FIELD NOTES FROM A PRESALES SOLUTIONS ARCHITECT

The 200-Day Certificate Rule — Automating a Paid Wildcard With ACME, EAB and a TSIG Key That Can Touch One Record

A field story about a paid wildcard certificate that now has to be reinstalled twice a year — the deadline almost nobody has put in a calendar — and the ACME design I now propose before the next one is due.

Daanial Farrukh · 29 November 2026 · https://quabin.com/wildcard-certificate-acme-eab-dns-01-tsig/

The email came from the certificate authority, not from the customer.

"Your certificate must be reissued and reinstalled before the date below."

The customer had bought a one-year wildcard certificate through us in the spring. Their own team runs the web server. We run their DNS.

I forwarded the notice and asked for a short call.

"It's a one-year certificate," their admin said. "We installed it in April. Why do we need to touch it now?"

I asked one question: "How many times do you think you will install it this year?"

"Once. It's a one-year certificate."

"Twice," I said. "And next year it will be more."

Silence on the line.

That's the trap. **The order is still one year. The certificate inside it is not.** You pay once, and you install again and again. The order date gives you a false sense of safety, because nothing in the order tells you the certificate itself will expire long before it.

What Changed — the Validity Limit, Not the Product

The CA/Browser Forum, the body whose rules every public certificate authority follows, voted to shorten the maximum lifetime of public TLS certificates in steps:

- From March 2026: about 200 days (this CA's notice said 199).
- From March 2027: 100 days.
- From March 2029: 47 days.

A "one-year" or "two-year" product still exists. It is now a subscription. Inside it, the CA reissues the certificate as many times as the rules require, and somebody has to install each one.

For a customer who installs certificates by hand, the arithmetic is simple:

- Today: 2 installs a year.
- 2027: about 4 installs a year.
- 2029: about 8 installs a year.

Every one of those is a chance to miss a date, pick the wrong file, or break the site at 4 p.m. on a Thursday.

Where this hits hardest:

- Paid wildcard and OV certificates bought for a year, installed by hand on Windows IIS or an appliance.
- Split responsibility: one party sells the certificate, another runs the server, a third runs DNS.
- Customers who renewed in the spring and have not received the CA's reissue notice yet.

The Manual Reissue — and the Windows Trap Inside It

We did this one by hand, because the deadline did not leave room for anything else. It is worth showing, because it is the same work the customer would repeat every few months.

The reissue reused the original CSR from April, so the new certificate carries the same public key as the old one. That is allowed, and it saves a step. On Windows it also creates a trap.

The usual way to finish a certificate on IIS is to accept the response file:

```
certreq -accept STAR_example_com.crt
```

It failed with 0x80092004 — "Cannot find object or property".

Here's why. certreq -accept looks for the **pending request** that was created on this server, so it can pair the certificate with the private key it generated. The April request had already been completed in April. There was no pending request left to match. The key was on the server; Windows just had no record linking this new file to it.

The fix is to import the certificate into the machine store and then ask Windows to find its private key:

```
certutil -addstore My STAR_example_com.crt

certutil -repairstore My <THUMBPRINT>
```

-repairstore searches the key store for a private key that matches the certificate's public key and links them. When it works, it prints Encryption test passed, and the certificate shows "You have a private key that corresponds to this certificate" in the MMC.

Then bind it in IIS Manager › Sites › your site › Bindings › https, for every hostname that uses SNI, and check from outside:

```
openssl s_client -connect example.com:443 -servername example.com </dev/null | openssl x509 -
noout -subject -enddate
```

The end date must be the new one. If it still shows the old date, a binding was missed — usually www or a second site on the same IP.

It worked. But the old certificate had expired a few days before the new one was live, and this is the exact work that now repeats every 200, then 100, then 47 days.

Half an hour of careful work, twice a year, is not a process. It is a risk with a calendar.

Why "We'll Just Set a Reminder" Misses It

Every customer's first answer is a reminder. It fails for three reasons:

1. The reminder lives with one person. People change roles; the certificate does not.
2. The CA's notice goes to whoever bought the certificate, not to whoever installs it. In this case it reached us, the provider, not the customer's web team.
3. The install itself is the risky part, not remembering it. The Windows key-pairing trap above happens on the day, under time pressure.

