Network Working Group
Internet-Draft
Updates: 7030 (if approved)
Intended status: Standards Track
Expires: 5 November 2026

L. Liao
NIO

4 May 2026

EST Lightweight Operations
draft-liao-lamps-est-lightweight-operations-01

Abstract

This document defines eight new operations for the Enrollment over Secure Transport (EST) protocol specified in RFC 7030: ucacaps, ucacert, ucacerts, ucrlinfo, ucrl, usimpleenroll, usimplereenroll, and userverkeygen. These operations deliver PKI objects, including CA certificates, certificate chains, CRLs, and enrolled certificates, as Base64-encoded DER or PEM, without the CMS encapsulation used by the corresponding EST operations in RFC 7030. The ucacaps operation enables EST clients to discover which operations the EST server supports. Eliminating the CMS wrapper for these responses reduces code size and parsing complexity.

This document updates RFC 7030.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 November 2026.

Copyright Notice

Table of Contents

1. Introduction

Enrollment over Secure Transport (EST) [RFC7030] defines a set of HTTPS-based operations for certificate enrollment and management. An EST server sits between a Certification Authority (CA) and an EST client and performs Registration Authority (RA) functions, as described in [RFC7030], Section 1. [RFC8951] updates RFC 7030 to clarify EST transfer-encoding behavior and ASN.1 details. [RFC9908] further updates RFC 7030 and RFC 9148 to clarify and enhance the csrattrs definition.

The responses to the standard EST operations (cacerts, simpleenroll, simplereenroll, and serverkeygen) are encapsulated in Cryptographic Message Syntax (CMS) structures [RFC5652], specifically using the

application/pkcs7-mime; smime-type=certs-only content type. While CMS provides a well-established container format, parsing it requires code that may be unnecessary or unavailable in some EST deployments [RFC7228].

EST-coaps [RFC9148] addresses EST over CoAP and DTLS by defining shorter resource paths, using CoAP Block-Wise Transfer for fragmentation, and allowing binary ASN.1 payloads. It also defines Content-Format 287 (application/pkix-cert) as a non-CMS single-certificate response format that can be selected using CoAP content negotiation. However, RFC 9148 primarily specifies EST semantics for CoAP-based networks and does not define the HTTPS operations or the CRL metadata and retrieval operations defined in this document. The relationship with RFC 9148 is discussed in Appendix D.

This document also addresses two functional gaps. First, [RFC7030] provides no capability-discovery operation for these new operations, so an EST client would otherwise need to rely on out-of-band configuration or probing. The ucacaps operation enables the client to discover server support explicitly. Second, [RFC7030] does not define in-band CRL metadata or retrieval operations; revocation information is normally obtained from CRL Distribution Point (CDP) URIs carried in certificates. The ucrlinfo operation supports lightweight freshness checks, and the ucrl operation provides an EST-based retrieval path when the full CRL is needed.

This document adds eight new operations to [RFC7030], Section 4 (Protocol Exchange Details), as summarized in Table 1:

New Operation	Based on RFC 7030 Operation	Auth	Response Format
ucacaps	_(new)_	No	text/plain keyword list
ucacert	_(new)_	No	application/pkix-cert (single CA cert)
ucacerts	cacerts ([RFC7030], Section 4.1)	No	application/pem-certificate-chain
ucrlinfo	_(new)_	No	text/plain (CRL metadata)
ucrl	_(new)_	No	application/pkix-crl
usimpleenroll	simpleenroll ([RFC7030], Section 4.2.1)	Yes	application/pkix-cert
usimplereenroll	simplereenroll ([RFC7030], Section 4.2.2)	Yes	application/pkix-cert
userverkeygen	serverkeygen ([RFC7030], Section 4.4)	Yes	application/x-pem-file (key + cert)

Table 1: New Operations Defined in This Document

These eight operations fall into four categories: capability discovery (ucacaps), CA certificate retrieval (ucacert, ucacerts), CRL metadata and retrieval (ucrlinfo, ucrl), and certificate enrollment and server-side key generation (usimpleenroll, usimplereenroll, userverkeygen).

The "u" prefix is used consistently across all eight new operations. For the six operations whose responses carry DER or PEM payloads (ucacert, ucacerts, ucrl, usimpleenroll, usimplereenroll, userverkeygen), it denotes "unencapsulated": their responses carry DER or PEM payloads without a CMS wrapper. For ucacaps (capability discovery) and ucrlinfo (CRL metadata), there is no CMS equivalent; the "u" prefix is used solely to maintain a consistent naming convention with the other operations in this document.

All security requirements imposed by [RFC7030] on the corresponding RFC 7030 operations (cacerts, simpleenroll, simplereenroll, serverkeygen) apply equally to the corresponding operations defined in this document. Refer to Section 8 for security considerations specific to these new operations.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms from [RFC7030] are used in this document:

EST client: The entity that contacts the EST server to obtain certificates or CA information, as defined in [RFC7030], Section 1.

EST server: The entity that processes EST requests, typically acting as an RA between the EST client and the CA, as defined in [RFC7030], Section 1.

CA: Certification Authority. The entity that issues X.509 certificates [RFC5280].

CSR: Certificate Signing Request. A PKCS#10 [RFC2986] message used by an EST client to request certificate issuance.

PoP: Proof of Possession. Verification that the requester holds the private key corresponding to the public key in the CSR, as described in [RFC7030], Section 3.4.

3. EST URL Structure and Path Components

The operations in this document follow the same URI path structure defined in [RFC7030], Section 3.2.2. Retrieval operations (ucacaps, ucacert, ucacerts, ucrlinfo, ucrl) use HTTP GET:

```
GET /.well-known/est/<operation>
GET /.well-known/est/<label>/<operation>
```

Enrollment operations (usimpleenroll, usimplereenroll, userverkeygen) use HTTP POST:

```
POST /.well-known/est/<operation>
POST /.well-known/est/<label>/<operation>
```

An EST server MUST return HTTP 405 (Method Not Allowed) if a client uses POST for a retrieval operation or GET for an enrollment operation.

In the URI templates above, <operation> is one of: ucacaps, ucacert, ucacerts, ucrlinfo, ucrl, usimpleenroll, usimplereenroll, or userverkeygen.

The optional <label> path segment follows [RFC7030], Section 3.2.2: an EST server MAY use an additional path segment before the operation name to distinguish services for multiple CAs or certificate profiles. The <label> MUST NOT be the same as any defined operation name. The internal structure and interpretation of <label> are outside the scope of this document; for example, it may be a single opaque token that identifies a CA and certificate profile, or a slash-separated pair of the form <ca-name>/<cert-profile>.

3.1. Base64 Transfer

[RFC8951] deprecates use of Content-Transfer-Encoding for EST. The new operations defined in this document follow that guidance.

For operations that return DER payloads (application/pkix-cert, application/pkix-crl), the response body MUST be the Base64 encoding defined in [RFC4648] of the DER object. Servers MUST NOT include a Content-Transfer-Encoding header.

