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 December 23, 2018.
Table of Contents

1. Introduction .................................................. 3
2. Terminology ................................................... 3
3. Conformance to RFC7925 profiles .......................... 3
4. Protocol Design ................................................. 4
   4.1. Payload format ............................................. 5
       4.1.1. Content Format application/multipart-core .... 6
   4.2. Message Bindings .......................................... 6
   4.3. CoAP response codes ....................................... 6
   4.4. Delayed Results .......................................... 7
   4.5. Server-side Key Generation .............................. 8
   4.6. Message fragmentation ................................... 9
   4.7. Deployment limits ....................................... 10
5. Discovery and URI .............................................. 10
6. DTLS Transport Protocol ..................................... 11
7. HTTPS-CoAPS Registrar ....................................... 13
8. Parameters ................................................... 15
9. IANA Considerations ........................................ 16
   9.1. Content-Format Registry ............................... 16
   9.2. Resource Type registry .................................. 17
10. Security Considerations .................................... 17
    10.1. EST server considerations .......................... 17
    10.2. HTTPS-CoAPS Registrar considerations ............. 18
11. Acknowledgements ............................................. 19
12. Change Log .................................................. 19
13. References .................................................. 20
    13.1. Normative References .................................. 20
    13.2. Informative References .............................. 21
Appendix A. EST messages to EST-coaps ....................... 23
    A.1. cacerts ................................................ 23
    A.2. csrattrs ................................................ 28
    A.3. enroll / reenroll ..................................... 29
1. 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 [RFC5246], 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. CoAP Observe options [RFC7641] are also used to convey delayed EST responses to clients.

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.

2. 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.

3. 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 key generation as described in section 4.4 of [RFC7030] and discussed in Section 4.5 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 defined in [RFC7251], and the the curve secp256r1 MUST be supported [RFC4492]; 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 applies.

DTLS1.2 implementations MUST use the Supported Elliptic Curves and Supported Point Formats Extensions [RFC4492]. 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 re-enrollment 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.

4. 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 4.6. The Figure 1 below shows the layered EST-coaps architecture.
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.

4.1. 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 9. The CoAP response codes are defined in Section 4.3.

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. Therefore, the content formats specification in Section 4.1.1 specifies that the binary payload is transported as a CBOR major type 2, a byte string, for all EST-coaps Content-Formats. 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.
4.1.1. 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.fossati-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, one with content format ID TBD5 and one with content format ID TBD2, looks like this in diagnostic CBOR notation: [TBD5,h’0123456789abcdef’,TBD2,h’fedcba9876543210’]. An example is shown in Appendix A.4.

4.2. 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.

4.3. 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. In Section 4.4 it is specified how EST requests over CoAP handle delayed messages.
All other HTTP 2xx response codes are not used by EST. For the following HTTP 4xx error codes that may occur: 400, 401, 403, 404, 405, 406, 412, 413, 415; the equivalent CoAP response code for EST-coaps is 4.xx. For the HTTP 5xx error codes: 500, 501, 502, 503, 504 the equivalent CoAP response code is 5.xx.

4.4. Delayed Results

If the server is slow providing the response, she can respond with an empty ACK, sending the content later, according to [RFC7252], section 5.2.2. If the response will be more than one packet (requiring block mode) then the client needs to send an empty ACK with code 0.00 for the first block and acknowledge the rest of the blocks accordingly. To demonstrate this situation below we show a client sending an enrollment request that will use more than one Block1 blocks to send the CSR to the server. The server on the other hand will need more than one Block2 blocks to respond, but will need take some time to provide the response. Thus the server will use a 0.00 ACK for the response which will be provided when ready by using 2.04 messages and Block2 options. Readers should note that the client asks for a decrease in the block size when acknowledging the first Block2.

