Abstract

Enrollment over Secure Transport (EST) is used as a certificate provisioning protocol over HTTPS. Low-resource devices often use the lightweight Constrained Application Protocol (CoAP) for message exchanges. This document defines how to transport EST payloads over secure CoAP (EST-coaps), which allows low-resource constrained devices to use existing EST functionality for provisioning certificates.

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 April 11, 2019.
Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Change Log .................................................. 3
2. Introduction ............................................... 4
3. Terminology ................................................ 5
4. Conformance to RFC7925 profiles .......................... 5
5. Protocol Design ............................................. 6
   5.1. Mandatory/optional EST Functions ..................... 7
   5.2. Payload format ....................................... 7
       5.2.1. Content Format application/multipart-core .......... 8
   5.3. Message Bindings ..................................... 8
   5.4. CoAP response codes .................................. 9
   5.5. Delayed Responses ................................... 9
   5.6. Server-side Key Generation .......................... 11
   5.7. Message fragmentation ............................... 12
   5.8. Deployment limits ................................... 13
6. Discovery and URI ........................................... 13
7. DTLS Transport Protocol ................................. 15
8. HTTPS-CoAPS Registrar .................................... 17
9. Parameters ................................................ 19
10. IANA Considerations ...................................... 20
   10.1. Content-Format Registry ............................ 20
   10.2. Resource Type registry .............................. 20
11. Security Considerations ................................. 21
   11.1. EST server considerations ......................... 21
   11.2. HTTPS-CoAPS Registrar considerations ............... 22
12. Acknowledgements ......................................... 22
13. References ............................................... 23
   13.1. Normative References ............................... 23
   13.2. Informative References ............................. 24
Appendix A. EST messages to EST-coaps ...................... 26
   A.1. cacerts ............................................. 26
   A.2. csrattrs ............................................ 31
1. Change Log

EDNOTE: Remove this section before publication

-06:

clarified discovery section, by specifying that no discovery may
be needed for /.well-known/est URI.

added resource type values for IANA

added list of compulsory to implement and optional functions.

Fixed issues pointed out by the idnits tool.

Updated COAP response codes section with more mappings between EST
HTTP codes and EST-coaps COAP codes.

Minor updates to the MTI EST Functions section.

Moved Change Log section higher.

-05:

repaired again

TBD8 removed from C-F registration, to be done in CT draft.

-04:

Updated Delayed response section to reflect short and long delay
options.

-03:

Removed observe and simplified long waits

Repaired content-format specification

-02:
2. Introduction

"Classical" Enrollment over Secure Transport (EST) [RFC7030] is used for authenticated/authorized endpoint certificate enrollment (and optionally key provisioning) through a Certificate Authority (CA) or Registration Authority (RA). EST messages run over HTTPS.

This document defines a new transport for EST based on the Constrained Application Protocol (CoAP) since some Internet of Things (IoT) devices use CoAP instead of HTTP. Therefore, this specification utilizes DTLS [RFC6347], CoAP [RFC7252], and UDP instead of TLS [RFC8446], HTTP [RFC7230] and TCP.
EST messages may be relatively large and for this reason this
document also uses CoAP Block-Wise Transfer [RFC7959] to offer a
fragmentation mechanism of EST messages at the CoAP layer.

This specification also profiles the use of EST to only support
certificate-based client Authentication. HTTP Basic or Digest
authentication (as described in Section 3.2.3 of [RFC7030] are not
supported.

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in [RFC2119].

Many of the concepts in this document are taken over from [RFC7030].
Consequently, much text is directly traceable to [RFC7030]. The same
document structure is followed to point out the differences and
commonalities between EST and EST-coaps.

4. Conformance to RFC7925 profiles

This section shows how EST-coaps fits into the profiles of low-
resource devices described in [RFC7925].

EST-coaps can transport certificates and private keys. Certificates
are responses to (re-)enrollment requests or request for a trusted
certificate list. Private keys can be transported as responses to a
request to a server-side keygeneration as described in section 4.4 of
[RFC7030] and discussed in Section 5.6 of this document.

As per [RFC7925] section 3.3 and section 4.4, the mandatory cipher
suite for DTLS in EST-coaps is TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
declared in [RFC7251], and the curve secp256r1 MUST be supported
[RFC8422]; this curve is equivalent to the NIST P-256 curve. Crypto
agility is important, and the recommendations in [RFC7925] section
4.4 and any updates to RFC7925 concerning Curve25519 and other CFRG
curves also apply.

DTLS1.2 implementations MUST use the Supported Elliptic Curves and
Supported Point Formats Extensions [RFC8422]. Uncompressed point
format MUST also be supported. [RFC6090] can be used as summary of
the ECC algorithms. DTLS 1.3 implementations differ from DTLS 1.2
because they do not support point format negotiation in favor of a
single point format for each curve and thus support for DTLS 1.3 does
not mandate point formation extensions and negotiation.
The EST-coaps client MUST be configured with at least an implicit TA database from its manufacturer. The authentication of the EST-coaps server by the EST-coaps client is based on certificate authentication in the DTLS handshake.

The authentication of the EST-coaps client is based on a client certificate in the DTLS handshake. This can either be

- a previously issued client certificate (e.g., an existing certificate issued by the EST CA); this could be a common case for simple reenrollment of clients;
- a previously installed certificate (e.g., manufacturer-installed certificate or a certificate issued by some other party); the server is expected to trust the manufacturer’s root CA certificate in this case.

5. Protocol Design

EST-coaps uses CoAP to transfer EST messages, aided by Block-Wise Transfer [RFC7959] to transport CoAP messages in blocks thus avoiding (excessive) fragmentation of UDP datagrams. The use of "Block" for the transfer of larger EST messages is specified in Section 5.7. The Figure 1 below shows the layered EST-coaps architecture.

```
+------------------------------------------------+
|    EST request/response messages               |
| +------------------------------------------------|
|    CoAP for message transfer and signalling    |
| +------------------------------------------------|
|    DTLS for transport security                 |
| +------------------------------------------------|
|    UDP for transport                            |
```

Figure 1: EST-coaps protocol layers

The EST-coaps protocol design follows closely the EST design. The actions supported by EST-coaps are identified by their message types:

- CA certificate retrieval, needed to receive the complete set of CA certificates.
- Simple enroll and reenroll, for CA to sign public client-identity key.
- Certificate Signing Request (CSR) Attributes request messages, informs the client of the fields to include in generated CSR.
Server-side key generation messages, to provide a private client-identity key when the client choses for an external entity to generate its private key.