For POST operations that submit a PKCS#10 CSR, the request body MUST be the Base64 encoding defined in [RFC4648] of the DER-encoded PKCS#10 object. Clients and servers implementing this document MUST NOT require or generate a Content-Transfer-Encoding header for these operations.

The ucacaps, ucrlinfo, ucacerts, and userverkeygen operations already return textual payloads and therefore do not require any additional transfer-encoding mechanism.

The ucacert, ucacerts, and ucrl retrieval operations support HTTP caching. EST servers SHOULD provide caching metadata on successful authoritative responses so that clients and HTTP intermediaries can reuse them. EST clients SHOULD cache authoritative responses locally and refresh them according to the advertised freshness information. Authoritative responses for these operations MUST NOT include Pragma: no-cache, Cache-Control: no-cache, or Cache-Control: no-store.

4. EST Server Capability Discovery

EST [RFC7030] does not provide a mechanism for EST clients to discover which operations an EST server supports before invoking them. This section defines the ucacaps operation for that purpose.

EST clients SHOULD issue a ucacaps request before using any of the other operations defined in this document, to confirm that the EST server supports the intended operation.

4.1. ucacaps

4.1.1. Request

The EST client requests the server's capability list using the HTTP GET method. No request body is sent. The EST server SHOULD NOT require client authentication for this operation, consistent with [RFC7030], Section 4.1.2.

GET /.well-known/est/<label>/ucacaps

4.1.2. Response

On success, the EST server returns HTTP status 200 with:

* Content-Type: text/plain

* Body: A plain-text list of capability keywords, one keyword per line. Lines MUST be terminated by <CR><LF> or <LF>. Keywords are unquoted and case-insensitive; EST clients MUST match them in a case-insensitive manner.

The EST server SHOULD use the canonical lowercase form defined in Section 4.1.4. The EST server MAY advertise additional, implementation-specific keywords not defined in this document. EST clients MUST ignore any unknown keywords.

If the EST server supports none of the capabilities listed in Section 4.1.4, it SHOULD return an empty body. An EST client that receives an empty body SHOULD interpret the response as if none of the capabilities are supported. An EST client that receives an HTTP error response to a ucacaps request SHOULD assume that only the operations defined in [RFC7030] are supported and MUST NOT attempt the operations defined in this document.

4.1.3. Choice of Capability-List Format

The ucacaps response uses text/plain rather than JSON or CBOR because the information being conveyed is intentionally limited to an unordered set of capability keywords. A line-oriented keyword list can be parsed with minimal code, can be inspected manually, and does not require a JSON, CBOR, or CMS parser. This is consistent with the purpose of this document: reducing parsing requirements in EST deployments that do not implement CMS.

JSON and CBOR are both suitable formats when a protocol needs structured values, nested data, typed fields, or extensible objects. The ucacaps response does not require those properties. Each advertised capability is a single token, unknown tokens are ignored, and absence of a token has a simple meaning. Adding a structured encoding for this operation would therefore increase implementation requirements without adding protocol semantics.

Future specifications that need to advertise parameters associated with a capability, such as algorithm lists, profile identifiers, size limits, or transport-specific options, can define additional capability keywords or a separate structured discovery operation. Such an extension would be preferable to overloading the simple ucacaps keyword list with structured data.

4.1.4. Capability Keywords

The following keywords are defined. An EST server MUST include a keyword in the ucacaps response if and only if it supports the corresponding operation.

RFC 7030 operations (see also Table 2):

Keyword	Description
cacerts	EST server supports cacerts ([RFC7030], Section 4.1)
simpleenroll	EST server supports simpleenroll ([RFC7030], Section 4.2.1)
simplereenroll	EST server supports simplereenroll ([RFC7030], Section 4.2.2)
fullcmc	EST server supports fullcmc ([RFC7030], Section 4.3)
serverkeygen	EST server supports serverkeygen ([RFC7030], Section 4.4)
csrattrs	EST server supports csrattrs ([RFC7030], Section 4.5, as updated by [RFC9908])

Table 2: ucacaps Keywords for RFC 7030 Operations

RFC 8295 operations (see also Table 3):

Keyword	Description
pal	EST server supports pal ([RFC8295], Section 2)
eecerts	EST server supports eecerts ([RFC8295], Section 3)
crls	EST server supports crls ([RFC8295], Section 4)
symmetrickeys	EST server supports symmetrickeys ([RFC8295], Section 5.1)
symmetrickeys/return	EST server supports symmetrickeys/return ([RFC8295], Section 5.2)
firmware	EST server supports firmware ([RFC8295], Section 6.1)
firmware/return	EST server supports firmware/return ([RFC8295], Section 6.2)
tamp	EST server supports tamp ([RFC8295], Section 7.1)
tamp/return	EST server supports tamp/return ([RFC8295], Section 7.2)
serverkeygen/return	EST server supports serverkeygen/return ([RFC8295], Section 8.2)

Table 3: ucacaps Keywords for RFC 8295 Operations

Operations defined in this document (see also Table 4):

Keyword	Description
ucacert	EST server supports ucacert (Section 5.1)
ucacerts	EST server supports ucacerts (Section 5.2)
ucrlinfo	EST server supports ucrlinfo (Section 6.1)
ucrl	EST server supports ucrl (Section 6.2)
usimpleenroll	EST server supports usimpleenroll (Section 7.2)
usimplereenroll	EST server supports usimplereenroll (Section 7.3)
userverkeygen	EST server supports userverkeygen (Section 7.4)

Table 4: ucacaps Keywords Defined in This Document

Capability keywords are case-insensitive; the canonical (registered) form is lowercase as shown in the tables above. An EST server SHOULD use the canonical lowercase form. An EST client MUST accept keywords in any case and MUST ignore unknown keywords.

Note: [RFC7030] designates cacerts, simpleenroll, and simplereenroll as MUST-implement operations. An EST server implementing this document SHOULD include these three keywords in every ucacaps response so that a client can enumerate all capabilities from a single request.

Example response:

```
cacerts
simpleenroll
simplereenroll
serverkeygen
crls
ucacert
ucacerts
ucrlinfo
ucrl
usimpleenroll
usimplereenroll
userverkeygen
```

This response indicates that the EST server supports the three mandatory RFC 7030 operations, serverkeygen, the RFC 8295 crls operation, and all operations defined in this document.

5. Distribution of CA Certificates

This section defines two operations that provide EST clients with CA certificate material. ucacert is a new operation with no RFC 7030 equivalent. ucacerts is an alternative encoding of the cacerts operation ([RFC7030], Section 4.1), returning the chain as a PEM-encoded certificate chain (application/pem-certificate-chain [RFC8555]) without a CMS wrapper.

5.1. ucacert

The ucacert operation returns the issuing CA certificate for the requested EST service as a single DER-encoded X.509 certificate. This differs from cacerts in two important ways: (1) only the issuing CA certificate is returned, not the full certificate chain; and (2) the certificate is returned without a CMS wrapper.

