ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

S. Gerdes
O. Bergmann
C. Bormann
Universitaet Bremen TZI

October 19, 2015

Delegated CoAP Authentication and Authorization Framework (DCAF)
draft-gerdes-ace-dcaf-authorize-04

Abstract

This specification defines a protocol for delegating client authentication and authorization in a constrained environment for establishing a Datagram Transport Layer Security (DTLS) channel between resource-constrained nodes. The protocol relies on DTLS to transfer authorization information and shared secrets for symmetric cryptography between entities in a constrained network. A resource-constrained node can use this protocol to delegate authentication of communication peers and management of authorization information to a trusted host with less severe limitations regarding processing power and memory.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] is a transfer protocol similar to HTTP which is designed for the special requirements of constrained environments. A serious problem with constrained devices is the realization of secure communication. The devices only have limited system resources such as memory, stable storage (such as disk space) and transmission capacity and often lack input/output devices such as keyboards or displays. Therefore, they are not readily capable of using common protocols. Especially authentication mechanisms are difficult to realize, because the lack of stable storage severely limits the number of keys the system can store. Moreover, CoAP has no mechanism for authorization.

[I-D.ietf-ace-actors] describes an architecture that is designed to help constrained nodes with authorization-related tasks by introducing less-constrained nodes. These Authorization Managers perform complex security tasks for their nodes such as managing keys for numerous devices, and enable the constrained nodes to enforce the authorization policies of their principals.

DCAF uses access tokens to implement this architecture. A device that wants to access an item of interest on a constrained node first has to gain permission in the form of a token from the node's Authorization Manager.

As fine-grained authorization is not always needed on constrained devices, DCAF supports an implicit authorization mode where no authorization information is exchanged.

The main goals of DCAF are the setup of a Datagram Transport Layer Security (DTLS) [RFC6347] channel with symmetric pre-shared keys (PSK) [RFC4279] between two nodes and to securely transmit authorization tickets.

1.1. Features

- Utilize DTLS communication with pre-shared keys.

- Authenticated exchange of authorization information.

- Simplified authentication on constrained nodes by handing the more sophisticated authentication over to less-constrained devices.

- Support of secure constrained device to constrained device communication.

- Authorization policies of the principals of both participating parties are ensured.

- Simplified authorization mechanism for cases where implicit authorization is sufficient.

- Using only symmetric encryption on constrained nodes.

1.2. 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 RFC 2119 [RFC2119].

Readers are expected to be familiar with the terms and concepts defined in [I-D.ietf-ace-actors].

1.2.1. Actors

Server (S): An endpoint that hosts and represents a CoAP resource.

Client (C): An endpoint that attempts to access a CoAP resource on the Server.

Server Authorization Manager (SAM): An entity that prepares and endorses authentication and authorization data for a Server.

Client Authorization Manager (CAM): An entity that prepares and endorses authentication and authorization data for a Client.

Authorization Manager (AM): An entity that is either a SAM or a CAM.

Client Overseeing Principal (COP): The principal that is in charge of the Client and controls permissions concerning authorized representations of a CoAP resource.

Resource Overseeing Principal (ROP): The principal that is in charge of the CoAP resource and controls its access permissions.

1.2.2. Other Terms

Resource (R): A CoAP resource.

Authorization information: Contains all information needed by S to decide if C is privileged to access a resource in a specific way.

Authentication information: Contains all information needed by S to decide if the entity in possession of a certain key is verified by SAM.

Access information: Contains authentication information and, if necessary, authorization information.

Access ticket: Contains the authentication and, if necessary, the authorization information needed to access a resource. A Ticket consists of the Ticket Face and the Client Information. The access ticket is a representation of the access information.

Ticket Face: The part of the ticket which is generated for the Server. It contains the authorization information and all information needed by the Server to verify that it was granted by SAM.

Client Information (CI): The part of the ticket which is generated for the Client. It contains the Verifier and optionally may contain authorization information that represent COP’s authorization policies for C.

Client Authorization Information (CAI): A data structure that describes the C’s permissions for S according to CAM, e.g., which actions C is allowed to perform on an R of S.

Server Authorization Information (SAI): A data structure that describes C’s permissions for S according to SAM, e.g., which actions C is allowed to perform on an R of S.

Verifier: The secret (e.g. a 128-bit PSK) shared between C and S. It enables C to validate that it is communicating with a certain S and vice versa.

Explicit authorization: SAM informs the S in detail which privileges are granted to the Client.

Implicit authorization: SAM authenticates the Client for the Server without specifying the privileges in detail. This can be used for flat or unrestricted authorization (cf section 4 of [I-D.ietf-ace-actors]).

2. System Overview

Within the DCAF Architecture each Server (S) has a Server Authorization Manger (SAM) which conducts the authentication and authorization for S. S and SAM share a symmetric key which has to be exchanged initially to provide for a secure channel. The mechanism used for this is not in the scope of this document.

To gain access to a specific resource on a S, a Client (C) has to request an access ticket from the SAM serving S either directly or, if it is a constrained device, using its Client Authorization Manager (CAM). In the following, we always discuss the CAM role separately, even if that is co-located within a (more powerful) C (see section Section 11 for details about co-located actors).

CAM decides if S is an authorized source for R according to the policies set by COP and in this case transmits the request to SAM. If SAM decides that C is allowed to access the resource according to the policies set by ROP, it generates a DTLS pre-shared key (PSK) for the communication between C and S and wraps it into an access ticket. For explicit access control, SAM adds the detailed access permissions to the ticket in a way that CAM and S can interpret. CAM checks if the permissions in the access ticket comply with COP's authorization policies for C, and if this is the case sends it to C. After C presented the ticket to S, C and S can communicate securely.

To be able to provide for the authentication and authorization services, an Authorization Manager has to fulfill several requirements:

- o AM must have enough stable storage (such as disk space) to store the necessary number of credentials (matching the number of Clients and Servers).
- o AM must possess means for user interaction, for example directly or indirectly connected input/output devices such as keyboard and

display, to allow for configuration of authorization information by the respective Principal.

- o AM must have enough processing power to handle the authorization requests for all constrained devices it is responsible for.

3. Protocol

The DCAF protocol comprises three parts:

1. transfer of authentication and, if necessary, authorization information between C and S;

2. transfer of access requests and the respective ticket transfer between C and CAM; and

3. transfer of ticket requests and the respective ticket grants between SAM and CAM.

3.1. Overview

In Figure 1, a DCAF protocol flow is depicted (messages in square brackets are optional):

```
 CAM                    C                    S                   SAM
  | <== DTLS chan. ==> |                    | <== DTLS chan. ==> |
  |                    | [Resource Req.-->] |                    |
  |                    |                    |                    |
  |                    | [<-- SAM Info.]    |                    |
  |                    |                    |                    |
  | <-- Access Req.    |                    |                    |
  |                    |                    |                    |
  | <==== TLS/DTLS channel (CAM/SAM Mutual Authentication) ====> |
  |                    |                    |                    |
  | Ticket Request   ----------------------------------------->  |
  |                    |                    |                    |
  | <------------------------------------------    Ticket Grant  |
  |                    |                    |                    |
  | Ticket Transf. --> |                    |                    |
  |                    |                    |                    |
  |                    | <== DTLS chan. ==> |                    |
  |                    | Auth. Res. Req. -> |                    |
```

Figure 1: Protocol Overview

To determine the SAM in charge of a resource hosted at the S, C MAY send an initial Unauthorized Resource Request message to S. S then denies the request and sends the address of its SAM back to C.

Instead of the initial Unauthorized Resource Request message, C MAY look up the desired resource in a resource directory (cf. [I-D.ietf-core-resource-directory]) that lists S's resources as discussed in Section 9.