5.1. Mandatory/optional EST Functions

This specification contains a set of required-to-implement functions, optional functions, and not specified functions. The latter ones are deemed too expensive for low-resource devices in payload and calculation times.

Table 1 specifies the mandatory-to-implement or optional implementation of the EST-coaps functions.

<table>
<thead>
<tr>
<th>EST Functions</th>
<th>EST-coaps implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>/cacerts</td>
<td>Mandatory</td>
</tr>
<tr>
<td>/simpleenroll</td>
<td>Mandatory</td>
</tr>
<tr>
<td>/simplereenroll</td>
<td>Mandatory</td>
</tr>
<tr>
<td>/fullcmc</td>
<td>Not specified</td>
</tr>
<tr>
<td>/serverkeygen</td>
<td>Optional</td>
</tr>
<tr>
<td>/csrattrs</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Table 1: list of EST-coaps functions

5.2. Payload format

The content-format (media type equivalent) of the CoAP message determines which EST message is transported in the CoAP payload. The media types specified in the HTTP Content-Type header (section 3.2.2 of [RFC7030]) are in EST-coaps specified by the Content-Format Option (12) of CoAP. The combination of URI path and content-format used for CoAP MUST map to an allowed combination of URI and media type as defined for EST. The required content-formats for these requests and response messages are defined in Section 10. The CoAP response codes are defined in Section 5.4.

EST-coaps is designed for use between low-resource devices and hence does not need to send base64-encoded data. Simple binary is more efficient (30% smaller payload) and well supported by CoAP.

The payload for a given media type follows the ASN.1 structure of the media-type and is transported as straight binary coding instead of the base64-encoded. The binary is wrapped in a CBOR major type 2 using h’xxx’ notation (to assure compatibility with multipart).
EDNote: suggestion to remove CBOR wrapping for not multipart.

In the examples of Appendix A, the base16 diagnostic notation is used for CBOR major type 2, where h’450aafbb’ represents an example binary payload. The content formats specification in Section 5.2.1 specifies the payload structure when multiple media types are present in the payload.

5.2.1. Content Format application/multipart-core

A representation with content format ID TBD8 contains a collection of representations along with their respective content format. The content-format identifies the media-type application/multipart-core specified in [I-D.ietf-core-multipart-ct].

The collection is encoded as a CBOR array [RFC7049] with an even number of elements. The second, fourth, sixth, etc. element is a binary string containing a representation. The first, third, fifth, etc. element is an unsigned integer specifying the content format ID of the following representation.

For example, a collection containing two representations in response to a server-side key generation, could include a private key in PKCS#8 with content format ID 284 and a certificate with content format ID 281, looks like this in diagnostic CBOR notation: [284,h’0123456789abcdef’,281,h’fedcba9876543210’]. The PKCS#8 key and the X.509 certificate representations will be ASN.1 encoded in binary format. An example is shown in Appendix A.4.

5.3. Message Bindings

The general EST CoAP message characteristics are:

- All EST-coaps messages expect a response from the server, thus the client MUST send the requests over confirmable CON COAP messages.

- The Ver, TKL, Token, and Message ID values of the CoAP header are not affected by EST.

- The CoAP options used are Uri-Host, Uri-Path, Uri-Port, Content-Format, and Location-Path in CoAP. These CoAP Options are used to communicate the HTTP fields specified in the EST REST messages.

- EST URLs are HTTPS based (https://), in CoAP these will be assumed to be transformed to coaps (coaps://)

Appendix A includes some practical examples of EST messages translated to CoAP.
5.4. CoAP response codes

Section 5.9 of [RFC7252] specifies the mapping of HTTP response codes to CoAP response codes. Every time the HTTP response code 200 is specified in [RFC7030] in response to a GET request, in EST-coaps the equivalent CoAP response code 2.05 or 2.03 MUST be used. Similarly, 2.01, 2.02 or 2.04 MUST be used in response to POST EST requests. Response code HTTP 202 has no equivalent in CoAP. Section 5.5 specifies how EST requests over CoAP handle delayed messages.

Other HTTP response codes EST makes use of, are 204 and 404 when a resource is not available for the client. The equivalent COAP error code to use in an EST-coaps response is 4.04. Additionally, EST’s 401 error translates to 4.01 in EST-coaps. Other HTTP error messages commonly used in EST are 400, 423 and 503. Their equivalent COAP errors are 4.00, 4.03 and 5.03 respectively.

5.5. Delayed Responses

Appendix B.2 shows an example of a server response that comes immediately after a client request. The example shows the flows of blocks as the large messages require fragmentation. But server responses can sometimes be delayed.

According to section 5.2.2 of [RFC7252], a slow server can acknowledge the request and respond later with the requested resource representation. In particular, a slow server can respond to a enroll request with an empty ACK with code 0.00, before sending the certificate to the server after a short delay. Consecutively, the server will need more than one "Block2" blocks to respond if the certificate is large. This situation is shown in Figure 2 where a client sends an enrollment request that uses more than one "Block1" blocks. The server uses an empty 0.00 ACK to announce the response which will be provided later with 2.04 messages containing "Block2" options. Having received the first 128 bytes in the first "block2" block, the client asks for a block reduction to 128 bytes in all following "block2" blocks, starting with the second block (NUM=1).
POST [2001:db8::2:1]:61616/est/sen (CON)(1:0/1/256) {CSR req} -->
-- (ACK) (1:0/1/256) (2.31 Continue)
POST [2001:db8::2:1]:61616/est/sen (CON)(1:1/1/256) {CSR req} -->
-- (ACK) (1:1/1/256) (2.31 Continue)
.
.
POST [2001:db8::2:1]:61616/est/sen (CON)(1:N1/0/256){CSR req} -->
-- (0.00 empty ACK)

...... short delay before certificate is ready....... 

-- (CON) (1:N1/0/256)(2:0/1/256) (2.04 Changed) {Cert resp}
(ACK) -->
POST [2001:db8::2:1]:61616/est/sen (CON)(2:1/0/128) -->
-- (ACK) (2:1/1/128) (2.04 Changed) {Cert resp}
.
.
POST [2001:db8::2:1]:61616/est/sen (CON)(2:N2/0/128) -->
-- (ACK) (2:N2/0/128) (2.04 Changed) {Cert resp}

Figure 2: EST-COAP enrolment with short wait

If the server is very slow providing the response (say minutes, possible when a manual intervention is wanted), the server SHOULD respond with an ACK containing response code 5.03 (Service unavailable) and a Max-Age option to indicate the time the client SHOULD wait to request the content later. After a delay of Max-Age, the client SHOULD resend the identical CSR to the server. As long as the server responds with response code 5.03 (Service Unavailable), the client can resend the enrolment request until the server responds with the certificate or the client abandons for other reasons.

