draft-ietf-acme-acme

IETF 97

Closed since last IETF meeting

We did some stuff!

38 pull requests merged since IETF 94, with **9** different authors!

Highlights...

Typos and Clarifications

#160	#163	#166	#169	#171
#173	#174	#175	#176	#178
#179	#180	#183	#184	#185
#186	#188	#192	#196	#197
#198	#200	#201	#202	#206

Developer Friendliness

#194 - Default to PEM with chain for certificates.

#168 - Add SHOULDs for User-Agent and Accept-Language.

#203 - Hard fail on invalid contacts

Terms of Service

#182 - Clarify flows around agreement to terms
#167 - Simplify terms-of-service flow.

- Server specifies ToS in directory
- Client sends “terms-of-service-agreed”: true
- If server changes ToS in a way that requires re-agreement, urn:ietf:params:acme:error:agreementRequired
- **Optimizes for the case where CAs do not require re-agreement, even when ToS changes**

Key Roll-over

#164 - Unparallelize signatures on key-change

#199 - Remove redundant oldKey field.

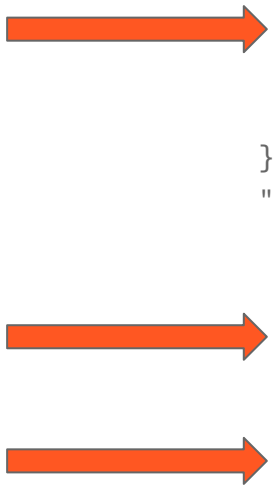
#189 - Fix signing order for key-change endpoint.

#187 - Remove spurious thumbprints para in key-change.

Outer JWS signed with old key

Inner JWS signed with new key

In payload:
Old key by reference
New key by value



```
POST /acme/key-change HTTP/1.1
Host: example.com
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "jwk": /* old key */,
    "nonce": "K60BWPrMQG9SDxBDS_xtSw",
    "url": "https://example.com/acme/key-change"
  }),
  "payload": base64url({
    "protected": base64url({
      "alg": "ES256",
      "jwk": /* new key */,
    }),
    "payload": base64url({
      "account": "https://example.com/acme/reg/asdf",
      "newKey": /* new key */
    })
    "signature": "Xe8B94RD30Azj2ea...8BmZIRtcSKPSd8gU"
  }),
  "signature": "5TWiqIYQfIDfALQv...x9C2mg8JGPxl5bI4"
}
```

Bigger Things

#165 - Re-add new-authz as pre-authorization

#181 - Add a new-nonce endpoint

#191 - Standardize on "proactive" certificate issuance

#193 - Specify account by kid (reg URL) rather than key.

Open issues

Account Management

#170 - Add a special token parameter in registration

#172 - Add an external secret field to registration.

(list) - Adding arbitrary CA-specific name-value pairs to registration object

- CAs have non-ACME customer accounts
- Want to associate those accounts with ACME accounts
- Current PR ------------------------------>
- Might need some security considerations
- E.g., recommending “counter-signing” to avoid UKS-like risks

```
POST /acme/new-reg HTTP/1.1

{
  "protected": "..."
  "payload": base64url({
    "terms-of-service-agreed": true,
    "external_secret": "Ld6G0vvDaF...7G2bkcnQ",
    "contact": [...]
  }),
  "signature": "..."
}
```