A wildcard adds one more constraint. A wildcard name like *.example.com can only be validated through DNS. HTTP validation is not allowed for wildcards. So any automation must be able to write a DNS record — and DNS here belongs to a different party than the web server.

That is the real design problem. Not "how do I renew", but "how do I let the customer's server prove control of the domain without giving it control of the domain".

The Design — ACME, EAB and a Key That Can Change One Record

Many commercial CAs, Sectigo and DigiCert among them, now offer ACME — the same protocol Let's Encrypt uses — for paid certificates. The difference is **External Account Binding (EAB)**: the CA gives you a key ID and an HMAC key that tie your ACME client to your paid account, so the certificates it issues are billed to your order.

The flow I proposed:

Customer server (ACME client) → CA ACME endpoint (EAB) CA → asks for TXT _acme-challenge.example.com Customer server → nsupdate with TSIG key → provider's BIND primary CA → reads TXT → issues certificate → client installs and reloads

Three things move between the parties, once, by a secure channel:

- From the provider: the CA's ACME directory URL, the EAB key ID and HMAC key.
- From the provider: a TSIG key for dynamic DNS updates.
- From the customer: nothing, except an outbound path to DNS and to the CA.

The important part is the second item. **The TSIG key is allowed to change exactly one name and one record type.** If the customer's server is ever compromised, the attacker can write a TXT record under _acme-challenge. They cannot change the A record, the MX record, or anything else in the zone.

Status, so you know what you are reading: the manual reissue above is done and verified. The ACME design below is what I handed over; it is not yet running in production, and step 1 is the check that

decides whether it can.

How to Actually Set This Up

Step 1: Confirm Your CA Account Really Offers ACME for This Product

Before any DNS work, log in to the account the certificate was bought through and look for ACME or EAB credentials on the order. Some CAs offer ACME only on certain products or only through certain reseller portals.

If there is no ACME option, stop here. The rest of the design does not help, and the honest answer for the customer is a manual calendar until the product changes — or a move to a product that has ACME.

Step 2: Create a TSIG Key on the BIND Primary

On the primary name server (BIND 9.18 in this case):

```
tsig-keygen -a hmac-sha256 example-acme > /etc/bind/example-acme.key
```

Include it from named.conf.local:

```
include "/etc/bind/example-acme.key";
```

Step 3: Scope the Key to One TXT Record

In the zone block, use update-policy instead of allow-update. allow-update lets the key change anything in the zone. update-policy lets you name exactly what it may change:

```
zone "example.com" {
    type primary;
    file "/var/lib/bind/db.example.com";
    update-policy { grant example-acme name _acme-challenge.example.com. TXT; };
};
```

Two details that trip people up:

- **The zone file must live under /var/lib/bind.** A dynamic zone writes a journal file (.jnl) next to the zone file. On Ubuntu, AppArmor lets named write in /var/lib/bind but not in /etc/bind. Leave the file in /etc/bind and the first update fails with a permission error.
- **Hand edits now need a freeze.** Once a zone is dynamic, BIND owns the file. To edit it by hand, run rndc freeze example.com, edit, bump the serial, then rndc thaw example.com.

Check and reload:

```
named-checkconf

rndc reload
```

Step 4: Prove the Key Works — and Prove It Cannot Do More

This step is where most tutorials stop and where mine starts.

From the customer's server, write a test record:

```
nsupdate -k example-acme.key
> server <NS_PRIMARY_IP>
```

```
> update add _acme-challenge.example.com. 60 TXT "test"
> send
```

Then read it back from the name server:

```
dig +short TXT _acme-challenge.example.com @<NS_PRIMARY_IP>
```

Expected output: "test".

Now try to change something the key must not touch:

```
nsupdate -k example-acme.key
> server <NS_PRIMARY_IP>
> update add www.example.com. 60 A 203.0.113.10
> send
```

Expected output: update failed: REFUSED.

This error is not a bug. It is the verification. If that second update succeeds, the policy is too wide — fix it before the key leaves your hands. Delete the test TXT record afterwards with update delete.

The customer's server needs UDP and TCP 53 to the primary, and TCP 443 out to the CA. Test both before the first real issue.