To demonstrate this situation, Figure 3 shows a client sending an enrolment request that will use more than one "Block1" block to send the CSR to the server. The server needs more than one "Block2" blocks to respond, but also needs to take a long delay (minutes) to provide the response. Consequently, the server will use a 5.03 ACK for the response. The client can be requested to wait multiple times for a period of Max-Age. Note that in the example below the server asks for a decrease in the block size when acknowledging the first Block2.

Figure 5 can be compared with Figure 3 to see the extra requests after a Max-Age wait.
Figure 3: EST-COAP enrolment with long wait

5.6. Server-side Key Generation

Constrained devices sometimes do not have the necessary hardware to generate statistically random numbers for private keys and DTLS ephemeral keys. Past experience has shown that low-resource endpoints sometimes generate numbers which could allow someone to decrypt the communication or guess the private key and impersonate as the device. Studies have shown that the same keys are generated by the same model devices deployed on-line.

EDNote: Is there a reference for these studies?

Additionally, random number key generation is costly, thus energy draining. Even though the random numbers that constitute the identity/cert do not get generated often, an endpoint may not want to spend time and energy generating keypairs, and just ask for one from the server.

In these scenarios, server-side key generation can be used. The client asks for the server or proxy to generate the private key and
the certificate which is transferred back to the client in the server-side key generation response.

[RFC7030] recommends for the private key returned by the server to be encrypted. The specification provides two methods to encrypt the generated key, symmetric and asymmetric. The methods are signalled by the client by using the relevant attributes (SMIMECapabilities and DecryptKeyIdentifier or AsymmetricDecryptKeyIdentifier) in the CSR request. In the symmetric key case, the key can be established out-of-band or alternatively derived by the established TLS connection as described in [RFC5705].

The server-side key generation response is returned using a CBOR array Section 5.2.1. The certificate part exactly matches the response from an enrollment response. The private key is placed inside of a CMS SignedData. The SignedData is signed by the party that generated the private key, which may or may not be the EST server or the EST CA. The SignedData is further protected by placing it inside of a CMS EnvelopedData as explained in Section 4.4.2 of [RFC7030].

5.7. Message fragmentation

DTLS defines fragmentation only for the handshake part and not for secure data exchange (DTLS records). [RFC6347] states that to avoid using IP fragmentation, which involves error-prone datagram reconstitution, invokers of the DTLS record layer SHOULD size DTLS records so that they fit within any Path MTU estimates obtained from the record layer. In addition, invokers residing on a 6LoWPAN over IEEE 802.15.4 network SHOULD attempt to size CoAP messages such that each DTLS record will fit within one or two IEEE 802.15.4 frames.

That is not always possible. Even though ECC certificates are small in size, they can vary greatly based on signature algorithms, key sizes, and OID fields used. For 256-bit curves, common ECDSA cert sizes are 500-1000 bytes which could fluctuate further based on the algorithms, OIDs, SANs and cert fields. For 384-bit curves, ECDSA certs increase in size and can sometimes reach 1.5KB. Additionally, there are times when the EST cacerts response from the server can include multiple certs that amount to large payloads. Section 4.6 of CoAP [RFC7252] describes the possible payload sizes: "if nothing is known about the size of the headers, good upper bounds are 1152 bytes for the message size and 1024 bytes for the payload size". Section 4.6 of [RFC7252] also suggests that IPv4 implementations may want to limit themselves to more conservative IPv4 datagram sizes such as 576 bytes. From [RFC0791] follows that the absolute minimum value of the IP MTU for IPv4 is as low as 68 bytes, which would leave only 40 bytes minus security overhead for a UDP payload. Thus, even with ECC certs, EST-coaps messages can still exceed sizes in MTU of...
1280 for IPv6 or 60-80 bytes for 6LoWPAN [RFC4919] as explained in section 2 of [RFC7959]. EST-coaps needs to be able to fragment EST messages into multiple DTLS datagrams. Fine-grained fragmentation of EST messages is essential.

To perform fragmentation in CoAP, [RFC7959] specifies the "Block1" option for fragmentation of the request payload and the "Block2" option for fragmentation of the return payload of a CoAP flow.

The BLOCK draft defines SZX in the Block1 and Block2 option fields. These are used to convey the size of the blocks in the requests or responses.

The CoAP client MAY specify the Block1 size and MAY also specify the Block2 size. The CoAP server MAY specify the Block2 size, but not the Block1 size. As explained in Section 1 of [RFC7959]), blockwise transfers SHOULD be used in Confirmable CoAP messages to avoid the exacerbation of lost blocks.

The Size1 response MAY be parsed by the client as a size indication of the Block2 resource in the server response or by the server as a request for a size estimate by the client. Similarly, Size2 option defined in BLOCK should be parsed by the server as an indication of the size of the resource carried in Block1 options and by the client as a maximum size expected in the 4.13 (Request Entity Too Large) response to a request.

Examples of fragmented messages are shown in Appendix B.

5.8. Deployment limits

Although EST-coaps paves the way for the utilization of EST for constrained devices on constrained networks, some devices will not have enough resources to handle the large payloads that come with EST-coaps. The specification of EST-coaps is intended to ensure that EST works for networks of constrained devices that choose to limit their communications stack to UDP/CoAP. It is up to the network designer to decide which devices execute the EST protocol and which do not.

6. Discovery and URI

EST-coaps is targeted to low-resource networks with small packets. Saving header space is important and a short EST-coaps URI (see Table 2) is specified that is shorter than the EST URI specified in [RFC7030]. The individual EST-coaps well-known server URIs differ from the EST URI by replacing the scheme https by coaps and by specifying shorter resource path names:
coaps://example.com:<port>/.well-known/est/<short-est>
coaps://example.com:<port>/.well-known/est/ArbitraryLabel/<short-est>

The ArbitraryLabel Path-Segment, if used, SHOULD be of the shortest length possible (See sections 3.1 and 3.2.2 of [RFC7030]. Following [RFC7030] discovery is not needed when the client is preconfigured with the /.well-known/est server URI and the coaps port 5684.