Once C knows SAM's address, it can send a request for authorization to SAM using its own CAM. CAM and SAM authenticate each other and each determine if the request is to be authorized. If it is, SAM generates an access ticket for C. The ticket contains keying material for the establishment of a secure channel and, if necessary, a representation of the permissions C has for the resource. C keeps one part of the access ticket and presents the other part to S to prove its right to access. With their respective parts of the ticket, C and S are able to establish a secure channel.

The following sections specify how CoAP is used to interchange access-related data between S and SAM so that SAM can provide C and S with sufficient information to establish a secure channel, and simultaneously convey authorization information specific for this communication relationship to S.

Note: Special implementation considerations apply when one single entity takes the role of more than one actors. Section 11 gives additional advice on some of these usage scenarios.

This document uses Concise Binary Object Representation (CBOR, [RFC7049]) to express authorization information as set of attributes passed in CoAP payloads. Notation and encoding options are discussed in Section 5. A formal specification of the DCAF message format is given in Appendix A.

3.2. Unauthorized Resource Request Message

The optional Unauthorized Resource Request message is a request for a resource hosted by S for which no proper authorization is granted. S MUST treat any CoAP request as Unauthorized Resource Request message when any of the following holds:

- o The request has been received on an unprotected channel.
- o S has no valid access ticket for the sender of the request regarding the requested action on that resource.

- S has a valid access ticket for the sender of the request, but this does not allow the requested action on the requested resource.

Note: These conditions ensure that S can handle requests autonomously once access was granted and a secure channel has been established between C and S.

Unauthorized Resource Request messages MUST be denied with a client error response. In this response, the Server MUST provide proper SAM Information to enable the Client to request an access ticket from S’s SAM as described in Section 3.3.

The response code MUST be 4.01 (Unauthorized) in case the sender of the Unauthorized Resource Request message is not authenticated, or if S has no valid access ticket for C. If S has an access ticket for C but not for the resource that C has requested, S MUST reject the request with a 4.03 (Forbidden). If S has an access ticket for C but it does not cover the action C requested on the resource, S MUST reject the request with a 4.05 (Method Not Allowed).

Note: The use of the response codes 4.03 and 4.05 is intended to prevent infinite loops where a dumb Client optimistically tries to access a requested resource with any access token received from the SAM. As malicious clients could pretend to be C to determine C’s privileges, these detailed response codes must be used only when a certain level of security is already available which can be achieved only when the Client is authenticated.

3.3. SAM Information Message

The SAM Information Message is sent by S as a response to an Unauthorized Resource Request message (see Section 3.2) to point the sender of the Unauthorized Resource Request message to S’s SAM. The SAM information is a set of attributes containing an absolute URI (see Section 4.3 of [RFC3986]) that specifies the SAM in charge of S.

An optional field A lists the different content formats that are supported by S.

The message MAY also contain a timestamp generated by S.

Figure 2 shows an example for an SAM Information message payload using CBOR diagnostic notation. (Refer to Section 5 for a detailed description of the available attributes and their semantics.)

```
4.01 Unauthorized
Content-Format: application/dcaf+cbor
{SAM: "coaps://sam.example.com/authorize", TS: 168537,
 A: [ TBD1, ct_cose_msg ] }
```

Figure 2: SAM Information Payload Example

In this example, the attribute SAM points the receiver of this message to the URI "coaps://sam.example.com/authorize" to request access permissions. The originator of the SAM Information payload (i.e. S) uses a local clock that is loosely synchronized with a time scale common between S and SAM (e.g., wall clock time). Therefore, it has included a time stamp on its own time scale that is used as a nonce for replay attack prevention. Refer to Section 4.1 for more details concerning the usage of time stamps to ensure freshness of access tickets.

The content formats accepted by S are TBD1 (identifying 'application/dcaf+cbor' as defined in this document), and 'application/cose+cbor' defined in [I-D.ietf-cose-msg].

Editorial note: ct_cose_msg is to be replaced with the numeric value assigned for 'application/cose+cbor'.

The examples in this document are written in CBOR diagnostic notation to improve readability. Figure 3 illustrates the binary encoding of the message payload shown in Figure 2.

```
a2                                   # map(2)
    00                               # unsigned(0) (=SAM)
    78 21                            # text(33)
       636f6170733a2f2f73616d2e6578
       616d706c652e636f6d2f617574686f72
       697a65            # "coaps://sam.example.com/authorize"
    05                               # unsigned(5) (=TS)
    1a 00029259                      # unsigned(168537)
    0a                               # unsigned(10) (=A)
    82                               # array(2)
       19 03e6                       # unsigned(998) (=dcaf+cbor)
       19 03e7                       # unsigned(999) (=cose+cbor)
```

Figure 3: SAM Information Payload Example encoded in CBOR

3.3.1. Piggybacked Protected Content

For some use cases (such as sleepy nodes) it might be necessary to store sensor data on a server that might not belong to the same security domain. A client can retrieve the data from that server.

To be able to achieve the security objectives of the principles the data must be protected properly.

The server that hosts the stored data may respond to GET requests for this particular resource with a SAM Information message that contains the protected data as piggybacked content. As the server may frequently publish updates to the stored data, the URI of the authorization manager responsible for the protected data MAY be omitted and must be retrieved from a resource directory.

Once a requesting client has received the SAM Information Message with piggybacked content, it needs to request authorization for accessing the protected data. To do so, it constructs an Access Request as defined in Section 3.4. If access to the protected data is granted, the requesting client will be provided with cryptographic material to verify the integrity and authenticity of the piggybacked content and decrypt the protected data in case it is encrypted.

3.4. Access Request

To retrieve an access ticket for the resource that C wants to access, C sends an Access Request to its CAM. The Access Request is constructed as follows:

1. The request method is POST.

2. The request URI is set as described below.

3. The message payload contains a data structure that describes the action and resource for which C requests an access ticket.

The request URI identifies a resource at CAM for handling authorization requests from C. The URI SHOULD be announced by CAM in its resource directory as described in Section 9.

Note: Where capacity limitations of C do not allow for resource directory lookups, the request URI in Access Requests could be hard-coded during provisioning or set in a specific device configuration profile.

The message payload is constructed from the SAM information that S has returned in its SAM Information message (see Section 3.3) and information that C provides to describe its intended request(s). The Access Request MUST contain the following attributes:

1. Contact information for the SAM to use.

2. An absolute URI of the resource that C wants to access.

3. The actions that C wants to perform on the resource.

4. Any time stamp generated by S.

An example Access Request from C to CAM is depicted in Figure 4. (Refer to Section 5 for a detailed description of the available attributes and their semantics.)

```
POST client-authorize
Content-Format: application/dcaf+cbor
{
  SAM: "coaps://sam.example.com/authorize",
  SAI: ["coaps://temp451.example.com/s/tempC", 5],
  TS: 168537
}
```

Figure 4: Access Request Message Example

The example shows an Access Request message payload for the resource "/s/tempC" on the Server "temp451.example.com". Requested operations in attribute SAI are GET and PUT.

The attributes SAM (that denotes the Server Authorization Manager to use) and TS (a nonce generated by S) are taken from the SAM Information message from S.

The response to an Authorization Request is delivered by CAM back to C in a Ticket Transfer message.

3.5. Ticket Request Message

When CAM receives an Access Request message from C and COP specified authorization policies for C, CAM MUST check if the requested actions are allowed according to these policies. If all requested actions are forbidden, CAM MUST send a 4.03 response.

If no authorization policies were specified or some or all of the requested actions are allowed according to the authorization policies, CAM either returns a cached response or attempts to create a Ticket Request message. The Ticket Request message MAY contain all actions requested by C since CAM will add CAI in the Ticket Transfer Message if COP specified authorization policies (see Section 3.7).

CAM MAY return a cached response if it is known to be fresh according to Max-Age. CAM SHOULD NOT return a cached response if it expires in less than a minute.