Additional CA Account Data

(list) - Adding arbitrary CA-specific name-value pairs to new-reg object

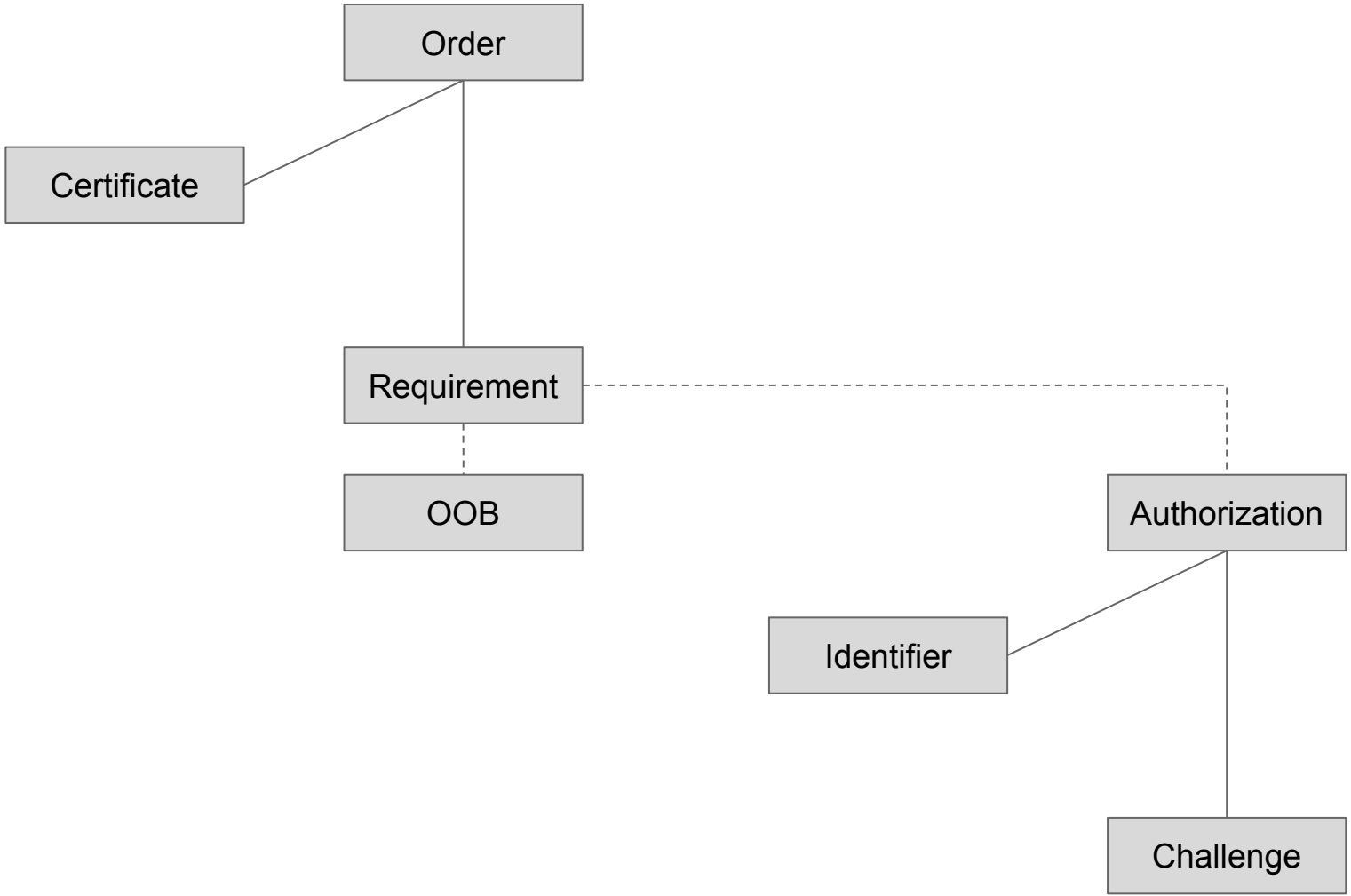
- CAs might want to collect additional data from subscribers
- Do we need to reflect that in ACME?
- If so, where?
  - Special slot, e.g., “ca-extension”
  - Just dump it in at the top level
- Does we need slots in new-reg?
- … or do these fields also get reflected in the registration object?
- Do we need a way for the CA to tell the client that certain extensions are needed?