CON | POST 1:0/1/256 (enroll request with CSR) -->
---  ACK | 2.31 1:0/1/256
CON | POST 1:1/1/256 (enroll request with CSR)
---  ACK | 2.31 1:1/1/256
CON | POST 1:2/0/256 (enroll request with CSR)
---  ACK (code 0.00, no payload,
     to signal delay in the response.
     When ready, the server transfers
     the response in Block2 blocks.)
CON | 2.04 1:2/0/256 & 2:0/1/128 (Certificate) -->
---  ACK (code 0.00, no payload)
---  CON | POST 2:1/0/128
ACK | 2.04 2:1/1/128 (Certificate) -->
---  CON | POST 2:2/0/128
ACK | 2.04 2:2/0/128 (Certificate) -->

[EDNOTE: To update this. HTTP 202 Retry-After in EST needs an equivalent mechanism in EST-coaps. Observe seems like a candidate but after the HTTP 202 the client needs to do a new POST, not a GET, so Observe is not the best option. We could use 2.04 or a new 2.0x with Max-Age to convey the EST Retry-After. ] It is possible that responses are not always directly available by the server, and may even require manual intervention to generate the certificate for the server response. Delays of minutes to hours are possible. EST requires the use of an HTTP 202 message with a Retry-After header by
the server which signals to the client to attempt the request in a certain amount of time. In EST, each GET request MUST be accompanied by the observe option. When the result is directly available, the client receives the result and forgets about the observe as specified in section 3.6 of [RFC7641]. When a POST response is delayed, the POST returns a 2.01 (Created) response code, having put a value in the Location-Path option. After reception of 2.01 the client does a GET request with the observe option to the newly returned location. Once the delayed result is notified by the server, the client forgets about the observe.

Next to the observe option the server MUST specify the Max-Age option that indicates the maximum waiting time in minutes.

4.5. 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.

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 sever-side key generation response is returned using a CBOR array Section 4.1.1. 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].

4.6. 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.

4.7. 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.

5. Discovery and URI

EST-coaps is targeted to low-resource networks with small packets. Saving header space is important and an additional EST-coaps URI is specified that is shorter than the EST URI.

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 individual EST-coaps server URIs differ from the EST URI by replacing the scheme https by coaps and by specifying shorter resource path names:


The ArbitraryLabel Path-Segment SHOULD be of the shortest length possible.
Figure 5 in section 3.2.2 of [RFC7030] enumerates the operations and corresponding paths which are supported by EST. Table 1 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 1

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. The example below shows the discovery of the presence and location of management data.

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

RES: 2.05 Content
</est>; rt="ace.est"
</est/crts>;ct=TBD2
</est/sen>;ct=TBD2 TBD7
</est/sren>;ct=TBD2 TBD7
</est/att>;ct=TBD6
</est/skg>;ct=TBD1 TBD7 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.

6. 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
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

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

* 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 [I-D.ietf-tls-tls13].
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.

Support for Observe CoAP options [RFC7641] is compulsory. The necessity of Observer for long delays (minutes - hours) is explained in Section 4.4. Observe options could also be used by the server to notify clients about a change in the cacerts or csr attributes (resources) and might be an area of future work.

7. 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), 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 can be used by an end-entity or client. In other words, 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 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.

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

![Diagram]

ESTcoaps-to-HTTPS Registrar at the CoAP boundary.

Table 1 contains the URI mapping between the EST-coaps and EST the Registrar SHOULD adhere to. Section 7 of [RFC8075] and Section 4.3 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 9. 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 SHOULD 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 5. 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].

When server-side key generation is used, if the private key is protected using symmetric keys then the Registrar needs to encrypt the private key down to the client with one symmetric key and decrypt it from the server with another. If no private key encryption takes place the Registrar will be able to see the key as it establishes a separate connection to the server. In the case of asymmetrically encrypted private key, the Registrar may not be able to decrypt it if the server encrypted it with a public key that corresponds to a private key that belongs to the client.