The additional EST-coaps server URIs, obtained through discovery of the EST root resource(s) as shown below, are of the form:

coops://example.com:<port>/<root-resource>/<short-est>
coaps://example.com:<port>/<root-resource>/ArbitraryLabel/<short-est>

In the context of CoAP, the presence and location of (path to) the management data are discovered by sending a GET request to "/.well-known/core" including a resource type (RT) parameter with the value "ace.est" [RFC6690]. Upon success, the return payload will contain the root resource of the EST resources. It is up to the implementation to choose its root resource; throughout this document the example root resource /est is used.

The optional additional EST-coaps server URIs, obtained through discovery of the EST root resource(s) as shown below, are of the form:

coops://example.com:<port>/<root-resource>/<short-est>
coaps://example.com:<port>/<root-resource>/ArbitraryLabel/<short-est>

Figure 5 in section 3.2.2 of [RFC7030] enumerates the operations and corresponding paths which are supported by EST. Table 2 provides the mapping from the EST URI path to the shorter EST-coaps URI path.

<table>
<thead>
<tr>
<th>EST</th>
<th>EST-coaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>/cacerts</td>
<td>/crts</td>
</tr>
<tr>
<td>/simpleenroll</td>
<td>/sen</td>
</tr>
<tr>
<td>/simplereenroll</td>
<td>/sren</td>
</tr>
<tr>
<td>/csrattrs</td>
<td>/att</td>
</tr>
<tr>
<td>/serverkeygen</td>
<td>/skg</td>
</tr>
</tbody>
</table>

Table 2: Short EST-coaps URI path

The short resource URIs MUST be supported. The corresponding longer URIs specified in [RFC7030] MAY be supported.
When discovering the root path for the EST resources, the server MAY return all available resource paths and the used content types. This is useful when multiple content types are specified for EST-coaps server and optional functions are available. The example below shows the discovery of the presence and location of EST-coaps resources. Linefeeds are included only for readability.

REQ: GET /.well-known/core?rt=ace.est*

RES: 2.05 Content
</est>; rt="ace.est",
</est/crts>; rt="ace.est.crts"; ct=281,
</est/sen>; rt="ace.est.sen"ct=281 286,
</est/sren>; rt="ace.est.sren"ct=281 286,
</est/att>; rt="ace.est.att"ct=285,
</est/skg>; rt="ace.est.skg"ct=280 286 TBD8

The first line of the discovery response MUST be returned. The five consecutive lines MAY be returned. The return of the content-types in the last four lines allows the client to choose the most appropriate one from multiple content types.

Port numbers, not returned in the example, are assumed to be the default numbers 5683 and 5684 for coap and coaps respectively (sections 12.6 and 12.7 of [RFC7252]. Discoverable port numbers MAY be returned in the <href> of the payload.

7. DTLS Transport Protocol

EST-coaps depends on a secure transport mechanism over UDP that can secure (confidentiality, authenticity) the exchanged CoAP messages.

DTLS is one such secure protocol. When "TLS" is referred to in the context of EST, it is understood that in EST-coaps, security is provided using DTLS instead. No other changes are necessary (all provisional modes etc. are the same as for TLS).

CoAP was designed to avoid fragmentation. DTLS is used to secure CoAP messages. However, fragmentation is still possible at the DTLS layer during the DTLS handshake when using ECC ciphersuites. If fragmentation is necessary, "DTLS provides a mechanism for fragmenting a handshake message over several records, each of which can be transmitted separately, thus avoiding IP fragmentation" [RFC6347].

CoAP and DTLS can provide proof of identity for EST-coaps clients and server with simple PKI messages conformant to section 3.1 of
[RFC5272]. EST-coaps supports the certificate types and Trust Anchors (TA) that are specified for EST in section 3 of [RFC7030].

Channel-binding information for linking proof-of-identity with connection-based proof-of-possession is optional for EST-coaps. When proof-of-possession is desired, a set of actions are required regarding the use of tls-unique, described in section 3.5 in [RFC7030]. The tls-unique information translates to the contents of the first "Finished" message in the (D)TLS handshake between server and client [RFC5929]. The client is then supposed to add this "Finished" message as a ChallengePassword in the attributes section of the PKCS#10 Request Info to prove that the client is indeed in control of the private key at the time of the TLS session when performing a /simpleenroll, for example. In the case of EST-coaps, the same operations can be performed during the DTLS handshake. For DTLS 1.2, in the event of handshake message fragmentation, the Hash of the handshake messages used in the MAC calculation of the Finished message

\[ \text{PRF(master_secret, finished_label, Hash(handshake_messages))} \]

\[ \text{[0..verify_data_length-1];} \]

MUST be computed as if each handshake message had been sent as a single fragment [RFC6347]. Similarly, for DTLS 1.3, the Finished message

\[ \text{HMAC(finished_key, Transcript-Hash(Handshake Context, Certificate*, CertificateVerify*))} \]

* Only included if present.

MUST be computed as if each handshake message had been sent as a single fragment following the algorithm described in 4.4.4 of [RFC8446].

In a constrained CoAP environment, endpoints can’t afford to establish a DTLS connection for every EST transaction. Authenticating and negotiating DTLS keys requires resources on low-end endpoints and consumes valuable bandwidth. The DTLS connection SHOULD remain open for persistent EST connections. For example, an EST cacerts request that is followed by a simpleenroll request can use the same authenticated DTLS connection. Given that after a successful enrollment, it is more likely that a new EST transaction will take place after a significant amount of time, the DTLS connections SHOULD only be kept alive for EST messages that are relatively close to each other. In some cases, such as NAT rebinding, keeping the state of a connection is not possible when
devices sleep for extended periods of time. In such occasions, [I-D.rescorla-tls-dtls-connection-id] negotiates a connection ID that can eliminate the need for new handshake and its additional cost.

8. HTTPS-CoAPS Registrar

In real-world deployments, the EST server will not always reside within the CoAP boundary. The EST-server can exist outside the constrained network in a non-constrained network that supports TLS/HTTP. In such environments EST-coaps is used by the client within the CoAP boundary and TLS is used to transport the EST messages outside the CoAP boundary. A Registrar at the edge is required to operate between the CoAP environment and the external HTTP network. The EST coaps-to-HTTPS Registrar MUST terminate EST-coaps and authenticate the client downstream and initiate EST connections over TLS upstream.

