Docs Menu

ACME Basics

`step-ca` supports the Automated Certificate Management Environment (ACME) protocol. You can get X.509 certificates from your own certificate authority (CA) using popular ACME clients and libraries, or via the `step` command's built-in ACME client.

ACME is a modern, standardized protocol for automatic validation and issuance of X.509 certificates from a CA to clients.

In this document

- Learn about the ACME certificate flow and the most common ACME challenge types.
- Learn how to use an ACME challenge to issue X.509 certificates to endpoints automatically.
- Configure `step-ca` to enable ACME, and get your first certificate via `step ca certificate` .

> If you run into any issues please let us know in Discord or in GitHub Discussions.

Requirements

- **Open source -** This tutorial assumes you have initialized and started up a `step-ca` server (see Getting Started).
- **Smallstep Certificate Manager -** follow the instructions provided in the Certificate Manager ACME documentation.

Overview

Why ACME?

With ACME, machines can get certificates from a CA without any human interaction involved.

For example, you can:

- Use ACME in production to issue X.509 certificates to internal workloads, proxies, queues, databases, etc. so you can use mutual TLS for authentication and encryption.
- Simulate Let's Encrypt's certificate authority in development and pre-production scenarios where connecting to Let's Encrypt's staging server is problematic.

A Typical ACME Flow

Here's how an ACME client might request an X.509 certificate from a CA.

1. The ACME client creates an account with an ACME CA server and submits a certificate order. You can think of an ACME account as a place to store open certificate requests for that particular client.
2. The CA responds with a set of challenges. To complete the challenges, the client must prove it controls each subject name (domain name, IP address, or hardware device ID) that it is requesting in the certificate.
3. The CA verifies the client's challenge responses.
4. Once the client successfully completes the ACME challenges, it submits a certificate signing request (CSR) to the CA.
5. The CA verifies that the client has control of the private key associated with the certificate request.
6. The CA issues a certificate to the client.

The ACME API for certificate requests requires an HTTPS connection between the client and CA.

ACME Challenge Types

There is not a single standard way to prove control over an identifier, so the core ACME specification makes this an extension point. Today, most commonly, ACME clients respond to challenges via HTTP, DNS, or TLS protocols.

Identifier Type	`http-01`	`dns-01`	`tls-alpn-01`	`device-attest-01`
IP address	✔	**X**	✔	**X**
Hostname	✔	✔	✔	**X**
Device ID	**X**	**X**	**X**	✔

A Device ID may be a device's serial number, or another identifier assigned at manufacture.

THE HTTP CHALLENGE (HTTP-01)

The ACME CA challenges the client to host a random number at a random URL under `/.well-known/acme-challenge` on port 80. The CA verifies client control by issuing an HTTP GET request to that URL.

When To Use It

This is a good general-purpose challenge type. By hosting the challenge response via HTTP on port 80, the client proves its control over a protected port on the domain being requested. The `http-01` challenge type is the easiest to set up, because any web server will let you host the challenge response as a static file.

THE DNS CHALLENGE (DNS-01)

The ACME CA challenges the client to provision a random DNS `TXT` record for the domain in question. It verifies the challenge by querying DNS for that `TXT` record.

When To Use It

The `dns-01` challenge type is good if your ACME server cannot reach the requested domain directly. The server only needs to be able to perform a DNS lookup to confirm the challenge. However, because the ACME client needs to modify DNS records, configuring a `dns-01` client is usually more involved.

THE TLS ALPN CHALLENGE (TLS-ALPN-01)

The ACME CA uses TLS to validate a challenge, leveraging application layer protocol negotiation (ALPN) in the TLS handshake. The client presents a self-signed TLS certificate containing the challenge response as a special X.509 certificate extension.

When To Use It

This challenge type is useful when a security policy requires the CA to reach the client via a TLS connection. This is a popular challenge type in cases where an ingress controller fronts clients attempting to receive a certificate.

THE DEVICE ATTESTATION CHALLENGE (DEVICE-ATTEST-01)

The device attestation challenge (`device-attest-01`) is designed to issue client certificates bound to a device identifier.

It allows clients with an attached security module (TPM, Secure Enclave, Yubikey, etc) to request a certificate bound to the security module's permanent hardware identifier. It can also be used to attest the hardware protection of a private key in the security module.

Further reading:

- Managed Device Attestation: ACME as the Bottom Turtle in Mobile Device Management
- ACME Device Attestation Explained focuses Device Attestation for MDM certificate enrollment on Apple devices.

Configure `step-ca` for ACME

Let's get an X.509 certificate via the ACME `http-01` challenge, using `step-ca` .

First, add an ACME provisioner to your `step-ca` configuration to enable ACME support:

```
step ca provisioner add acme --type ACME
```

You may need to restart `step-ca` to pick up the configuration changes.

Finally, use `step` as an ACME client to request a certificate. The client will start a web server on port 80 to respond to the ACME challenge.

```
step ca certificate --provisioner acme example.com example.crt example
```

Output:

```
✔ Provisioner: acme (ACME)
Using Standalone Mode HTTP challenge to validate example.com . done!
Waiting for Order to be 'ready' for finalization .. done!
Finalizing Order .. done!
✔ Certificate: example.crt
✔ Private Key: example.key
```

Next Steps

- Start tailoring `step-ca` to your infrastructure. See Provisioners, Production Considerations, and our Configuration Guide.
- Tutorial: Learn how to configure the most popular ACME clients to connect to a `step-ca` server.

Try for free

Register for demo

Products

Certificate Manager

Smallstep SSH

ACME Registration Authority

Integrations

Documentation

Certificate Manager

Smallstep SSH

step-ca

Tutorials

Step command reference

Open Source

step-ca

Step CLI

About

About

Support

Status

Careers

Privacy

Terms of use

Privacy Policy

Privacy Center

Security