8. 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 2: 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 then 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]:

van der Stok, et al. Expires December 23, 2018
o DEFAULT_LEISURE: This setting is only relevant in multicast scenarios, outside the scope of the EST-coaps draft.

o 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.

o 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.

9. IANA Considerations

9.1. Content-Format Registry

Additions to the sub-registry "CoAP Content-Formats", within the "CoRE Parameters" registry are specified in Table 2. These can be registered either in the Expert Review range (0-255) or IETF Review range (256-9999).
### 9.2. Resource Type registry

Additions to the sub-registry "CoAP Resource Type", within the "CoRE Parameters" registry are needed for a new resource type.

- rt="ace.est" needs registration with IANA.

### 10. Security Considerations

#### 10.1. 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 6 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.

10.2. HTTPS-CoAPS Registrar considerations

The Registrar proposed in Section 7 must be deployed with care, and only when the recommended connections are impossible. When POP is used the Registrar terminating the TLS connection establishes a new one with the upstream CA. Thus, it is impossible for POP to be enforced throughout 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, depending on the private key encryption method, 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.

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. Since the client has 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.

11. 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.

12. Change Log

-02:

  Added parameter discussion in section 8

  Concluded content-format specification using multipart-ct draft

  examples updated

-01:

  Editorials done.

  Redefinition of proxy to Registrar in Section 7. Explained better the role of https-coaps Registrar, instead of "proxy"

  Provide "observe" option examples
extended block message example.

inserted new server key generation text in Section 4.5 and motivated server key generation.

Broke down details for DTLS 1.3

New media type uses CBOR array for multiple content-format payloads

provided new content format tables

new media format for IANA

-00
copied from vanderstok-ace-coap-04

13. References

13.1. Normative References

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

[I-D.ietf-tls-tls13]


13.2. Informative References


van der Stok, et al. Expires December 23, 2018


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 (Observe)
  Option Delta = 0x1 (option nr = 3+3=6)
  Option Length = 0x1
  Option Value = 0 (register)
- Option3 (Uri-Port) [optional]
  Option Delta = 0x1 (option nr = 6+1=7)
  Option Length = 0x4
  Option Value = 8085
- Option4 (Uri-Path)
  Option Delta = 0x4 (option nr = 7+4= 11)
  Option Length = 0x5
  Option Value = "est"
- Option5 (Uri-Path)
  Option Delta = 0x0 (option nr = 11+0= 11)
  Option Length = 0x6
  Option Value = "crts"
- Option6 (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: TBD2) {payload}

with CoAP fields

Ver = 1
T = 2 (ACK)
Code = 0x45 (2.05 Content)
Token = 0x9a (copied by server)
Options

Option 1 (Observe)
  Option Delta = 0x6 (option nr = 6)
  Option Length = 0x1
  Option Value = 12 (12 > 0)

Option 2 (Content-Format)
  Option Delta = 0xC (option nr = 6 + 6 = 12)
  Option Length = 0x2
  Option Value = TBD2 (defined in this document)