If CAM does not send a cached response, it checks whether the request payload is of type "application/dcaf+cbor" and contains at least the fields SAM and SAI. CAM MUST respond with 4.00 (Bad Request) if the type is "application/dcaf+cbor" and any of these fields is missing or does not conform to the format described in Section 5.

If the payload is correct, CAM creates a Ticket Request message from the Access Request received from C as follows:

1. The destination of the Ticket Request message is derived from the "SAM" field that is specified in the Access Request message payload (for example, if the Access Request contained 'SAM: "coaps://sam.example.com/authz"', the destination of the Ticket Request message is sam.example.com).

2. The request method is POST.

3. The request URI is constructed from the SAM field received in the Access Request message payload.

4. The payload is copied from the Access Request sent by C.

To send the Ticket Request message to SAM a secure channel between CAM and SAM MUST be used. Depending on the URI scheme used in the SAM field of the Access Request message payload (the less-constrained devices CAM and SAM do not necessarily use CoAP to communicate with each other), this could be, e.g., a DTLS channel (for "coaps") or a TLS connection (for "https"). CAM and SAM MUST be able to mutually authenticate each other, e.g. based on a public key infrastructure. (Refer to Section 8 for a detailed discussion of the trust relationship between Client Authorization Managers and Server Authorization Managers.)

3.6. Ticket Grant Message

When SAM has received a Ticket Request message it has to evaluate the access request information contained therein. First, it checks whether the request payload is of type "application/dcaf+cbor" and contains at least the fields SAM and SAI. SAM MUST respond with 4.00 (Bad Request) for CoAP (or 400 for HTTP) if the type is "application/dcaf+cbor" and any of these fields is missing or does not conform to the format described in Section 5.

SAM decides whether or not access is granted to the requested resource and then creates a Ticket Grant message that reflects the result. To grant access to the requested resource, SAM creates an access ticket comprised of a Face and the Client Information as described in Section 4.

The Ticket Grant message then is constructed as a success response indicating attached content, i.e. 2.05 for CoAP, or 200 for HTTP, respectively. The payload of the Ticket Grant message is a data structure that contains the result of the access request. When access is granted, the data structure contains the Ticket Face and the Client Information. Face contains the SAI and the Session Key Generation Method. The CI at this point only consists of the Verifier.

The Ticket Grant message MAY provide cache-control options to enable intermediaries to cache the response. The message MAY be cached according to the rules defined in [RFC7252] to facilitate ticket retrieval when C has crashed and wants to recover the DTLS session with S.

SAM SHOULD set Max-Age according to the ticket lifetime in its response (Ticket Grant Message).

Figure 5 shows an example Ticket Grant message using CoAP. The Face/ Verifier information is transferred as a CBOR data structure as specified in Section 5. The Max-Age option tells the receiving CAM how long this ticket will be valid.

```
2.05 Content
Content-Format: application/dcaf+cbor
Max-Age: 86400
{ F: {
        SAI: [ "/s/tempC", 7 ],
        TS: 0("2013-07-10T10:04:12.391"),
        L:  86400,
        G: hmac_sha256
  },
  V: h’f89947160c73601c7a65cb5e08812026
       6d0f0565160e3ff7d3907441cdf44cc9’
}
```

Figure 5: Example Ticket Grant Message

A Ticket Grant message that declines any operation on the requested resource is illustrated in Figure 6. As no ticket needs to be issued, an empty payload is included with the response.

```
2.05 Content
Content-Format: application/dcaf+cbor
```

Figure 6: Example Ticket Grant Message With Reject

3.7. Ticket Transfer Message

A Ticket Transfer message delivers the access information sent by SAM in a Ticket Grant message to the requesting client C. The Ticket Transfer message is the response to the Access Request message sent from C to CAM and includes the ticket data from SAM contained in the Ticket Grant message.

The Authorization Information provided by SAM in the Ticket Grant Message may grant more permissions than C has requested. The authorization policies of COP and ROP may differ: COP might want restrict the resources C is allowed to access, and the actions that C is allowed to perform on the resource.

If COP defined authorization policies that concern the requested actions, CAM MUST add Authorization Information for C (CAI) to the CI that reflect those policies. Since C and CAM use a DTLS channel for communication, the autorization information does not need to be encrypted.

CAM includes the Face and the CI containing the verifier sent by SAM in the Ticket Transfer message. However, CAM MUST NOT include additional information SAM provided in CI. In particular, CAM MUST NOT include any CAI information provided by SAM, since CAI represents COP’s authorization policies that MUST NOT be provided by SAM.

Figure 7 shows an example Ticket Transfer message that conveys the permissions for actions GET, POST, PUT (but not DELETE) on the resource "/s/tempC" in field SAI. As CAM only wants to permit outbound GET requests, it restricts C’s permissions in the field CAI accordingly.

```
2.05 Content
Content-Format: application/dcaf+cbor
Max-Age: 86400
{ F: {
        SAI: [ "/s/tempC", 7 ],
        TS: 0("2013-07-10T10:04:12.391"),
        L:  86400,
        G: hmac_sha256
  },
  V: h'f89947160c73601c7a65cb5e08812026
       6d0f0565160e3ff7d3907441cdf44cc9'
  CAI: [ "/s/tempC", 1 ],
  TS: 0("2013-07-10T10:04:12.855"),
  L:  86400
}
```

Figure 7: Example Ticket Transfer Message

3.8. DTLS Channel Setup Between C and S

When C receives a Ticket Transfer message, it checks if the payload contains a face and a Client Information. With this information C can initiate establishment of a new DTLS channel with S. To use DTLS with pre-shared keys, C follows the PSK key exchange algorithm specified in Section 2 of [RFC4279], with the following additional requirements:

1. C sets the psk_identity field of the ClientKeyExchange message to the ticket Face received in the Ticket Transfer message.

2. C uses the ticket Verifier as PSK when constructing the premaster secret.

Note1: As S cannot provide C with a meaningful PSK identity hint in response to C's ClientHello message, S SHOULD NOT send a ServerKeyExchange message.

Note2: According to [RFC7252], CoAP implementations MUST support the ciphersuite TLS_PSK_WITH_AES_128_CCM_8 [RFC6655]. C is therefore expected to offer at least this ciphersuite to S.

Note3: The ticket is constructed by SAM such that S can derive the authorization information as well as the PSK (refer to Section 6 for details).

3.9. Authorized Resource Request Message

If the Client Information in the Ticket Transfer message contains CAI, C MUST ensure that it only sends requests that according to them are allowed. C therefore MUST check CAI, L and TS before every request. If CAI is no longer valid according to L, C MUST terminate the DTLS connection with S and re-request the CAI from CAM using an Access Request Message.

On the Server side, successful establishment of the DTLS channel between C and S ties the SAM authorization information contained in the psk_identity field to this channel. Any request that S receives on this channel is checked against these authorization rules. Incoming CoAP requests that are not Authorized Resource Requests MUST be rejected by S with 4.01 response as described in Section 3.2.

S SHOULD treat an incoming CoAP request as Authorized Resource Request if the following holds:

1. The message was received on a secure channel that has been established using the procedure defined in Section 3.8.

2. The authorization information tied to the secure channel is valid.

3. The request is destined for S.

4. The resource URI specified in the request is covered by the authorization information.

5. The request method is an authorized action on the resource with respect to the authorization information.

Note that the authorization information is not restricted to a single resource URI. For example, role-based authorization can be used to authorize a collection of semantically connected resources simultaneously. Implicit authorization also provides access rights to authenticated clients for all actions on all resources that S offers. As a result, C can use the same DTLS channel not only for subsequent requests for the same resource (e.g. for block-wise transfer as defined in [I-D.ietf-core-block] or refreshing observe-relationships [RFC7641]) but also for requests to distinct resources.