5.1.1. Request

The EST client requests the CA certificate using the HTTP GET method. No request body is sent. The EST server SHOULD NOT require client authentication for this operation, consistent with [RFC7030], Section 4.1.2.

GET /.well-known/est/<label>/ucacert

5.1.2. Response

On success, the EST server returns HTTP status 200 with:

* Content-Type: application/pkix-cert

* Body: The Base64 encoding of the DER-encoded X.509 certificate [RFC5280] of the issuing CA.

The returned certificate is the issuing CA certificate for the requested EST service context, as selected by the EST server for the target <label> (if any). The response contains exactly one certificate.

Successful ucacert responses SHOULD include HTTP caching metadata. In particular, EST servers SHOULD include ETag and Last-Modified headers. EST servers SHOULD include Cache-Control and Expires headers when the certificate lifetime or the local CA-certificate replacement policy provides a meaningful freshness bound.

5.1.3. Relationship to cacerts

The cacerts operation ([RFC7030], Section 4.1) returns the full certificate chain (issuing CA plus any intermediate and root CAs) wrapped in a CMS certs-only structure (application/pkcs7-mime; smime-type=certs-only).

ucacert differs in two respects:

1. *Scope*: ucacert returns only the issuing CA certificate; the full chain is not included. EST clients that need the full chain for path validation should use ucacerts (Section 5.2) or cacerts ([RFC7030], Section 4.1).

2. *Format*: The certificate is returned as the Base64 encoding of a single DER-encoded X.509 certificate [RFC5280], with no CMS wrapper.

5.2. ucacerts

The ucacerts operation returns the CA certificate chain for the requested EST service ([RFC7030], Section 4.1.2) as a PEM-encoded certificate chain (application/pem-certificate-chain), without a CMS wrapper.

5.2.1. Request

The EST client requests the CA certificate chain using the HTTP GET method. No request body is sent. The EST server SHOULD NOT require client authentication for this operation, consistent with [RFC7030], Section 4.1.2.

GET /.well-known/est/<label>/ucacerts

5.2.2. Response

On success, the EST server returns HTTP status 200 with:

* Content-Type: application/pem-certificate-chain

* Body: A PEM-encoded certificate chain [RFC8555] containing one or more certificates. The first certificate is the issuing CA certificate; subsequent certificates are the trust chain up to and including the root CA, each encoded as a CERTIFICATE PEM block.

Successful ucacerts responses SHOULD include HTTP caching metadata. In particular, EST servers SHOULD include ETag and Last-Modified headers. EST servers SHOULD include Cache-Control and Expires headers when the chain lifetime or the local CA-certificate replacement policy provides a meaningful freshness bound.

5.2.3. Comparison to cacerts

The cacerts operation ([RFC7030], Section 4.1) wraps the certificate chain in a CMS certs-only structure. ucacerts delivers the same chain as concatenated PEM blocks that many TLS and PKI libraries can parse directly without a CMS library.

6. Distribution of Certificate Revocation Lists

This section defines the ucrlinfo and ucrl operations. Neither operation has an equivalent in [RFC7030]. [RFC8295] defines the crls EST extension for distribution of one or more CRLs (and ARLs), but it does not define a lightweight CRL-metadata operation corresponding to ucrlinfo. Revocation status is also commonly obtained via the CRL Distribution Points (CDP) extension in issued certificates. The ucrlinfo operation returns lightweight metadata for the current CRL, including CRL number, validity window, issuer, and SHA-256 fingerprint, without downloading the full CRL body. The ucrl operation provides an in-band retrieval path for that same current CRL when the full object is needed.

6.1. ucrlinfo

The ucrlinfo operation returns metadata about the current CRL for the target CA without downloading the full CRL body. This enables an EST client to check whether its locally cached CRL is still current (by comparing nextupdate, crlnumber, or sha256sum) before deciding to issue a ucrl request. Unlike the RFC 8295 crls operation, ucrlinfo does not describe a set of CRLs; it describes only the current CRL selected by the EST server for the target CA.

6.1.1. Request

The EST client requests the CRL metadata using the HTTP GET method. No request body is sent. The EST server SHOULD NOT require client authentication for this operation, consistent with [RFC7030], Section 4.1.2.

GET /.well-known/est/<label>/ucrlinfo

6.1.2. Response

On success, the EST server returns HTTP status 200 with:

* Content-Type: text/plain

* Body: A plain-text list of CRL metadata fields, one field per line. Lines MUST be terminated by <CR><LF> or <LF>. Each line has the form <name>=<value>. Field order is not significant; EST clients MUST accept fields in any order and MUST ignore unknown field names.

Field names are case-insensitive; the canonical (registered) form is lowercase as shown in Table 5. EST servers SHOULD use the canonical lowercase form. EST clients MUST match field names in a case-insensitive manner.

The defined fields are listed in Table 5. Date/time values are expressed as Unix time (seconds since 1970-01-01T00:00:00Z).

Field name	Value	Presence
sha256sum	Base64-encoded SHA-256 digest of the DER-encoded CRL	MUST
issuer	Single-line Base64-encoded DER encoding of the CRL issuer field ([RFC5280], Section 5.1.2.3)	MUST
thisupdate	thisUpdate time of the CRL as Unix time (seconds since 1970-01-01T00:00:00Z) ([RFC5280], Section 5.1.2.4)	MUST
nextupdate	nextUpdate time of the CRL as Unix time (seconds since 1970-01-01T00:00:00Z) ([RFC5280], Section 5.1.2.5)	MUST if present in CRL
crlnumber	Decimal representation of the CRL Number extension value ([RFC5280], Section 5.2.3)	MUST if present in CRL

Table 5: ucrlinfo Response Fields

6.1.3. Choice of CRL-Metadata Format

The ucrlinfo response uses text/plain rather than JSON or CBOR because the response is a small set of independent name=value fields. The values are already encoded as simple strings: Unix time values for freshness checks, decimal text for the CRL number, and Base64 text for binary DER or digest values. A line-oriented format lets an EST client compare freshness indicators and decide whether to download the CRL without adding a JSON, CBOR, CMS, or ASN.1 parser for the metadata response itself.

JSON or CBOR would be useful if ucrlinfo needed nested objects, arrays, typed alternatives, or complex policy information. This operation deliberately avoids those semantics. Field order is insignificant, unknown fields are ignored, and each defined field has a single textual representation, so a structured container would add implementation cost without changing the protocol semantics.

Future specifications that need richer revocation metadata can define additional fields with simple textual encodings or a separate structured operation. Such an extension would be preferable to changing the base ucrlinfo response format.

Example response:

```
sha256sum=fcWyNz1VMfF2wj4/ozla0+Aqj5s0c4fHkkjY8d26oF0=
issuer=MIHxMQswCQYDVQQGEwJERTEP...
thisupdate=1735732800
nextupdate=1736337600
crlnumber=42
```

If no current CRL is available, the EST server MUST return HTTP status 503 (Service Unavailable). The server MAY include a Retry-After header to indicate when the CRL is expected to become available.

