for example:

SPID AuthnRequest

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="494E729C-A56D-42F5-8A32-D6C3C064D767"
    Version="2.0"
    IssueInstant="2015-01-29T10:00:31Z"
    Destination="https://spid.identityprovider.it/redirect-Post-saml2sso"
    AssertionConsumerServiceURL="http://spid.serviceprovider.it"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    AttributeConsumingServiceIndex="1">

   ...

</samlp:AuthnRequest>
```

At the end of the SPID authentication, the IdentityProviderService (i.e., the provider used by the user to log in) will return an electronically signed XML file (SAML Assertion) that constitutes proof of successful authentication.

The document will contain the same certificateApplicationCode in the InResponseTo attribute: .SPID Response

```
<samlp:Response Destination="https://that.spid.example.org/saml2/acs/post"
        ID="_5e728601-9ad4-4686-b269-81d107a8194a"
        InResponseTo="494E729C-A56D-42F5-8A32-D6C3C064D767"
        IssueInstant="2021-02-04T15:41:59Z"
        Version="2.0"
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">

    ...

</samlp:Response>
```

This document must be transmitted to the eTRUST service to be associated with the certificate request.

The transmission is performed by invoking the following API, which accepts the document as a multi-part attachment in the request:

```
POST tsp/api/v1/apps/{certificateApplicationCode}/validate
```

SPID authentication - Response

```
{
  "code": "8F45ACFE-F075-470D-9E7D-C3F737EB87DE", ①
  "application": {
    "code": "494E729C-A56D-42F5-8A32-D6C3C064D767", ②
    "description": "Paul Reds"
  },
  "qscd": {
    "code": "ABF2EFD4-4681-4E4E-A4FB-DE69A0163649", ③
    "description": "TSP managed QSCD for ephemeral certificates"
  },
  "status": "I", ④
  "issuing_date": "2022-03-10", ⑤
  "expiring_date": "2022-03-10", ⑥
  "csr": "Y2lhbyBjaWFvIGNpYW8K", ⑦
  "crt": "Y2lhbyBjaWFvIGNpYW8gY2ljY2lvCg==", ⑧
  "operationType": "C" ⑨
}
```

① Unique code that identifies the certificate in subsequent calls.

② The `certificateApplicationCode` that identifies the certificate request.

③ The device (QSCD) that manages the certificate's private key.

④ `I` (ISSUED), indicates that the certificate has been issued.
⑤ Date and time when the certificate becomes valid (slightly earlier than the actual issuance time).
⑥ Date and time when the certificate expires.
⑦ Certificate Signing Request (PKCS#10) represented in Base64 format.
⑧ x509 certificate represented in Base64 format.
⑨ `C` (CREATED), indicates that the last operation performed on the certificate was its creation.

The SigningApplication must store the unique code that identifies the certificate, which will be used in subsequent operations.

5.3.3. Document Signature

The signing phase is carried out using the CSC APIs already discussed in previous section and consists of the following 3 steps:

1. OTP Sending Request
2. Authorization Request (with OTP)
3. Hash Signature Request or Document Signature Request