Incoming CoAP requests received on a secure channel according to the procedure defined in Section 3.8 MUST be rejected

1. with response code 4.03 (Forbidden) when the resource URI specified in the request is not covered by the authorization information, and

2. with response code 4.05 (Method Not Allowed) when the resource URI specified in the request covered by the authorization information but not the requested action.

Since SAM may limit the set of requested actions in its Ticket Grant message, C cannot know a priori if an Authorized Resource Request will succeed. If C repeatedly gets SAM Information messages as response to its requests, it SHOULD NOT send new Access Requests to CAM.

3.10. Dynamic Update of Authorization Information

Once a security association exists between a Client and a Resource Server, the Client can update the Authorization Information stored at the Server at any time. To do so, the Client creates a new Access Request for the intended action on the respective resource and sends this request to its CAM which checks and relays this request to the Server's SAM as described in Section 3.4.

Note: Requesting a new Access Ticket also can be a Client's reaction on a 4.03 or 4.05 error that it has received in response to an Authorized Resource Request.

Figure 8 depicts the message flow where C requests a new Access Tickets after a security association between C and S has been established using this protocol.

```
    CAM                    C                    S                   SAM
     | <== DTLS chan. ==> | <== DTLS chan. ==> | <== DTLS chan. ==> |
     |                    |                    |                    |
     |                    | [Unauth. R. Req->] |                    |
     |                    |[<- 4.0x+SAM Info.] |                    |
     |                    |                    |                    |
     | <-- Access Req.    |                    |                    |
     |                    |                    |                    |
     | <==== TLS/DTLS channel (CAM/SAM Mutual Authentication) ====> |
     |                    |                    |                    |
     | Ticket Request   ------------------------------------------> |
     |                    |                    |                    |
     | <------------------------------------------    Ticket Grant  |
     |                    |                    |                    |
     | Ticket Transf. --> |                    |                    |
     |                    |                    |                    |
     |                    | <== Update SAI ==> |                    |
```

Figure 8: Overview of Dynamic Update Operation

Processing the Ticket Request is done at the SAM as specified in Section 3.6, i.e. the SAM checks whether or not the requested operation is permitted by the Resource Principal’s policy, and then return a Ticket Grant message with the result of this check. If access is granted, the Ticket Grant message contains an Access Ticket comprised of a public Ticket Face and a private Ticket Verifier. This authorization payload is relayed by CAM to the Client in a Ticket Transfer Message as defined in Section 3.7.

The major difference between dynamic update of Authorization Information and the initial handshake is the handling of a Ticket Transfer message by the Client that is described in Section 3.10.1.

3.10.1. Handling of Ticket Transfer Messages

If the security association with S still exists and S has indicated support for session renegotiation according to [RFC5746], the ticket Face SHOULD be used to renegotiate the existing DTLS session. In this case, the ticket Face is used as psk_identity as defined in Section 3.8. Otherwise, the Client MUST perform a new DTLS handshake according to Section 3.8 that replaces the existing DTLS session.

After successful completion of the DTLS handshake S updates the existing SAM Authorization Information for C according to the contents of the ticket Face.

Note: No mutual authentication between C and S is required for dynamic updates when a DTLS channel exists that has been established as defined in Section 3.8. S only needs to verify the authenticity and integrity of the ticket Face issued by SAM which is achieved by having performed a successful DTLS handshake with the ticket Face as psk_identity. This could even be done within the existing DTLS session by tunneling a CoDTLS [I-D.schmertmann-dice-codtls] handshake.

4. Ticket

Access tokens in DCAF are tickets that consist of two parts, namely the Face and the Client Information (CI). SAM generates the ticket Face for S and the verifier that corresponds to the ticket Face for C. The verifier is included in the CI.

The Ticket is transmitted over CAM to C. C keeps the CI and sends the Face to S. CAM can add Client authorization information (CAI) for C to the CI if necessary.

S uses the information in the ticket Face to validate that it was generated by SAM and to authenticate and authorize the client. No additional information about the Client is needed, S keeps the Ticket Face as long as it is valid.

C uses the verifier to authenticate S. If CAM specified CAI, the client uses it to authorize the server.

The ticket is not required to contain a client or a server identifier. The ticket Face MAY contain an SAI identifier for revocation. The CI MAY contain a CAI identifier for revocation.

4.1. Face

Face is the part of the ticket that is generated by SAM for S. Face MUST contain all information needed for authorized access to a resource:

- o SAM Authorization Information (SAI)
- o A nonce

Optionally, Face MAY also contain:

- o A lifetime (optional)
- o A DTLS pre-shared key (optional)

- A SAI identifier (optional)

S MUST verify the integrity of Face, i.e. the information contained in Face stems from SAM and was not manipulated by anyone else. The integrity of Face can be ensured by various means. Face may be encrypted by SAM with a key it shares with S. Alternatively, S can use a mechanism to generate the DTLS PSK which includes Face. S generates the key from the Face it received. The correct key can only be calculated with the correct Face (refer to Section 6 for details).

Face MUST contain a nonce to verify that the contained information is fresh. As constrained devices may not have a clock, nonces MAY be generated using the clock ticks since the last reboot. To circumvent synchronization problems the timestamp MAY be generated by S and included in the first SAM Information message. Alternatively, SAM MAY generate the timestamp for the nonce. In this case, SAM and S MUST use a time synchronization mechanism to make sure that S interprets the timestamp correctly.

Face MAY contain an SAI identifier that uniquely identifies the SAI for S and SAM and can be used for revocation.

Face MAY be encrypted. If Face contains a DTLS PSK, the whole content of Face MUST be encrypted.

The ticket Face does not need to contain a client identifier.

4.2. Client Information

The CI part of the ticket is generated for C. It contains

- The Verifier generated by SAM

CI MAY additionally contain:

- CAI generated by CAM
- A nonce generated by CAM
- A lifetime generated by CAM
- A SAI identifier generated by CAM

CI MUST contain the verifier, i.e. the DTLS PSK for C. The Verifier MUST NOT be transmitted over unprotected channels.

Additionally, CI MAY contain CAI to provide the COP's authorization policies to C. If the CI contains CAI, CAM MUST add a nonce that enables C to validate that the information is fresh. CAM MAY use a timestamp as the nonce (see Section 4.1). CAM SHOULD add a lifetime to CI to limit the lifetime of the CAI. CAM MAY additionally add a CAI identifier to CI for revocating the CAI. The CAI identifier MUST uniquely identify the CAI for C and CAM.

4.3. Revocation

The existence of access tickets SHOULD be limited in time to avoid stale tickets that waste resources on S and C. This can be achieved either by explicit Revocation Messages to invalidate a ticket or implicitly by attaching a lifetime to the ticket.

The SAI in the ticket Face and the CAI in the CI need to be protected separately. CAM decides about the validity of the CAI while SAM is in charge of the validity of SAI. To be able to revoke the CAI, CAM SHOULD include a CAI identifier in the CI. SAM SHOULD include a SAI identifier in FACE to be able to revocate the SAI.

4.4. Lifetime

SAI and CAI MAY each have lifetime. SAM is responsible for defining the SAI lifetime, CAM is responsible for the CAI lifetime. If SAM sets a lifetime for SAI, SAM and S MUST use a time synchronization method to ensure that S is able to interpret the lifetime correctly. S SHOULD end the DTLS connection to C if the lifetime of a ticket has run out and it MUST NOT accept new requests. S MUST NOT accept tickets with an invalid lifetime.

If CAM provides CAI in the CI part of the ticket, CAM MAY add a lifetime for this CAI. If CI contains a lifetime, CAM and C MUST use a time synchronization method to ensure that C is able to interpret the lifetime correctly. C SHOULD end the DTLS connection to S and MUST NOT send new requests if the CAI in the ticket is no longer valid. C MUST NOT accept tickets with an invalid lifetime.