Step 5: Register the ACME Client With EAB

Any ACME client that supports EAB and RFC 2136 dynamic updates fits here. I use acme.sh as the example:

```
acme.sh --register-account --server <ACME_DIRECTORY_URL> --eab-kid <EAB_KID> --eab-hmac-key
<EAB_HMAC>
```

Step 6: Issue Through DNS-01

acme.sh's dns_nsupdate hook reads its settings from the environment:

```
export NSUPDATE_SERVER="<NS_PRIMARY_IP>"
export NSUPDATE_KEY="/root/example-acme.key"
export NSUPDATE_ZONE="example.com"

acme.sh --issue --server <ACME_DIRECTORY_URL> --dns dns_nsupdate -d example.com -d '*.example.
com'
```

The client writes the TXT record, the CA reads it, the client removes it. Nothing else in the zone moves.

Step 7: Install With a Reload Command

```
acme.sh --install-cert -d example.com --key-file /etc/ssl/private/example.com.key --fullchain-
file /etc/ssl/certs/example.com.pem --reloadcmd "systemctl reload nginx"
```

acme.sh remembers this, so every future renewal installs and reloads on its own. Then run the same outside check as before:

```
openssl s_client -connect example.com:443 -servername example.com </dev/null | openssl x509 -
noout -issuer -enddate
```

The issuer must be your paid CA, not a free one. That proves the EAB binding was used.

When You Have to Redo It

Automation removes the routine install. It does not remove every human step. Plan for these:

1. **Order renewal.** When the yearly order is renewed, expect new EAB credentials. That is my working assumption, not yet confirmed with this CA; check it on the first renewal.
2. **Server rebuild.** The TSIG key, the ACME account and the install path live on the server. Rebuild it and you repeat steps 5 to 7.
3. **Web server change.** A new path or a different web server means a new --install-cert line.
4. **Key rotation.** If the TSIG key may have leaked, generate a new one, swap it in BIND, and send the new key the same secure way.

The Checklist I Use

1. Order and certificate are separate dates. openssl x509 -noout -enddate -in cert.pem gives the one that matters.
2. Wildcard means DNS-01. Ask who runs DNS before you promise automation.
3. ACME on the paid account exists. Check the order page for EAB before designing anything.
4. TSIG key uses update-policy, not allow-update. Test that an A-record update is refused.
5. Dynamic zone file sits under /var/lib/bind. ls /var/lib/bind shows the .jnl after the first update.
6. Outbound 53 (UDP and TCP) and 443 work from the web server. dig @<NS_PRIMARY_IP> example.com SOA must answer.
7. A reload command is stored. The certificate on port 443 changes date after a forced renewal (acme.sh --renew -d example.com --force).

What I Tell Every Customer Now

When a customer buys a certificate for a year, I ask one question in the same meeting:

"Who installs the second one, and how?"

If the answer is "our ACME client, through a DNS key that can only write the challenge record", they're in good shape. If the answer is "we'll set a reminder", we're not done yet — because in 2029 that reminder rings eight times a year.

I write about the questions I ask in real customer rooms — the ones that change architectures and surface the problems no one else noticed. If you found this useful, drop your email below and I'll send the next one when it's ready.

FAQs

Why do I have to reinstall a one-year SSL certificate after six months?

Because the CA/Browser Forum limited public certificates to about 200 days from March 2026. The order still runs for a year, but the CA must reissue the certificate inside it, and the new file has to be installed on your server. The limit drops to 100 days in March 2027 and 47 days in March 2029.

What is ACME External Account Binding (EAB)?

EAB is how a commercial CA links an ACME client to a paid account. The CA gives you a key ID and an HMAC key, you register your ACME client with them once, and every certificate the client then requests is issued under your paid order instead of a free one.

Can a wildcard certificate be validated over HTTP?

No. A wildcard name such as *.example.com can only be validated with DNS-01, where the CA checks a TXT record under _acme-challenge. That is why wildcard automation always needs some way to write to DNS.

How do I stop a TSIG key from changing my whole DNS zone?

Use update-policy with a grant <key> name _acme-challenge.example.com. TXT; rule instead of allow-update. Then test it: an nsupdate that adds an A record with that key must return REFUSED. If it succeeds, the policy is too wide.

Why does certreq -accept fail with 0x80092004 after a certificate reissue?

certreq -accept looks for a pending request on the server to pair the certificate with its private key. If the reissue reused an old CSR whose request was already completed, there is nothing pending. Import the file with certutil -addstore My and link the key with certutil -repairstore My <thumbprint>.

Read online: https://quabin.com/wildcard-certificate-acme-eab-dns-01-tsig/