ACE Working Group
Internet Draft
Intended status: Informational
Expires: January 2016

S. Gerdes
O. Bergmann
C. Bormann
Universität Bremen TZI
July 06, 2015

Examples for Using DCAF with less constrained devices
draft-gerdes-ace-dcaf-examples-00

Abstract

Constrained nodes are devices which are limited in terms of processing power, memory, non-volatile storage and transmission capacity. Due to these constraints, commonly used security protocols are not easily applicable. Nevertheless, an authentication and authorization solution is needed to ensure the security of these devices.

The Delegated CoAP Authorization Framework (DCAF) specifies how resource-constrained nodes can delegate defined authentication- and authorization-related tasks to less-constrained devices called Authorization Managers, thus limiting the hardware requirements of the security solution for the constrained devices.

To realize the vision of “one Internet for all”, constrained devices need to securely establish trust relationships with less constrained devices. This document lists examples for using DCAF with less constrained devices.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

See abstract.

2. Terminology

- Readers should be familiar with the concepts introduced in [I-D.gerdes-ace-dcaf-authorize]

To reduce the confusion between OpenID and DCAF services, we are using the following terminology:

- 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.
- Server (S): An endpoint that hosts and represents a resource.
- Client (C): An endpoint that attempts to access a resource on the Server.

3. Example 1: Posting Sensor Values on a Website using OpenID Connect

To illustrate this example, we assume the following scenario: Sarah has a constrained device that is equipped with a temperature sensor. During a heat wave, she wants her device to continuously post the current room temperature in her office in the blog of her social media account to inform others of her working conditions. She wants to use OpenID Connect to log into CAM and establish a trust relationship between the temperature sensor and her blog.

The temperature sensor that is acting as a CoAP client (C) in this scenario is associated with a Client Authorization Manager (CAM) that helps C with authentication and authorization and provides a user interface to Sarah for configuring C. The blog on her social media account acts as a DCAF server (S).

In this first example, Sarah is registered with an OpenID provider (OP) that CAM accepts for authentication and which additionally acts as an Authorization Server (AS) for her social media service. Moreover, AS is compatible with DCAF and acts as a DCAF server authorization manager (SAM). For requesting access to the blog, OP defines the scope parameter coaps://blog.socialmedia.example.com. Sarah uses the browser on her smartphone as a User Agent (UA) to initiate the authentication and authorization process.

This example illustrates the authentication and authorization using the OpenID Connect Authorization Code Flow as described in section 3.1 of [OpenID-Connect].

Sarah uses her UA to request CAM to configure C to access her blog. To authenticate Sarah, CAM generates an Authentication Request (cf. section 3.1.2.1 of [OpenID-Connect]). Since CAM needs to obtain authorization for accessing Sarah’s Blog for C, it also adds the scope parameter coaps://blog.socialmedia.example.com. CAM sends the Authentication Request to Sarah’s UA, thereby triggering it to send an Authentication Request to OP’s Authorization Endpoint.

```
HTTP/1.1 302 Found
Location: https://server.example.com/authorize?
  response_type=code
  &scope=openid%20coaps://blog.socialmedia.example.com
  &client_id=s6BhdRkqt3
  &state=af0ifjsldkj
  &redirect_uri=https%3A%2F%2Fcam.example.org%2Flogin
```

After authenticating Sarah and requesting her permission (for details please consult sections 3.1.2.3 and 3.1.2.4 of [OpenID-Connect]), the OP sends back a successful Authentication Response that contains the parameter code. The code can then be used by CAM to send a Token Request to the Token Endpoint which responds with an ID token, an access token and a refresh token. CAM uses the ID token and access token to authenticate Sarah. The refresh token is then used to request an additional access token for accessing the blog.

CAM sends the refresh token to the Token Endpoint with the scope coaps://blog.socialmedia.example.com. The Token Endpoint validates the refresh token and makes sure that the requested scope does fit to the original grant. The Token Endpoint speaks both DCAF and OAuth and acts as SAM. It generates a ticket including a verifier and sends it back to CAM. CAM transmits the ticket to its client and instructs it to access the blog. The client then uses the ticket to establish a DTLS connection with the blog.

Figure 1 illustrates the resulting message flow.

