Certigna

Digital trust by Tessi

ACME Technical Document

Innovation&trust

Tessi's digital factory

Document history

Version	Date	Author	Modifications
1.0	2024-07-24	J.Dusautois	Creation
1.1	2024-10-10	J.Dusautois	Add error code
1.2	2024-11-21	J.Dusautois	Add certbot example for challenge DNS
1.3	2025-01-24	J.Dusautois	Precise result code Add comment for certbot usage Add acme.sh sample config
1.4	2025-02-25	J.Dusautois	Precise certificate issuance and dcv validation Add acme client “lego” example
1.5	2025-03-28	J.Dusautois	Fix acme test url

Document validation

Entity	Validator	Date
CERTIGNA	J.DUSAUTOIS	2024-07-24

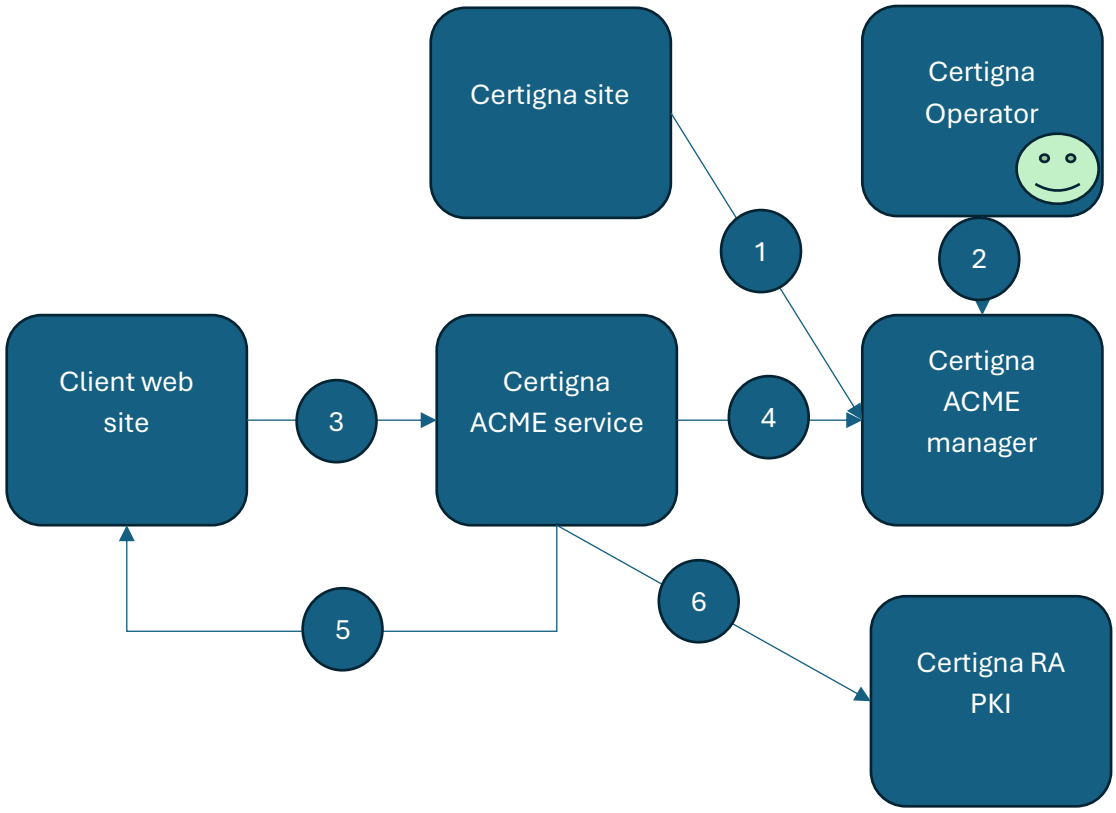
TABLE OF CONTENTS

Introduction

Purpose

This document describes the functionalities, and the implementation of the functions offered by the "ACME" service developed and hosted by CERTIGNA. This service is used to automatically issue TLS certificates.

Schema



1. Account is created
2. Operator valid account
3. Client ACME request certificate with account ID
4. ACME server verify the validity of the request and the validity of the account
5. ACME server verify challenge (http or dns)
6. ACME server request PKI to issue certificate

Usage context

This service complies with the RFC8555 for Automatic Certificate Management Environment. It offers the following features:

- Account binding

- Order management
- Authorization and challenge for DNS and HTTP
- Certificate issuance
- Certificate revocation
- Key change (coming soon)

The service URL are:

- For test: https://acme-ov-test.certigna.com/directory
- For production: https://acme-ov.certigna.com/directory

Prerequisites

To use the ACME service, an ACME account is required. This account must be created on the Certigna website, before a certificate can be issued.