Payload =
h'3023906092a6206734107028c2a30232620201013100300b06092a6206734107018
c0c30232b063c2010202090a61e75193b7acc2d06092a6206734107050030
1b311193017b0603550431301657374578616d706c654341204f774f30e170d313
30335039303353333315a170d313430353039333333315a301b3119301706
0355040313106573744578616d706c654341204f774f302062300d60092a6206734107034
1010100050024f0030204a0220103a923a2668bae44ae16ca4e2512c5020680
358482ac39df640e4574e654a35f481e054c5da3732877f7a1e42f9edf39584
32efb2106591d3eb78c1034709f2f251fc86566dbda2d41c79283e5ac4e9e18f4b
9f596e5ef2679cc321542bb1337f9a44dfc385f1516561fa968a191f4265bc0b8b2
76eb3106a79d97d34c837c79ef1c30b39642646a42c6284a9f6022e2693943
6880adafd95c98ca1d2fc2e6d75319b85d0458de289a13bf6d20ff7541f6a25d
7daf004355020301000130b040300f0603551d30101f05030030101fc1d0603551
d0e0014041084d321ca0135e77217a486b68b334000e0603551d0f010f4030
20106300d0609a62067341010500032041002370b365746a0c2c798666d7787
94f89b49a511fd369b28936ec2475c0f0855c8e83f823f2b871a1d92282f323c45
904ba008579216cf5223bb8bc425a0677262047f7700240631c17f3035d1c3780b
2385241ca1f4a6e96e6820306b3a786de5a557795d1893822347bf5f825d34a7
ad2876f8fe6ba4d525b310666f6505796f71530003431a3e6bfafe788b465029a7e20
a51107677552586152d051e8eebf383e922888983421d5c5652a4870c3af74b9dbde
6b462e2263d0f63d020c30206bc2010202010300d06092a620673410105000
301b31193017b0603550431301657374578616d706c654341204f774f30e170d313
1333050393033533333325a170d3134303530393333333325a301b3119301706
036355040313106573744578616d706c654341204f774f302062300d60092a6206734107034
3410101005003204f0030204a0220100e6f677a3247c1f03d9b9af113e5e7e1
1f49e402112eb8384e60f202630f544d5d056235713c79a7229283a790
8751a6324a2e3e2a4b1f01519d046f02f5a5d6d7670c948235660e767b79d49338
d1faa3b3ddd483060a207b0a09706070e45b052b60fdaee56b65e6a4f5b7b7
b0cfc87a979d221f1127313c53371ce1245d63db45a1203a2340a8042c7680d3b8
076a028d3a51187d2e87bbd62305ce5e6766872402f7276d9c14467c37de0f
55033f1b92a5ad2f19a2a71c2c301000134b050300e0603551d0f010f4030204c
1d6003551d0e041604112966e304761732f2bf6a2823c03f0603551d2304183
016508d321ca0135e77217a486b68b334b00d06092a62067341010500032041
00b382ba3355a50e8278bae15758b3b2eff63d34d3e357b90031495d018868e49589b
9faf46a4ad49b1d35b06ef38016677440934663c2c11c18365f5d4c10c3401
123d35387398db91fe1b4131b16c291d35730b3f933c7475124851555fe5fc647
e8fd029605367c7e01281f6617110021b0d10947de0e9f0ca6c764b6334784055
172c3983d3e3a3a82301a54fcc9b0670c543a1c747164619101ff23b24b0b2a26394

van der Stok, et al. Expires December 23, 2018
The hexadecimal dump of the CBOR payload looks like:

59 09CD # bytes (2509)
30233906092A6206734107028C2A3023260201013100300B6092A62067341070
18C0C3020BB302063CC0102020900A1E75193B7AC00D06092A6206734101050
00301B31913017060355040313106573445758616D706C65431204F774F301E1
70D31333305303930333533333335A170D3134303530393033353333333331A01B31
193017060353040313106573445758616D706C65431204F774F30262300D060
After reception of the 2.05 response, the client can forget the observe.

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: TBD6)(observe =0)(Max-Age =1)

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 TBD7) 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 4.4, if the server is not able to provide a response, then it ACKs the GET (with no payload), and the payload will be sent later as part of the OBSERVE processing.