6.2. ucrl

The ucrl operation returns the current CRL for the target CA as a DER-encoded CRL. Unlike the RFC 8295 crls operation, which distributes one or more CRLs in a CRL package, ucrl returns only the single current CRL selected by the EST server for the target CA.

6.2.1. Request

The EST client requests the current CRL using the HTTP GET method. No request body is sent. The EST server SHOULD NOT require client authentication for this operation, consistent with [RFC7030], Section 4.1.2.

GET /.well-known/est/<label>/ucrl

6.2.2. Response

On success, the EST server returns HTTP status 200 with:

* Content-Type: application/pkix-crl

* Body: The Base64 encoding of the DER-encoded X.509 CRL [RFC5280] of the target CA.

Successful ucrl responses SHOULD include HTTP caching metadata. In particular, EST servers SHOULD include Last-Modified, ETag, Expires, and Cache-Control headers. When present, Last-Modified SHOULD reflect the CRL thisUpdate value, and Expires SHOULD reflect the CRL nextUpdate value. The Cache-Control header SHOULD include a max-age value no later than the advertised nextUpdate, together with the public, no-transform, and must-revalidate directives.

If no current CRL is available, the EST server MUST return HTTP status 503 (Service Unavailable). The server MAY include a Retry-After header to indicate when the CRL is expected to become available.

7. Client Certificate Request Functions

This section defines three enrollment operations that extend the Client Certificate Request Functions of [RFC7030], Section 4.2 and the Server-Side Key Generation operation of [RFC7030], Section 4.4.

7.1. Client Authentication

The enrollment operations defined in this section (usimpleenroll, usimplereenroll, and userverkeygen) require client authentication. The client authentication requirements are unchanged from [RFC7030]. EST clients and EST servers MUST follow [RFC7030], Section 3.2.3 and [RFC7030], Section 3.3.2 for client authentication, respectively.

7.2. usimpleenroll

The usimpleenroll operation requests a new certificate from the EST server ([RFC7030], Section 4.2.1) and returns the issued certificate as a DER-encoded X.509 certificate.

7.2.1. Request

An authenticated EST client submits a PKCS#10 CSR [RFC2986] using the HTTP POST method.

The request body is the Base64 encoding of the DER-encoded PKCS#10 CSR.

```
POST /.well-known/est/<label>/usimpleenroll
Content-Type: application/pkcs10

<Base64-encoded DER CSR>
```

The CSR MUST include a valid PoP signature. The EST server MUST verify the PoP signature against the public key in the CSR before issuing a certificate, as required by [RFC7030], Section 3.4.

7.2.2. Response

On success, the EST server returns HTTP status 200 with:

* Content-Type: application/pkix-cert

* Body: The Base64 encoding of the DER-encoded X.509 certificate [RFC5280] issued for the subject in the CSR.

7.2.3. Comparison to simpleenroll

The simpleenroll response ([RFC7030], Section 4.2.3) wraps the issued certificate in a CMS certs-only structure (application/pkcs7-mime; smime-type=certs-only). usimpleenroll returns that certificate as the Base64 encoding of a single DER-encoded certificate.

7.3. usimplereenroll

The usimplereenroll operation renews or rekeys an existing certificate ([RFC7030], Section 4.2.2) and returns the renewed certificate as a DER-encoded X.509 certificate.

7.3.1. Request

An authenticated EST client submits a PKCS#10 CSR for re-enrollment using the HTTP POST method.

The request body is the Base64 encoding of the DER-encoded PKCS#10 CSR.

```
POST /.well-known/est/<label>/usimplereenroll
Content-Type: application/pkcs10

<Base64-encoded DER CSR>
```

Re-enrollment processing follows [RFC7030], Section 4.2.2.

7.3.2. Response

On success, the EST server returns HTTP status 200 with:

* Content-Type: application/pkix-cert

* Body: The Base64 encoding of the DER-encoded renewed X.509 certificate.

7.3.3. Comparison to simplereenroll

The simplereenroll response ([RFC7030], Section 4.2.3) wraps the renewed certificate in a CMS certs-only structure. usimplereenroll returns that certificate as the Base64 encoding of a DER-encoded certificate.

7.4. userverkeygen

The userverkeygen operation requests server-side key generation and returns the generated private key and certificate ([RFC7030], Section 4.4).

7.4.1. Request

An authenticated EST client requests server-side key generation by submitting a PKCS#10 CSR using the HTTP POST method. Because the key pair is generated by the EST server, the SubjectPublicKeyInfo in the CSR SHOULD be absent or contain a placeholder. EST servers MUST NOT require a specific public key to be present. Likewise, the CSR signature is only a placeholder because the client possesses no corresponding private key. The EST client MAY use the id-alg-unsigned algorithm (OID 1.3.6.1.5.5.7.6.36) defined in [RFC9925] to signal explicitly that the signature field is intentionally unsigned.

EST servers MUST accept CSRs that use id-alg-unsigned.

The request body is the Base64 encoding of the DER-encoded PKCS#10 CSR.

```
POST /.well-known/est/<label>/userverkeygen
Content-Type: application/pkcs10

<Base64-encoded DER CSR>
```

Unlike usimpleenroll, the EST server MUST NOT verify a PoP signature for userverkeygen because no client-generated public key is being certified (consistent with [RFC7030], Section 4.4.1). If the CSR uses id-alg-unsigned ([RFC9925]), there is no signature value to check.

7.4.2. Response

On success, the EST server returns HTTP status 200 with:

* Content-Type: application/x-pem-file

* Body: A sequence of PEM textual encodings [RFC7468] containing exactly two label-value pairs in the following order:

  1. PRIVATE KEY: the PKCS#8 [RFC5958] DER-encoded private key, Base64-encoded within the PEM encapsulation boundary.

  2. CERTIFICATE: the DER-encoded X.509 certificate [RFC5280] issued for the generated key pair, Base64-encoded within the PEM encapsulation boundary.

  Note: The media type application/x-pem-file is a widely deployed convention for PEM-encoded ([RFC7468]) cryptographic objects. It predates the IETF media type registration process and is not registered in the IANA Media Types registry. No IANA-registered media type currently exists for a single message body carrying both a private key and a certificate in PEM textual encoding. This document uses application/x-pem-file to remain consistent with existing deployments. A future revision MAY register an appropriate media type with IANA and update this specification accordingly.

Example response body:

```
-----BEGIN PRIVATE KEY-----
<Base64-encoded PKCS#8 private key>
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
<Base64-encoded X.509 certificate>
-----END CERTIFICATE-----
```

The EST server SHOULD delete the private key from its storage as soon as the response has been transmitted successfully, unless the deployment policy requires retention for key escrow or disaster recovery (see Section 8). The private key is protected only by the TLS channel; no additional encryption is applied.