Since Certigna only provides validated certificates for the organization, the account must be activated by the Certigna operator before use. The operator verifies all the mandatory documents to prove that the organization is authorized to request certificates for a specified domain name.

The following checks are carried out before certificate can be requested:

- The organization set in certificates is authorized
- The certificate manager is authorized by their company
- The domain name is under the control of the company
- All necessary documents are provided
- The company has sufficient credit

The following checks are carried out during certificate issuance:

- The requester can prove it manages the domain name by placing a file with dynamic content in a well know directory on his/her server or in DNS entry.
- The DNS entry CAA contains certigna.com or DNS CAA entry does not exist for the requested domain name.

If more than one domain name is requested, the above checks are performed for each domain.

ACME account

The ACME account must be created on the web site. It is identified by an Id and an hmac key. This information is provided by the certificate manager when the account is created.

The id is a UUDI V4 and looks like “6136f1b5-39c8-4417-b8a7-c4114b092be6” and the hmac key looks like “fh2fRV_yZ3NJZyjAQzSlWcPv1z91FEtO2pO62PRgZXk”. Do not use the padding ‘=’ characters if present.

The id and the key are used to bind the account with the acme client tool. After first used, the hmac key is no longer used.

Security

The acme account is protected by the hmac key during account activation. During this step, an RSA key pair is generated by an ACME client and the public part is stored in the ACME server.

All subsequent exchanges with the ACME server will be signed with the private key. The account id and hmac key is no longer used. Public key can be change with the ACME keyChange endpoint.

Revocation

Certificate can be revoked with ACME client. Certificates can also be revoked by the administrator on the web site.

For example, with certbot, the command revoke can be used to revoke a certificate.

Account revocation

If the ACME administrator revoked the ACME account, all not expired certificates issued with the revoked account are also revoked.

Notifications

Multiple notifications can be sent by email to the contact associated with the administrator account. The following emails can be sent:

- Certificate issued: with the list of certificates issued in the last 24 hours.
- Missing documents: with the list of missing documents to validate the account. Without these documents, the account remains PENDING, and no certificate can be issued.
- Expiring documents: with the list and expiration date of each document. When the document expires, if no one replaces it, the account is put on PENDING status and no further certificates can be issued.
- Expired certificates: With the list of certificates near the end of the validity period.

Issuing certificates

To issue certificates, you must use an ACME client.

You must provide the keyId and hamcKey to authenticate the organization.

All domain names you want to put in the certificate must be defined in the domain list. For SSL_RGS certificate, all FQDN must be present in the domain list.

During issuance, the domain is validated (DCV). Two methods are available:

- dns-01: A challenge must be set in TXT entry in the path of the domain to set in certificate prefixing by _acme-challenge (ie _acme-challenge.mondomain.com). You must set one TXT entry per domain in the certificate with the challenges provided by ACME server during the certificate request.
- http-01: A challenge must be set in a file. This file is accessible with an http request on http://**domainname**/ .well-known/acme-challenge/**challengeId**. The domainname is the name you want to set in certificate. ChallengeId is the challenge provided by ACME server during the certificate request.

Result code

If certificate is correctly issued, the response is the certificate and the chain.

If an error occurred, the following code can be returned:

- DEACTIVATED_ACCOUNT: The ACME account is deactivated. You cannot use any more this account.
- DEACTIVATED_EXTERNAL_ACCOUNT: External account is deactivated. You cannot use any ACME account referenced by the deactivated external account.
- DOMAIN_NOT_FOUND: The requested domain is not in the domain list for the AMCE account used. Remember, for SSL_RGS certificate, you must define all FQDN in domain list.
- EXPIRED_ACCOUNT: The ACME account has expired. You must regenerate the key to reusing this account.
- EXPIRED_EXTERNAL_ACCOUNT: The external account referenced by ACME account is expired. You certainly provide new documents to extend the account validity.
- EXTERNAL_ACCOUNT_NOT_FOUND: The external account referenced by ACME account is not found.
- INVALID_DNSCAA: One or more of domain name required in certificate is blocked by CAA entry in DNS. You must specify a CAA entry with the value “issue certigna.com”.
- INVALID_DOMAIN: The requested domain is invalid.
- INVALID_DOMAIN_FOR_WILDCARD_PROFILE: Your ACME account is configured to issue wildcard certificates. It cannot be used to generate a non-wildcard certificate.
- INVALID_PROFILE_FOR_WILDCARD: Your ACME account is configured to issue non-wildcard certificate, and you try to generate wildcard certificates.
- NO_DOCUMENT_FORM_FOUND: A valid form document is missing.
- NO_VALID_ORGANIZATION_DOCUMENT_FOUND: No document is provided for the organization.
- NOT_SAME_ORGANIZATION: In the requested certificate there are two domains managed by two different organizations.
- ORGANIZATION_IS_NOT_VALID: The status of the organization is not valid. Perhaps a document is missing, expired, or invalid.

- OUT_OF_CREDIT: The is not enough credit to generate the certificate.
- PENDING_EXTERNAL_ACCOUNT: The status of the external account is pending. Perhaps a document is missing, expired, or invalid.
- REVOKED_ACCOUNT: The ACME account is revoked. You can no longer issue certificates with this account. Please create a new ACME account.
- REVOKED_EXTERNAL_ACCOUNT: The external account is revoked. You can no longer use the ACME account managed by this account. Please create a new external account and ACME account.
- SUSPENDED_ACCOUNT: The ACME account is suspended. Please reactivate the account to issue certificates.
- SUSPENDED_EXTERNAL_ACCOUNT: The external account is suspended. Please reactivate the external account to be able to use the ACME account.
- UNAUTHORIZED_DOMAIN: The domain name you try to set in certificate is not authorized. Please verify that all documents are provided to authorize the ACME account to issue certificates with this domain name.
- ACME_ORDER_NOT_READ: Internal error when trying to retrieve the certificate order.
- ACME_REQUEST_NOT_FOUND: Internal error when trying to retrieve the certificate request.
- ACME_ORDER_FINALIZE_ERROR: Error during the generation of the certificate.
- ACME_CA_ERROR: Error during the generation of the certificate by the Certificate Authority.
- ACME_OTHER_ERROR: Other internal server error.
- ACME_DEACTIVATED: We receive an end-of-generation provided by le ACME client.

ACME client

ACME Client is the common name for identifying programs and APIs that comply with RFC 8555.

Certbot client

certbot is a tool provided by Letsencrypt to request and install certificates.

The following options must be provided to request a certificate on Certigna ACME server:

- server: The url of the directory end point (ie. https://acme-ov-test.certigna.fr/directory)
- eab-kid: The account id (ie. 6136f1b5-39c8-4417-b8a7-c4114b092be6)
- eab-hmac-key: The hmac key base64 encoded (ie. fh2fRV_yZ3NJZyjAQzSlWcPv1z91FEtO2pO62PRgZXk)
- d: The domain name for which the certificate is requested. You can use this option multiple times to request a certificate with an additional SAN name.
- key-type: rsa (the only supported). Only key length 2048 and 3072 are supported.

There are several other options to set the installation method or the validation method of the challenge.

Sample to use certbot command in challenge http with a certbot http server:

```
certbot certonly --eab-kid 6136f1b5-39c8-4417-b8a7-c4114b092be6 --eab-hmac-key
fh2fRV_yZ3NJZyjAQzSlWcPv1z91FEtO2pO62PRgZXk --server https://acme-ov-
test.certigna.fr/directory --email certificate@certigna.com --standalone --cert-path
/etc/certigna/acme --key-type rsa -d test.certigna.com
```

If you need to use challenge dns, you can use this command:

```
certbot certonly --eab-kid 6136f1b5-39c8-4417-b8a7-c4114b092be6 --eab-hmac-key
fh2fRV_yZ3NJZyjAQzSlWcPv1z91FEtO2pO62PRgZXk --server http://acme-ov-
test.certigna.com/directory --cert-path /etc/certigna/acme --key-type rsa --manual-auth-hook
./createtxt.sh --manual-cleanup-hook ./cleantxt.sh --preferred-challenges dns-01 --manual -d
test.certigna.com
```