The Registrar SHOULD authenticate the client downstream and it should be authenticated by the EST server or CA upstream. The Registration Authority (re-)creates the secure connection from DTLS to TLS and vice versa. A trust relationship SHOULD be pre-established between the Registrar and the EST servers to be able to proxy these connections on behalf of various clients.

When enforcing Proof-of-Possession (POP) linking, the (D)TLS tls-unique value of the (D)TLS session needs to be used to prove that the private key corresponding to the public key is in the possession of and was used to establish the connection by an end-entity or client. To do that the CSR the client is using needs to include information from the DTLS connection the client establishes with the server. In EST, that information is the (D)TLS tls-unique value of the (D)TLS session. In the presence of ESTcoaps-to-HTTPS Registrar, the EST-coaps client MUST be authenticated and authorized by the Registrar and the Registrar MUST be authenticated as an EST Registrar client to the EST server. Thus the POP linking information is lost between the EST-coaps client and the EST server. The EST server becomes aware of the presence of an EST Registrar from its TLS client certificate that includes id-kp-cmcRA [RFC6402] extended key usage extension. As explained in Section 3.7 of [RFC7030], the EST server SHOULD apply an authorization policy consistent with a Registrar client. For example, it could be configured to accept POP linking information that does not match the current TLS session because the authenticated EST client Registrar has verified this information when acting as an EST server.

For some use cases, clients that leverage server-side key generation might prefer for the enrolled keys to be generated by the Registrar if the CA does not support server-side key generation. In these
cases the Registrar MUST support the random number generation using proper entropy and is responsible for generating a new CSR signed by a new key which will be returned to the client along with the certificate from the CA.

One possible use-case, shown in one figure below, is expected to be deployed in practice:

![Diagram showing the flow of communication between the Registrar, EST server, CoAPS, and EST client in a constrained network.]

Table 2 contains the URI mapping between the EST-coaps and EST the Registrar SHOULD adhere to. Section 7 of [RFC8075] and Section 5.4 define the mapping between EST-coaps and HTTP response codes, that determines how the Registrar translates CoAP response codes from/to HTTP status codes. The mapping from Content-Type to media type is defined in Section 10. The conversion from CBOR major type 2 to base64 encoding needs to be done in the Registrar. Conversion is possible because a TLS link exists between EST-coaps-to-HTTP Registrar and EST server and a corresponding DTLS link exists between EST-coaps-to-HTTP Registrar and EST client.

Due to fragmentation of large messages into blocks, an EST-coaps-to-HTTP Registrar MUST reassemble the BLOCKs before translating the binary content to Base-64, and consecutively relay the message upstream.

For the discovery of the EST server by the EST client in the coap environment, the EST-coaps-to-HTTP Registrar MUST announce itself according to the rules of Section 6. The available actions of the Registrars MUST be announced with as many resource paths. The discovery of EST server in the http environment follow the rules specified in [RFC7030].
9. Parameters

This section addresses transmission parameters described in sections 4.7 and 4.8 of the CoAP document [RFC7252].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK_TIMEOUT</td>
<td>2 seconds</td>
</tr>
<tr>
<td>ACK_RANDOM_FACTOR</td>
<td>1.5</td>
</tr>
<tr>
<td>MAX_RETRANSMIT</td>
<td>4</td>
</tr>
<tr>
<td>NSTART</td>
<td>1</td>
</tr>
<tr>
<td>DEFAULT_LEISURE</td>
<td>5 seconds</td>
</tr>
<tr>
<td>PROBING_RATE</td>
<td>1 byte/second</td>
</tr>
</tbody>
</table>

Figure 4: EST-COAP protocol parameters

EST does not impose any unique parameters that affect the CoAP parameters in Table 2 and 3 in the CoAP draft but the ones in CoAP could be affecting EST. For example, the processing delay of CAs could be less than 2s, but in this case they should send a CoAP ACK every 2s while processing.

The main recommendation, based on experiments using Nexus Certificate Manager with Californium for CoAP support, communicating with a ContikiOS and tinyDTLS based client, from RISE SICS, is to start with the default CoAP configuration parameters.

However, depending on the implementation scenario, resending and timeouts can also occur on other networking layers, governed by other configuration parameters.

Some further comments about some specific parameters, mainly from Table 2 in [RFC7252]:

- **DEFAULT_LEISURE**: This setting is only relevant in multicast scenarios, outside the scope of the EST-coaps draft.

- **NSTART**: Limit the number of simultaneous outstanding interactions that a client maintains to a given server. The default is one, hence is the risk of congestion or out-of-order messages already limited.

- **PROBING_RATE**: A parameter which specifies the rate of re-sending non-confirmable messages. The EST messages are defined to be sent as CoAP confirmable messages, hence the PROBING_RATE setting is not applicable.

Finally, the Table 3 parameters are mainly derived from the more basic Table 2 parameters. If the CoAP implementation allows setting
them directly, they might need to be updated if the table 2 parameters are changed.

10. IANA Considerations

10.1. Content-Format Registry

Additions to the sub-registry "CoAP Content-Formats", within the "CoRE Parameters" registry are specified in Table 3. These have been registered temporarily in the Expert Review range (0-255).

<table>
<thead>
<tr>
<th>HTTP Media-Type</th>
<th>Encoding</th>
<th>ID</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/pkcs7-mime; smime-type=server-generated-key</td>
<td>-</td>
<td>280</td>
<td>[I-D.ietf-lamps-rfc5751-bis] [RFC7030]</td>
</tr>
<tr>
<td>application/pkcs7-mime; smime-type=certs-only</td>
<td>-</td>
<td>281</td>
<td>[I-D.ietf-lamps-rfc5751-bis]</td>
</tr>
<tr>
<td>application/pkcs7-mime; smime-type=CMC-request</td>
<td>-</td>
<td>282</td>
<td>[I-D.ietf-lamps-rfc5751-bis] [RFC5273]</td>
</tr>
<tr>
<td>application/pkcs7-mime; smime-type=CMC-response</td>
<td>-</td>
<td>283</td>
<td>[I-D.ietf-lamps-rfc5751-bis] [RFC5273]</td>
</tr>
<tr>
<td>application/pkcs8</td>
<td>-</td>
<td>284</td>
<td>[I-D.ietf-lamps-rfc5751-bis] [RFC5958]</td>
</tr>
<tr>
<td>application/csrattrs</td>
<td>-</td>
<td>285</td>
<td>[RFC7030] [RFC7231]</td>
</tr>
<tr>
<td>application/pkcs10</td>
<td>-</td>
<td>286</td>
<td>[I-D.ietf-lamps-rfc5751-bis] [RFC5967]</td>
</tr>
</tbody>
</table>

Table 3: New CoAP Content-Formats

10.2. Resource Type registry

This memo registers a new Resource Type (rt=) Link Target Attributes in the "Resource Type (rt=) Link Target Attribute Values" subregistry under the "Constrained RESTful Environments (CoRE) Parameters" registry.