7.4.3. Comparison to serverkeygen

The serverkeygen response ([RFC7030], Section 4.4.2) uses a multipart/mixed MIME response containing a PKCS#8 part and a CMS certs-only part. userverkeygen delivers the same material in a single PEM file, eliminating the need for a MIME multipart parser and a CMS library.

8. Security Considerations

The security requirements of [RFC7030] apply in full to all operations defined in this document. The following subsections address security considerations specific to the new operations.

8.1. Transport Security

All operations defined in this document MUST be carried out over HTTPS (HTTP over TLS) as required by [RFC7030], Section 3. Implementations MUST NOT fall back to plain HTTP. EST servers MUST present a TLS certificate that EST clients can validate against a configured trust anchor, and EST clients SHOULD reject server certificates that cannot be validated.

Implementers SHOULD consult [RFC7925] for guidance on TLS version, cipher suite selection, and certificate profiles when those recommendations are applicable to the deployment.

8.2. Client Authentication

The ucacaps, ucacert, ucacerts, ucrlinfo, and ucrl retrieval operations follow the same authentication policy as the cacerts operation defined in [RFC7030], Section 4.1.2: the EST server SHOULD NOT require client authentication for these operations. CRLs and related CRL metadata are public information that relying parties must be able to retrieve without impediment; requiring authentication for

ucrlinfo or ucrl would hinder revocation checking, consistent with the design intent of CRL distribution points in [RFC5280].

Enrollment operations (usimpleenroll, usimplereenroll, userverkeygen) MUST require client authentication as specified in Section 7.1.

When HTTP Basic authentication is used, the considerations in [RFC7030], Section 3.2.3 apply. In particular, the user:password credential is Base64-encoded and visible to any observer that can read the TLS plaintext. Basic authentication MUST therefore only be used inside a TLS session. EST servers SHOULD rate-limit failed authentication attempts to mitigate brute-force attacks against user credentials.

When TLS client certificate authentication is used, the requirements and considerations in [RFC7030], Section 3.3.2 apply.

8.3. Proof of Possession

usimpleenroll and usimplereenroll MUST verify the PoP signature in the PKCS#10 CSR before issuing a certificate, as required by [RFC7030], Section 3.4. An EST server that skips PoP verification may issue certificates for public keys that the requester does not control, undermining the binding between the certified key pair and the subscriber.

The userverkeygen operation does not require PoP verification because the EST server generates the key pair itself (see [RFC7030], Section 4.4.1). The authenticity of the request is therefore established solely by client authentication (Section 8.2). EST servers MUST enforce client authentication at least as strongly for userverkeygen as for the other enrollment operations.

8.4. Private Key Delivery

The userverkeygen operation delivers a generated private key to the EST client over TLS in a PEM file. This follows the same model as the serverkeygen operation in [RFC7030], Section 4.4. The following additional considerations apply:

* The EST server SHOULD delete the generated private key immediately after the response has been transmitted successfully. If the deployment requires the server to retain a copy (e.g., for key escrow or disaster recovery), the private key MUST be stored in encrypted form with access controls that restrict retrieval to authorized personnel only.

* EST clients MUST verify the EST server’s TLS certificate before transmitting the enrollment request, as the TLS channel is the only confidentiality protection for the delivered private key.

* Server-side key generation is generally discouraged when the EST client is capable of generating its own key pair; see [RFC7030], Section 6 for further guidance.

* The PEM response contains the private key in a PRIVATE KEY block with no encryption. EST clients MUST store the key material securely immediately upon receipt (e.g., import it into a hardware security module or an encrypted key store) and MUST NOT write it to unprotected storage.

8.5. Response Integrity and Trust Anchor Bootstrapping

The ucacert, ucacerts, and ucrl operations return PKI objects over a TLS-protected channel without an additional application-layer signature. EST clients MUST validate received certificates and CRLs against an independently configured trust anchor.

When bootstrapping a device that does not yet possess a trust anchor, the initial ucacert or ucacerts exchange is subject to the same considerations as the cacerts bootstrap operation described in [RFC7030], Section 4.1.1. EST clients SHOULD follow the guidance in [RFC7030], Section 3.6 (Server Authorization) for trust anchor selection, and SHOULD use an Implicit Trust Anchor ([RFC7030], Section 3.6.2) only when an Explicit Trust Anchor is not available.

8.6. Subject Name and SAN Validation for Re-enrollment

The usimplereenroll operation identifies the certificate to be renewed by the subject DN (and optionally SAN) carried in the CSR, consistent with [RFC7030], Section 4.2.2. EST servers MUST ensure that the authenticated requester is authorized to renew the certificate identified by that subject identity.

When the id-cmc-changeSubjectName attribute is present in the CSR, the EST server MUST validate that the requested new subject DN and SAN values are within the policy permitted for the authenticated requester and the selected certificate profile.

8.7. Capability Discovery Security

The ucacaps operation SHOULD NOT require client authentication (consistent with [RFC7030], Section 4.1.2) and returns its response without any application-layer signature. Therefore, an on-path attacker that can intercept the TLS session could return a manipulated capability list. For example, the attacker could omit userverkeygen to prevent key generation or omit ucrlinfo and ucrl to interfere with CRL checking.

EST clients SHOULD treat the ucacaps response as advisory information only. EST clients SHOULD NOT rely on it as a security enforcement mechanism. In particular, a negative ucacaps response (e.g., an absent keyword) MUST NOT cause an EST client to fall back to less secure behavior.

EST clients that obtain the ucacaps response via a TLS session in which the server certificate has been validated against an Explicit Trust Anchor MAY treat the response as reliable.

8.8. Denial-of-Service Considerations

The ucrl operation returns the full CRL, which may be large. EST servers SHOULD implement rate limiting and response size limits to prevent resource exhaustion from concurrent CRL download requests. EST clients SHOULD call ucrlinfo before ucrl to determine whether the locally cached CRL is still current. If the crlnumber, nextupdate, or sha256sum fields indicate no change since the last download, the client SHOULD refrain from issuing a ucrl request. EST clients that cache the CRL locally SHOULD use the nextupdate field from ucrlinfo (or from the cached CRL itself) to determine when to refresh, and SHOULD NOT poll more frequently than necessary.

9. IANA Considerations

This document makes no requests to IANA.

9.1. Media Types (Informative)

The operations defined in this document use the media types listed in Table 6. All types except application/x-pem-file are already registered in the IANA Media Types registry; no new registrations are requested by this document.

Media Type	Used by	Registered in
application/pkcs10	Request body for enrollment operations	[RFC5652]
application/pkix-cert	Response body for ucacert, usimpleenroll, usimplereenroll	[RFC2585]
application/pkix-crl	Response body for ucrl	[RFC2585]
application/pkcs7-mime	(existing cacerts/ simpleenroll format, for reference)	[RFC8551]
application/pem-certificate-chain	Response body for ucacerts	[RFC8555]
application/x-pem-file	Response body for userverkeygen (key + cert)	Unregistered; see note below
text/plain	Response body for ucacaps and ucrlinfo	[RFC2046]

Table 6: Media Types Used by Lightweight EST Operations

Note: application/pkix-cert and application/pkix-crl are registered in the IANA Media Types registry (see [RFC2585]). Implementers should consult the IANA registry for authoritative definitions. The media type application/x-pem-file is not registered with IANA, but it is a widely deployed convention for PEM-encoded ([RFC7468]) cryptographic objects. It predates the IETF media type registration process. No IANA-registered type currently exists for a response body containing both a private key and a certificate in PEM textual encoding. A future revision of this document MAY register an appropriate media type and update the userverkeygen specification accordingly.