Note: Defining reasonable ticket lifetimes is difficult to accomplish. How long a client needs to access a resource depends heavily on the application scenario and may be difficult to decide for SAM.

4.4.1. Revocation Messages

SAM MAY revoke tickets by sending a ticket revocation message to S. If S receives a ticket revocation message, it MUST end the DTLS connection to C and MUST NOT accept any further requests from C.

If ticket revocation messages are used, S MUST check regularly if SAM is still available. If S cannot contact SAM, it MUST end all DTLS connections and reject any further requests from C.

Likewise, CAM MAY revoke tickets by sending a ticket revocation message to C. If C receives a CAI revocation message, it MUST end the DTLS connection to S and MUST NOT send any further requests to S.

If CAI revocation messages are used, C MUST check regularly if CAM is still available. If C cannot contact CAM, it MUST end all DTLS connections and MUST NOT send any more requests to S.

Note: The loss of the connection between S and SAM prevents all access to S. This might especially be a severe problem if SAM is responsible for several Servers or even a whole network.

5. Payload Format and Encoding (application/dcaf+cbor)

Various messages types of the DCAF protocol carry payloads to express authorization information and parameters for generating the DTLS PSK to be used by C and S. In this section, a representation in Concise Binary Object Representation (CBOR, [RFC7049]) is defined.

DCAF data structures are defined as CBOR maps that contain key value pairs. For efficient encoding, the keys defined in this document are represented as unsigned integers in CBOR, i. e. major type 0. For improved reading, we use symbolic identifiers to represent the corresponding encoded values as defined in Table 1.

Encoded Value	Key
0	SAM
1	SAI
2	CAI
3	E
4	K
5	TS
6	L
7	G
8	F
9	V
10	A
11	D
12	N

Table 1: DCAF field identifiers encoded in CBOR

The following list describes the semantics of the keys defined in DCAF.

SAM: Server Authorization Manager. This attribute denotes the Server Authorization Manager that is in charge of the resource specified in attribute R. The attribute’s value is a string that contains an absolute URI according to Section 4.3 of [RFC3986].

SAI: SAM Authorization Information. A data structure used to convey authorization information from SAM to S. It describes C’s permissions for S according to SAM, e.g., which actions C is allowed to perform on an R of S. The SAI attribute contains an AIF object as defined in [I-D.bormann-core-ace-aif]. C uses SAI for its Access Request messages.

CAI: CAM Authorization Information. A data structure used to convey authorization information from CAM to C. It describes the C's permissions for S according to CAM, e.g., which actions C is allowed to perform on an R of S. The CAI attribute contains an AIF object as defined in [I-D.bormann-core-ace-aif].

A: Accepted content formats. An array of numeric content formats from the CoAP Content-Formats registry (c.f. Section 12.3 of [RFC7252].

D: Protected Data. A binary string containing data that may be encrypted.

E: Encrypted Ticket Face. A binary string containing an encrypted ticket Face.

K: Key. A string that identifies the shared key between S and SAM that can be used to decrypt the contents of E. If the attribute E is present and no attribute K has been specified, the default is to use the current session key for the secured channel between S and SAM.

TS: Time Stamp. A time stamp that indicates the instant when the access ticket request was formed. This attribute can be used by the Server in an SAM Information message to convey a time stamp in its local time scale (e.g. when it does not have a real time clock with synchronized global time). When the attribute's value is encoded as a string, it MUST contain a valid UTC timestamp without time zone information. When encoded as integer, TS contains a system timestamp relative to the local time scale of its generator, usually S.

L: Lifetime. When in included in a ticket face, the contents of the L parameter denote the lifetime of the ticket. In combination with the protected data field D, this parameter denotes the lifetime of the protected data. When encoded as a string, L MUST denote the ticket's expiry time as a valid UTC timestamp without time zone information. When encoded as an integer, L MUST denote the ticket's validity period in seconds relative to TS.

N: Nonce. An initialization vector used in combination with piggybacked protected content.

G: DTLS PSK Generation Method. A numeric identifier for the method that S MUST use to derive the DTLS PSK from the ticket Face. This attribute MUST NOT be used when attribute V is present within the contents of F. This specification uses symbolic identifiers for improved readability. The corresponding numeric values encoded in

CBOR are defined in Table 2. A registry for these codes is defined in Section 13.1.

F: Ticket Face. An object containing the fields SAI, TS, and optionally G, L and V.

V: Ticket Verifier. A binary string containing the shared secret between C and S.

Encoded Value	Mnemonic	Support
0	hmac_sha256	mandatory
1	hmac_sha384	optional
2	hmac_sha512	optional

Table 2: CBOR encoding for DTLS PSK Key Generation Methods

5.1. Examples

The following example specifies a SAM that will be accessed using HTTP over TLS. The request URI is set to "/a?ep=%5B2001:DB8::dcaf:1234%5D" (hence denoting the endpoint address to authorize). TS denotes a local timestamp in UTC.

```
POST /a?ep=%5B2001:DB8::dcaf:1234%5D HTTP/1.1
Host: sam.example.com
Content-Type: application/dcaf+cbor
{SAM: "https://sam.example.com/a?ep=%5B2001:DB8::dcaf:1234%5D",
 SAI: ["coaps://temp451.example.com/s/tempC", 1],
 TS: 0("2013-07-14T11:58:22.923")}
```

The following example shows a ticket for the distributed key generation method (cf. Section 6.2), comprised of a Face (F) and a Verifier (V). The Face data structure contains authorization information SAI, a client descriptor, a timestamp using the local time scale of S, and a lifetime relative to S’s time scale.

The DTLS PSK Generation Method is set to hmac_sha256 denoting that the distributed key derivation is used as defined in Section 6.2 with SHA-256 as HMAC function.

The Verifier V contains a shared secret to be used as DTLS PSK between C and S.

```
HTTP/1.1 200 OK
Content-Type: application/dcaf+cbor
{
  F: {
       SAI: [ "/s/tempC", 1 ],
       TS: 2938749,
       L:  3600,
       G: hmac_sha256
     },
  V: h'48ae5a81b87241d81618f56cab0b65ec
       441202f81faabbe10075b20cb57fa939'
}
```

The Face may be encrypted as illustrated in the following example. Here, the field E carries an encrypted Face data structure that contains the same information as the previous example, and an additional Verifier. Encryption was done with a secret shared by SAM and S. (This example uses AES128_CCM with the secret { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f } and S's timestamp { 0x00, 0x2C, 0xD7, 0x7D } as nonce.) Line breaks have been inserted to improve readability.

The attribute K describes the identity of the key to be used by S to decrypt the contents of attribute E. Here, The value "key0" in this example is used to indicate that the shared session key between S and SAM was used for encrypting E.

```
{
  E: h'2e75eeae01b831e0b65c2976e06d90f4
       82135bec5efef3be3d31520b2fa8c6fb
       f572f817203bf7a0940bb6183697567c
       e291b03e9fca5e9cbdfa7e560322d4ed
       3a659f44a542e55331a1a9f43d7f',
  K: "key0",
  V: h'48ae5a81b87241d81618f56cab0b65ec
       441202f81faabbe10075b20cb57fa939'
}
```

The decrypted contents of E are depicted below (whitespace has been added to improve readability). The presence of the attribute V indicates that the DTLS PSK Transfer is used to convey the session key (cf. Section 6.1).

```
{
  F: {
       SAI: [ "/s/tempC", 1 ],
       TS: 2938749,
       L:  3600,
       G: hmac_sha256
     },
  V: h'48ae5a81b87241d81618f56cab0b65ec
       441202f81faabbe10075b20cb57fa939'
}
```

6. DTLS PSK Generation Methods

One goal of the DCAF protocol is to provide for a DTLS PSK shared between C and S. SAM and S MUST negotiate the method for the DTLS PSK generation.