- rt="ace.est". This EST resource is used to query and return the supported EST resources of a CoAP server.

- rt="ace.est.certs". This resource depicts the support of EST get cacerts.

- rt="ace.est.sen". This resource depicts the support of EST simple enroll.
11. Security Considerations

11.1. EST server considerations

The security considerations of Section 6 of [RFC7030] are only partially valid for the purposes of this document. As HTTP Basic Authentication is not supported, the considerations expressed for using passwords do not apply.

Given that the client has only limited resources and may not be able to generate sufficiently random keys to encrypt its identity, it is possible that the client uses server generated private/public keys to encrypt its certificate. The transport of these keys is inherently risky. A full probability analysis MUST be done to establish whether server side key generation enhances or decreases the probability of identity stealing.

When a client uses the Implicit TA database for certificate validation, the client cannot verify that the implicit database can act as an RA. It is RECOMMENDED that such clients include "Linking Identity and POP Information" Section 7 in requests (to prevent such requests from being forwarded to a real EST server by a man in the middle). It is RECOMMENDED that the Implicit Trust Anchor database used for EST server authentication be carefully managed to reduce the chance of a third-party CA with poor certification practices from being trusted. Disabling the Implicit Trust Anchor database after successfully receiving the Distribution of CA certificates response (Section 4.1.3 of [RFC7030]) limits any risk to the first DTLS exchange.

In accordance with [RFC7030], TLS cipher suites that include "_EXPORT_" and "_DES_" in their names MUST NOT be used. More information about recommendations of TLS and DTLS are included in [RFC7525].

As described in CMC, Section 6.7 of [RFC5272], "For keys that can be used as signature keys, signing the certification request with the private key serves as a POP on that key pair". The inclusion of tls-unique in the certification request links the proof-of-possession to...
the TLS proof-of-identity. This implies but does not prove that the authenticated client currently has access to the private key.

Regarding the Certificate Signing Request (CSR), an adversary could exclude attributes that a server may want, include attributes that a server may not want, and render meaningless other attributes that a server may want. The CA is expected to be able to enforce policies to recover from improper CSR requests.

Interpreters of ASN.1 structures should be aware of the use of invalid ASN.1 length fields and should take appropriate measures to guard against buffer overflows, stack overruns in particular, and malicious content in general.

11.2. HTTPS-CoAPS Registrar considerations

The Registrar proposed in Section 8 must be deployed with care, and only when the recommended connections are impossible. When POP linking is used the Registrar terminating the TLS connection establishes a new one with the upstream CA. Thus, it is impossible for POP linking to be enforced end-to-end for the EST transaction. The EST server could be configured to accept POP linking information that does not match the current TLS session because the authenticated EST Registrar client has verified this information when acting as an EST server. The introduction of an EST-coaps-to-HTTP Registrar assumes the client can trust the registrar using its implicit or explicit TA database. It also assumes the Registrar has a trust relationship with the upstream EST server in order to act on behalf of the clients.

In a server-side key generation case, if no end-to-end encryption is used, the Registrar may be able see the private key as it acts as a man-in-the-middle. Thus, the clients puts its trust on the Registrar not exposing the private key.

Clients that leverage server-side key generation have no knowledge if the Registrar will be generating the keys and enrolling the certificates with the CA or if the CA will be responsible for generating the keys, the existence of a Registrar requires the client to put its trust on the registrar doing the right thing if it is generating they private keys.

12. Acknowledgements

The authors are very grateful to Klaus Hartke for his detailed explanations on the use of Block with DTLS and his support for the content-format specification. The authors would like to thank Esko Dijk and Michael Verschoor for the valuable discussions that helped
in shaping the solution. They would also like to thank Peter Panburana for his feedback on technical details of the solution. Constructive comments were received from Benjamin Kaduk, Eliot Lear, Jim Schaad, Hannes Tschofenig, Julien Vermillard, and John Manuel.

13. References

13.1. Normative References

[I-D.ietf-core-multipart-ct]

[I-D.ietf-lamps-rfc5751-bis]


13.2. Informative References

[I-D.rescorla-tls-dtls-connection-id]


Appendix A. EST messages to EST-coaps

This section takes all examples from Appendix A of [RFC7030], changes the payload from Base64 to binary and replaces the http headers by their CoAP equivalents.

The corresponding CoAP headers are only shown in Appendix A.1. Creating CoAP headers are assumed to be generally known.

Binary payload is a CBOR major type 2 (byte array), that is shown with a base16 (hexadecimal) CBOR diagnostic notation.

[EDNOTE: The payloads of the examples need to be re-generated with appropriate tools and example certificates.]

A.1. cacerts

These examples assume that the resource discovery, returned a short URL of "/est".

In EST-coaps, a coaps cacerts IPv4 message can be:

GET coaps://192.0.2.1:8085/est/crts

The corresponding CoAP header fields are shown below. The use of block and DTLS are worked out in Appendix B.
Ver = 1
T = 0 (CON)
Code = 0x01 (0.01 is GET)
Token = 0x9a (client generated)
Options
  Option1 (Uri-Host) [optional]
    Option Delta = 0x3 (option nr = 3)
    Option Length = 0x9
    Option Value = 192.0.2.1
  Option2 (Uri-Port) [optional]
    Option Delta = 0x4 (option nr = 3+4=7)
    Option Length = 0x4
    Option Value = 8085
  Option3 (Uri-Path)
    Option Delta = 0x4 (option nr = 7+4= 11)
    Option Length = 0x5
    Option Value = "est"
  Option4 (Uri-Path)
    Option Delta = 0x0 (option nr = 11+0= 11)
    Option Length = 0x6
    Option Value = "crtst"
  Option5 (Max-Age)
    Option Delta = 0x3 (option nr = 11+3= 14)
    Option Length = 0x1
    Option Value = 0x1 (1 minute)
Payload = [Empty]