10. References

10.1. Normative References

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/rfc/rfc2046>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, DOI 10.17487/RFC2585, May 1999, <https://www.rfc-editor.org/rfc/rfc2585>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/rfc/rfc2986>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/rfc/rfc5652>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/rfc/rfc5958>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/rfc/rfc7030>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/rfc/rfc7468>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/rfc/rfc8551>.

[RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <https://www.rfc-editor.org/rfc/rfc8555>.

[RFC8951] Richardson, M., Werner, T., and W. Pan, "Clarification of Enrollment over Secure Transport (EST): Transfer Encodings and ASN.1", RFC 8951, DOI 10.17487/RFC8951, November 2020, <https://www.rfc-editor.org/rfc/rfc8951>.

[RFC9925] Benjamin, D., "Unsigned X.509 Certificates", RFC 9925, DOI 10.17487/RFC9925, February 2026, <https://www.rfc-editor.org/rfc/rfc9925>.

10.2. Informative References

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/rfc/rfc7228>.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/rfc/rfc7925>.

[RFC8295] Turner, S., "EST (Enrollment over Secure Transport) Extensions", RFC 8295, DOI 10.17487/RFC8295, January 2018, <https://www.rfc-editor.org/rfc/rfc8295>.

[RFC9148] van der Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol", RFC 9148, DOI 10.17487/RFC9148, April 2022, <https://www.rfc-editor.org/rfc/rfc9148>.

[RFC9908] Richardson, M., Ed., Friel, O., von Oheimb, D., and D. Harkins, "Clarification and Enhancement of the CSR Attributes Definition in RFC 7030", RFC 9908, DOI 10.17487/RFC9908, January 2026, <https://www.rfc-editor.org/rfc/rfc9908>.

Appendix A. Message Flow Diagrams

This appendix illustrates the HTTP message flows for each operation defined in this document. All exchanges occur within a TLS session; the TLS handshake is not shown. For DER-carrying operations, the examples show the base64-encoded form required by this specification.