6.1. DTLS PSK Transfer

The DTLS PSK is generated by AS and transmitted to C and S using a secure channel.

The DTLS PSK transfer method is defined as follows:

- o SAM generates the DTLS PSK using an algorithm of its choice
- o SAM MUST include a representation of the DTLS PSK in Face and encrypt it together with all other information in Face with a key K(SAM,S) it shares with S. How SAM and S exchange K(SAM,S) is not in the scope of this document. SAM and S MAY use their preshared key as K(SAM,S).
- o SAM MUST include a representation of the DTLS PSK in the Verifier.
- o As SAM and C do not have a shared secret, the Verifier MUST be transmitted to C using encrypted channels.
- o S MUST decrypt Face using K(SAM,S)

6.2. Distributed Key Derivation

SAM generates a DTLS PSK for C which is transmitted using a secure channel. S generates its own version of the DTLS PSK using the information contained in Face (see also Section 4.1).

The distributed key derivation method is defined as follows:

- SAM and S both generate the DTLS PSK using the information included in Face. They use an HMAC algorithm on Face with a shared key K(SAM,S). The result serves as the DTLS PSK. How SAM and S exchange K(SAM,S) is not in the scope of this document. They MAY use their preshared key as K(SAM,S). How SAM and S negotiate the used HMAC algorithm is also not in the scope of this document. They MAY however use the HMAC algorithm they use for their DTLS connection.

- SAM MUST include a representation of the DTLS PSK in the Verifier.

- As SAM and C do not have a shared secret, the Verifier MUST be transmitted to C using encrypted channels.

- SAM MUST NOT include a representation of the DTLS PSK in Face.

- SAM MUST NOT encrypt Face.

7. Authorization Configuration

For the protocol defined in this document, proper configuration of CAM and SAM is crucial. The principals that are in charge of the resource, S and SAM, and the principals that are in charge of C and CAM need to define the respective permissions. The data representation of these permissions are not in the scope of this document.

8. Trust Relationships

The constrained devices may be too constrained to manage complex trust relationships. Thus, DCAF does not require the constrained devices to perform complex tasks such as identifying a formerly unknown party. Each constrained device has a trust relationship with its respective AM. These less constrained devices are able to perform the more complex security tasks and can establish security associations with formerly unknown parties. The AMs hand down these security associations to their respective constrained device. The constrained devices require the help of their AMs for authentication and authorization.

C has a trust relationship with CAM: C trusts CAM to act in behalf of COP. S has a trust relationship with SAM: S trusts SAM to act in behalf of ROP. CAM trusts C to handle the data according to the CAI. SAM trusts S to protect resources according to the SAI. How the trust relationships between AMs and their respective constrained devices are established, is not in the scope of this document. It may be achieved by using a bootstrapping mechanism similar to [bergmann12] or by the means introduced in [I-D.gerdes-ace-a2a].

Additionally, SAM and CAM need to have established a trust relationship. Its establishment is not in the scope of this document. It fulfills the following conditions:

1. SAM and CAM have means to mutually authenticate each other (e.g., they might have a certificate of the other party or a PKI in which it is included)

2. If SAM requires information about the client from SAM, e.g. if SAM only wans to authorize certain types of devices, it can be sure that CAM correctly identifies these clients towards SAM and does not leak tickets that have been generated for a specific client C to another client.

SAM trusts C indirectly because it trusts CAM and CAM vouches for C. The DCAF Protocol does not provide any means for SAM to validate that a resource request stems from a specific C.

C indirectly entrusts SAM with some potentially confidential information, and trusts that SAM correctly represents S, because CAM trusts SAM.

CAM trusts S indirectly because it trusts SAM and SAM vouches for S.

C implicitly entrusts S with some potentially confidential information and trusts it to correctly represent R because it trusts CAM and because S can prove that it shares a key with SAM.

```
CAM <------------------> SAM

/|\                      /|\
 |                        |
\|/                      \|/

 C ....................   S
```

9. Listing Authorization Manager Information in a Resource Directory

CoAP utilizes the Web Linking format [RFC5988] to facilitate discovery of services in an M2M environment. [RFC6690] defines specific link parameters that can be used to describe resources to be listed in a resource directory [I-D.ietf-core-resource-directory].

9.1. The "auth-request" Link Relation

This section defines a resource type "auth-request" that can be used by clients to retrieve the request URI for a server's authorization service. When used with the parameter rt in a web link, "auth-request" indicates that the corresponding target URI can be used in a POST message to request authorization for the resource and action that are described in the request payload.

The Content-Format "application/dcaf+cbor with numeric identifier TBD1 defined in this specification MAY be used to express access requests and their responses.

The following example shows the web link used by CAM in this document to relay incoming Authorization Request messages to SAM. (Whitespace is included only for readability.)

```
<client-authorize>;rt="auth-request";ct=TBD1
                  ;title="Contact Remote Authorization Manager"
```

The resource directory that hosts the resource descriptions of S could list the following description. In this example, the URI "ep/node138/a/switch2941" is relative to the resource context "coaps://sam.example.com/", i.e. the Server Authorization Manager SAM.

```
<ep/node138/a/switch2941>;rt="auth-request";ct=TBD1;ep="node138"
                         ;title="Request Client Authorization"
                         ;anchor="coaps://sam.example.com/"
```

10. Examples

This section gives a number of short examples with message flows for the initial Unauthorized Resource Request and the subsequent retrieval of a ticket from SAM. The notation here follows the actors conventions defined in Section 1.2.1. The payload format is encoded as proposed in Section 5. The IP address of SAM is 2001:DB8::1, the IP address of S is 2001:DB8::dcaf:1234, and C's IP address is 2001:DB8::c.

10.1. Access Granted

This example shows an Unauthorized PUT request from C to S that is answered with a SAM Information message. C then sends a POST request to CAM with a description of its intended request. CAM forwards this request to SAM using CoAP over a DTLS-secured channel. The response from SAM contains an access ticket that is relayed back to CAM.

```
C --> S
PUT a/switch2941 [Mid=1234]
Content-Format: application/senml+json
{"e": [{"bv": "1"}]}

C <-- S
4.01 Unauthorized  [Mid=1234]
Content-Format: application/dcaf+cbor
{SAM: "coaps://[2001:DB8::1]/ep/node138/a/switch2941"}

C --> CAM
POST client-authorize [Mid=1235,Token="tok"]
Content-Format: application/dcaf+cbor
{
  SAM: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
  SAI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 4]
}

CAM --> SAM [Mid=23146]
POST ep/node138/a/switch2941
Content-Format: application/dcaf+cbor
{
  SAM: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
  SAI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 4]
}

CAM <-- SAM
2.05 Content  [Mid=23146]
Content-Format: application/dcaf+cbor
{ F: {
       SAI: ["a/switch2941", 5],
       TS: 0("2013-07-04T20:17:38.002"),
       G: hmac_sha256
     },
  V: h'7ba4d9e287c8b69dd52fd3498fb8d26d
       9503611917b014ee6ec2a570d857987a'
}

C <-- CAM
2.05 Content  [Mid=1235,Token="tok"]
Content-Format: application/dcaf+cbor
{ F: {
       SAI: ["a/switch2941", 5],
       TS: 0("2013-07-04T20:17:38.002"),
       G: hmac_sha256
     },
  V: h'7ba4d9e287c8b69dd52fd3498fb8d26d
       9503611917b014ee6ec2a570d857987a'
```

```
}

C --> S
ClientHello (TLS_PSK_WITH_AES_128_CCM_8)

C <-- S
ServerHello (TLS_PSK_WITH_AES_128_CCM_8)
ServerHelloDone

C --> S
ClientKeyExchange
  psk_identity=0xa301826c612f73776974636832393431
               0x0505c077323031332d30372d30345432
               0x303a31373a33382e3030320700

(C decodes the contents of V and uses the result as PSK)
ChangeCipherSpec
Finished

(S calculates PSK from SAI, TS and its session key
   HMAC_sha256(0xa301826c612f73776974636832393431
               0x0505c077323031332d30372d30345432
               0x303a31373a33382e3030320700,
               0x736563726574)
= 0x7ba4d9e287c8...
)

C <-- S
ChangeCipherSpec
Finished
```