Where createtxt.sh and cleantxt.sh are script to manage your dns server. This script can be something like:

```
#! /bin/bash

echo "Create dns TXT"

echo "domain: $CERTBOT_DOMAIN"
echo "challenge: $CERTBOT_VALIDATION"
echo "file: $CERTBOT_TOKEN"
echo "auth: $CERTBOT_AUTH_OUTPUT"
echo "remaining: $CERTBOT_REMAINING_CHALLENGES"
echo "all domains: $CERTBOT_ALL_DOMAINS"

domain=$CERTBOT_DOMAIN
challenge=$CERTBOT_VALIDATION

tmpFile="/tmp/nsupdate.txt"
```

```
echo "server 127.0.0.1" > $tmpFile
echo "zone soapui.web" >> $tmpFile
#echo "update delete _acme-challenge.$domain 10" >> $tmpFile
echo "update add _acme-challenge.$domain 10 TXT $challenge" >> $tmpFile
echo "send" >> $tmpFile
echo "quit" >> $tmpFile
nsupdate -k "/etc/bind/keys" -v "$tmpFile"
```

The challenge and other information are provided by environment variable. CERTBOT_VALIDATION contains the challenge to put in TXT entry on the DNS server.

In my example, I use nsupdate to update the DNS server. You can use other tools or api call using curl to set up the entry in your DNS server.

If you cannot have permission to modify TXT entry in your DNS, you can create a static entry _acme-challenge in your DNS with a CNAME redirecting to another public DNS you managed. In this public DNS you can add TXT entry when script is activated by --manual-auth-hook option.

If you need multiple ACME account in one machine, you must specifiy the account path with the argument --config-dir.

You should not use the same account on different machines unless you copy the account key to the configuration directory.

WACS client

wacs is a tool for Windows. It is an open-source project written in csharp. wacs is also an API.

The following options must be provided to request certificate with wacs:

- baseuri: The service base url (ie https://acme-ov-test.certigna.fr/directory)
- eab-key-identifier: The account id (ie 6136f1b5-39c8-4417-b8a7-c4114b092be6)
- eab-key: The account hmac key base64 encoded (ie fh2fRV_yZ3NJZyjAQzSlWcPv1z91FEtO2pO62PRgZXk)
- host: The domain name for which the certificate is requested

Sample to use wacs command:

wacs --verbose --baseuri https://acme-ov-test.apitest.web/directory --nocache --id test --friendlyname apitest --closeonfinish --eab-key-identifier 6136f1b5-39c8-4417-b8a7-c4114b092be6 --eab-key fh2fRV_yZ3NJZyjAQzSlWcPv1z91FEtO2pO62PRgZXk --commonname test.certigna.com --host test.certigna.com

acme.sh

Avec acme.sh vous devez effectuer deux commande. La première n'est à faire qu'une seule fois.

1. **Account Registration**
   acme.sh --register-account --eab-kid XXXX --eab-hmac-key XXXX --server https://acme-ov-test.certigna.com/directory

2. **Certificat (en mode webroot)**
   acme.sh --issue -d test.monsite.com -w /var/www/default/test-monsite --server https://acme-ov-test.certigna.com/directory \
   --k 3072 # Nécessaire pour utiliser une clé RSA avec une des valeurs suivantes : 2048 ou 3072 \
   --cert-file /etc/ssl/acme/test.monsite.com.pem \
   --key-file /etc/ssl/acme/test.monsite.com.key \
   --fullchain-file /etc/ssl/acme/test.monsite.com.fullchain.pem \
   --reloadcmd "systemctl reload apache2"

lego

lego est un client acme écrit en GO.

Voici un exemple de parameter à fournir pour demander l'émission d'un certificat.

lego --server http://acme-ov-test.certigna.com/directory -d mondomain.com --email monemail@tessi.fr --eab --kid 9cf3b207-2c6e-4ccf-bc82-2d25fef94021 --hmac WHgzMSZnXkVIJ2F1PDdWfEdTcUJ1dSh9KTwlLk9bKCpbODxXOkxiJg --http --http.webroot "/var/www/html " --accept-tos --key-type rsa3072 run

Avec cette commande, une demande de certificat ser aenvoyé à notre server de test. Le dcv sera validé par http.

ACME API

amce4j

This api provides acme capabilities for java.