A.1. ucacaps

Figure 1: ucacaps message flow

A.2. ucacert

Figure 2: ucacert message flow

A.3. ucacerts

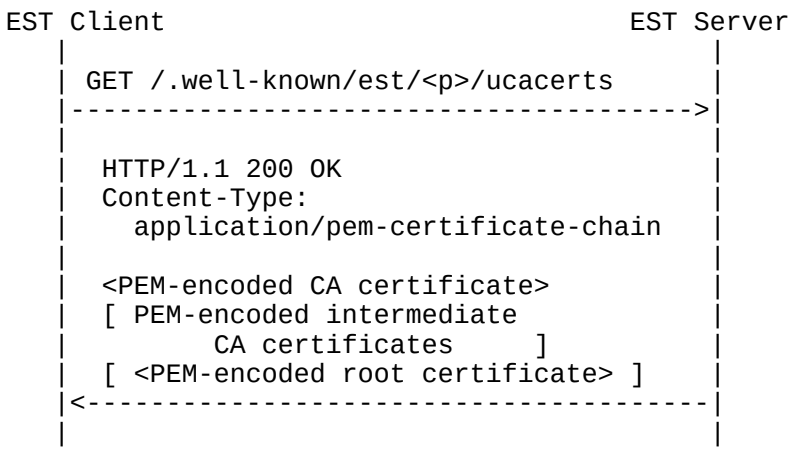

Figure 3: ucacerts message flow

A.4. ucrlinfo

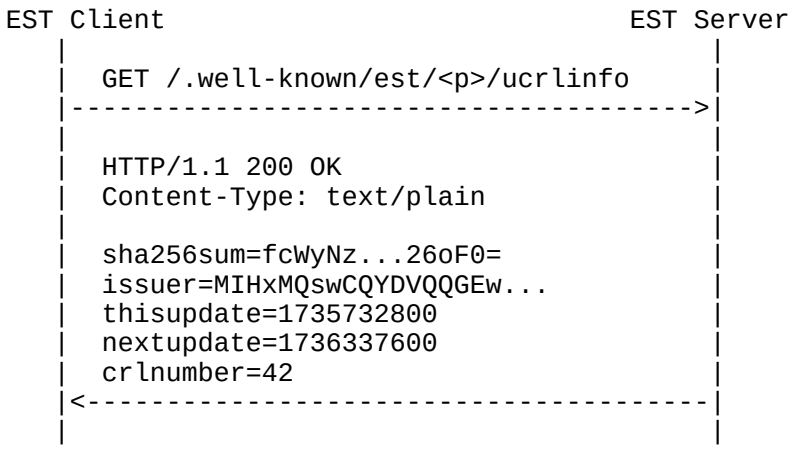

Figure 4: ucrlinfo message flow

A.5. ucrl

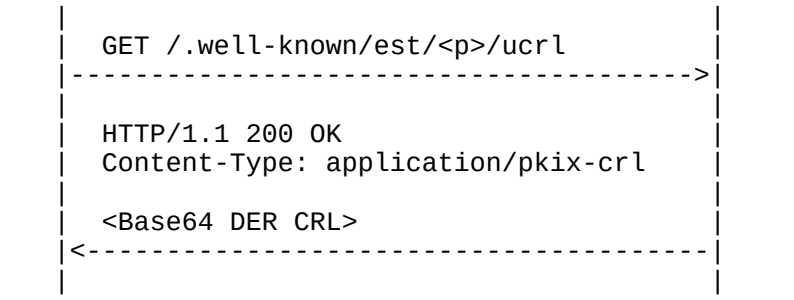

Figure 5: ucrl message flow

A.6. usimpleenroll

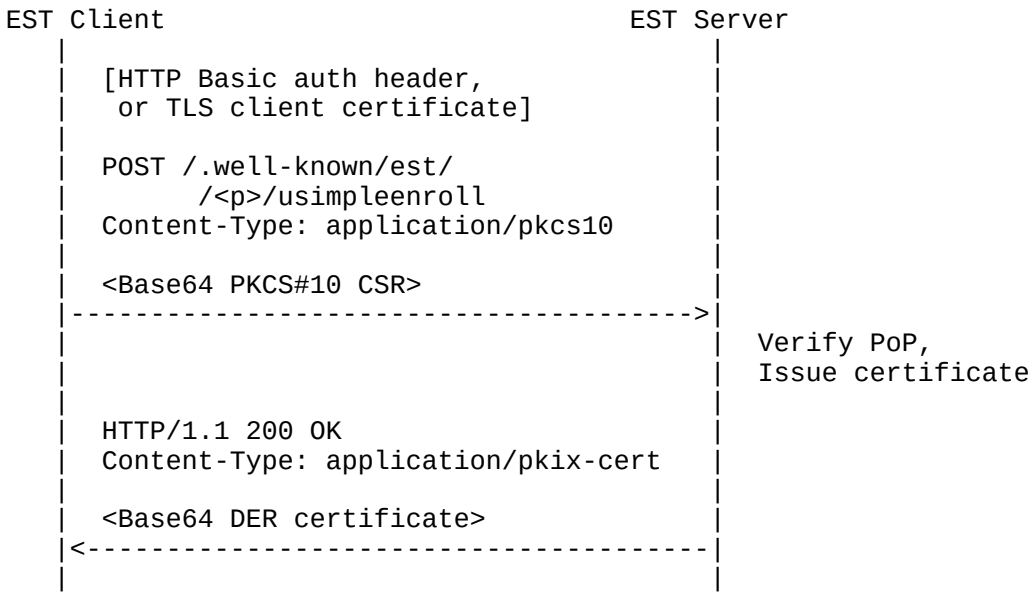

Figure 6: usimpleenroll message flow

A.7. usimplereenroll

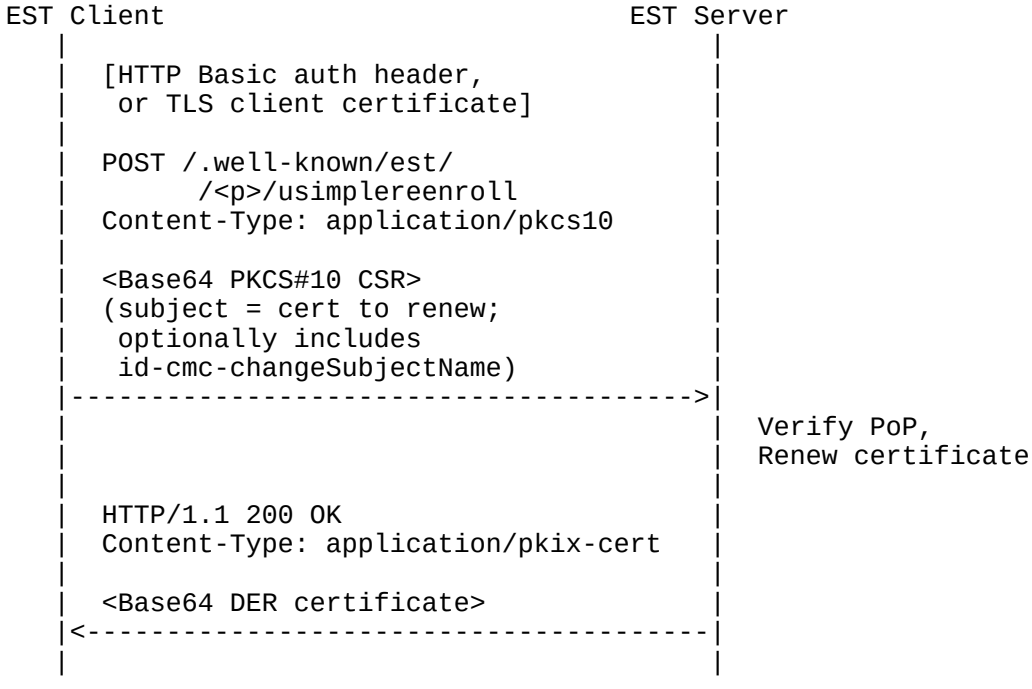

Figure 7: usimplereenroll message flow

A.8. userverkeygen

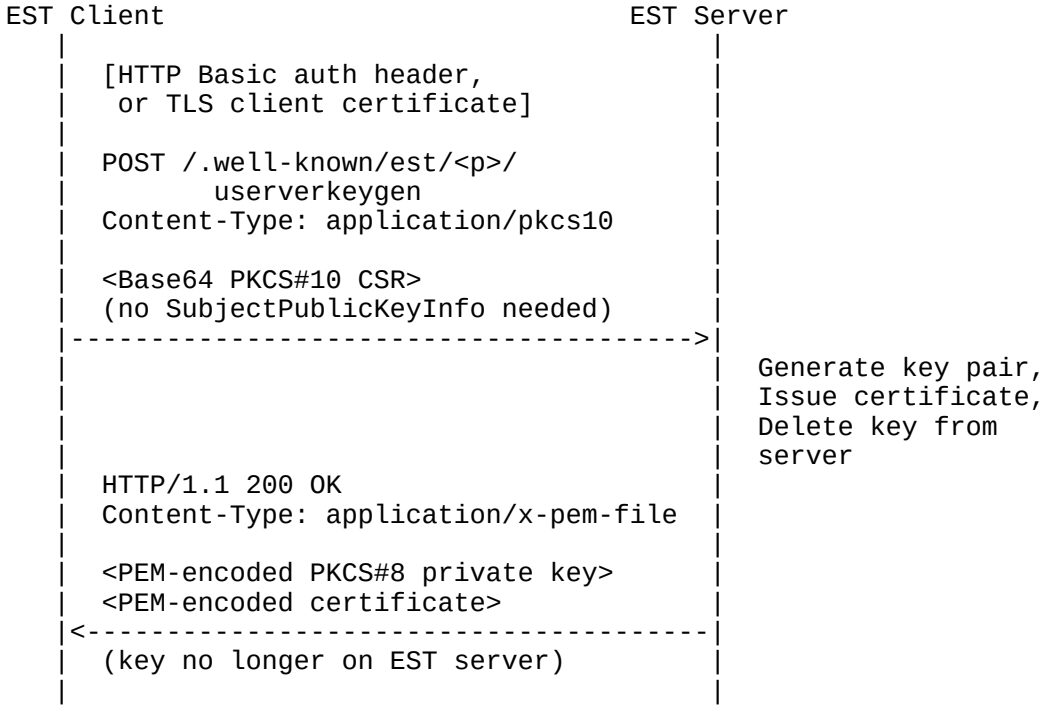

Figure 8: userverkeygen message flow

Appendix B. Comparison with RFC 7030 Operations

The following table summarizes how each new operation relates to its RFC 7030 counterpart, the changes in response format, and the corresponding RFC 7030 section (see Table 7).