A 2.05 Content response with a cert in EST-coaps will then be:

2.05 Content (Content-Format: 281)
  {payload}

with CoAP fields

Ver = 1
T = 2 (ACK)
Code = 0x45 (2.05 Content)
Token = 0x9a (copied by server)
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option nr =12)
    Option Length = 0x2
    Option Value = 281 (defined in this document)

Payload =
h’30233906092a6206734107028c2a3023260201013100300b06092a6206734107018c0c3020bb302063c20102020900a61e75193b7acc0d06092a620673410105050030
The hexadecimal dump of the CBOR payload looks like:

<table>
<thead>
<tr>
<th>09CD</th>
<th># bytes (2509)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3023906092A6206734107028C2A032326202013100300B6092A620673410702</td>
<td></td>
</tr>
</tbody>
</table>

The hexadecimal dump of the CBOR payload looks like:

The hexadecimal dump of the CBOR payload looks like:
A.2.  csrattrs

In the following valid /csrattrs exchange, the EST-coaps client authenticates itself with a certificate issued by the connected CA.

The initial DTLS handshake is identical to the enrollment example. The IPv6 CoAP GET request looks like:

REQ:
GET coaps://[2001:db8::2:1]:61616/est/att
(Content-Format: 285)

A 2.05 Content response contains attributes which are relevant for the authenticated client. In this example, the EST-coaps server returns two attributes that the client can ignore when they are unknown to him.

A.3.  enroll / reenroll

During the Enroll/Reenroll exchange, the EST-coaps client uses a CSR (Content-Format 286) request in the POST request payload.

After verification of the CSR by the server, a 2.05 Content response with the issued certificate will be returned to the client. As described in Section 5.5, if the server is not able to provide a response immediately, it sends an empty ACK with response code 5.03 (Service Unavailable) and the Max-Age option. See Figure 3 for an example exchange.

[EDNOTE: When redoing this example, given that POP linking is also used, make sure it is obvious that the ChallengePassword attribute in the CSR is valid HMAC output. HMAC-REAL.]
Internet-Draft                  EST-coaps                   October 2018

[23x778]POST [2001:db8::2:1]:61616/est/sen
(token 0x45)

h'30208530206d020100301031d301b0603550403131464656d6f7374657034203
133368313433335320362300d6092a620673410101050003204f0030204a
202401005d9f4dfdfd3c5949f64a9584367778560950b355c35b8e34726d37674
54321734795b4c09b9c6d75d408311307a81f7adeff7f5d241f7d5b8e85620c5d44
38bb4422cf215c167f2ccf36c364ea2618a62f0536576369d604e6a96877224
7d86824f079faac7a6f694cfda5b84c42087dc062d462190c525813f210a036a77
37b4f30d889f4b75559fb72752453146332d51c937557716ccec624f5125c3a4
44ad3115020048113efef54ad554ee88af09a2583aac920475113db499b0b786
b871691e0f0203010001870f0609a2620673410907311213102b7272436972f2
372b455975350343400d6092a6206734101050003204100441b0177a3a6
5501487735a8ad5d3827a4eaa867013920e2afcd87aa81733c7c0353be471ebf
a7cda7e67ccc6e22ae03498588d5f2e3b143f2b1a6175e5c44e8e7625af6b
836fd4416894c2e55eaa9c660f69075d653475d410729aa6d806affb9986caf
7b844b5b3e4545f1907186ada00760cdad6db26a592d4a7bda7d586b68110962
17071103407553155ccddc75481e272b5ed553a859fb7e251006f7605085dab4
fc7e0731f0e7fe305703791362d5157e92e6b5c2e3edcbad4b0'

RET:

h'3020f806092a620673410702839202e50201013100300b06092a62067341070
1830b3020c730206fcd2010200115300d06092a620673410105000301b311930
17060355040313106573474578616d706c65434120e774e301e170d31330353
0393233133533535a170d313430353093323313353353a301f31d301b060355
0403131464656df773746570342031336383134313335320206230d06092a6
20673410101050003204f0030204a02204100459df4dfdd3c5949f64a95843677
78560950b355c35b8e34726dd376454231734795b4c09b9c6d7540831107a81
f7adef7f5d241f7d5be85620c5d4438bb4242cf215c167f2ccf36c364ea2618a
62f0536576369d6304e6a968772247d86824f079faac7a6f694cfda5b84c42084
dc062d462190c525813f210a036a737b4f3d0889f4fb75559fb72752453146332
51c937557716ccec624f5125c3a4447ad3115020048113efef54ad554ee88af09
a2583aac920475113db499b0b7g6e3071691e0f020301000134b050300e60355
10d1f001040320401cd0603551d0e0416041e81d0788aa27103045c5edd41e
065701f0603551d203418031653112966e3047671372fbeb6a2c823c300d06092a
62067341010500032041002910d86f2ffeeb914c046816871d601567d291b4
3fabeef0e0f8ef81cea27302a7133e20e9d04029866a9863c7d14e26fbeb8a9ab1b
77fbb12414bb9c906fbce381137ec1de685f790403e416b8482997174b691637
5a4e1c4bf7474c7572b4b2c6bade9f35da786392ee0d9e3970542565f3886ad6
7746db12484bb02616ee3302dc371c6006e431fb7c457598dd204b36b0b3d3
258760a303f1102db26327f9297b5c7a60173e1799491b69150248756026b80553
171e4733ad3d13c0103100'

A.4. serverkeygen

During this valid /serverkeygen exchange, the EST-coaps client authenticates itself using the certificate provided by the connected CA.

The initial DTLS handshake is identical to the enrollment example. The CoAP GET request looks like:

[EDNOTE: same comment as HMAC-REAL above applies.]

[EDNOTE: Suggestion to have only one example with complete encrypted payload (the short one) and point out the different fields. Update this example according to the agreed upon solution from Section 5.6.]