Sample code

```
{
   final String domainTest = "test.certigna.com";
   // Account identification
   final String accountId = "6136f1b5-39c8-4417-b8a7-c4114b092be6";
```

```
    final byte[] hmacKey = " fh2fRV_yZ3NJZyjAQzSlWcPv1z91FEtO2pO62PRgZXk "

    // Initialize key generator
    final SecureRandom secureRandom = new SecureRandom();
    secureRandom.setSeed(new byte[0]);
    final KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
    keyPairGenerator.initialize(new RSAKeyGenParameterSpec(3072, RSAKeyGenParameterSpec.F4),
secureRandom);

    // Start ACME session
    final Session session = new Session(target(AcmeDirectoryResource.PATH).getUri());

    // login with account identification
    final Login login = new AccountBuilder()//
      .agreeToTermsOfService()//
      .onlyExisting()//
      .useKeyPair(keyPairGenerator.generateKeyPair())//
      .withKeyIdentifier(accountId, new SecretKeySpec(hmacKey, "HMAC"))//
      .createLogin(session);
    final Account account = login.getAccount();


    // Create order with domain name
    final Order order = account.newOrder().domains(domainTest).create();
    Assertions.assertEquals(Status.PENDING, order.getStatus(), "Order status not pending");

    // Query authorization challenges until authorization is valid
    for (final Authorization authorization : order.getAuthorizations())
    {
      if (authorization.getStatus() == Status.VALID)
      {
        // Authorization is valid, certificate can be issuance can be requested
        continue;
      }

      try (final Connection conn = session.connect())
      {
        final JSONBuilder claims = new JSONBuilder();
        final Identifier identifier = new Identifier(Identifier.TYPE_DNS,
authorization.getLocation().toString());
        claims.put("identifier", identifier.toMap());
        conn.sendSignedRequest(authorization.getLocation(), /* claims */null, login);

        final Authorization auth = login.bindAuthorization(authorization.getLocation());

        for (final Challenge challenge : auth.getChallenges())
        {
```

```
          if (challenge.getStatus() == Status.VALID)
          {
            continue;
          }

          switch (challenge.getType())
          {
            case "dns-01":
              final Dns01Challenge dns01Challenge = (Dns01Challenge)challenge;
              // Put here call to function to set dns challenge
              break;
            case "http-01":
              final Http01Challenge http01Challenge = (Http01Challenge)challenge;
              // Put here function to set challenge in wellknown path
              break;
            default:
              break;
          }

          // Send notification to ACME server to check challenge
          challenge.trigger();

          // Wait until challenge is validated by ACMLE server
          while (challenge.getStatus() != Status.VALID)
          {
            Thread.sleep(1_000);
            authorization.update();
          }
        }
      }
    }

    // Generate CSR
    final KeyPair keyPair = KeyPairGenerator.getInstance("RSA").generateKeyPair();
    final ExtensionsGenerator extGen = new ExtensionsGenerator();
    extGen.addExtension(Extension.subjectAlternativeName, false, new GeneralNames(new
GeneralName(GeneralName.dNSName, domainTest)));

    final CertificationRequestInfo requestInfo = new CertificationRequestInfo(//
      new X500NameBuilder().build(), //
      SubjectPublicKeyInfo.getInstance(keyPair.getPublic().getEncoded()), //
      new DERSet(new Attribute(PKCSObjectIdentifiers.pkcs_9_at_extensionRequest, new
DERSet(extGen.generate()))))//
    );

    final Signature signature = Signature.getInstance("SHA256withRSA");
    signature.initSign(keyPair.getPrivate());
```

```
    signature.update(requestInfo.getEncoded(ASN1Encoding.DER));

    final CertificationRequest pkcs10CertificationRequest = new CertificationRequest(//
      requestInfo, //
      new AlgorithmIdentifier(PKCSObjectIdentifiers.sha256WithRSAEncryption,
DERNull.INSTANCE), //
      new DERBitString(signature.sign())//
    );

    // Send csr
    session.networkSettings().setTimeout(Duration.ofMinutes(2));
    order.execute(pkcs10CertificationRequest.getEncoded(ASN1Encoding.DER));

    // Wait certificate issuance
    while (order.getStatus() != Status.VALID)
    {
      Assertions.assertNotEquals(Status.INVALID, order.getStatus());
      Thread.sleep(1_000);
      order.update();
    }


    // download certificate
    Certificate cert = order.getCertificate();
    cert.download();
    X509Certificate certificate = cert.getCertificate();

    return certificate;
  }
```


Certigna
la confiance numérique by Tessi