[EDNOTE: When redoing this example, given that proof of possession (POP) is also used, make sure it is obvious that the ChallengePassword attribute in the CSR is valid HMAC output. HMAC-REAL.]
POST [2001:db8::2:1]:61616/est/sen
(token 0x45)
(Content-Format: TBD7)(observe 0)
h'30208530206d020100301f311d301b060350403131464656d6f737465703420
1333683134333532506262300d6092a62067341010105003204f0030204a022041005d9f4dfddf3c599f4f646a904367778560950b355c35b8e8276dd37674
54231734795b4c09b9c675d408311307a81f7adef7f5d241f7f5be85620c5d44
38bb4242cf215c167f2ccf363c364ea2618a62f0536576369d6304e6a96877224
7d86824f079faac7a6f694cfada5b84c42087dc062d462190c525813f210a036a7
37b4f30d8891f4b75559fb72752453146332d51c93757716c662c6125c3a4
447ad3115020048113fc5f4d554ee88af09a2583a9c024705113db4990b1876
b871691e0fo0203010018701f6092a62067341090731121302b77272436972f2
372b45597593503543200d6092a62067341010505003204100441b0177a3a6
5501487735a8ad5d387a4ea867013920e2afcd87a81733c7c0353be471ebf
a7cda5fe7cc6be22aae03498588df52de3b143b21a6175ec544e8e7625af6b
836fd441e694c2e55ea99c6606f69075d54374d5410729a6d806afbb9986caf
7b84b5bae455f419071865ada00760cad6d62a6592d4a7bda7586b68110962
1707110340755315ccddc75481e272b5ed553a859fb7e251006f7605085db4
fc7e0731f0e7fe305703791362d5157e92e6b5c2e3edcbad4’
RET:
(Content-Format: TBD2)(token =0x45)(observe =12)
2.01 Created
h'3020f806092a620673410728329020e50201013100300b06092a62067341070
1830b3020c730206fc20102020115300d6092a6206734101050500301b311930
17060355040313106573744578616d706c654341204e77e301e170d31333035
039323313535353a170d313430353093233313535353a301f31d301b060355
4003311464565d6f7374657034201333368313433133532302062300d6092a6
2067341010105003204f0030204a022041005d9f4dfddf3c5949f6646a958436
78560950b355c35b8e34726dd37645425213734795b4c09b9c675d408311307a81
f7adef7f5f241f7d5be85620c5d44383d4bb4242cf215c167f2ccf363c364ea2618a
62f0536576369d6304e6a968772247d86824f079faac7a6f694cfda5b84c42087
c062d462190c525813f210a036a7a737b4f3d08891f4b75559fb72752453146332
d15c93757716c6624f5125ca4447ad3115020048133ef5ad554ee88af09
a2583a9c024705113db4990b1876b871691e0f020301001340b05300e060355
1d0f10f0104020041d6063551d0e0416041e81d0788aa27130405c5edc41d
065701f6035051d2043180165311296e630476173f2fe6a2c823c300d6092a6
20673410105050003204102910d682f6f2f8e0b14c046816871d601567d291b4
3fafeb0f0e8f81cea27302a7133c2e09d04029866a9863c7d14e6febe8a0ab1b
77ffbb124bc99c7c96fbc381137ec1de685f9706c3e416b8d8297174bc691637
5a4e1c4bf774c7572b4b2c6bae9f35d3a876392ee0d95e3970542565f3886ad6
7746db12484bb02616e6302dc317c4e6331fb7c457598d204b367bb03d3
258760a303f1102db26327f929b7c5a60173e1799491b6915024875602680553
171e4733ad3d13c0103100’

After reception of the 2.01 response the client can forget the observe registration

van der Stok, et al. Expires December 23, 2018
The same example when delays occur (omitting the payloads in the examples) has a different behavior. The response to the POST is an empty payload with response code 2.01 (Created) that also returns the resource to query. The client issues a GET with an observe and a new token value, and waits for the notification after possibly receiving an empty payload first.