10.2. Access Denied

This example shows a denied Authorization request for the DELETE operation.

```
C --> S
DELETE a/switch2941

C <-- S
4.01 Unauthorized
Content-Format: application/dcaf+cbor
{SAM: "coaps://[2001:DB8::1]/ep/node138/a/switch2941"}

C --> CAM
POST client-authorize
Content-Format: application/dcaf+cbor
{
  SAM: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
  SAI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 8]
}

CAM --> SAM
POST ep/node138/a/switch2941
Content-Format: application/dcaf+cbor
{
  SAM: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
  SAI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 8]
}

CAM <-- SAM
2.05 Content
Content-Format: application/dcaf+cbor

C <-- CAM
2.05 Content
Content-Format: application/dcaf+cbor
```

10.3. Access Restricted

This example shows a denied Authorization request for the operations GET, PUT, and DELETE. SAM grants access for PUT only.

```
CAM --> SAM
POST ep/node138/a/switch2941
Content-Format: application/dcaf+cbor
{
  SAM: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
  SAI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 13]
}

CAM <-- SAM
2.05 Content
Content-Format: application/dcaf+cbor
{ F: {
       SAI: ["a/switch2941", 5],
       TS: 0("2013-07-04T21:33:11.930"),
       G: hmac_sha256
     },
  V: h'c7b5774f2ddcbd548f4ad74b30a1b2e5
       b6b04e66a9995edd2545e5a06216c53d'
}
```

10.4. Implicit Authorization

This example shows an Authorization request using implicit authorization. CAM initially requests the actions GET and POST on the resource "coaps://[2001:DB8::dcaf:1234]/a/switch2941". SAM returns a ticket that has no SAI field in its ticket Face, hence implicitly authorizing C.

```
CAM --> SAM
POST ep/node138/a/switch2941
Content-Format: application/dcaf+cbor
{
   SAM: "coaps://[2001:DB8::1]/ep/node138/a/switch2941",
   SAI: ["coaps://[2001:DB8::dcaf:1234]/a/switch2941", 3]
}

CAM <-- SAM
2.05 Content
Content-Format: application/dcaf+cbor
{ F: {
       TS: 0("2013-07-16T10:15:43.663"),
       G: hmac_sha256
      },
  V: h'4f7b0e7fdcc498fb2ece648bf6bdf736
       61a6067e51278a0078e5b8217147ea06'
}
```

11. Specific Usage Scenarios

The general DCAF architure outlined in Section 3.1 illustrates the various actors who participate in the message exchange for authenticated authorization. The message types defined in this document cover the most general case where all four actors are separate entities that may or may not reside on the same device.

Special implementation considerations apply when one single entity takes the role of more than one actor. This section gives advice on the most common usage scenarios where the Client Authorization Manager and Client, the Server Authorization Manager and Server or both Authorization Managers reside on the same (less-constrained) device and have a means of secure communication outside the scope of this document.

11.1. Combined Authorization Manager and Client

When CAM and C reside on the same (less-constrained) device, the Access Request and Ticket Transfer messages can be substituted by other means of secure communication. Figure 9 shows a simplified message exchange for a combined CAM+C device.

```
CAM+C                 S                 SAM
 |                    | <== DTLS chan. ==> |
 | [Resource Req.-->] |                    |
 |                    |                    |
 |  [<-- SAM Info.]   |                    |
 |                    |                    |
 | <==== TLS/DTLS chan. (Mutual Auth) ===> |
 |                    |                    |
 | Ticket Request   ---------------------> |
 |                    |                    |
 | <---------------------    Ticket Grant  |
 |                    |                    |
 | <== DTLS chan. ==> |                    |
 | Auth. Res. Req. -> |                    |
```

Figure 9: Combined Client Authorization Manager and Client

11.1.1. Creating the Ticket Request Message

When CAM+C receives an SAM Information message as a reaction to an Unauthorized Request message, it creates a Ticket Request message as follows:

1. The destination of the Ticket Request message is derived from the authority information in the URI contained in field "SAM" of the SAM Information message payload.

2. The request method is POST.

3. The request URI is constructed from the SAM field received in the SAM Information message payload.

4. The payload contains the SAM field from the SAM Information message, an absolute URI of the resource that CAM+C wants to access, the actions that CAM+C wants to perform on the resource, and any time stamp generated by S that was transferred with the SAM Information message.

11.1.2. Processing the Ticket Grant Message

Based on the Ticket Grant message, CAM+C is able to establish a DTLS channel with S. To do so, CAM+C sets the psk_identity field of the DTLS ClientKeyExchange message to the ticket Face received in the Ticket Grant message and uses the ticket Verifier as PSK when constructing the premaster secret.

11.2. Combined Client Authorization Manager and Server Authorization Manager

In certain scenarios, CAM and SAM may be combined to a single entity that knows both, C and S, and decides if their actions are authorized. Therefore, no explicit communication between CAM and SAM is necessary, resulting in omission of the Ticket Request and Ticket Grant messages. Figure 10 depicts the resulting message sequence in this simplified architecture.

```
     C                 CAM+SAM                S
     | <== DTLS chan. ==> | <== DTLS chan. ==> |
     |                    |                    |
     | [Resource Req.----------------------->] |
     |                    |                    |
     | [<-------------------- SAM Information] |
     |                    |                    |
     | Access Request --> |                    |
     |                    |                    |
     | <-- Ticket Transf. |                    |
     |                    |                    |
     | <===========  DTLS channel ===========> |
     |                    |                    |
     | Authorized Resource Request ----------> |
```

Figure 10: Combined Client Authorization Manager and Server Authorization Manager

11.2.1. Processing the Access Request Message

When receiving an Access Request message, CAM+SAM performs the checks specified in Section 3.5 and returns a 4.00 (Bad Request) response in case of failure. Otherwise, if the checks have succeeded, CAM+SAM evaluates the contents of Access Request message as described in Section 3.6.

The decision on the access request is performed by CAM+SAM with respect to the stored policies. When the requested action is permitted on the respective resource, CAM+SAM generates an access ticket as outlined in Section 4.1 and creates a Ticket Transfer message to convey the access ticket to the Client.

11.2.2. Creating the Ticket Transfer Message

A Ticket Transfer message is constructed as a 2.05 response with the access ticket contained in its payload. The response MAY contain a Max-Age option to indicate the ticket’s lifetime to the receiving Client.

This specification defines a CBOR data representation for the access ticket as illustrated in Section 3.6.

11.3. Combined Server Authorization Manager and Server

If SAM and S are colocated in one entity (SAM+S), the main objective is to allow CAM to delegate access to C. Accordingly, the authorization information could be replaced by a nonce internal to SAM+S. (TBD.)

```
   CAM                   C                  SAM+S
     | <== DTLS chan. ==> |                    |
     |                    | [Resource Req.-->] |
     |                    |                    |
     |                    |  [<-- SAM Info.]   |
     |                    |                    |
     | <-- Access Req.    |                    |
     |                    |                    |
     | <========= TLS/DTLS channel  =========> |
     |                    |                    |
     | Ticket Request   ---------------------> |
     |                    |                    |
     | <----------------------    Ticket Grant  |
     |                    |                    |
     | Ticket Transf. --> |                    |
     |                    |                    |
     |                    | <== DTLS chan. ==> |
     |                    | Auth. Res. Req. -> |
```

Figure 11: Combined Server Authorization Manager and Server

12. Security Considerations

As this protocol builds on transitive trust between Authorization Managers as mentioned in Section 8, SAM has no direct means to validate that a resource request originates from C. It has to trust CAM that it correctly vouches for C and that it does not give authorization tickets meant for C to another client nor disclose the contained session key.