POST coaps://192.0.2.1:8085/est/skg
(token 0xa5)
(Content-Format: 286) (Max-Age=120)

```
h’302081302069020100305b313e303c0603550403133537657265724b65794765e2072657120627920636c69656e742066656e646f776e73696f6e696e6720696e206465626f6d656564656e74696d656e7473203133363831343139353119301706035504053105104943a5769646765726520643a313030262300d06092a62067341010105000320f003020a02204100f4
```

RET:

2.01 Content (Content-Format: TBD8)
(token=0xa5)

```
h’30213e020100305b313e303c0603550403133537657265724b65794765e2072657120627920636c69656e742066656e646f776e73696f6e696e6720696e206465626f6d656564656e74696d656e7473203133363831343139353119301706035504053105104943a5769646765726520643a313030262300d06092a62067341010105000320f003020a02204100f4
```

RET:

2.01 Content (Content-Format: TBD8)
(token 0xa5)

```
h’30213e020100305b313e303c0603550403133537657265724b65794765e2072657120627920636c69656e742066656e646f776e73696f6e696e6720696e206465626f6d656564656e74696d656e7473203133363831343139353119301706035504053105104943a5769646765726520643a313030262300d06092a62067341010105000320f003020a02204100f4
```

Without the DecryptKeyIdentifier attribute, the response has no additional encryption beyond DTLS.

The response contains first a preamble that can be ignored. The EST-coaps server can use the preamble to include additional explanations, like ownership or support information.

Appendix B. EST-coaps Block message examples

Two examples are presented: (1) a cacerts exchange shows the use of Block2 and the block headers, and (2) a enroll exchange shows the Block1 and Block2 size negotiation for request and response payloads.

B.1. cacerts block example

This section provides a detailed example of the messages using DTLS and BLOCK option Block2. The minimum PMTU is 1280 bytes, which is the example value assumed for the DTLS datagram size. The example block length is taken as 64 which gives an SZX value of 2.

The following is an example of a valid /cacerts exchange over DTLS. The content length of the cacerts response in appendix A.1 of [RFC7030] is 4246 bytes using base64. This leads to a length of 2509 bytes in binary. The CoAP message adds around 10 bytes, the DTLS record 29 bytes. To avoid IP fragmentation, the CoAP block option is used and an MTU of 127 is assumed to stay within one IEEE 802.15.4 packet. To stay below the MTU of 127, the payload is split in 39 packets with a payload of 64 bytes each, followed by a packet of 13 bytes. The client sends an IPv6 packet containing the UDP datagram with the DTLS record that encapsulates the CoAP Request 40 times. The server returns an IPv6 packet containing the UDP datagram with the DTLS record that encapsulates the CoAP response. The CoAP request-response exchange with block option is shown below. Block option is shown in a decomposed way (block-option:NUM/M/size) indicating the kind of Block option (2 in this case because used in the response) followed by a colon, and then the block number (NUM), the more bit (M = 0 in lock2 response means last block), and block size with exponent (2**(SZX+4)) separated by slashes. The Length 64 is used with SZX= 2 to avoid IP fragmentation. The CoAP Request is sent with confirmable (CON) option and the content format of the Response is /application/cacerts.
GET /192.0.2.1:8085/est/crts   (2:0/0/64)    -->
   (2:0/1/64) 2.05 Content
GET /192.0.2.1:8085/est/crts   (2:1/0/64)    -->
   (2:1/1/64) 2.05 Content

GET /192.0.2.1:8085/est/crts    (2:39/0/64)  -->
   (2:39/0/64) 2.05 Content

40 blocks have been sent with partially filled block NUM=39 as last block.

For further detailing the CoAP headers, the first two blocks are written out.

The header of the first GET looks like:

Ver = 1
T = 0 (CON)
Code = 0x01 (0.1 GET)
Token = 0x9a    (client generated)
Options
  Option1 (Uri-Host)            [optional]
    Option Delta = 0x3  (option nr = 3)
    Option Length = 0x9
    Option Value = 192.0.2.1
  Option2 (Uri-Port)            [optional]
    Option Delta = 0x4   (option nr = 3+4=7)
    Option Length = 0x4
    Option Value = 8085
  Option3 (Uri-Path)
    Option Delta = 0x4    (option nr = 7+4=11)
    Option Length = 0x5
    Option Value = "est"
  Option4 (Uri-Path)
    Option Delta = 0x0    (option nr = 11+0=11)
    Option Length = 0x6
    Option Value = "crts"
Payload = [Empty]

The header of the first response looks like:
Ver = 1
T = 2 (ACK)
Code = 0x45 (2.05 Content)
Token = 0x9a (copied by server)

Options
  Option1 (Content-Format)
    Option Delta = 0xC (option nr = 12)
    Option Length = 0x2
    Option Value = 281
  Option2 (Block2)
    Option Delta = 0xB (option 23 = 12 + 11)
    Option Length = 0x1
    Option Value = 0x0A (block number = 0, M=1, SZX=2)

Payload =
h'30233906092a6206734107028c2a3023260201013100300b06092a6206734107018c0c3020bb302063c2010202000a61e75193b7acc0d06092a6206734101'
Ver = 1
T = 2 (means ACK)
Code = 0x45      (2.05 Content)
Token = 0x9a     (copied by server)
Options
  Option1 (Content-Format)
    Option Delta = 0xC  (option nr =12)
    Option Length = 0x2
    Option Value = 281
  Option2 (Block2)
    Option Delta = 0xB  (option 23 = 12 + 11)
    Option Length = 0x2
    Option Value = 0x272 (block number = 39, M=0, SZX=2)
Payload = h’73a30d0c006343116f58403100’

B.2. enroll block example

In this example the requested block2 size of 256 bytes, required by
the client, is transferred to the server in the very first request
message. The request/response consists of two parts: part1
containing the CSR transferred to the server, and part2 contains the
certificate transferred back to the client. The block size
256=(2**(SZX+4)) which gives SZX=4. The notation for block numbering
is the same as in Appendix B.1. It is assumed that CSR takes N1+1
blocks and Cert response takes N2+1 blocks. The header fields and
the payload are omitted to show the block exchange. The type of
payload is shown within curly brackets.
POST [2001:db8::2:1]:61616/est/sen (CON) (1:0/1/256) {CSR req} -->
   <-- (ACK) (1:0/1/256) (2.31 Continue)
POST [2001:db8::2:1]:61616/est/sen (CON) (1:1/1/256) {CSR req} -->
   <-- (ACK) (1:1/1/256) (2.31 Continue)

Figure 5: EST-COAP enrolment with multiple blocks

N1+1 blocks have been transferred from client to server and N2+1 blocks have been transferred from server to client.

Authors’ Addresses

Peter van der Stok
Consultant
Email: consultancy@vanderstok.org

Panos Kampanakis
Cisco Systems
Email: pkampana@cisco.com

Sandeep S. Kumar
Philips Lighting Research
High Tech Campus 7
Eindhoven  5656 AE
NL
Email: ietf@sandeep.de