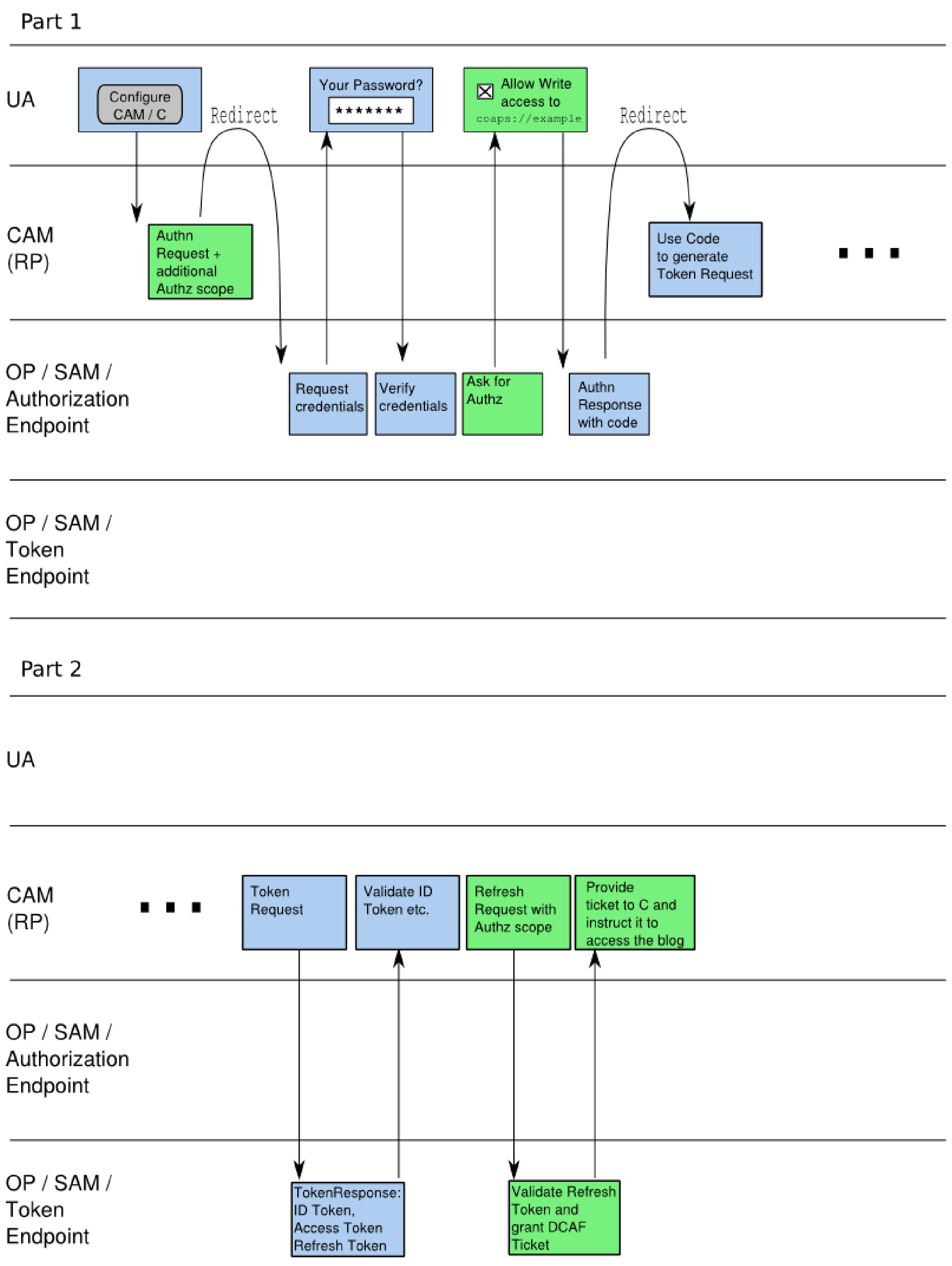
Part 1
UA
CAM (RP)
OP / SAM / Authorization Endpoint
OP / SAM / Token Endpoint
Configure CAM / C
Redirect
Your Password?
****
Allow Write access to coaps://example
Redirect
Authn Request + additional Authz scope
Use Code to generate Token Request
Request credentials
Verify credentials
Ask for Authz
Authn Response with code
Part 2
UA
CAM (RP)
OP / SAM / Authorization Endpoint
OP / SAM / Token Endpoint
Token Request
Validate ID Token etc.
Refresh Request with Authz scope
Provide ticket to C and instruct it to access the blog
TokenResponse: ID Token, Access Token Refresh Token
Validate Refresh Token and grant DCAF Ticket

Figure 1: Authentication and Authorization Flow

4. Security Considerations

TBD

5. IANA Considerations

None

6. References

6.1 Normative References

[I-D.gerdes-ace-dcaf-authorize]	Gerdes, S., Bergmann, O., and C. Bormann, "Delegated CoAP Authentication and Authorization Framework (DCAF)", Internet-Draft draft-gerdes-ace-dcaf-authorize-02 (work in progress), March 2015.

6.2 Informative References

[OpenID-Connect]	Sakimura, N., Bradley, J., Jones, M., de Madeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014.

Authors' Addresses

Stefanie Gerdes
Universität Bremen TZI
Postfach 330440
Bremen, D-28359
Germany
Phone: +49-421-218-63906
EMail: gerdes@tzi.org

Olaf Bergmann
Universität Bremen TZI
Postfach 330440
Bremen, D-28359
Germany
Phone: +49-421-218-63904
EMail: bergmann@tzi.org

Carsten Bormann
Universität Bremen TZI
Postfach 330440
Bremen, D-28359
Germany
Phone: +49-421-218-63921
EMail: cabo@tzi.org