The Authorization Managers also could constitute a single point of failure. If the Server Authorization Manager fails, the resources on all Servers it is responsible for cannot be accessed any more. If a Client Authorization Manager fails, all clients it is responsible are not able to access resources on a Server. Thus, it is crucial for large networks to use Authorization Managers in a redundant setup.

13. IANA Considerations

The following registrations are done following the procedure specified in [RFC6838].

Note to RFC Editor: Please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification.

13.1. DTLS PSK Key Generation Methods

A sub-registry for the values indicating the PSK key generation method as contents of the field G in a payload of type application/dcaf+cbor is defined. Values in this sub-registry are numeric integers encoded in Concise Binary Object Notation (CBOR, [RFC7049]). This document follows the notation of [RFC7049] for binary values, i.e. a number starts with the prefix "0b". The major type is separated from the actual numeric value by an underscore to emphasize the value’s internal structure.

Initial entries in this sub-registry are as follows:

Encoded Value	Name	Reference
0b000_00000	hmac_sha256	[RFC-XXXX]
0b000_00001	hmac_sha384	[RFC-XXXX]
0b000_00010	hmac_sha512	[RFC-XXXX]

Table 3: DTLS PSK Key Generation Methods

New methods can be added to this registry based on designated expert review according to [RFC5226].

(TBD: criteria for expert review.)

13.2. dcaf+cbor Media Type Registration

Type name: application

Subtype name: dcaf+cbor

Required parameters: none

Optional parameters: none

Encoding considerations: Must be encoded as using a subset of the encoding allowed in [RFC7049]. Specifically, only the primitive data types String and Number are allowed. The type Number is restricted to unsigned integers (i.e., no negative numbers, fractions or exponents are allowed). Encoding MUST be UTF-8. These restrictions simplify implementations on devices that have very limited memory capacity.

Security considerations: TBD

Interoperability considerations: TBD

Published specification: [RFC-XXXX]

Applications that use this media type: TBD

Additional information:

Magic number(s): none

File extension(s): dcaf

Macintosh file type code(s): none

Person & email address to contact for further information: TBD

Intended usage: COMMON

Restrictions on usage: None

Author: TBD

Change controller: IESG

13.3. CoAP Content Format Registration

This document specifies a new media type application/dcaf+cbor (cf. Section 13.2). For use with CoAP, a numeric Content-Format identifier is to be registered in the "CoAP Content-Formats" sub-registry within the "CoRE Parameters" registry.

Note to RFC Editor: Please replace all occurrences of "RFC-XXXX" with the RFC number of this specification.

Media type	Encoding	Id.	Reference
application/dcaf+cbor	-	TBD1	[RFC-XXXX]

14. Acknowledgements

The authors would like to thank Renzo Navas for his valuable input and feedback.

15. References

15.1. Normative References

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <http://www.rfc-editor.org/info/rfc4279>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <http://www.rfc-editor.org/info/rfc5746>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <http://www.rfc-editor.org/info/rfc6838>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

15.2. Informative References

[I-D.bormann-core-ace-aif]
Bormann, C., "An Authorization Information Format (AIF) for ACE", draft-bormann-core-ace-aif-03 (work in progress), July 2015.

[I-D.gerdes-ace-a2a]
Gerdes, S., "Managing the Authorization to Authorize in the Lifecycle of a Constrained Device", draft-gerdes-ace-a2a-01 (work in progress), September 2015.

[I-D.greevenbosch-appsawg-cbor-cddl]
Vigano, C. and H. Birkholz, "CBOR data definition language (CDDL): a notational convention to express CBOR data structures", draft-greevenbosch-appsawg-cbor-cddl-07 (work in progress), October 2015.

[I-D.ietf-ace-actors]
Gerdes, S., Seitz, L., Selander, G., and C. Bormann, "An architecture for authorization in constrained environments", draft-ietf-ace-actors-02 (work in progress), October 2015.

[I-D.ietf-core-block]
Bormann, C. and Z. Shelby, "Block-wise transfers in CoAP", draft-ietf-core-block-18 (work in progress), September 2015.

[I-D.ietf-core-resource-directory]
Shelby, Z., Koster, M., Bormann, C., and P. Stok, "CoRE Resource Directory", draft-ietf-core-resource-directory-05 (work in progress), October 2015.

[I-D.ietf-cose-msg]
Schaad, J., "CBOR Encoded Message Syntax", draft-ietf-cose-msg-06 (work in progress), October 2015.

[I-D.schmertmann-dice-codtls]
Schmertmann, L., Hartke, K., and C. Bormann, "CoDTLS: DTLS handshakes over CoAP", draft-schmertmann-dice-codtls-01 (work in progress), August 2014.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, DOI 10.17487/RFC6655, July 2012, <http://www.rfc-editor.org/info/rfc6655>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <http://www.rfc-editor.org/info/rfc6690>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <http://www.rfc-editor.org/info/rfc7641>.

[bergmann12] Bergmann, O., Gerdes, S., Schaefer, S., Junge, F., and C. Bormann, "Secure Bootstrapping of Nodes in a CoAP Network", IEEE Wireless Communications and Networking Conference Workshops (WCNCW), April 2012.

Appendix A. CDDL Specification

This appendix shows a formal specification of the DCAF messaging format using the CBOR data definition language (CDDL) [I-D.greevenbosch-appsawg-cbor-cddl]:

```
dcaf-msg = sam-information-msg
         / access-request-msg
         / ticket-transfer-msg
         / ticket-grant-msg

sam-information-msg = { sam, ? full-timestamp, ? accepted-formats,
                        ? piggybacked }

access-request-msg = { sam, sam-ai, full-timestamp }

ticket-transfer-msg = { face-or-encrypted, verifier }
face-or-encrypted = ( face | encrypted-face )
face = ( F => { sam-ai, limited-timestamp, lifetime, psk-gen } )
verifier = ( V => shared-secret )
shared-secret = bstr
F   = 8
V   = 9

encrypted-face = ( E => bstr, K => tstr )
E   = 3
K   = 4
```

```
   ticket-grant-msg    = { face-or-encrypted, verifier, ? client-info }
   client-info = ( cam-ai, full-timestamp, lifetime)

   sam = (SAM => abs-uri)
   SAM = 0
   abs-uri = tstr ; .regexp "______"

   sam-ai = ( SAI => [* auth-info])
   SAI = 1
   auth-info = ( uri : tstr, mask : 0..15 )

   cam-ai = ( CAI => [* auth-info])
   CAI = 2

   full-timestamp = ( TS => date)
   TS  = 5
   date = tdate / localdate
   localdate = uint
   limited-timestamp = ( TS => localdate)

   accepted-formats = ( A => [+ content-format] )
   content-format = uint ; valid entry from CoAP content format registry
   A=10

   piggybacked = ( data, lifetime, nonce )
   data = ( D => bstr )
   none = ( N => bstr )
   lifetime = ( L => period)
   period = uint ; in seconds
   L   = 6
   D   = 11
   N   = 12

   psk-gen = ( G => mac-algorithm)
   G   = 7
   mac-algorithm = &( hmac-sha256: 0, hmac-sha384: 1, hmac-sha512: 2 )
```

Authors' Addresses

Stefanie Gerdes
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63906
Email: gerdes@tzi.org

Olaf Bergmann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63904
Email: bergmann@tzi.org

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org