POST [2001:db8::2:1]:61616/est/sen
(token 0x45)(observe = 0)
(Content-Format: TBD7)(Max-Age=120)
[payload]
RET:
(token =0x45)(observe =12)(Location-Path=/est/1245)
2.01 Created
[empty payload]

GET [2001:db8::2:1]:61616/est/1245
(token 0x53)
(observe =0)(Max-Age=120)

RET:
(token =0x53)(observe = 5)
2.01 Created
[empty payload]

RET:
(token =0x53)(observe = 6)
(Content-Format: TBD2)
2.04 Changed
[payload]

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 4.5. ]
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 headers, and (2) a enroll exchange shows the Block1 and Block2 size negotiation for request and response payloads.

van der Stok, et al. Expires December 23, 2018
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 (Observe)
    Option Delta = 0x3  (option nr = 3+3=6)
    Option Length = 0x1
    Option Value = 0     (register)
  Option3 (Uri-Port) [optional]
    Option Delta = 0x4   (option nr = 3+4=7)
    Option Length = 0x4
    Option Value = 8085
  Option4 (Uri-Path)
    Option Delta = 0x4   (option nr = 7+4=11)
    Option Length = 0x5
    Option Value = "est"
  Option5 (Uri-Path)
    Option Delta = 0x0   (option nr = 11+0=11)
    Option Length = 0x6
    Option Value = "crls"
  Option6 (Max-Age)
    Option Delta = 0x3   (option nr = 11+3=14)
    Option Length = 0x1
    Option Value = 0x1   ( 1 minute)
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 (Observe)
    Option Delta = 0x6 (option nr=6)
    Option Length = 0x1
    Option Value = 12 (12 > 0)
  Option2 (Content-Format)
    Option Delta = 0xC (option nr =6+6=12)
    Option Length = 0x2
    Option Value = TBD2
  Option2 (Block2)
    Option Delta = 0xB (option 23 = 12 + 11)
    Option Length = 0x1
    Option Value = 0x0A (block number = 0, M=1, SZX=2)
Payload =
  h'30233906092a6206734107028c2a3023260201013100300b06092a6206734107018c0c3020bb302063c20102020900a61e75193b7acc0d06092a6206734101''

The second Block2:

Ver = 1
T = 2 (means ACK)
Code = 0x45 (2.05 Content)
Token = 0x9a (copied by server)
Options
  Option1 (Observe)
    Option Delta = 0x6 (option nr=6)
    Option Length = 0x1
    Option Value = 16 (16 > 12)
  Option2 (Content-Format)
    Option Delta = 0xC (option nr =6+6=12)
    Option Length = 0x2
    Option Value = TBD2
  Option2 (Block2)
    Option Delta = 0xB (option 23 = 12 + 11)
    Option Length = 0x1
    Option Value = 0x1A (block number = 1, M=1, SZX=2)
Payload =
  h'050500301b311930170603550403131065734578616d706c654341204f774f301e170d31333035303930333333333315a170d313430353039303333333315a'
Ver = 1
T = 2 (means ACK)
Code = 0x45 (2.05 Content)
Token = 0x9a (copied by server)
Options
  Option1 (Observe)
    Option Delta = 0x6 (option nr=6)
    Option Length = 0x1
    Option Value = 55 (55 > 12)
  Option2 (Content-Format)
    Option Delta = 0xC (option nr =6+6=12)
    Option Length = 0x2
    Option Value = TBD2
  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)

.
.
.

POST [2001:db8::2:1]:61616/est/sen (CON)(1:N1/0/256) {CSR req} -->

  <-- (ACK) (1:N1/0/256) (2:0/1/256) (2.04 Changed) (Cert resp)

POST [2001:db8::2:1]:61616/est/sen (CON)(2:1/0/256) -->

  <-- (ACK) (2:1/1/256) (2.04 Changed) (Cert resp)

.
.
.

POST [2001:db8::2:1]:61616/est/sen (CON)(2:N2/0/256) -->

  <-- (ACK) (2:N2/0/256) (2.04 Changed) (Cert resp)

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
Michael C. Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/

Martin Furuhed
Nexus Group

Email: martin.furuhed@nexusgroup.com

Shahid Raza
RISE SICS
Isafjordsgatan 22
Kista, Stockholm 16440
SE

Email: shahid@sics.se