New Operation	RFC 7030 Operation	Auth	New Content-Type	New Format
ucacaps	_(none)_	No	text/plain	Keyword list
ucacert	_(none)_	No	application/pkix-cert	Single DER cert
ucacerts	cacerts ([RFC7030], Section 4.1)	No	application/pem-certificate-chain	PEM cert chain
ucrlinfo	_(none)_	No	text/plain	CRL metadata
ucrl	_(none)_	No	application/pkix-crl	DER CRL
usimpleenroll	simpleenroll ([RFC7030], Section 4.2.1)	Yes	application/pkix-cert	DER cert
usimplereenroll	simplereenroll ([RFC7030], Section 4.2.2)	Yes	application/pkix-cert	DER cert
userverkeygen	serverkeygen ([RFC7030], Section 4.4)	Yes	application/x-pem-file	PEM: PKCS#8 + certificate

Table 7: Comparison of New Operations with RFC 7030 Counterparts

Key differences from RFC 7030 equivalents:

* *New single-certificate retrieval operation*: ucacert returns the issuing CA certificate as the Base64 encoding of a single DER-encoded X.509 certificate, with no certificate chain and no CMS wrapper. [RFC7030] has no equivalent. cacerts always returns the full chain in CMS format.

* *No CMS wrapper*: The certificate- and CRL-returning operations deliver their payloads without CMS encapsulation. EST clients therefore do not need a CMS library to parse those responses.

* *PEM instead of multipart*: userverkeygen returns the private key and certificate in a single PEM file (application/x-pem-file) rather than a multipart/mixed MIME response ([RFC7030], Section 4.4.2). EST clients do not need a MIME multipart parser or a CMS library.

* *New capability discovery operation*: ucacaps enables EST clients to discover which operations the EST server supports. [RFC7030] has no equivalent operation.

* *New CRL operations*: ucrlinfo provides lightweight CRL metadata (number, validity window, issuer, and SHA-256 fingerprint) without downloading the full CRL body, and ucrl extends EST with in-band CRL retrieval when the full CRL is needed. [RFC7030] does not define CRL distribution or CRL status operations; CRL retrieval is normally handled via the CDP extension in issued certificates.

Appendix C. Comparison with RFC 8295

[RFC8295] defines EST extensions for package distribution, including the crls operation for distributing one or more CRLs (and ARLs). This document addresses a narrower use case for the latest CRL of the selected CA.

The main differences are summarized in Table 8.

Topic	RFC 8295 crls	This Document
Object scope	Returns one or more CRLs in a CRL package and may also distribute ARLs.	ucrl returns a single current CRL; ucrlinfo returns metadata for that current CRL.
Metadata-only check	Not defined.	ucrlinfo allows freshness checks without downloading the CRL body.
Response format	Follows the RFC 7030 CMS-based response model for CRL packages.	Uses Base64-encoded DER for ucrl and plain text for ucrlinfo, without CMS.
Intended use	General EST extension for CRL/ ARL distribution.	Lightweight retrieval of the current CRL and its metadata.

Table 8: Comparison with RFC 8295

Appendix D. Comparison with RFC 9148 (EST-coaps)

RFC 9148 defines EST-coaps: a profile and transport mapping that carries EST messages over secure CoAP instead of HTTPS. It changes the transfer substrate by using CoAP, DTLS, short CoAP resource paths, CoAP resource discovery, CoAP Block-Wise Transfer, and CoAP Content-Format identifiers.

RFC 9148 is important prior art for reducing EST payload overhead. In particular, it defines Content-Format 287 for application/pkix-cert, allowing a client to request a single DER-encoded certificate rather than a CMS certs-only object for certificate-returning EST-coaps functions. RFC 9148 performs this selection with CoAP content negotiation, using the Accept Option, rather than by defining separate resources for each response format.

This document addresses a related but different problem. It keeps the HTTPS-based EST model of [RFC7030] and defines additional EST operations with simpler response payloads. The goal is to reduce application-layer parsing requirements for EST clients that still use HTTP/TLS and to add operations that are not present in RFC 7030 or RFC 9148.

The main differences are summarized in Table 9.

Topic	RFC 9148	This Document
Transport	Defines EST over CoAP secured with DTLS.	Uses the existing EST HTTPS/TLS transport from RFC 7030.
URI design	Maps EST operations to short CoAP paths such as /crts, /sen, /sren, /skg, /skc, and /att.	Adds HTTP EST operations under the RFC 7030 /.well-known/est/ path structure.
Discovery	Uses CoRE Link Format resource discovery for EST-coaps resources.	Defines ucacaps as an EST operation that returns supported operation keywords.
Response selection	Uses CoAP content negotiation; a client uses the Accept Option to request an available Content-Format.	Uses distinct HTTP EST resources for the unencapsulated operations, with ucacaps for capability discovery.
Certificate response format	Supports PKCS#7 certs-only Content-Format 281 and defines Content-Format 287 (application/pkix-cert) for a single non-CMS certificate response.	Defines unencapsulated HTTP operations whose successful responses use base64-encoded DER certificates or PEM certificate chains without CMS.
CA certificate retrieval	Maps the EST cacerts function to /crts; Content-Format 287 can carry a single certificate when negotiated.	Adds ucacert for a single issuing CA certificate and ucacerts for a PEM certificate chain.
Enrollment responses	Maps simpleenroll and simplereenroll to CoAP and permits single-certificate responses when Content-Format 287 is negotiated.	Defines usimpleenroll and usimplereenroll as HTTP/TLS operations that directly return a DER certificate.

Server-side key generation	Defines CoAP resources for server-side key generation and uses application/multipart-core for key-and-certificate responses.	Defines userverkeygen, returning a PEM-encoded private key and certificate in a single response body.
CRL support	Does not define EST-coaps CRL metadata or CRL retrieval operations.	Defines ucrlinfo and ucrl for CRL freshness checks and CRL retrieval.

Table 9: Comparison with RFC 9148

RFC 9148 is therefore complementary to this document rather than a substitute for it. It already solves part of the same problem for EST-coaps by defining a non-CMS single-certificate format and negotiating it with Accept. That mechanism may also inform implementations that expose the same semantics over transports other than CoAP. However, RFC 9148 does not update the HTTPS resource model of RFC 7030, does not define the ucacaps, ucacert, ucacerts, ucrlinfo, ucrl, usimpleenroll, usimplereenroll, or userverkeygen HTTP operations, and does not define EST-based CRL metadata or CRL retrieval.

An implementation can support both specifications. For example, a server could offer EST-coaps resources for CoAP clients while also offering the operations defined in this document for HTTPS EST clients. Another implementation might choose to apply RFC 9148-style content-negotiation semantics across multiple transports. The interfaces can share the same CA, RA policy, authentication checks, and certificate issuance logic, but their standardized protocol bindings and payload formats are distinct.

Acknowledgements

The authors want to thank Michael Richardson for reviewing and commenting on intermediate versions of the draft.

Implementation Notes

> *Editor’s Note:* This section is for author reference only. It MUST be removed before RFC publication.

A reference implementation of the operations described in this document is available in the gateway module of the XiPKI open-source PKI project (https://github.com/xipki/xipki).

The reference implementation supports HTTP Basic authentication and TLS client certificate authentication, consistent with [RFC7030], Section 3.2.3 and [RFC7030], Section 3.3.2. Its path routing also supports both the two-segment <alias>/<operation> form and the three-segment <ca-name>/<cert-profile>/<operation> form described in Section 3.

Author's Address

Lijun Liao
NIO
Email: lijun.liao@nio.io