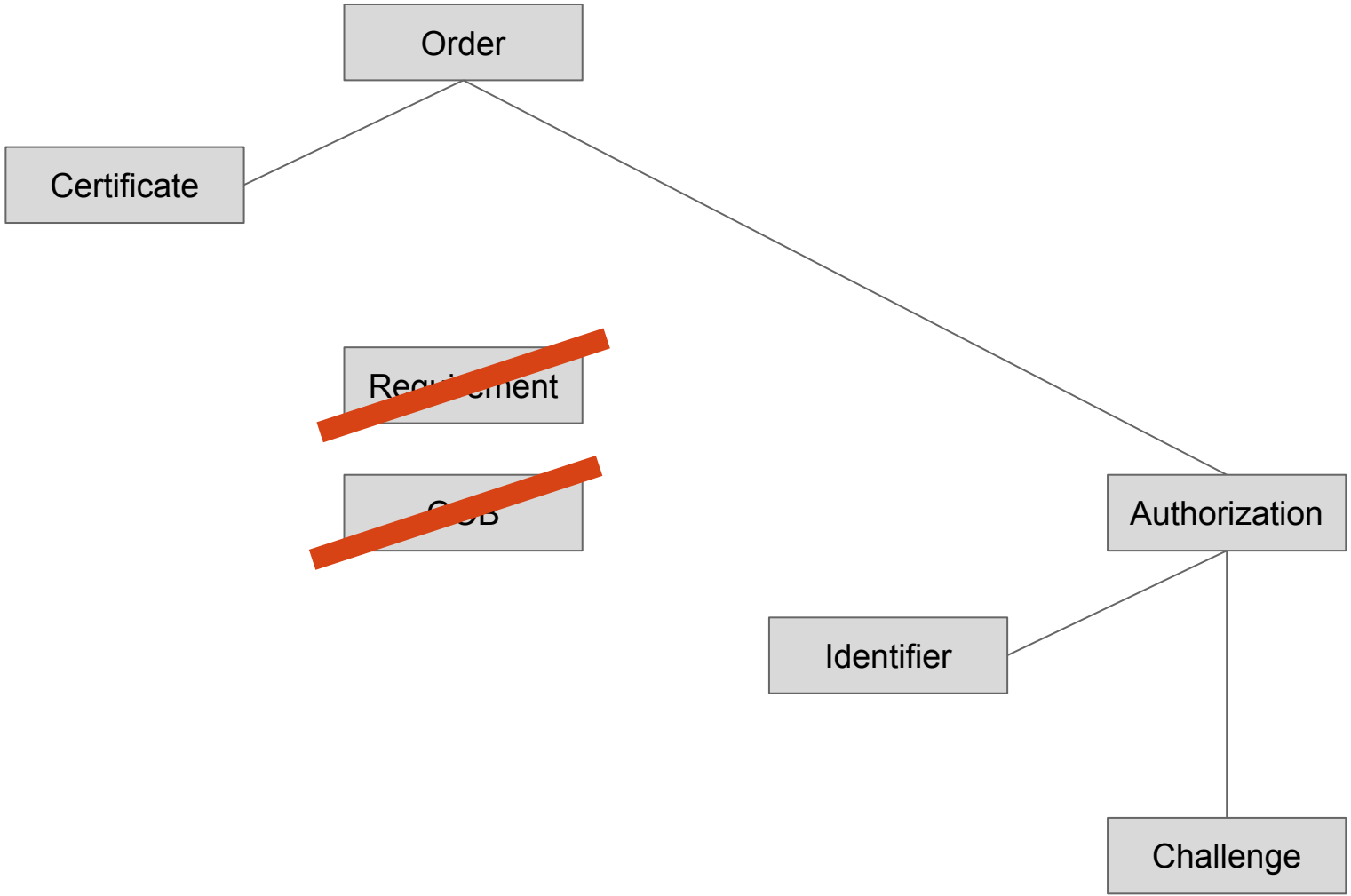
```
POST /acme/new-reg HTTP/1.1

{
  "protected": "..."
  "payload": base64url({
    "terms-of-service-agreed": true,
    "ca-extension": {
      "<ca-ext-name-1>": "<ca-ext-value-1>",
      "<ca-ext-name-2>": "<ca-ext-value-2>"
    },
    "contact": [...]
  }),
  "signature": "..."
}
```

Object Model

#195 - Combine "requirements" and "authorizations."

- Goals:
  - Avoid unnecessary duplication of “status” fields and layers of indirection
  - Keep the authorization flow as a reusable concept for different identifiers
- Proposal:
  - Remove the “requirement” abstraction
  - In particular remove the “out-of-band” requirement type
  - Represent authorization dependencies directly in the order object (unless “valid”)


Order
Certificate
Requirement
OOB
Authorization
Identifier
Challenge


Order
Certificate
Authorization
Identifier
Challenge

Before:

```
{
  "status": "pending",
  "expires": "2015-03-01T14:09:00Z",
  "csr": "jcRf4uXra7F...rhlnznwy8Ybp",

  "requirements": [
    {
      "type": "authorization",
      "status": "valid",
      "url": "https://example.com/acme/..."
    },
    {
      "type": "out-of-band",
      "status": "pending",
      "url": "https://example.com/acme/..."
    }
  ]
}
```

After:

```
{
  "status": "pending",
  "expires": "2015-03-01T14:09:00Z",
  "csr": "jcRf4uXra7F...rhlnznwy8Ybp",

  "authorizations": [{
      "status": "pending",
      "url": "https://example.com/acme/...",
      "identifier": {...},
      "challenges": [...]
    }, {
      "status": "valid",
      "url": "https://example.com/acme/..."
    }]
  ]
}
```


shed

Terminology

Before	After
Application	Order
Registration	Account

Objections? Other bids?

Where do we go from here?

To WGLC or not to WGLC

Want to get this work done as soon as possible … and no sooner

Heard some requests to get some implementation experience before we finalize

Propose following the model TLS is doing right now:

- Have a WGLC and declare the spec basically stable
- Give some time to do implementation / interop / analysis
- See what we've learned; fix bugs; ship